Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Dokumentácia k tímovému projektu

Riadenie projektu

Vedúci tímu: Mgr. Martin Sabo, PhD., Ing. Marta Šoltésová Prnová, PhD.
Číslo tímu: 05
Kontakt: mikasa.fiit@gmail.com

Obsah

1 Úvod

Tento dokument opisuje postupy pri riadení tímu na projekte Automatické rozpoznávanie spektier.

V nasledujúcich kapitolách sú opísané role jednotlivých členov tímu a ich manažérska pozícia, z ktorej vyplývajú ich zodpovednosti na projekte, prehľad podielu práce na dokumentácii, opis spôsobov aplikácie manažmentov a zavádzania postupov v tíme, pokrok tímu na projekte počas jednotlivých šprintov a evidencia dodaných používateľských scenárov a globálna retrospektíva.

2 Role členov a podiel práce na dokumentácii

2.1 Role členov tímu

Meno a Priezvisko	Manažér	Rola
Simona Klučková	Manažér dátovej analýzy	Data scientist
Maroš Kollár	Manažér testovania a spracovania obrazových dát	Data scientist
Jakub Kučečka	Manažér kvality kódu	Developer
Zuzana Popovcová	Manažér úloh a neurónových sietí	Software architect
Mária Rajníková	Manažér vývoja klasifikačných metód	Data scientist
Alena Valová	Manažér dokumentácie	Scrum master

2.2 Podiel práce na dokumentácii

Meno a Priezvisko	Percentuálny podiel
Simona Klučková	17%
Maroš Kollár	17%
Jakub Kučečka	16%
Zuzana Popovcová	17%
Mária Rajníková	16%
Alena Valová	17%

3 Aplikácie manažmentov

Pre aplikáciu manažmentov v tíme využívame metodológie. Ide o dokument, v ktorom si definujeme pravidlá a postupy a procesy v tíme. Zoznam metodík nájdete na stránke tímu.

Metodiky sú priebežne aktualizované a dopĺňané. V každej metodike sa preto nachádza história verzií, ktorá nesie informácie o vykonaných zmenách v dokumente.

3.1 Manažment komunikácie

Na komunikáciu v tíme využívame viacero komunikačných nástrojov. Hlavným nástrojom je Microsoft Teams. Tento nástroj je využívaný na komunikáciu členov tímu a ich stretnutia.

Najvyužívanejším kanálom Microsoft Teams je kanál *Stand-up*, ktorý používame namiesto daily stand-up.

Na komunikáciu s vedúcimi projektu (ďalej len PO) využívame viacero komunikačných nástrojov. Primárne však využívame Google Meet.

Všetky komunikačné nástroje a ich kanály, ktoré v tíme využívame nájdete v prílohe C.

3.2 Manažment vývoja

Hlavným programovacím jazykom pre vývoj projektu je jazyk Python. Odporúčaným vývojovým prostredím pre členov tímu je Pycharm.

Pre tvorbu kvalitného a dobre štrukturovaného kódu, sme si zadefinovali pravidlá vývoja v Metodike vývoja v Pythone. V rámci metodiky sme určili štruktúru pracovných adresárov projektu a základné pravidlá pre tvorbu jednotného a dobre štrukturovaného kódu.

Podobne popísané pravidlá vývoja v jazyku Python sú popísané v dokumente Metodika vývoja v Python-e. Tento dokument nájdete v prílohe D.

3.3 Manažment testovania

Dôležitou súčasťou vývoja je testovanie. Napomáha k doručeniu validných a nezávadných častí softvéru. Pre validáciu dodaných riešení sú využívané akceptačné testy a pre odhalenie chýb sú vytvárané jednotkové testy. Jednotkové testy budú vykonávané pomocou knižnice *pytest*.

Presnejšie zadefinované pravidlá použitia týchto testov a ich správa v projekte je zadefinovaná v dokumente Metodika testovania. Metodiku nájdete v prílohe E

3.4 Manažment splnenia úlohy

Pre zadefinovanie pravidiel, kedy je činnosť na úlohe dokončená sme vytvorili dokument Metodika splnenej úlohy, ktorý nájdete v prílohe F. V metodike sme si zadefinovali pravidlá, kedy bude úloha, používateľský príbeh a projekt považovaný za úplne dokončený.

3.5 Manažment práce s úlohami

Hlavným nástrojom pre manažment úloh na projekte je Jira. Tento nástroj využívame na plánovanie práce na jednotlivých šprintoch. Vytvárame v ňom úlohy, ktoré je potrebné realizovať na projekte. Všetky úlohy sú vytvárané v backlogu. Následne na spoločnom stretnutí ohodnocujeme úlohy bodmi a zaraďujeme ich podľa priority do šprintu.

Pre bližšie zadefinovanie stavov úloh, ich manipuláciu a spôsobu vytvárania bola vytvorená Metodika práce s úlohami. Celú metodiku nájdete v prílohe G.

3.6 Manažment verziovania programu

Pre umožnenie vytvárania verzii programu a paralelného pracovania na projekte používame GitHub.

Všetky pravidlá pre manipuláciu s týmto nástrojom, pomenovanie vetiev a potvrdzujúcich správ sú bližšie definované v dokumente Metodika verziovania programu. Celú metodiku nájdete v prílohe H.

3.7 Manažment vývoja aplikácie

Unifikovanie a nastavenie pravidiel v rámci vývoja je dôležitou súčasťou projektu, nakoľko napomáha všetkým vývojárom lepšie zladenie a prehľad v projekte aj kóde samotnom. V rámci metodiky vývoja aplikácie sú spísané normy vývoja, ktorými sa musí riadiť celý tím pre ďalšie zjednodušenie manipulácie. Konkrétne sa jedná o názvoslovie súborov, rozvrhnutie adresárov, normy kódových konštrukcií a súborov na základe programovacieho jazyka a iné. Celú metodiku nájdete v prílohe I.

4 Sumarizácie šprintov

4.1 Šprint č.1

Názov šprintu	Starohutský vodopád
Trvanie	14.10.2020 - 23.10.2020

Prvý šprint bol zameraný na zavádzanie procesov v tíme, vytvorenie reprezentačného webu a na urgenciu vedúceho (ďalej PO) bola do šprintov zaradená aj práca na klasifikátore. User stories zahrnuté do šprintu boli ohodnotené na 67 story points (ďalej SP).

Spoločne sme v šprinte zaviedli procesy prostredníctvom napísania metodík. Pravidlá v napísaných metodikách sa podarilo počas šprintu rýchlo integrovať s výnimkou metodiky vývoja v Pythone a metodiky verziovania programu. Integrácia pravidiel týchto metodík bola podmienená ukončením všetkých implementačných úloh. Počas šprintu sa podarilo úspešne vytvoriť a odovzdať prihlášku na TP-cup.

V rámci práce na projekte bolo do šprintu zaradené spracovanie dát a vytvorenie prvotného klasifikátora. Všetky úlohy na projekte boli zvládnuté výborne. PO bol s výsledkami prvotného klasifikátora spokojný a nemal žiadne výhrady.

Najviac SP bolo pridelených v šprinte na vytvorenie webovej stránky. Stránka bola úspešne vytvorená a nasadená.

Najväčší podiel bodov v šprinte získal Jakub Kučečka, ktorý vytváral webovú stránku a odviedol pri tom skvelú prácu.

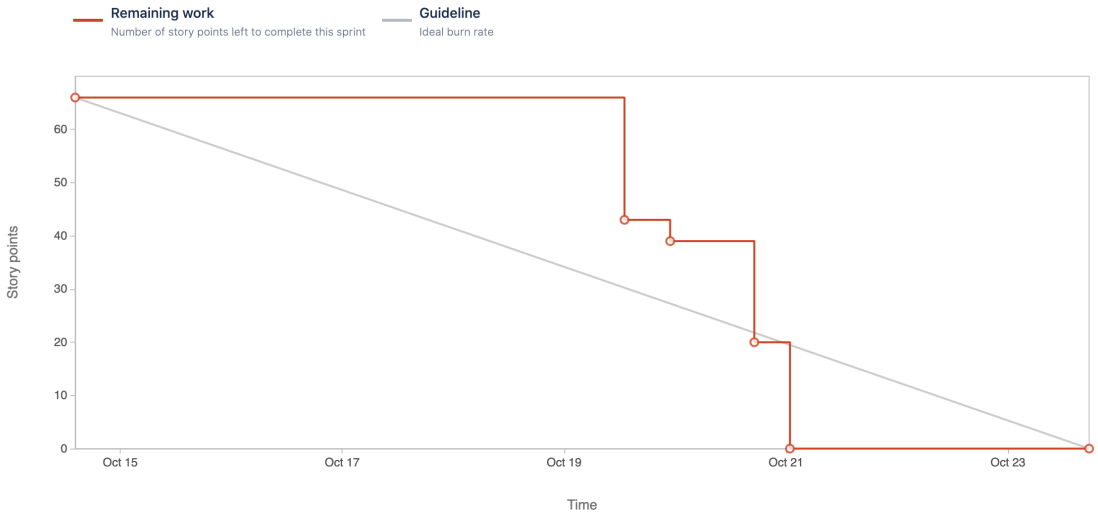
User stories

Názov	**SP**	**Schválená**
Ako produkt owner potrebujem rozoznať jednotlivé druhy olejov pomocou tradičných klasifikátorov	21	Áno
Ako tím, potrebujeme vytvoriť frontend pre prezentáciu tímu	23	Áno
Ako tím potrebujeme metodiky pre jasné definovanie pravidiel v tíme aby sme zefektívnili a zkvalitnili našu prácu	19	Áno
Ako product owner chcem aby sa tím zúčastnil súťaže TP cup pre reprezentáciu výsledkov projektu	4	Áno

Práca na projekte

Meno a Priezvisko	Počet úloh	Percentuálny podiel na šprinte
Simona Klučková	4	15%
Maroš Kollár	5	15%
Jakub Kučečka	1	23%
Zuzana Popovcová	3	17%
Mária Rajníková	3	15%
Alena Valová	4	15%

Burndown chart



Obr. 1: Burndown chart

Export úloh

Dokument s exportom úloh 1. šprintu.

4.2 Šprint č.2

Názov šprintu	Vodopád Skok
Trvanie	23.10.2020 - 6.11.2020

Šprint bol zameraný na vylepšovanie a rozširovanie počtu klasifikátorov, vytváranie obrazových dát a implementovanie administratívneho rozhrania pre webovú stránku. User stories zahrnuté do šprintu boli ohodnotené na 60 story points (ďalej SP).

Počas šprintu boli vytvorené 4 nové modely tradičných klasifikátorov a to Naive bayes, decision tree, KNN a SVM. Vytvoril sa prvý model neurónovej siete a návrh konvolučnej siete. Taktiež sme v šprinte pokračovali v implementácií nových čŕt. V šprinte sa podarilo vytvoriť aj obrazovú reprezentáciu dát. Všetky úlohy súvisiace s klasifikáciou na projekte boli zvládnuté perfektne.

K reprezentačnej stránke tímu bolo úspešne vytvorené a nasadené admin rozhranie, ktoré umožňuje jednoduché pridávanie nových dokumentov na webovú stránku.

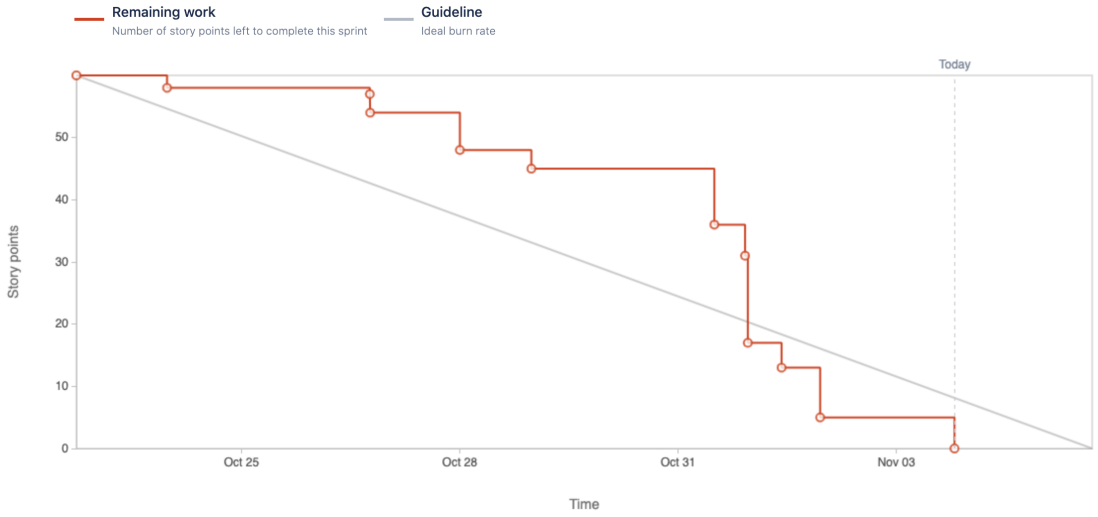
User stories

Názov	**SP**	**Schválená**
Ako tím potrebujeme vytvoriť správy o riešení projektu pre úspešné absolvovanie predmetu	3	Áno
Ako tím potrebujeme zrefaktorovať kód z predošlých šprintov	3	Áno
Ako tím potrebujeme v 2.týždni šprintu zabezpečiť prípravu a dokumentáciu stretnutí	5	Áno
Ako product owner potrebujem implementovať neurónové siete	8	Áno
Ako product owner potrebujem implementovať tradičné klasifikátory	4	Áno
Ako product owner potrebujem dosiahnuť najlepšie výsledky pri klasifikácií, preto potrebujem rozšíriť zoznam čŕt	14	Áno
Ako tím potrebujeme v 1.týždni šprintu zabezpečiť prípravu a dokumentáciu stretnutí	6	Áno
Ako správca webovej domény potrebujem implementovať backend webu	9	Áno

Práca na projekte

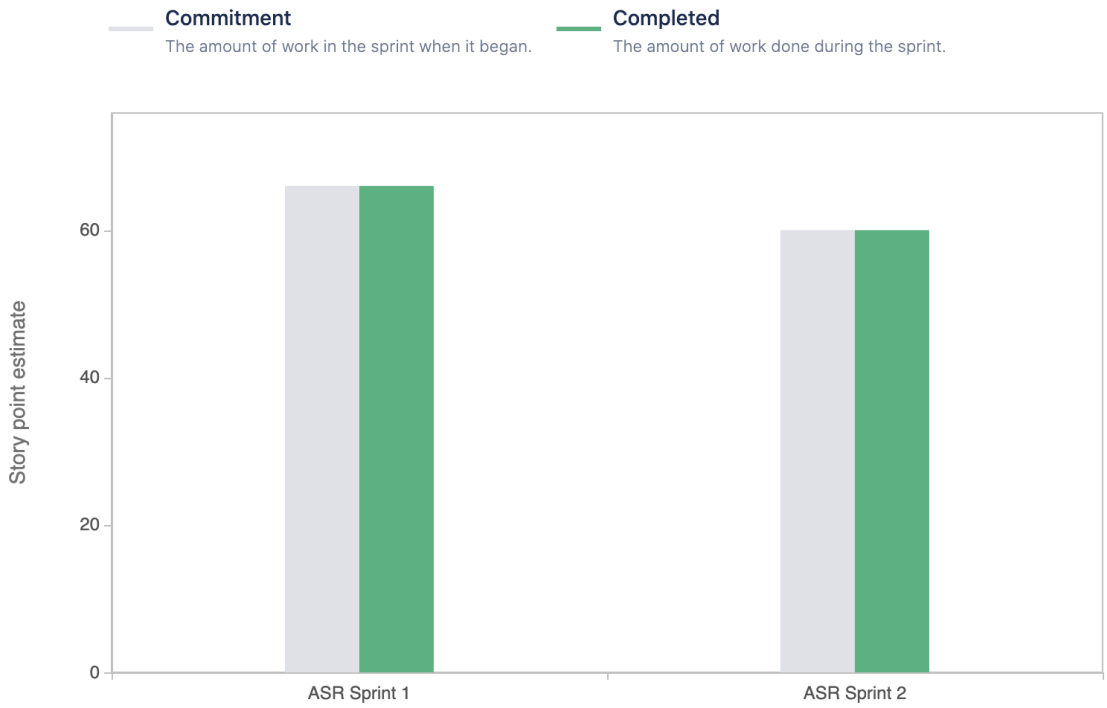
Meno a Priezvisko	Počet úloh	Percentuálny podiel na šprinte
Simona Klučková	6	17%
Maroš Kollár	2	17%
Jakub Kucecka	4	18%
Zuzana Popovcová	3	17%
Mária Rajníková	3	15%
Alena Valová	6	17%

Burndown chart



Obr. 2: Burndown chart

Velocity report



Obr. 3: Velocity report

Export úloh

Dokument s exportom úloh 2. šprintu.

4.3 Šprint č.3

Názov šprintu	Tomašovský vodopád
Trvanie	6.11.2020 - 20.11.2020

Šprint bol zameraný na vytváranie dokumentácie k projektu a spisovanie požiadaviek na aplikáciu. Pokračovali sme vo vytváraní čŕt a ďalších klasifikátorov. User stories zahrnuté do šprintu boli ohodnotené na 64 story points (ďalej SP).

Počas šprintu boli vytvorené 3 dokumenty.

- **Dokumentácia k riadeniu** - ”big picture”riadenia projektu. Obsahom dokumentu je prevažne prehľad rolí členov v tíme a prehľad manažmentov.
- **Dokumentácia modulov systému** - Obsahom dokumentu je popis analýzy, návrhu, implementácie a testovania.
- **Inžinierske dielo** - ”big picture”cieľov semestra. Obsahom dokumentu sú globálne ciele semestra a celkový pohľad na systém.

V šprinte sme začali definovať požiadavky na systém a boli vytvorené a spísané hlavné prípady použitia aplikácie.

Pri klasifikácii sme pokračovali vo vytváraní nových čŕt, ako napríklad diferenčné mapy, median, std a ďalšie. Bola vytvorená a otestovaná prvá konvolučná sieť. Do šprintu sme zahrnuli aj analytické úlohy nad dátami a úlohu o vyhľadávaní existujúcich riešení pre klasifikáciu látok.

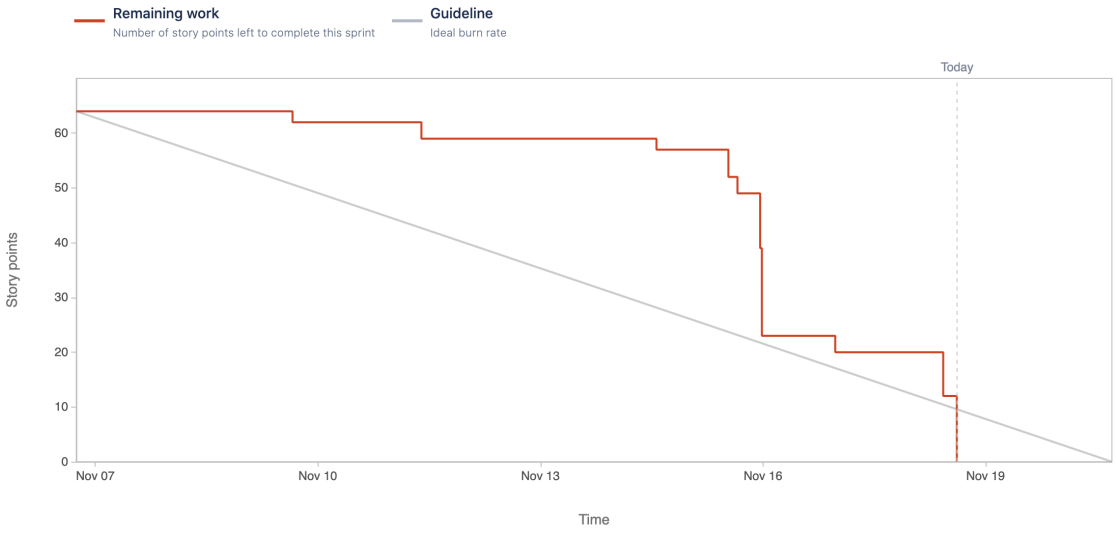
User stories

Názov	**SP**	**Schválená**
Črty	5	Áno
Analýza dát	8	Áno
Konvolučná neurónová sieť	3	Áno
Predspracovanie dát	7	Áno
Diferenčná mapa	10	Áno
Vytvorenie dokumentu Inžinierske dielo	3	Áno
Vytvorenie dokumentu Modulov systému	8	Áno
Vytvorenie dokumentu Riadenie projektu	2	Áno
Zabezpečenie prípravy dokumentácie a stretnutí - 2. týždeň	5	Áno
Zabezpečenie prípravy dokumentácie a stretnutí - 1. týždeň	3	Áno
Špecifikácia aplikácie	8	Áno

Práca na projekte

Meno a Priezvisko	Počet úloh	Percentuálny podiel na šprinte
Simona Klučková	4	19%
Maroš Kollár	4	17%
Jakub Kucecka	4	12%
Zuzana Popovcová	4	17%
Mária Rajníková	4	19%
Alena Valová	7	16%

Burndown chart



Obr. 4: Burndown chart

Velocity report

Obr. 5: Velocity report

Export úloh

Dokument s exportom úloh 3. šprintu.

4.4 Šprint č.4

Názov šprintu	Vodopády Studeného potoka
Trvanie	20.11.2020 - 4.12.2020

Šprint bol zameraný na refaktorizáciu klasifikačných metód, ktoré boli vytvorené v predošlých šprintoch, vylepšovanie modelov a grafický návrh aplikácie. User stories zahrnuté do šprintu boli ohodnotené na 52 story points (ďalej SP).

Počas šprintu sme sa venovali rozdeleniu dát, ktoré sme dostali od product owner-a, na trénovacie a testovacie dáta. Z týchto dát sme odstránili stĺpce, ktoré sme v analýzach v predošlých šprintoch, identifikovali ako nepodstatné pre navrhnuté klasifikátory.

V šprinte sme sa venovali vyhodnocovaniu triedy pomocou viacerých klasifikátorov súčasne. Do tohto šprintu sme zahrnuli aj úlohu na hľadanie čŕt, ktoré sú pre klasifikátory pri rozhodovaní podstatné. Zároveň sme hľadali optimálne hyperparametre pre navrhnuté klasifikátory.

Do šprintu sme zahrnuli aj vytvorenie konfiguračného súboru a verziovanie klasifikátorov. Taktiež sme v rámci refaktorizácie kódu vytvorili funkcie, ktoré spúšťajú trénovanie, predikovanie tried a vyhodnocovanie. Tieto funkcie sme vytvárali tak, aby ich bolo možné využiť v aplikácii. Okrem toho sme po normalizácii aktualizovali masky a šablóny diferenčných máp.

Skompletizovali sme dokumenty, ktoré sa odovzdávali v 10.týždni semestra. V šprinte sme vytvorili grafické návrhy obrazoviek aplikácie a zároveň sme začali tieto obrazovky implementovať.

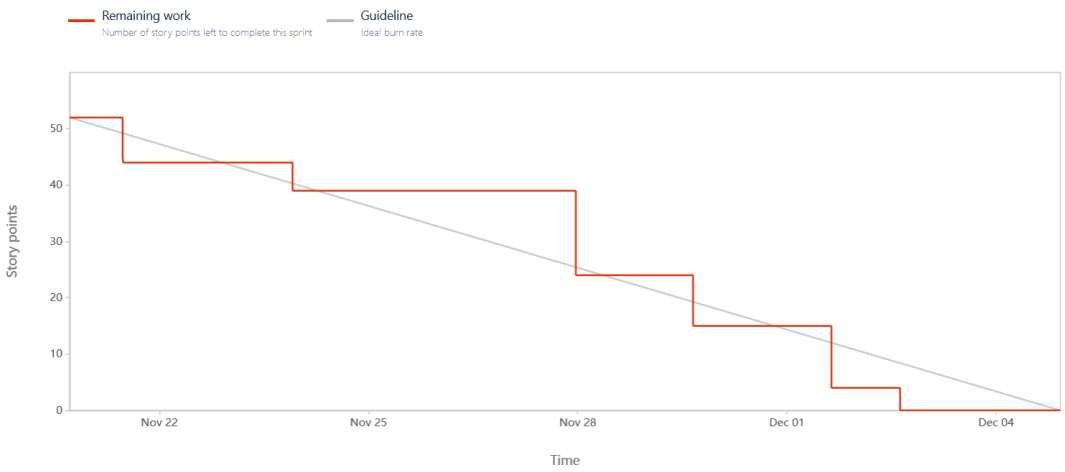
User stories

Názov	**SP**	**Schválená**
Obrazovky	9	Áno
Zabezpečenie prípravy dokumentácie a stretnutí - 2. týždeň	4	Áno
Zabezpečenie prípravy dokumentácie a stretnutí - 1. týždeň	3	Áno
Kompletizovanie odovzdania v 9-tom týždni	2	Áno
Úprava vstupných súborov	7	Áno
Refaktorizácia klasifikácie	15	Áno
Vylepšenie modelov	11	Áno

Práca na projekte

Meno a Priezvisko	**Počet úloh**	**Percentuálny podiel na šprinte**
Simona Klučková	2	15%
Maroš Kollár	2	15%
Jakub Kucecka	4	15%
Zuzana Popovcová	4	17%
Mária Rajníková	4	19%
Alena Valová	4	17%

Burndown chart



Obr. 6: Burndown chart

Velocity report

Obr. 7: Velocity report

Export úloh

Dokument s exportom úloh 4. šprintu.

4.5 Šprint č.5

Názov šprintu	Vodopád Lúčky
Trvanie	4.12.2020 - 13.12.2020

Šprint bol zameraný najmä na dokončovanie dokumentácie k finálnemu odovzdaniu. User stories, ktoré boli zahrnuté do šprintu boli ohodnotené na 18 story points (SP). Taktiež sme mali ešte jeden task zameraný na finalizáciu modelov, konkrétne pridanie najlepších hyperparametrov.

Počas šprintu sme sa venovali viacerým častiam dokumentácie, ktoré bolo treba doplniť alebo upraviť po predchádzajúcom odovzdaní, keďže sme postupne dopĺňali dokumentáciu ku kódu alebo doplňovali metodiky.

Ako počas každého šprintu sme pripravovali prezentácie a vytvárali zápisnice, ktoré boli vždy následne pridané na webovú stránku nášho tímu.

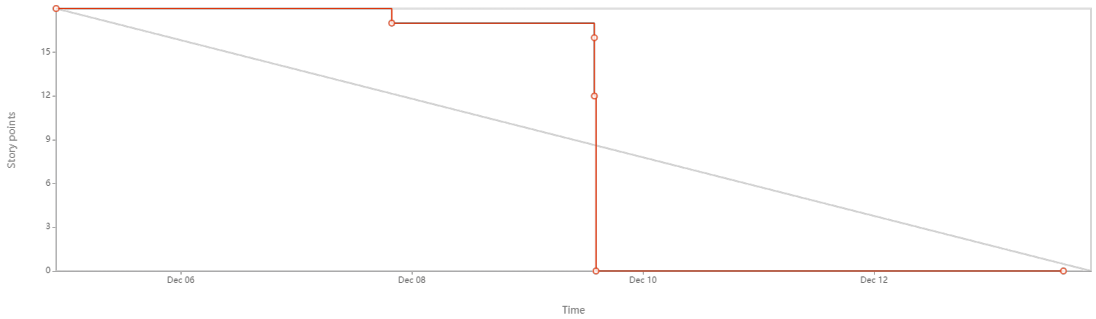
User stories

Názov	**SP**	**Schválená**
Finalizovanie modelov	1	Áno
Zabezpečenie prípravy dokumentácie a stretnutí	4	Áno
Dokumentácia	12	Áno

Práca na projekte

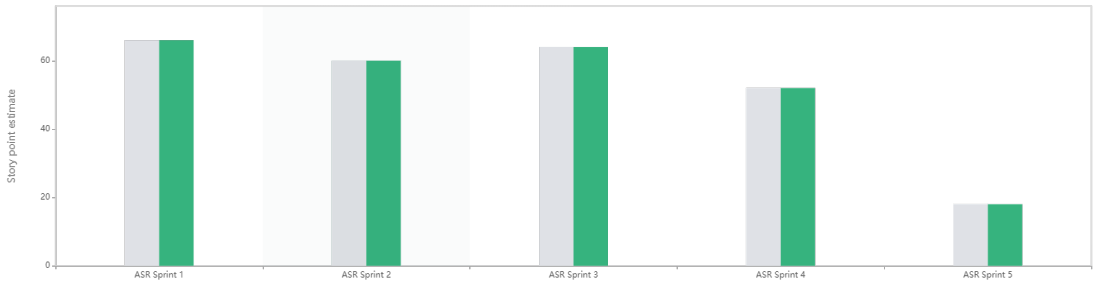
Meno a Priezvisko	**Počet úloh**	**Percentuálny podiel na šprinte**
Simona Klučková	3	16,6%
Maroš Kollár	2	16,6%
Jakub Kucecka	2	16,6%
Zuzana Popovcová	1	16,6%
Mária Rajníková	2	16,6%
Alena Valová	2	16,6%

Burndown chart



Obr. 8: Burndown chart

Velocity report



Obr. 9: Velocity report

Export úloh

Dokument s exportom úloh 5. šprintu.

4.6 Šprint č.6

Názov šprintu	Vodopád z Batmanovej jaskyne
Trvanie	15.2.2021 - 1.3.2021

Šiesty šprint bol zameraný na refaktorizáciu kódu a to najmä častí testov a vyhodnocovania klasifikácie. Ďalej sa pracovalo na optimalizácii úspešnosti klasifikácie, kde sa

nám podarilo vylepšiť úspešnosť zo zimného semestra. V tomto šprinte sa spravila aj veľká časť na aplikácii, kde boli vytvorené obrazovky a následne vznikol UX test k tejto aplikácii.

Keďže sa blížilo odovzdanie prihlášky na IIT.SRC, tak sme zahrnuli do šprintu aj prípravu článku a ten sme konzultovali aj s našimi vedúcimi.

Taktiež ako počas každého šprintu sme pripravovali rôzne prezentácie, ktoré boli prezentované aj našim vedúcim tímového projektu a vytárali sme zápisnice, ktoré boli vždy pridané na webovú stránku nášho tímu.

User stories, ktoré boli zahrnuté do šprintu boli ohodnotené na 48 story points (SP).

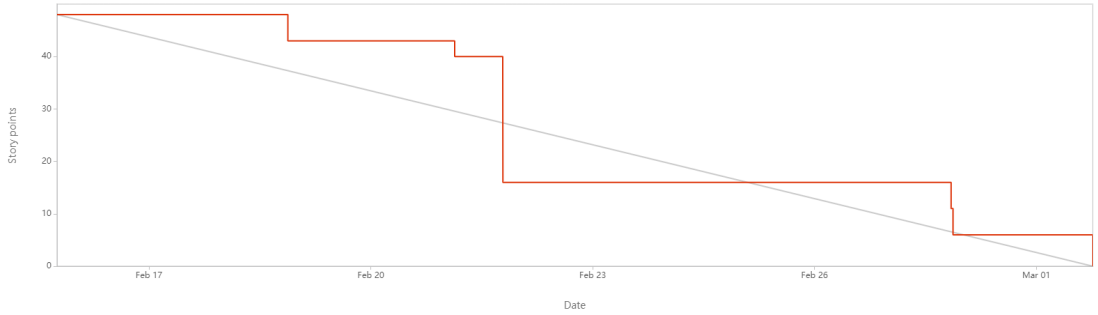
User stories

Názov	**SP**	**Schválená**
Optimalizácia úspešnosti kalsifikácie	3	Áno
Refaktorizácia vyhodnocovania	5	Áno
Refaktorizácia klasifikácie	3	Áno
Obrazovky	21	Áno
UX testovanie	5	Áno
Článok na IIT.SRC	5	Áno
Administrácie šprintu 6	6	Áno

Práca na projekte

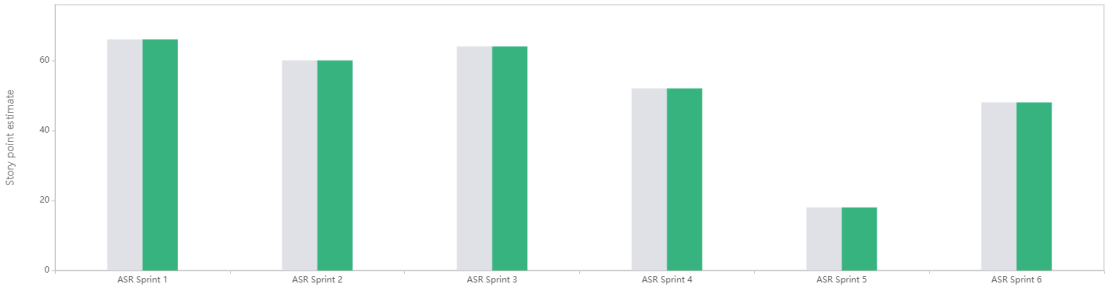
Meno a Priezvisko	**Počet úloh**	**Percentuálny podiel na šprinte**
Simona Klučková	3	16,6%
Maroš Kollár	2	16,6%
Jakub Kučečka	1	16,6%
Zuzana Popovcová	2	16,6%
Mária Rajníková	3	16,6%
Alena Valová	1	16,6%

Burndown chart



Obr. 10: Burndown chart

Velocity report



Obr. 11: Velocity report

Export úloh

Dokument s exportom úloh 6. šprintu.

4.7 Šprint č.7

Názov šprintu	Dúhový vodopád
Trvanie	1.3.2021 - 15.3.2021

Siedmy šprint bol zameraný na dokončenie grafického rozhrania aplikácie. Dokončili sa všetky existujúce obrazovky. Zároveň sa upravili chyby, ktoré sme identifikovali pri

implementácií obrazoviek a pripravovaní UX testov. Následne sme sa v aplikácií zamerali na refaktoring niektorých častí kódu.

V tomto šprinte boli vykonané aj UX testy s piatimi testermi. Medzi týmito testermi sa nachádzali aj vedúci nášho tímového projektu. Po vyhodnotení testov bol spísaný ich priebeh a odhalené nedostatky a pozitíva nami navrhnutej aplikácie.

Do tohto šprintu sme zahrnuli aj analýzu dát, ktoré sme získali v minulom šprinte. Taktiež sme vyhodnotili úspešnosť našich klasifikátorov na nových dátach.

Ďalej sme sa v venovali klasifikácií len extra panenského olivového oleja, teda klasifikovali sme 2 triedy a to EVOO a iné.

User stories, ktoré boli zahrnuté do šprintu boli ohodnotené na 84 story points (SP).

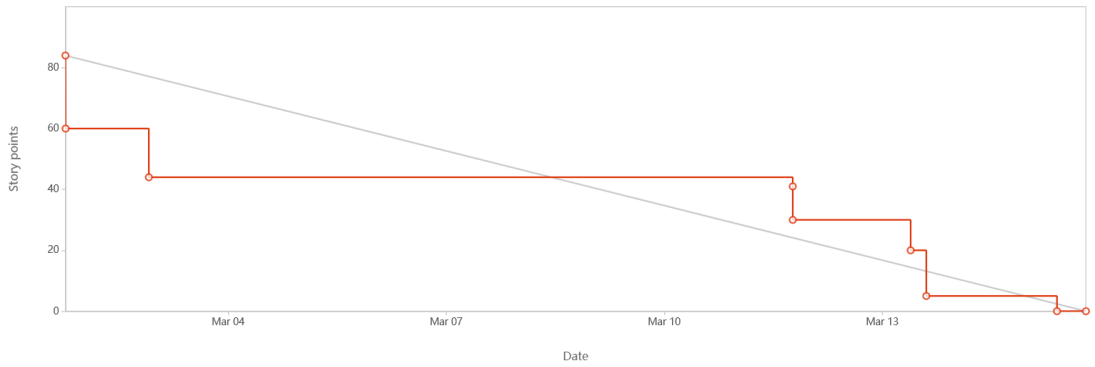
User stories

Názov	**SP**	**Schválená**
API front-end	24	Áno
Backend aplikácie	16	Áno
UX testy	11	Áno
Application	10	Áno
Prispôsobenie na nové dáta	15	Áno
Vyhodnocovanie tried	3	Áno
Administrácie 7. šprintu	5	Áno

Práca na projekte

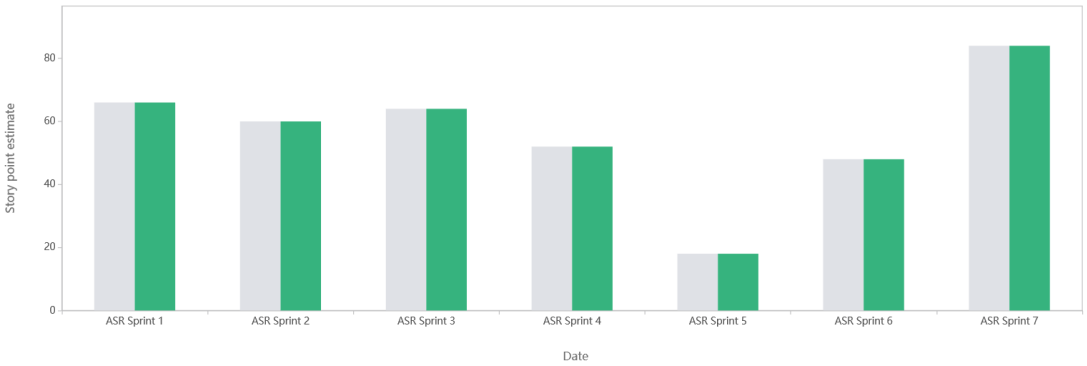
Meno a Priezvisko	**Počet úloh**	**Percentuálny podiel na šprinte**
Simona Klučková	2	13%
Maroš Kollár	2	10%
Jakub Kučečka	6	32%
Zuzana Popovcová	3	12%
Mária Rajníková	3	12%
Alena Valová	6	21%

Burndown chart



Obr. 12: Burndown chart

Velocity report



Obr. 13: Velocity report

Export úloh

Dokument s exportom úloh 7. šprintu.

4.8 Šprint č.8

Názov šprintu	Šútovský vodopád
Trvanie	15.3.2021 - 29.3.2021

V ôsmom šprinte sme sa venovali hlavne napájaniu backend-u aplikácie na zvyšok, teda klasifikáciu, trénovanie modelov a vytváranie obrázkov. Napísali sa teda funkcie, ktoré z backend-u volajú jednotlivé segmenty kódu na funkcionalitu, ktorú potrebujú vykonať. Minimálne sa upravil vzhľad aplikácie a pridalo sa označenie súborov, ktoré používateľ načítava.

Venovali sme sa aj optimalizácii klasifikácie dát a to tým spôsobom, že sa pridalo vyhodnocovanie EVOO alebo iné. Taktiež sme upravili funkciu na spracovávanie chýbajúcich spektier a dané spektrá sme dopočítali.

Do šprintu sme mali ako pravidelne zaradené administratívne tasky.

User stories, ktoré boli zahrnuté do šprintu boli ohodnotené na 39 story points (SP).

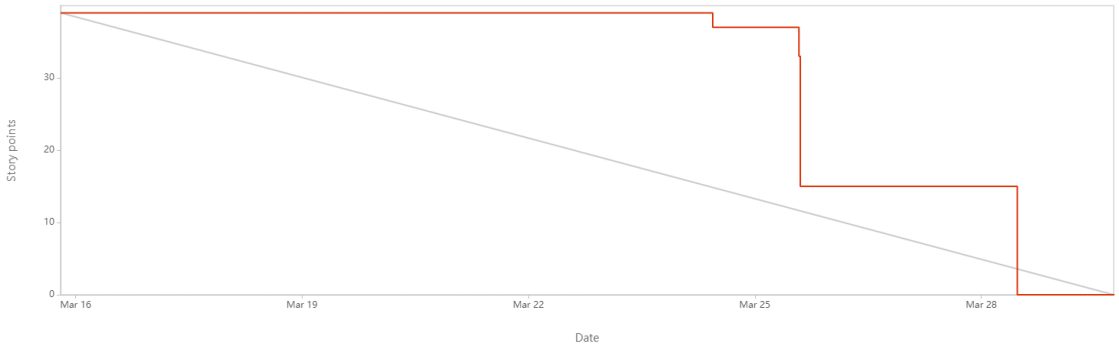
User stories

Názov	**SP**	**Schválená**
Obrazovky	10	Áno
Backend aplikácie	18	Áno
Predspracovanie	4	Áno
Optimalizácia úspešnosti klasifikácie	2	Áno
Administrácie 8. šprintu	5	Áno

Práca na projekte

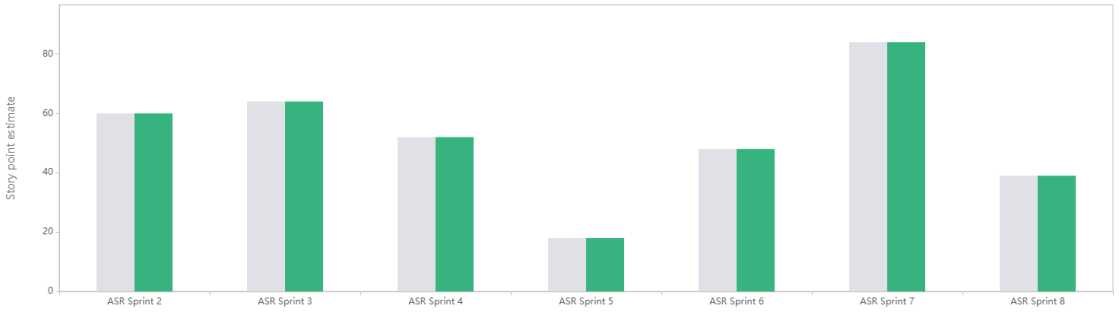
Meno a Priezvisko	**Počet úloh**	**Percentuálny podiel na šprinte**
Simona Klučková	2	15%
Maroš Kollár	2	26%
Jakub Kučečka	2	13%
Zuzana Popovcová	2	18%
Mária Rajníková	2	18%
Alena Valová	1	10%

Burndown chart



Obr. 14: Burndown chart

Velocity report



Obr. 15: Velocity report

Export úloh

Dokument s exportom úloh 8. šprintu.

4.9 Šprint č.9

Názov šprintu	Kľacký vodopád
Trvanie	29.3.2021 - 12.4.2021

V deviatom šprinte sme sa venovali viacerým častiam, backend, obrazovky, klasifikácia, administrácia. Do aplikácie sme pridali vizualizáciu obrázkov pomocou heat mapy,

ukazovanie istoty jednotlivých modelov pre danú vzorku a taktiež sme doplnili loading screen. V aplikácii sme pridali aj možnosť mazania vybraných dát. Na konci šprintu sme testovali celú funkčnosť vytvoreného EXE súboru.

Zlepšili sme časovú optimalizáciu klasifikácie takým spôsobom, že sme pridali multiprocesing do viacerých častí klasifikácie, taktiež sme dopredu špecifikovali typy, ktoré sa vkladajú do dataframe. Pri všeobecnej optimalizcácii sme upravili normalizáciu a zlepšili sme rozhodovanie klasifikátorov na olivové oleje typov VOO a LOO.

Do šprintu sme mali ako pravidelne zaradené administratívne tasky a taktiež sme pracovali na príprave posteru na IIT.SRC. Začali sme pracovať na príprave finálnej dokumentácie, kde sme prekontrolovali či stále sedia požiadavky a testy aplikácie a taktiež aj UC aplikácie.

User stories, ktoré boli zahrnuté do šprintu boli ohodnotené na 44 story points (SP).

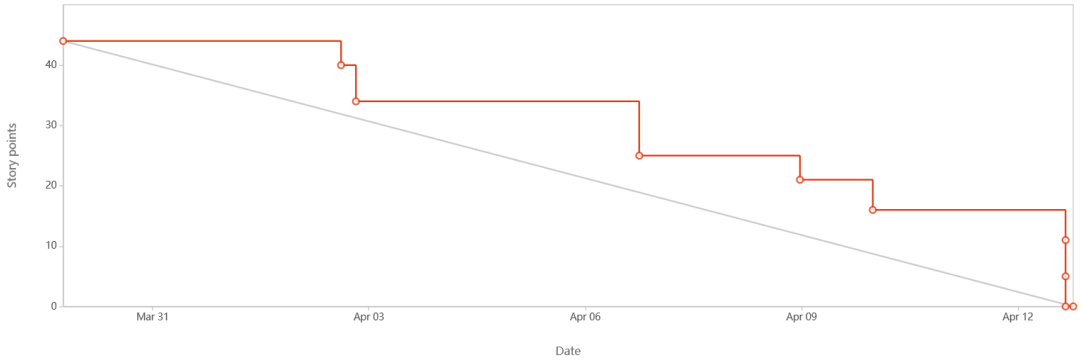
User stories

Názov	**SP**	**Schválená**
Obrazovky	5	Áno
Backend aplikácie	6	Áno
Uprava	5	Áno
Optimalizácia klasifikácie	4	Áno
Časová optimalizácia klasifikácie	9	Áno
UC aplikácia	4	Áno
TP cup	6	Áno
Administratíva 9. šprintu	5	Áno

Práca na projekte

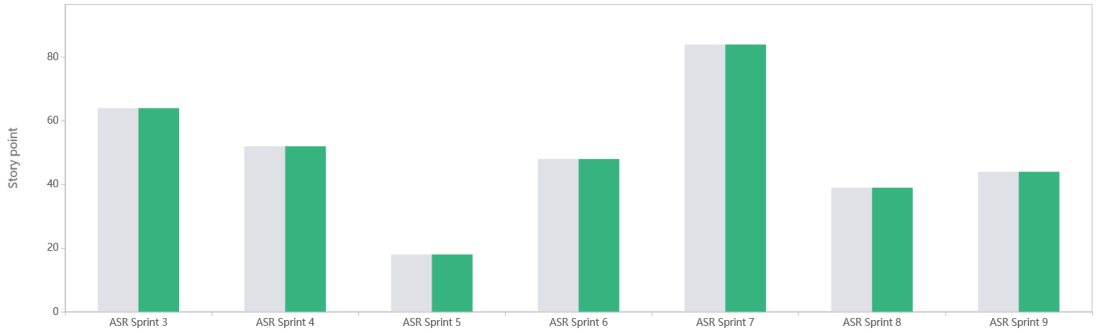
Meno a Priezvisko	**Počet úloh**	**Percentuálny podiel na šprinte**
Simona Klučková	4	20%
Maroš Kollár	4	20%
Jakub Kučečka	3	11%
Zuzana Popovcová	3	15%
Mária Rajníková	1	18%
Alena Valová	3	16%

Burndown chart



Obr. 16: Burndown chart

Velocity report



Obr. 17: Velocity report

Export úloh

Dokument s exportom úloh 9. šprintu.

4.10 Šprint č.10

Názov šprintu	Vodopád Bystrého potoka
Trvanie	12.4.2021 - 26.4.2021

V desiatom šprinte sme sa venovali záverečným úpravám aplikácie. Optimalizovali sme zapínanie aplikácie a venovali sa refaktoringu programu. Pridali do aplikácie upozornenia a vylepšili zobrazovanie tabuliek.

Počas šprintu sme získali požiadavky na úpravu článku, ktoré Zuzka Popovcová zapracovala. Taktiež se sa zúčastnili súťaže TP cup, na ktorej Maroš Kollár odprezentoval poster s pripraveným videom funkcionality aplikácie.

Ďalej sme sa v šprinte venovali aj prípravám na nadchádzajúce odovzdanie predmetu.

User stories, ktoré boli zahrnuté do šprintu boli ohodnotené na 53 story points (SP).

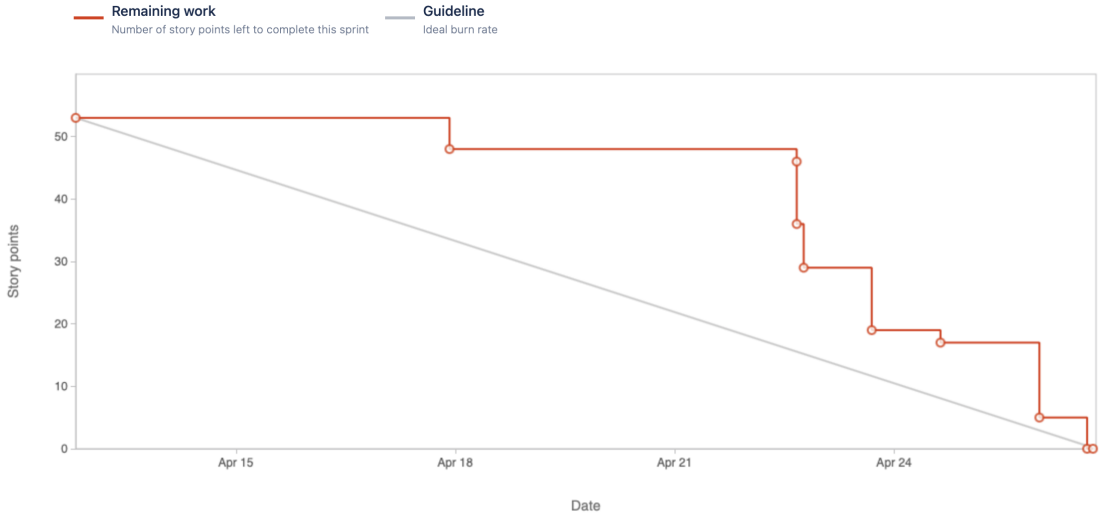
User stories

Názov	**SP**	**Schválená**
Refaktor backend app	12.0	Áno
Administratíva 10.šprintu	5.0	Áno
Príručky	2.0	Áno
Moduly systému	2.0	Áno
Optimalizácia vyhodnotenia	5.0	Áno
TP cup	10.0	Áno
Príprava konečnej dokumentácie	10.0	Áno
Obrazovky	7.0	Áno

Práca na projekte

Meno a Priezvisko	**Počet úloh**	**Percentuálny podiel na šprinte**
Simona Klučková	4	17%
Maroš Kollár	4	19%
Jakub Kucecka	3	13%
Zuzana Popovcová	2	19%
Mária Rajníková	3	17%
Alena Valová	3	15%

Burndown chart



Obr. 18: Burndown chart

Velocity report

Obr. 19: Velocity report

Export úloh

Dokument s exportom úloh 10. šprintu.

4.11 Šprint č.11

Názov šprintu	Kmeťov vodopád
Trvanie	26.4.2021 - 10.5.2021

V jedenástom šprinte sme sa venovali finalizácií odovzdania projektu. Na začiatku šprintu sme upravili posledné chybičky krásy v aplikácií. Vykonali sme akceptačné testy s jendým z product owner-ov, s Ing. Martou Šoltésovou Prnovou, PhD. Spísali sme spätnú väzbu z nasadenia aplikácie vo firme MaSa Tech s.r.o. Doplnili a upravili sme dokumenty, ktoré sa budú odovzdávať. Vytvorili sme záverečné prezentácie.

Počas tohto špintu sme vypracovali dokumenty, ktoré boli potrebné pre ďalšie pokračovanie v súťaži TP cup.

User stories, ktoré boli zahrnuté do šprintu boli ohodnotené na 35 story points (SP).

V šprinte sme nespravili statickú webovú stránku, túto úlohu sme dokončili po šprinte, pretože sme vkladali na web všetky potrebné dokumenty.

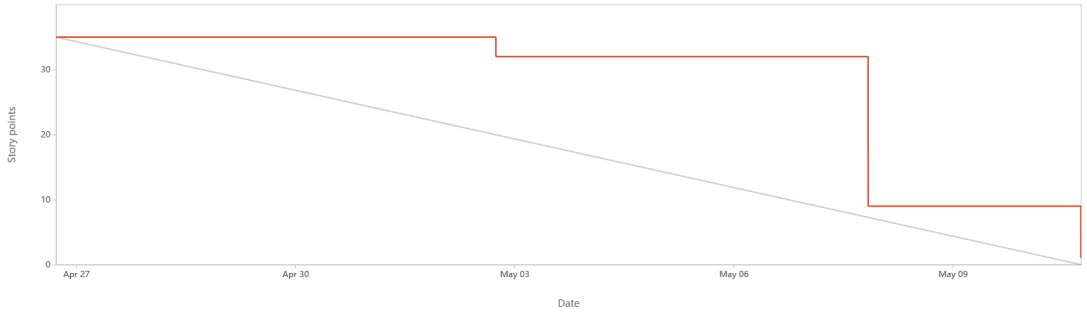
User stories

Názov	**SP**	**Schválená**
Inžinierske dielo	3	Áno
Vytvorenie statickej webovej stránky	1	Áno
Riadenie projektu	4	Áno
Moduly systému	13	Áno
Kompletizácia	10	Áno
Administratíva 11.šprintu	4	Áno

Práca na projekte

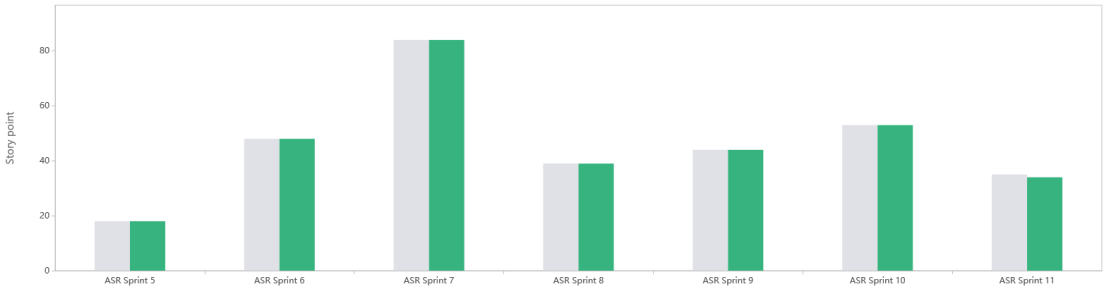
Meno a Priezvisko	**Počet úloh**	**Percentuálny podiel na šprinte**
Simona Klučková	3	20%
Maroš Kollár	2	11.43%
Jakub Kucecka	2	5.71%
Zuzana Popovcová	5	28.57%
Mária Rajníková	2	22.86%
Alena Valová	2	11.43%

Burndown chart



Obr. 20: Burndown chart

Velocity report



Obr. 21: Velocity report

Export úloh

Dokument s exportom úloh 11. šprintu.

5 Globálna retrospektíva

5.1 Retrospektíva šprintu č.1

Retrospektívy sa zúčastnili všetci členovia tímu.

Meno a Priezvisko	Zúčastnil/a sa na stretnutí
Simona Klučková	Áno
Maroš Kollár	Áno
Jakub Kučečka	Áno
Zuzana Popovcová	Áno
Mária Rajníková	Áno
Alena Valová	Áno

Čo sme spravili dobre?

- Začali sme s klasifikáciou pomerne skoro.
- Dokončili sme písanie metodík.
- Dali sme do poriadku git.
- Dobre sme zvládli komunikáciu v tíme.
- Super komunikácia s PO.
- Hotový web po prvom šprinte.
- Odovzdaná prihláška na TP cup.
- Podarilo sa nakonfigurovať server.

Čo sme mohli urobiť lepšie?

- Písanie úloh. Trvalo nám to moc dlho.
- Vkladanie odkazov na dokončený výstup práce, keď ju posúvame na review.
- Zaviesť stand-up, aby sme vedeli na čom, kedy a kto robí.
- Viac plánovať implementáciu a menej riešiť úlohy.
- Častejšie riešiť na stretnutiach implementačné detaily.
- Častejšie zapájať vedúcich do vývoja.

Čo ideme zaviesť?

- Zmeniť pomenovanie úloh zadaných používateľom. Aktuálny nadpis pôjde do opisu a nadpis bude podobný ako pri ostatných úlohách.
- Lepšia príprava na stretnutia, na ktorých sa plánuje šprint. Vopred si pripraviť úlohy, ktoré úlohy chcem do šprintu zaradiť.
- Vkladanie odkazov do úloh, ktoré sme dokončili.
- Zaviesť stand-up prostredníctvom komunikačného kanála vytvoreného na Microsoft teams.
- Každú 2. stredu, aspoň jeden krát za šprint sa budú na stretnutí riešiť spôsoby implementácie.
- Dohodnúť si vedúcimi stretnutie a spísať s nimi plán na 2 semestre.
- Zahrnúť do úloh prípravu prezentácii.

5.2 Retrospektíva šprintu č.2

Retrospektívy sa zúčastnili všetci členovia tímu.

Meno a Priezvisko	Zúčastnil/a sa na stretnutí
Simona Klučková	Áno
Maroš Kollár	Áno
Jakub Kučečka	Áno
Zuzana Popovcová	Áno
Mária Rajníková	Áno
Alena Valová	Áno

Čo sme spravili dobre?

- Zaviedli sme Standup a RRF.
- Zavedenie retrospektívy pomocou Trella.
- Plánovanie šprintov.
- Splnili sme všetky úlohy v dohodnutom čase.
- Zapracovali sme všetky pripomienky z minulej retrospektívy.
- Zapojenie vedúcich do tvorby príbehov.

- Konzultovanie obsahu dokumentácie.
- Zaviedli sme implementačné stretnutia.

Čo sme mohli urobiť lepšie?

- Viac SP pre tasky s testovaním.
- Viac farebných grafov.
- Nezabúdať na písanie standup.
- Prestať robiť okamžité implementácie. Viac analýzy.
- Menej dlhých a nezáživných stretnutí.
- Menej dokumentácie.

Čo ideme zaviesť?

- Začať označovať ľudí v skupine.
- Veci týkajúce malej skupiny ľudí riešiť v osobnom chate.
- Začať viac analyzovať.
- Zohľadňovať pri planning pokri úlohy, v ktorých sa programuje.
- Pomenovávanie šprintov podľa vodopádov. A pridanie obrázkov na web.
- Pridávanie nálepiek k nápadom, ktoré sa nám v trelle páčia a nepáčia.
- Do názvu kartičky v trelle napísať na začiatok svoje iniciály.

5.3 Retrospektíva šprintu č.3

Retrospektívy sa zúčastnili všetci členovia tímu.

Meno a Priezvisko	Zúčastnil/a sa na stretnutí
Simona Klučková	Áno
Maroš Kollár	Áno
Jakub Kučečka	Áno
Zuzana Popovcová	Áno
Mária Rajníková	Áno
Alena Valová	Áno

Čo sme spravili dobre?

- Máme progress na klasifikácii.
- Prezentácia výsledkov na sprint review.
- Práca pod stresom.
- Samostatné riešenie problémov na úlohách.
- Výborná komunikácia pri riešení problémov.
- Zvládnutie formátovania UC do latex.

Čo sme mohli urobiť lepšie?

- Potrebujeme viac ľudí na review.
- Tvorba backlogu.
- Lepšie plánovanie backlogu.
- Lepšie písanie opisu.
- Pre-planning a planning rozdeliť do dvoch stretnutí.
- Pridanie viac odkazov k úlohe (tuto to nájdeš...), s lepším vysvetlením.
- Lepší systém pre hodnotenie úloh.
- Nepísať správy v noci.
- Pri udeľovaní úloh rátať s rizikom.

Čo ideme zaviesť?

- Prideľovať k úlohám jednotlivo Akceptora.
- Pridať stĺpec Ready to Approve.
- Určiť ľudí zodpovedných za backlog epic.
  - Klasifikácia - Majka
  - Aplikácia - Jakub
  - Administrácia - Ala
- Nočný režim 23:00 - 10:00

5.4 Retrospektíva šprintu č.4

Retrospektívy sa zúčastnili všetci členovia tímu.

Meno a Priezvisko	Zúčastnil/a sa na stretnutí
Simona Klučková	Áno
Maroš Kollár	Áno
Jakub Kučečka	Áno
Zuzana Popovcová	Áno
Mária Rajníková	Áno
Alena Valová	Áno

Čo sme spravili dobre?

- Zavedenie Approver-a k úlohám.
- Venovanie sa implementačným problémom na stretkách.
- Skorá komunikácia pri vzniknutých problémoch.
- Rýchle review.
- Zvládanie blokujúcich taskov.
- Efektívne stretnutia. Naučili sme sa riešiť podstatné veci na stretnutiach a nevenovať sa témam, ktoré sa netýkajú tímaku.
- Dobrý burndown chart - dopracovali sme sa k tomu, ako by to malo vyzerať.

Čo sme mohli urobiť lepšie?

- Identifikácia taskov, na ktoré treba pohľad viacerých ľudí.
- Skôr identifikovať technológie, ktoré budeme využívať pri implementácii aplikácie.

- Celkovo začať skôr riešiť aplikáciu.
- Viac myslieť na to, že treba písať dokumentáciu k vytvoreným funkciám.
- Menej CVS, viac CSV.

Čo ideme zaviesť?

- Zaviesť team buildingy, na ktorých si sadneme a zabavíme sa pri jedle a pití.
- Dbať na psychické zdravie členov tímu.

5.5 Retrospektíva šprintu č.5

Retrospektívy sa zúčastnili všetci členovia tímu.

Meno a Priezvisko	Zúčastnil/a sa na stretnutí
Simona Klučková	Áno
Maroš Kollár	Áno
Jakub Kučečka	Áno
Zuzana Popovcová	Áno
Mária Rajníková	Áno
Alena Valová	Áno

Čo sme spravili dobre?

- Dať vedieť, že mám problém a riešiť ich spolu.
- Rýchle odpovede počas nočného režimu.

Čo sme mohli urobiť lepšie?

- Všetci sme sa mohli pozrieť na konfigurák.
- Definovať formát dokumentácie kódu predtým ako sme ju vytvorili.
- Zamyslieť sa pred kódením nad tým: prečo to robím a ako sa to má používať.
- Pýtať sa: ”Treba to?”
- Jasne povedať, kto má čo spraviť v tasku.

Čo ideme zaviesť?

- Spájať prezentácie a zápisnicu pre retrospektívu do jedného tasku.
- Taktiež spájať ostatné zápisnice a prezentácie.
- Dodržiavať formát dokumentácie kódu tak ako je teraz naformátovaná dokumentácia, zodpovednosť.

5.6 Retrospektíva šprintu č.6

Retrospektívy sa zúčastnili všetci členovia tímu.

Meno a Priezvisko	Zúčastnil/a sa na stretnutí
Simona Klučková	Áno
Maroš Kollár	Áno
Jakub Kučečka	Áno
Zuzana Popovcová	Áno
Mária Rajníková	Áno
Alena Valová	Áno

Čo sme spravili dobre?

- Zvládli sme napísať aplikáciu.
- Rýchly sprint planning.
- Vytvorili sme exe súbor aplikácie a vyriešili vzniknuté problémy.
- Vylepšili sme úspešnosť klasifikácie oproti zimnému semestru.
- Získali sme nové dáta.
- Urobili sme refaktorizáciu klasifikácie.
- Vyrobili sme svg ikonky do aplikácie.
- Odovzdali sme článok IIT.SRC.
- Pripravili sme UX testy k aplikácii.
- Máme pekné alertify.

Čo sme mohli urobiť lepšie?

- Napísať najskôr metodiku k aplikácii.
- Nepísať random kód, riadiť sa metodikou (ešte neexistovala).
- Písať dokumentáciu k aplikácii.
- Upratanie kódu aplikácie.
- Pripraviť backlog pre aplikáciu pred planningom.
- Nekomplikovať si veci a zamerať sa na to, čo treba spraviť prioritne.

- Do gitu pridávať zmeny do vlastnej vetvy.
- Zaškolenie viacerých ľudí do aplikácie, aby na nej mohlo začať pracovať viac ľudí.

Čo ideme zaviesť?

- Neimplementovať, pokým nemáme v Jire spísané tasky.
- Nerobiť to čo v tasku nie je napísané.
- Nerobiť na taskoch, ktoré nie sú v šprinte.

5.7 Retrospektíva šprintu č.7

Retrospektívy sa zúčastnili všetci členovia tímu.

Meno a Priezvisko	Zúčastnil/a sa na stretnutí
Simona Klučková	Áno
Maroš Kollár	Áno
Jakub Kučečka	Áno
Zuzana Popovcová	Áno
Mária Rajníková	Áno
Alena Valová	Áno

Čo sme spravili dobre?

- Sima, skvelá práca s UX testami.
- Veľká pochvala za obrázky, vyzerajú super.
- Zuzka super rýchle review a spätná väzba.
- Použili stack-overflow.
- Skvelá podpora pri plnení úloh.
- Maroš super zredukoval CSS.
- Jakub, super navrhnutá aplikácia.

Čo sme mohli urobiť lepšie?

- Prinútiť vedúcich viac komunikovať.
- Lepšie informovať o tom, keď príde nový mail.
- Menej sa drieť.
- Širší vedomostný záber k fungovaniu aplikácie.

Čo ideme zaviesť?

- Keď príde nový mail napísať do teamsov správu (kanál General).

5.8 Retrospektíva šprintu č.8

Retrospektívy sa zúčastnili všetci členovia tímu.

Meno a Priezvisko	Zúčastnil/a sa na stretnutí
Simona Klučková	Áno
Maroš Kollár	Áno
Jakub Kučečka	Áno
Zuzana Popovcová	Áno
Mária Rajníková	Áno
Alena Valová	Áno

Čo sme spravili dobre?

- Rýchle schvaľovanie.
- Maroš rýchle splnenie tasku.
- Zuzka a Jakub sú skvelí bug fixers.
- Skvelé vyhýbanie sa bugom.
- Poctivé dodržiavanie nočného režimu.

Čo sme mohli urobiť lepšie?

- Mali sme spraviť lepšiu správu tabuliek.

Čo ideme zaviesť?

- Zvážiť tagovanie všetkých.

5.9 Retrospektíva šprintu č.9

Retrospektívy sa zúčastnili všetci členovia tímu.

Meno a Priezvisko	Zúčastnil/a sa na stretnutí
Simona Klučková	Áno
Maroš Kollár	Áno
Jakub Kučečka	Áno
Zuzana Popovcová	Áno
Mária Rajníková	Áno
Alena Valová	Áno

Čo sme spravili dobre?

- Písanie pekných popiskov do Jiry.
- Zachovanie chladnej hlavy.
- Používanie oneline riešení.
- Zuzka, Jakub dobrá práca pri supporte a review.
- Pekný loading bar.
- Spojazdnenie SVM.
- Dobrá práca pri implementácii multiprocessing.
- Zredukovanie textu na posteri.

Čo sme mohli urobiť lepšie?

- Intenzívnejšie komunikovať s vedúcimi pri tvorbe postera.
- Ala nauč sa opraviť si GIT.
- Rozvážnejšie zvažovať kľúčové rozhodnutia.
- Nezabudnúť na zmenu času stretnutia.
- Treba písať dokumentáciu.

Čo ideme zaviesť?

- Oznámime si, že stretnutie je skôr.
- EXE do produkcie :)

5.10 Retrospektíva šprintu č.10

Retrospektívy sa zúčastnili všetci členovia tímu.

Meno a Priezvisko	Zúčastnil/a sa na stretnutí
Simona Klučková	Áno
Maroš Kollár	Áno
Jakub Kučečka	Áno
Zuzana Popovcová	Áno
Mária Rajníková	Áno
Alena Valová	Áno

Čo sme spravili dobre?

- Zuzka super práca na článku.
- Maroš super práca pri prezentovaní a tvorbe videa. Pekné animácie.
- Jakub skvelá práca pri vytvorení postera.
- Turbo zrýchlenie aplikácie.
- Koordinácia medzi tímom.

Čo sme mohli urobiť lepšie?

- Pozornejšie prideľovať a hodnotiť úlohy.
- Venovať zvýšenú pozornosť dokumentom.

Čo ideme zaviesť?

- Pozornejšie prideľovať a hodnotiť úlohy.
- Venovať zvýšenú pozornosť dokumentom.

5.11 Retrospektíva šprintu č.11

Retrospektívy sa zúčastnili všetci členovia tímu.

Meno a Priezvisko	Zúčastnil/a sa na stretnutí
Simona Klučková	Áno
Maroš Kollár	Áno
Jakub Kučečka	Áno
Zuzana Popovcová	Áno
Mária Rajníková	Áno
Alena Valová	Áno

Čo sme spravili dobre?

- Majka, skvelá práca pri ohandlovaní procesov, aby appka nezamrzla.
- Ala, super diagramy pri opise frontendu.
- Jakub vytvoril krásny dizajn prezentácie.
- Sima, super práca pri testovaní appky s Martou.
- Maroš, výborná práca pri tvorbe videa.
- Zuzka, skvelá práca vo finalizovaní dokumentácie.

Čo sme mohli urobiť lepšie?

- Nemazať potrebné warningy pri úprave kódu.

6 Webové sídlo

V rámci tímového projektu sme vytvorili stránku tímu, ktorá sa nachádza na URL adrese: https://team05-20.studenti.fiit.stuba.sk.

A Motivačný list

A.1 Predstavenie tímu

Sme mladý a flexibilný tím pripravený prekonať každú výzvu. Voláme sa Mikasa po postave z anime a po značke výrobcu volejbalových lôpt.

Každý člen tímu sa užšie špecializuje na niečo iné, ale naše predstavy o tom, čomu sa plánujeme venovať sú približne rovnaké. Zaujímajú nás neurónové siete, dátová analýza a vývoj aplikácií a softvérov. Zoznam technológií, ktoré ovládame.

Spoločným cieľom je vytvárať produkty s vysokou kvalitou. Základom vysokej kvality je:

- analýza - každý člen tímu vypracoval už minimálne 3 analytické dokumenty.
- návrh - počas štúdia sme vytvorili rôzne návrhy softvérov v rôznych notáciach.
- program - záleží nám na kvalite výsledného produktu, a fungujúci program je jeho základom. Každý jeden z nás už napísal tisíce riadkov kódu v minimálne 4 rôznych jazykoch.
- praktickosť uplatniteľniteľnosť vo svete - všetky programy sú tvorené pre ľudí, a preto nám záleží na prínose do ich životov.

Vyššie uvedené body kvality vieme zabezpečiť aj vďaka:

- Sime, ktorá sa zaujíma o dátovú analýzu, strojové učenie a virtuálnu realitu. V rámci svojej bakalárskej práce získala cenu dekana za ohodnocovanie biometrických charakteristík na základe gest rukami, kde vytvorila vlastnú hru, pomocou ktorej získala dáta od používateľov a tie následne spracovala. Použila machine learning na zisťovanie možnosti identifikácie používateľa na základe gest rukami a faktu, či je možné tieto behaviorálne črty napodobniť a "ukradnúť".
- Marošovi, ktorý sa zaujíma o neurónové siete a na väčšine školských projektov a aj v rámci brigády programoval backend. Vo svojej bakalárskej práci na základe vizuálnych charakteristík určoval komplementárne páry produktov pomocou siamskej neurónovej siete pre ich možné využitie v rámci personalizovaného odporúčania.
- Jakubovi, ktorý sa zaujíma o tvorbu webových stránok a aplikácií. Taktiež je flexibilný a vie sa chytiť modelovania. Vo svojej bakalárskej práci Rekurentné

vzťahy v grafických objektoch, vysvetľoval fungovanie rekurentných vzťahov pre postupnosti triviálnym spôsobom na rekurentných krivkách. Tiež má skúsenosti z práce so správou core-backendu a scriptovacími jazykmi.

- Zuzke, ktorú najviac zo všetkého baví modelovanie softvéru. Je flexibilná a v tíme pomôže, kde sa dá. Má skúsenosti s tvorením webov a s neurónovými sieťami. V rámci bakalárskej práce sa Zuzka venovala spracovaním medicínskych dát pomocou konvolučných neurónových sietí.
- Majke, ktorá sa zaujíma o dátovú analýzu a strojové učenie, taktiež bola šikovnou výskumníčkou. V rámci svojej bakalárskej práce získala cenu dekana za vytvorenie modelu, ktorý dokáže rozoznať ukradnutie mobilného zariadenia od bežného používania. Využila pritom dáta zo senzorov zaznamenané vlastnou mobilnou aplikáciou.
- Ale, ktorá sa zaujíma tvorbu webových stránok, modelovanie softvéru a má skúsenosti aj s tvorbou hier. V jej bakalárskej práci Meranie a navrhovanie algoritmických zručností, pomocou jednoduchej hry zisťovala, ktoré metriky vplývajú na výpočtové myslenie. Z hry získavala dáta a následne ich spracovala.

A.2 Motivácia: Automatické rozpoznávanie spektier (08)

Pri skúmaní súčasného informatického sveta a spôsobu, akým sa v posledných rokoch uberajú spoločnosti si v rámci nášho tímu myslíme, že spracovanie dát a strojové učenie sú oblasti informatiky, ktoré budú postupne ešte viac žiadané ako je tomu v súčasnosti.

Mnohí členovia nášho tímu by tak radi využili nadobudnuté vedomosti z predmetu Inteligentná Analýza Údajov alebo bakalárskych prác. Všetci by sme však pomocou práce na tomto projekte svoje vedomosti radi prehĺbili, prípadne sa naučili niečo úplne nové.

Téma Automatického rozpoznávania spektier nás zaujala najmä vďaka tomu, že sa primárne jedná o výskumne orientovanú tému. Súčasne však téma vytvára priestor spojiť výskum s praxou, na čom by sme sa radi podieľali.

Veríme, že aplikáciu umožňujúcu automatické rozpoznávanie spektier by prax uvítala, či už v rámci výrobných procesov alebo v rámci dodatočných špecializovaných rozborov.

Tému tímového projektu tak berieme ako možnosť získania nových vedomostí z oblasti dátovej vedy či skúseností z práce v tíme venujúcemu sa dátam a strojovému učeniu. Možnosť podieľať sa na tvorbe systému, ktorý by odborníkom ušetril množstvo času by však taktiež uspokojila našu chuť zjednodušovať prácu ľuďom z iných oblastí a vytvorilo skúsenosti pre nasledujúci profesný život.

Dúfame, že naše doterajšie skúsenosti so spracovaním dát a strojovým učením v kombinácií s našim zanietením pre túto oblasť informatiky z nás robia vhodných kandidátov pre prácu na tímovom projekte Automatického rozpoznávania spektier.

A.3 Motivácia: Podporný informačný systém pre študijné oddelenie (19)

Téma Podporný informačný systém pre študijné oddelenie náš tím zaujala najmä z dôvodu, že sa jedná o komplexnú webovú aplikáciu. Ako sme už spomínali, náš tím má bohaté skúsenosti s vývojom webových aplikácií ako aj návrhom softvéru. Pri práci na tejto aplikácií by sme radi využili naše vedomosti z predmetu Webové technológie, ktorý absolvovali 3 členovia nášho tímu a tiež z predmetu Webové publikovanie. Taktiež sa môžeme spoľahnúť aj na Alu a Jakuba, ktorí majú v letnom semestri zapísaný predmet Vývoj webových aplikácií v prostredí cloudu. Zároveň by sme si veľmi radi vyskúšali nasadenie funkčnej aplikácie do používania.

Ďalším dôležitým faktorom pri výbere tejto témy je aj možnosť pomôcť nášmu študijnému oddeleniu. Študijné referentky sú veľmi podstatnou súčasťou bezproblémového fungovania fakulty, a preto majú často práce vyše hlavy. Študenti sa na nich obracajú s rôznymi otázkami a oni im na každú veľmi ochotne odpovedajú. Preto by sme im chceli uľahčiť ich prácu systémom, ktorý ponúkne odpovede na opakujúce sa otázky študentov a vybavenie rôznych požiadaviek, či už zo strany študentov ale aj iných zamestnancov fakulty.

Myslíme si, že práve náš tím má potenciál na to, aby zvládol prekonať všetky problémy, ktoré so sebou táto téma prinesie a vytvoriť tak funkčnú a používateľsky príjemnú aplikáciu.

A.4 Motivácia: Korekcia dynamických vlastností virtuálnych modelov komponentov vozidiel (13)

Autonómne vozidlá sú jedným z najzaujímavejších smerov, do ktorého sa informatici môžu ponoriť. Čiastočná účasť na tomto projekte by bola neobyčajnou príležitosťou pre náš tím.

Celý tím sa primárne zaujíma o metódy machine learningu a ich prepojenie s praxou. Veľkú výhodu vidíme aj v spolupráci so Strojníckou fakultou s firmou Slovakia Ring a so zahraničnou firmou Siemens Belgium. Pre Slovensko je v budúcnosti kľúčová spolupráca na medzinárodných projektoch, a preto je dôležité vytvorenie "dobrej" vizitky počas práce na týchto projektoch, ktorú náš tím vie zabezpečiť.

Na tejto téme nás zaujali nasledujúce 2 smery:

- AI - Testovanie matematických modelov pomocou AI by nemuselo byť v budúcnosti využívané iba pri autonómnych vozidlách. Náš tím môže poskytnúť kvalitu odvedenej roboty, skúsenosti a 100% nasadenie počas celej doby projektu. Tím vo svojich doterajších prácach opakovane používa umelú inteligenciu, ktorej sa plánujeme venovať aj v profesijnom živote. Pracovanie na tomto projekte by bolo jedinečnou príležitosťou, získavať nových znalostí a rozšírenia rozhľadov pri práci s ML.

- Simulačné prostredia - simulačné prostredia sú pre našu budúcnosť kľúčové. Pre vykonávanie niektorých činností sa zavedením simulátorov ušetrí čas a náklady na testovanie.

Tešíme sa na spoločné stretnutie a prípadne na našu ďalšiu spoluprácu.

A.5 Príloha A

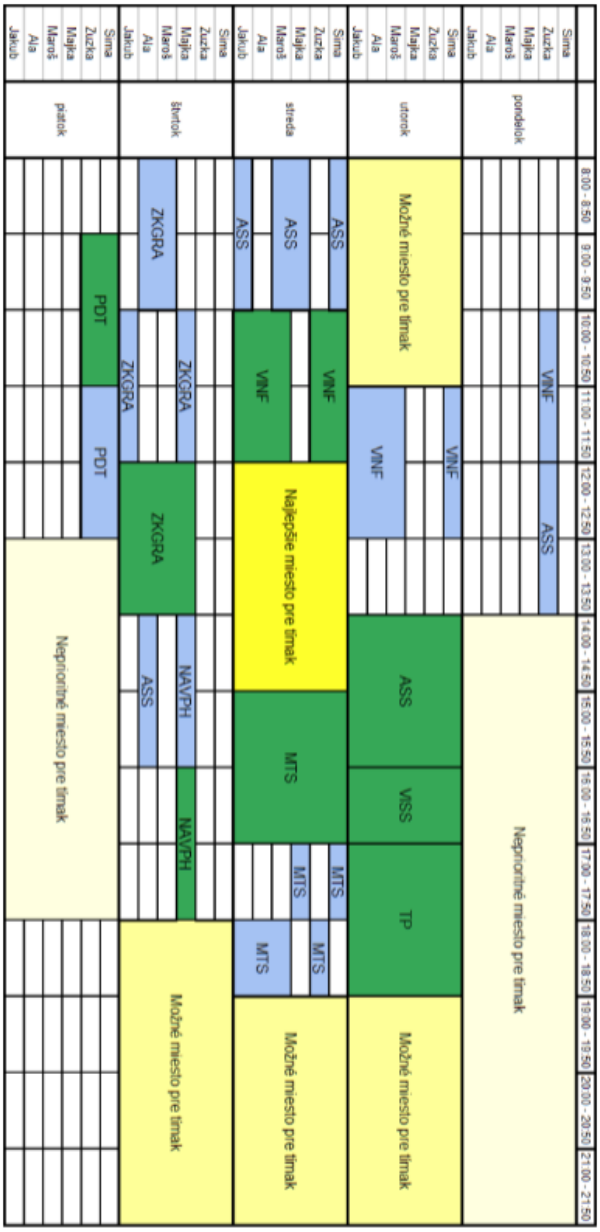
Zoradenie tém podľa priority

Priorita	Téma	Číslo témy
1.	Automatické rozpoznávanie spektier [ARS]	08
2.	Podporný informačný systém pre študijné oddelenie [CROSS-CHECK]	19
3.	Korekcia dynamických vlastností virtuálnych modelov komponentov vozidiel [CarComponents]	13
4.	VR laboratórium pre dištančné vzdelávanie [VRLab]	17
5.	Educational and coworking driven orchestration portal [EDUCO]	05
6.	Inteligentný informačný systém zameraný na projektový manažment a automatizáciu procesu verejného obstarávania [iPP]	16
7.	FIFé International Cat Show [MIAOW]	18
8.	Transformácia priestorov na bezpečné a inteligentné miesta na prácu [SmartSpace]	06
9.	Vzdialené monitorovanie zdravotného stavu človeka pomocou E-Health	09
10.	Safety panel a spätná analýza údajov pre vývoj autonómneho vozidla [avPANEL]	12
11.	Webové IDE pre ASIC [ASICDE]	02
12.	Platforma pre sledovanie dodávateľského reťazca s využitím technológie blockchain [S-Chain]	14

Tabuľka 1: Zoradenie tém podľa priority.

A.6 Príloha B

Tímový rozvrh



Obr. 22: Rozvrh. Miesta pre tímové stretnutia / spoločnú prácu sa bližšie špecifikujú podľa preferencií vedúceho.

B Prihláška na TP cup 2020

B.1 Náš tím

Sme mladý a flexibilný tím pripravený prekonať každú výzvu. Voláme sa Mikasa po postave z anime a po značke výrobcu volejbalových lôpt. Sme ochotní na sebe pracovať a získavať nové skúsenosti z rôznych oblastí informatiky. Väčšina členov nášho tímu sa zaujíma o strojové učenie a neurónové siete. V oblasti umelej inteligencie máme nadobudnuté skúsenosti z predmetu Inteligentná analýza údajov, Umelá inteligencia a z bakalárskych prác. O kvalite našej práce svedčia aj 2 bakalárske práce, ktoré boli ocenené dekanom našej fakulty. Okrem toho máme skúsenosti s návrhom softvéru, vývojom aplikácií, či virtuálnou realitou.
Nebojíme sa výskumu rôznych oblastí informatiky, akou je napríklad automatické rozpoznávanie spektier. V rámci rôznych pracovných príležitostí sme nadobudli skúsenosti s prácou v tíme a s vývojom aplikácií, preto veríme, že sa v tíme dokážeme popasovať s akýmkoľvek problémom.

B.2 Motivácia

Kvalita olivových olejov je posudzovaná primárne na základe chuti, vzhľadu a vône. V súčasnosti je hodnotenie kvality oleja regulované normami Európskej únie. Tieto nariadenia definujú 3 triedy olivového oleja, „extra panenský“ (EVOO), „panenský“ (VOO) a „lampantový“ (LOO). Každá vzorka oleja je klasifikovaná do 1 z týchto tried pomocou panelového testu. Experti v danej oblasti posudzujú oleje v rôznych kategóriách a prideľujú mu určité skóre. Aplikovaním štatistickej analýzy na skóre, ktoré bolo udelené účastníkmi testu sa vyhodnotí trieda oleja. Tento proces býva časovo náročný a drahý, preto sa hľadajú iné alternatívy vyhodnocovania kvality olivových olejov. Cieľom nášho projektu je umožniť rýchlejšiu analýzu oleja a vyhodnotenie bez nutnosti časovo a finančne náročnej manuálnej analýzy spektra expertom v danej oblasti.

B.3 Náš projekt

Náplňou nášho projektu je automatické rozpoznávanie spektier, ktoré nám umožní analyzovať kvalitu vzoriek olivových olejov. Pre tento účel vytvárame aplikáciu, ktorá bude

využívať algoritmy strojového učenia a neurónových sietí. Využitím rôznych metód chceme dosiahnuť čo najvyššiu možnú presnosť pri klasifikovaní olivových olejov. Pomocou aplikácie bude možné zobraziť namerané vzorky vo forme diagramov, aby boli dáta zrozumiteľné aj pre ľudí, ktorí nie sú expertami v oblasti analýzy spektier.

B.4 Ciele projektu

Hlavným cieľom nášho projektu je vytvoriť systém na automatické rozpoznávanie spektier, ktorý využíva metódy strojového učenia a neurónových sietí na klasifikáciu olivových olejov.
Navrhnutý systém by mal vedieť spracovávať spektrum, ktoré je výstupom merania meracieho zariadenia AMIS (Advanced Ion Mobility Spectrometer). Spektrum si môžeme predstaviť ako časovú sekvenciu meraní spektrálnych koeficientov uloženú vo formáte *.txt. Systém by mal vedieť nájsť rozličné črty pre olivové oleje, na základe ktorých bude môcť vyhodnotiť triedu oleja.
Na nami identifikované požiadavky sme navrhli nasledujúce riešenie:

- Poskytnuté dáta obsahujú vzorky 200 olejov, ktoré sú rozdelené do 3 tried. Každá vzorka oleja obsahuje viac ako 10 000 spektier uložených v samostatnom súbore. Z týchto súborov vytvoríme celistvú tabuľku, ktorá bude obsahovať všetky informácie o vzorkách.

- Každý olej je charakterizovaný nameranými hodnotami v spektrách. Zo spracovaných dát vypočítame črty, ktoré následne použijeme pri klasifikácií. V projekte sa zameriame na základné črty, akými sú priemer alebo maximum, taktiež vyhľadáme aj komplexnejšie črty. Pozrieme sa aj na hodnoty v špecifických spektrách, kedy môže nastať zmena oproti inej triede.

- V projekte sa zameriame na niekoľko algoritmov strojového učenia, ktoré budeme využívať na klasifikáciu olivových olejov. Chceme vyskúšať algoritmy, akými sú Náhodný les (Random Forest), KNN, SVM a neurónové siete. Natrénované modely vyhodnotíme a tie najúspešnejšie z nich použijeme vo výslednom produkte. Budeme pritom dohliadať na celkovú úspešnosť, ako aj úspešnosť pre jednotlivé triedy.

- Vytvoríme systém na rozpoznávanie tried olejov, pomocou ktorého bude môcť používateľ zobrazovať dáta vo forme diagramov a klasifikovať vzorku oleja. Klasifikačné modely a spôsoby vyhodnotenia si bude môcť používateľ prispôsobiť.

Našim cieľom je vytvoriť funkčnú a spoľahlivú aplikáciu, ktorá bude môcť byť v praxi využívaná, aby sa zabránilo zamieňaniu kvalitných a nekvalitných olejov, aby nedochádzalo k stratám ziskov výrobcov olivových olejov. Veríme, že kombináciou našich skúseností, nápadov a moderných technológií dokážeme vytvoriť presný a spoľahlivý produkt.

C Metodika komunikácie

C.1 Komunikačné nástroje

Metodika komunikácie definuje pravidlá komunikácie medzi jednotlivými členmi tímu. V tejto metodike sú predstavené komunikačné kanály, ktoré sa používajú v tíme, na to aby sa predchádzalo vzniku komunikačnému ruchu.

Google Meet

Nástroj Google Meet sa používa na pravidelné online stretnutia členov tímu s vedúcim tímu z dôvodu dištančnej výučby. Tieto stretnutia sú naplánované v rozvrhu členov tímu.

MsTeams

Nástroj MsTeams sa používa na pravidelné online stretnutia členov tímu.

Komunikačné kanály

- General - určený na komunikáciu všeobecných vecí a riešenie problémov v projekte
- Classification - určený na komunikáciu o spracovávaní dát, vytváraní klasifikačných metód a vyhodnocovaní klasifikátorov
- Application - určený na komunikáciu k vytváraniu wireframov, frontendu a backenduaplikácia, vznikol premenovaním kanálu pre web, ktorý bo už nepotrebný
- Stand-up - kanál, ktorý nahrádza ranné stand-upy, členovia tímu sem napíšu čo idú robiť alebo aké problémy aktuálne riešia
- Ready for review - kanál, kde sa pridávajú hotové tasky pripravené na review

Členovia tímu poznajú zameranie jednotlivých komunikačných kanálov a využívajú ich podľa spomenutých pravidiel. Vzniknuté problémy rieši tím spoločne, so vzniknutými problémami sa člen tímu netrápi sám. Ak chce člen spomenúť v komunikácií iného člena tímu, označí ho pomocou @. Každý člen tímu môže vytvoriť nový kanál. Nový kanál

musí byť výstižne pomenovaný a musí byť predpoklad, že sa bude v budúcnosti využívať. Po vytvorení nového komunikačného kanálu ho daný člen pridá a primerane opíše v metodike.

Jira

Komunikácia o úlohách (otázky, riešenie bugov, ...) sa rieši prostredníctvom pridávania komentárov k úlohám v nástroji Jira. Každý člen tímu môže pridať komentár k akejkoľvek úlohe. Komentáre musia byť stručné a týkať sa konkrétnej úlohy. V prípade, že je komentár príliš dlhý je vhodné využiť komunikačný kanál v násroji MsTeams.

Email

Komunikácia tímu o projekte s ďalšími stranami prebieha prostredníctvom emailovej adresy mikasa.fiit@gmail.com. Každý člen tímu má k emailu prístup a je povinný ho pravidelne sledovať. V prípade, že člen tímu príjme novú správu, informuje o nej zvyšok tímu.

Messenger

Messenger slúži na súkromnú komunikáciu členov tímu. Pomocou tohto komunikačného kanálu sa dohadujú stretnutia členov tímu. Členovia tímu môžu použiť tento kanál na upozornenie ostatných členov o dokončení úlohy.

D Metodika vývoja v Pythone

D.1 Manipulácia s projektom

Pre zefektívnenie a zachovanie rovnorodosti práce na projekte sú v tomto dokumente definované pravidlá manipulácie s projektom.

Odporúčaným IDE pre písanie programov v jazyku Python je Pycharm. Zvoleným jazykom pre vytváranie pomenovaní je anglický jazyk.

Názvoslovie pre súbory a adresáre

Všetky priečinky budú pomenované:

- v anglickom jazyku
- s veľkým začiatočným písmenom

Súbory s koncovkou *.py* sa budú pomenovávať v závislosti od ich obsahu:

- **Triedy** - V prípade, že obsahom súboru bude definícia triedy, súbory sa pomenujú cez cammelCase, s výnimkou veľkého začiatočného písmena. Podobne ako súbory. Napríklad trieda, ktorá definuje príspevky pre používateľov sa bude volať *UserPost*.
- **Scripty** - Ak obsahom súboru nie sú triedy, súbory budú pomenované cez *lower_case*, t.j. všetky slová v názve budú začínať malým začiatočným písmenom a budú oddelené cez podčiarkovník. Napríklad, pre funkcie upravujúce reťazce, by bol názov súboru *string_modifier*.

Adresáre v projekte Clasisfication

Pre zefektívnenie, sprehľadnenie a zjednodušenie práce na projekte budú v hlavnom projekte existovať nasledujúce adresáre:

- **Predprocessing** - všetky súbory z predspracovania.
- **FeatureSelection** - funkcie pre vytváranie čŕt.
- **Neurons** - neurónové siete.
- **StandardClasification** - štandardné klasifikátory.

- **Data** - adresár obsahuje všetky dáta.
  - **Test** - adresár obsahuje testovaciu vzorku dát.
    - **Raw** - predspracované dáta.
    - **Features** - dáta s vypočítanými črtami.
  - **Train** - adresár obsahuje trénovaciu vzorku dát a menšie vzorky
    - **Raw** - predspracované dáta.
    - **Features** - dáta s vypočítanými črtami, adresáre s obrázkami a diffmapami a súbormi šablóna a maska pre diff.
- **Statistics** - adresár obsahujúci štatistické údaje, grafy a iné.
  - **Charts** - adresár obsahujúci obrázky výsledkov modelov a matíc zámen.
- **UnitTests** - adresár obsahujúci unit testy.
- **Models** - adresár obsahujúci klasifikačné modely.

D.2 Písanie programov

Hlavička dokumentu - Importy a globálne premenné

Všetky globálne importy sa budú nachádzať na začiatku súboru. Prvý import sa umiestni na riadok č.1. V prípade že sa za importom globálne premenné:

- nachádzajú - Nasledujú za importom 1 prázdny riadok.
- nenachádzajú - Nasledujú za importom 2 prázdne riadky.

Následne budú umiestnené deklarácie globálnych premenných. Názvy globálnych premenných budú napísané veľkými písmenami, oddelených podčiarkovníkom. Napríklad globálna premenná pre maximálnu hodnotu bude označená ako *MAX_VALUE*. Príklad je uvedený na obrázku č. 26

```
import csv

MAX_VALUE = 1000000


def function:
    # some code to add
```

Obr. 23: Príklad deklarácie hlavičky súboru

Telo dokumentu

Funkcie

Medzi funkciami budú 2 voľné riadky.
V prípade, ak sa jedná o spustiteľný súbor, je povinná funkcia main v tvare:
if __name__ == "__main__":

Podmienky

Uprednostnenie *or* a *and* pred || a *&&*.
Pre podmienky *if, elif, else* bude blok kódu začínať vždy na novom riadku.

Cykly

Blok kódu začínať vždy na novom riadku.

Polia s kľúčovou hodnotou

Pri deklarácii polí s kľúčovou hodnotou sa odporúča nepridávať čiarku na koniec posledného riadku. Ukážka je uvedený na obrázku č. 27

```
thisdict =  {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
```

Obr. 24: Príklad deklarácie polí s kľúčovou hodnotou

Reťazce

Deklarácie reťazcov budú primárne začínať a končiť dvojitou úvodzovkou ”.

Tvorba pomenovaní

Črty

Názvy čŕt sa skladajú z dvoch častí, ktoré sú oddelené podčiarkovníkom. Prvá časť je názov funkcie, ktorá danú črtu ráta. Druhú časť tvorí stĺpec alebo viacero stĺpcov (riadkov), nad ktorými sa daná črta ráta.
Príklad črty priemer všetkých hodnôt zo stĺpca v_1:

mean_v_1

Ak je v názve viac ako jeden stĺpec (riadok) oddeľujú sa pomocou pomlčky. V prípade názvov, kde sa nachádzajú viaceré stĺpce v kombinácii s riadkami píšu sa iba čísla daného stĺpca (riadku). Ak názov funckie tvorí viac slov spájajú sa pomocou pomlčky.
Príklad výpočtu maxima z bodu (v-s-n):

max-point_1-7-10

V prípade, že chceme pri nejakej funkcii vyrátať črtu nad všetkými stĺpcami použijeme ako názov stĺpca slovo „ALL“.
Príklad výpočtu minima nad všetkými stĺpcami:

min_ALL

Verziovanie modelov

Názvy modelov konkrétnych klasifikátorov sa budú písať cez lower_case, oddelené od časovej pečiatky pomocou pomlčky. Každý model končí príponou *.joblib*.

Časová pečiatka je v tvare "RRRR-MM-DD HH-MM-SS". Rok mesiac a deň sú navzájom oddelné pomlčkou. Rok mesiac a deň sú oddelené od hodín minút a sekúnd medzerou. Hodiny minúty a sekundy sú taktiež navzájom oddelené pomlčkou.

Príklady

random_forest-2020-11-25 07-22-36.joblib

svm-2020-11-25 22-02-06.joblib

Premenné

Názvy premenných v programe sa budú písať cez lower_case. Názvy premenných by nemali obsahovať slovesá, preferované sú podstatné mená. Odporúča sa používať jednotné číslo, s malými výnimkami. Napríklad názov premennej, ktorá uchováva príspevky používateľov sa bude volať user_posts.

Triedy

Triedy budú pomenované cez cammelCase s veľkým začiatočným písmenom. Napríklad trieda uchovávajúca príspevky používateľov sa bude volať *UserPosts*.

V prípade nedefinovaných pravidiel sa budeme spoliehať na štandardy PEP-8 a PEP-257. Pravidlá definované v tomto dokumente sú nadradené pravidlám týchto štandardov.

E Metodika testovania

E.1 Testovanie

Testovanie je dôležitou súčasťou vývoja, nakoľko pomáha validácií riešenia a identifikácií chýb. Preto by malo zastrešovať čo najväčšiu časť vytvoreného produktu.

Vzhľadom na druh produktu, ktorý vyvíjame a dátovú prácu s ním súvisiacu, sme sa rozhodli využiť najmä jednotkové testy a akceptačné testy.

Jednotkové testy

Jednotkové testy budú vykonávané pomocou knižnice **pytest**. V rámci využívania tejto knižnice je preto nutné dodržať konvenciu pomenovávania.

- **Súbory:** názov súboru sa musí začínať slovom „test". Aj napriek metodike vývoja v Pythone sa budú súbory pomenovavávať cez *lower_case* nezávisle na tom, či obsahujú triedy alebo nie. Príklad: *test_features.py*
- **Triedy:** Podľa metodiky vývoja v Pythone sa triedy budú pomenovávať pomocou *CammelCase*. Prvé slovo v triede musí byť slovo „Test".
- **Metódy:** Metóda musí začať slovom „test". Dodržaná bude metodika vývoja v Pythone, čo znamená, že metódy budú písané ako lower_case.

Príklad jednotkového testu:

```
def test_mean_normal(self):
    value = add_two_numbers(5, 2)
    assert value == 7
```

Nutnosť jednotkových testov

Jednotlivé komponenty/metódy budú testované jednotkovými testmi podľa uváženia osoby, ktorá daný komponent/metódu napísala, čím musí zvážiť nutnosť jej iného ako fyzického testovania.

Odporúča sa však mať jednotkovými testami pokrytú čo najväčšiu časť kódu, vďaka čomu si môžu vývojári byť neustále istí správnym fungovaním komponentov/metód.

Existencia jednotkových testov taktiež napomôže review nových častí kódu, vďaka čomu sa ušetrí čas na manuálnom testovaní.

Taktiež sa odporúča, aby jednotkový test k určitému komponentu/metóde napísala osoba, ktorá daný komponent/metódu vytvorila. Ušetrí sa tak čas potrebný pre naštudovanie komponentu/metódy iným členom tímu.

Organizácia testov

Všetky súbory s testmi budú v špeciálnom adresári „*UnitTests*" z dôvodu ich jednoduchšej organizácie. Testy budú organizované do súborov a tried v závislosti od ich logickej organizácie.

Každá metóda v súboroch s testmi zobrazuje samostatný test.

Spustenie jednotkových testov

Testovanie pomocou jednotkových testov sa bude spúšťať príkazom „*python -m pytest -v*" pomocou konzoly z root adresáru. Prepínač *-v* slúži pre detailnejší výpis úspešnosti jednotlivých testov a je voliteľný.

Akceptačné testy

Na konci fázy tvorby softvéru, ktorý bude možné použiť pre používateľsky jednoduchšiu klasifikáciu vzoriek oleja sa vykoná akceptačný test.

Akceptačný test bude vytvorený na začiatku fázy tvorby softvéru. Test bude spísaný na základe dohody s product ownerom. Zadané budú možné vstupy a predpokladané výsledky k nim. Testované musia byť aj hraničné vstupy, ktorými sa overí schopnosť softvéru reagovať na chyby a zotaviť sa z chybových stavov.

Pre akceptovanie akceptačného testu je nutné bez výnimky splniť všetky dohodnuté body testu.

F Metodika splnenej úlohy

F.1 Definition of Done

V rámci vývoja softvéru viacerými členmi, je nutné si v rámci tímu dohodnúť jasné pravidlá, ktoré okrem iného označujú aj to, kedy je možné určitú činnosť považovať za dokončenú. Takéto pravidlá nazývame Definition of Done.

Definition of Done v rámci nášho tímu je prispôsobená podľa veľkosti celku, ktorý je potrebné označiť ako „dokončený". V rámci projektu teda rozlišujeme definíciu dokončenia pre:

- Task
- User Story
- Projekt

Definition of Done pre Task

Task je najmenšia jednotka, ktorú je možné považovať za dokončenú. Ako task sa označuje činnosť, ktorá je vykonateľná jednou osobou v rámci User story.

Task má jasne stanovený názov, ktorý označuje čo je nutné vykonať. Činnosť však lepšie opisuje popis tasku, ktorý je viac detailnejší ako samotný popis.

Pre považovanie tasku za dokončený je nutné:

- Vykonať popísanú činnosť v rámci kontextu projektu
- V prípade kódu:
  - Otestovať samotný kód (jednotkové testy / manuálne)
  - Otestovať správanie kódu v spolupráci s inými komponentmi
  - Opísať novovytvorený kód
- Prejsť si review tasku s iným členom tímu
- Zapracovať prípadné zmeny z review a opäť vykonať review

Definition of Done pre User Story

User Story je celok, ktorý má v rámci projektu priniesť product ownerovi určitú pridanú hodnotu. Je zložený z Taskov, ktoré presne popisujú činnosti, ktoré je nutné vykonať pre splnenie User Story.

Aby sme mohli User Story považovať za dokončené, je nutné:

- Považovať všetky Tasky súvisiace s User Story za dokončené
- Priniesť funkcionalitu opísanú v zadaní User Story
- Overiť funkcionalitu ako celok
- Zhodnotenie všetkých členov tímu o schválení User Story za dokončené

Definition of Done pre Projekt

Projekt je možné považovať za dokončený, keď:

- Dodá sa plánovaná funkcionalita
- Presnosť rozpoznávania oleja bude na najlepšej možnej hranici
- Tím zrealizuje všetky možné nápady zlepšovania výsledkov
- Nebude viac možné posunúť projekt na lepšiu úroveň

G Metodika práce s úlohami

G.1 Práca s úlohami

Pre zachovanie rovnorodosti práce, podporu spolupráce tímu na projekte a prevencii vzniku nezrovnalostí pri práci s úlohami na projekte sú v tomto dokumente definované pravidlá manipulácie s úlohami, spolu s definíciou typov úloh.

Zvoleným softvérom pre vytváranie a správu rôznych typov úloh je Jira. Zvoleným jazykom pre vytváranie úloh je jazyk slovenský, jazyk pre názvy stavov úloh je jazyk anglický.

Issue types

Epic

V projekte sa vyskytujú nasledovné epiky:

- **Administrácia** - združuje všetky administratívne úlohy a user stories. Patria sem všetky issue types, ktoré sa týkajú administratívnej časti projektu, ako napríklad TP cup, udržiavanie metodík a príprava na stretnutia.
- **Klasifikácia** - združuje všetky úlohy súvisiace s klasifikačnou časťou projektu. Napríklad rozpoznávanie druhov olejov a vytváranie nových čŕt.
- **Web** - združuje všetky úlohy súvisiace s webovou reprezentáciou tímu. Napríklad vytvorenie wireframov pre web, alebo vytvorenie frontendu. Tiež zastrešuje spravovanie servera.
- **Aplikácia** - združuje všetky úlohy súvisiace s aplikačnou časťou projektu. Napríklad špecifikácia požiadaviek, návrh prípadov použitia (Use case), vytvorenie aplikácie.

Používateľský príbeh

Používateľské príbehy, anglicky user stories, sa budú písať s vedúcimi projektu, v prípade, že sa tak nestane, musia byť príbehy akceptované vedúcimi. Podúlohy sú tvorené členmi tímu.

Do opisu príbehu sa doporučuje zahrnúť:

- **Úvod** - Ako KTO?, potrebuje ČO? a PREČO?
- **Analýza** - jednoduchú analýzu úlohy.
- **Podúlohy** - zoznam podúloh potrebných na splnenie príbehu.
- **Must be** - zoznam bodov, ktoré musia byť v rámci úlohy splnené pri dodaní, aby mohlo byť dodanie akceptované.

Podúloha

Podúlohy, anglicky subtasky, sú tvorené členmi tímu, na spoločných stretnutiach. Story pointy podúlohy sú prideľované na spoločných stretnutiach metódou planning poker.

Do opisu podúlohy sa doporučuje napísať:

- **Opis** - Stručný opis úlohy.
- **Vstupy a výstupy** - V prípade implementačných úloh sa môže nachádzať príklad vstupu a výstupu práce alebo pridanej funkcionality.
- **Testy** - Rozhodnutie, či je potrebné pre implementačnú úlohu vytvárať testy.
- **Kontrolovanie výstupu** - Zoznam bodov, ktoré treba skontrolovať, keď sa úloha nachádza v stave *ready for review*.

Manipulácia s úlohami

Vytváranie úloh a presúvanie úloh do šprintu

Úlohy sa prednostne vytvárajú a presúvajú do šprintov na tímových stretnutiach. Pridávanie úloh do šprintu počas jeho priebehu sa neodporúča. Výnimkou je nahlásenie chýb v projekte.

Stavy v úloh

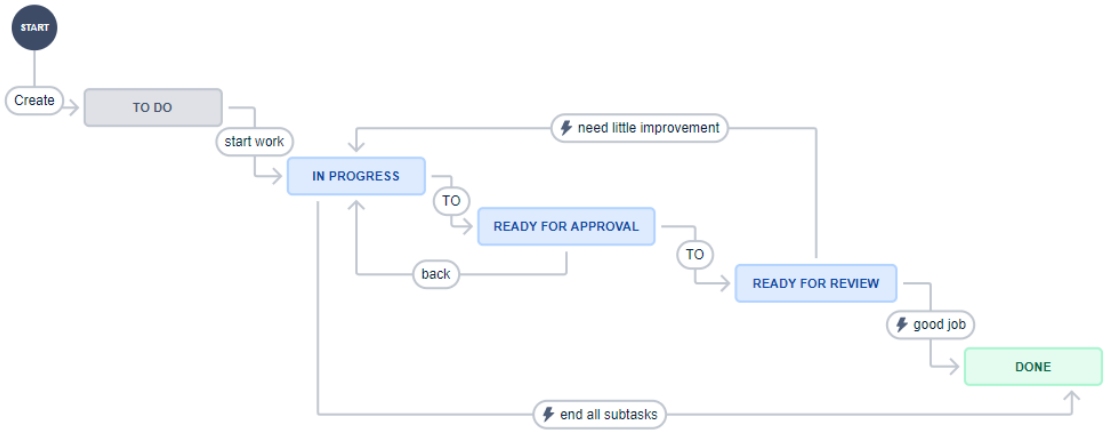
Zoznam stavov úloh:

- **To Do** - Zoznam úloh, ktoré treba dokončiť počas šprintu.
- **In Progress** - Zoznam rozpracovaných úloh alebo úloh, na ktorých sa začína pracovať.

- **Ready For Approval** - Zoznam dokončených úloh, ktoré sú pripravené na akceptáciu zo strany iného člena tímu. Uľahčuje review a zodpovedá za funkčnosť úlohy.

- **Ready For Review** - Zoznam dokončených úloh, ktoré sú pripravené na kontrolu. Úloha musí byť dokončená v zmysle pravidiel Metodiky splnenej úlohy. Kontrolu úloh vykonáva Jakub Kučečka.
- **Done** - Hotové úlohy, ktoré prešli kontrolou.

Presun medzi stavmi je povolený len v jednom smere, zľava doprava. Úlohy môžu presúvať ľubovoľní členovia tímu s výnimkou stavu *Ready For Review*. Z tohto stavu úlohy môžu presúvať aj spätne do stavu *In Progress* a môže ich presúvať len člen tímu zodpovedný za kontrolu. Ďalšou výnimkou je stav *Ready For Approval* z ktorého sa úloha môže taktiež vrátiť do stavu *In Progress*.



Obr. 25: Diagram prechodov medzi stavmi úloh

H Metodika verziovania programu

H.1 Práca v nástroji GitHub

Pre zachovanie rovnorodosti práce, podporu spolupráce tímu na projekte a prevencii vzniku nezrovnalostí vo verziách programu na projekte sú v tomto dokumente definované pravidlá manipulácie s verziami projektu.

Zvoleným kolaboratívnym nástrojom na projekte je Github. Slovenský jazyk sa bude využívať pri vytváraní:

- potvrdzujúcich správ, (ďalej commit message)
- názvov vetiev patriacich úlohám

V anglickom jazyku budú pomenované vetvy master a dev.

Vetvy

Master

Hlavnou vetvou je vetva master. Do tejto vetvy sa budú pridávať len hotové funkcionality spustiteľného programu. V tejto vetve by sa **nesmie** nachádzať program s nedokončenou funkcionalitou a s chybami.

Nad touto vetvou **môže** existovať len vetva dev. Do tejto vetvy sa bude vykonávať merge vetvy dev a to **prednostne** na konci šprintu po odsúhlasení v tíme. Pridaná časť programu **musí** mať hodnotu pre zákazníka. Nedokončená funkcionalita sa do tejto vetvy nepridáva.

Dev

Vetva určená na vývoj je vetva dev. Do tejto vetvy sa budú pridávať len ukončené funkcionality z pridelených úloh. Ak si obsah úlohy žiada testovanie, nesmie byť do vetvy pridaná neotestovaná časť programu. Výnimkou tohto prípadu je scenár, kedy je dané testovanie súčasťou inej úlohy.

Nad touto vetvou **môžu** vznikať len vetvy, ktoré sú vytvorené za účelov splnenia pridelenej úlohy. Do vetvy dev sa môže vykonať merge len s vetvou, ktorej:

- zodpovedajúca úloha je splnená na 100%

- pridaný program zodpovedajúci úlohe prešiel kontrolou druhej strany.

Spájanie vetiev bude vykonávať Jakub Kučečka. Riešenie konfliktov počas spájania bude riešiť osoba, ktorej úloha bola pridelená.

Vetvy úloh

Pre každú úlohu bude existovať vetva, ktorá sa vytvorí z vetvy dev tesne pred začatím programovania a zanikne po ukončení úlohy vykonaním merge do vetvy dev. Pre pod úlohy sa nemusí vytvárať vetva.

Vetvy budú nazývané podľa čísla úloh, priradené systémom Jira (bez diakritiky). Vzor:

ASR-1_vytvorenie_metodiky_verziovania_programu

Okolo pomlčky na nenachádzajú medzery. Za id úlohy nasleduje podčiarkovník a názov úlohy v slovenskom jazyku. Všetky začiatočné písmená sú malé a slová sú rozdelené podčiarkovníkom.

Commit messages

Príklad commit message:

ASR-1 pridanie funkcionality ....

Začína sa id úlohy, nasleduje popis k pridanej funkcionalite. Odporúča sa podrobný popis.

I Metodika vývoja aplikácie

I.1 Manipulácia s aplikáciou

Pre zefektívnenie a zachovanie rovnorodosti práce na projekte sú v tomto dokumente definované pravidlá manipulácie s aplikáciou.

Odporúčaným IDE pre písanie programov v jazyku Python je Pycharm. Zvoleným jazykom pre vytváranie pomenovaní je anglický jazyk. Na integráciu medzi Python-om a JavaScript-om budeme používať Eel

Názvoslovie pre súbory a adresáre

Všetky priečinky budú pomenované: *v anglickom jazyku* Keďže aplikácia je vyvíjaná pomocou technológii Python, JavaScript, CSS a HTML, jednotlivé súbory sa budú pomenováva v závislosti od ich obsahu:

- **Python triedy**. V prípade, že obsahom súboru bude definícia triedy, súbory sa pomenujú cez cammelCase, s výnimkou veľkého začiatočného písmena. Napríklad trieda, ktorá definuje príspevky pre používateľov sa bude volať *Sample*.

- **Python scripty**. Ak obsahom súboru nie sú triedy, súbory budú pomenované cez *lower_case*, t.j. všetky slová v názve budú začínať malým začiatočným písmenom (okrem štartovacieho scriptu) a budú oddelené cez podčiarkovník. Napríklad, pre funkcie upravujúce reťazce, by bol názov súboru *ASR_app*.

- **JavaScript scripty**. Rovnako ako pri Pythone, všetky subory budú pomenované *lower_case*, t.j. všetky slová v názve budú začínať malým začiatočným písmenom a budú oddelené cez podčiarkovník.

- **CSS**. Súbory budú začínať slovom ”stylesheet”nasledované podčiarkovníkom a ľubovoľným *lower_case* pomenovaním.

- **HTML**. Aplikácia bude obsahovať iba jeden HTML súbor a to index.html, ten musí byť umiestnený v Application/public/index.html.

Adresáre v projekte Application

Pre zefektívnenie, sprehľadnenie a zjednodušenie práce na projekte budú v projekte Application existovať nasledujúce adresáre:

- **eel**. Súbory potrebné pre fungovanie knižnice eel.
- **lib**. Scripty pre správy a jazyky.
  - **Classification**. Pôvodné súbory klasifikácie.
- **log**. Adresár pre ukladanie logov za behu programu.
- **public**. Obsahuje súbory rôznych typov, ako napríklad .css, .js, .html a ďalšie.
  - **images**. Adresár pre obrázky potrebné pre aplikáciu, teda ikony a obrázky na pozadí.
  - **models**. Adresár pre natrénované modely, načítajú sa pri štarte aplikácie.
  - **tmp**. Adresár pre dočasné súbory.
    - **featureCSV**. Adresár pre CSV súbory určené pre klasifikáciu.
    - **imageCSV**. Adresár pre CSV súbory určené pre tvorbu obrázkov.
    - **images**. Adresár pre vytvorené PNG obrázky.

I.2 Písanie programov

Hlavička dokumentu - Importy a globálne premenné

Všetky globálne importy sa budú nachádzať na začiatku súboru. Prvý import sa umiestni na riadok č.1. V prípade že sa za importom globálne premenné:

- nachádzajú. Nasledujú za importom 1 prázdny riadok.
- nenachádzajú. Nasledujú za importom 2 prázdne riadky.

Následne budú umiestnené deklarácie globálnych premenných. Názvy globálnych premenných budú napísané veľkými písmenami, oddelených podčiarkovníkom. Napríklad globálna premenná pre maximálnu hodnotu bude označená ako *MAX_VALUE*. Príklad je uvedený na obrázku č. 26

```
import csv

MAX_VALUE = 1000000


def function:
    # some code to add
```

Obr. 26: Príklad deklarácie hlavičky súboru

Telo dokumentu

Funkcie

Medzi funkciami budú 2 voľné riadky.
V prípade, ak sa jedná o spustiteľný súbor, je povinná funkcia main v tvare:
if __name__ == "__main__":

Podmienky

Uprednostnenie *or* a *and* pred || a *&&*.
Pre podmienky *if, elif, else* bude blok kódu začínať vždy na novom riadku.

Cykly

Blok kódu začínať vždy na novom riadku.

Polia s kľúčovou hodnotou

Pri deklarácii polí s kľúčovou hodnotou sa odporúča nepridávať čiarku na koniec posledného riadku. Ukážka je uvedený na obrázku č. 27

```
thisdict =  {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
```

Obr. 27: Príklad deklarácie polí s kľúčovou hodnotou

Reťazce

Deklarácie reťazcov budú primárne začínať a končiť dvojitou úvodzovkou ".

JSON

Objekty JSON budú obklopené zloženými zátvorkami {}, zapisujú sa do párov kľúč - hodnota. Kľúče musia byť reťazce a hodnoty musia byť platným typom údajov JSON (reťazec, číslo, objekt, pole, logická hodnota alebo nulová hodnota). Kľúče a hodnoty sú oddelené dvojbodkou. Každý pár kľúč - hodnota je oddelený čiarkou.

Tvorba pomenovaní

Python premenné

Názvy premenných v programe sa budú písať cez lower_case. Názvy premenných by nemali obsahovať slovesá, preferované sú podstatné mená. Odporúča sa používať jednotné číslo, s malými výnimkami.

JavaScript premenné

Názvy premenných v programe sa budú písať cez camelCase. Názvy premenných by nemali obsahovať slovesá, preferované sú podstatné mená. Odporúča sa používať jednotné číslo, s malými výnimkami.

Python funkcie

Názvy funkcií v programe sa budú písať cez lower_case. Názvy funkcií by mali obsahovať kombinácie slovesa a podstatného mena. Odporúča sa používať jednotné číslo, s malými výnimkami.

JavaScript funkcie

Názvy funkcií v programe sa budú písať cez camelCase. Názvy funkcií by mali obsahovať kombinácie slovesa a podstatného mena. Odporúča sa používať jednotné číslo, s malými výnimkami.

Integrácia Python vs. JavaScript

Pri vývoji aplikácie pomocou jazykou Python a JavaScriptom pomocou rozhrania Eel je spojených zopár zaujímavostí. Ak deklarujeme funkcie v jednom či druhom jazyku a chceme ich použiť v inom, pred nazvom funkcie použijeme deklaráciu *@eel.expose*. Dôležité je ešte načítanie premenných krížom cez jazyky. Nevieme si výstup z Python scriptu uložiť do premennej v JavaScripte. Odporúča sa teda zavolať Python funkciu v init funkcii a poslať ju do konkrétnej JavaScript funkcii ako vidíme na obr. 28.

Python

```python
@eel.expose
def get_lang(language):
    set_conf("lang", language)

    if language == 'sk':
        return lang.svk_lang()
    elif language == 'en':
        return lang.eng_lang()
    else:
        print_log("Unknown language " + language)
        return lang.eng_lang()
```

JavaScript

```javascript
function init(lang) {
    eel.get_lang(lang)(initScene);
}

    ...

function initScene(lang) {

    setLang(lang);

    document.body.style.width = window.innerWidth + 'px';
    document.body.style.height = window.innerHeight + 'px';
    document.getElementById("disableContent").style.width = window.innerWidth + 'px';
    document.getElementById("disableContent").style.height = window.innerHeight + 'px';

    eel.get_conf("theme")(setTheme);
}
```

Obr. 28: Príklad integrácie funkcie get_lang() medzi Pythonom a JavaScriptom

Dict Python objekt bude prekonvertovaný na JSON pomocou funkcie json.dump(). V JavaScripte bude pársovať daný objekt funkciou JSON.parse(), tá ho zmení na objekt. K jednotlivým údajom pristupujeme pomocou *[]* (napr. *obj[”key”]*)). Naopak JSON objekt netreba preložiť na Python dict (napr. let data = ”models”: models, ”samples”: samples pošleme priamo ako data do Python funkcie).

CSS

V CSS súboroch sú použité premenné definované v *:root*{}, použité sú pomocou *var(meno premennej)*]. Názvy tried a identifikátory sú písane camelCase. Rovnaké atribúty vyberáme do samostatných tried a tie používame pre konkrétne komponenty. U url

sú použité relatívne cesty.

Jednotlivé css komponenty sú oddelené voľným riadkom a súbory nie sú minifikované kvôli prehľadnosti. Jednotlivé atribúty CSS komponentu sú na začiatku oddelené tabulátorom.

HTML

Triedy a id sú písane camelCase. Súbor musí obsahovať *¡!DOCTYPE html¿* a samotný *html* tag. V *head* tagu sú definované cesty k *stylesheet* a *js* súborom, tiež je zadefinovaná ikona aplikácie a jej názov.

Body tag obsahuje sekciu *nav* nasledovanú sekciou *div* s *id* = *content*. Daný div bude fungovať ako dynamický komponent. Jeho obsah sa zobrazí a skrýva podľa potreby.

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Dokumentácia k tímovému projektu

Inžinierske dielo

Vedúci tímu: Mgr. Martin Sabo, PhD., Ing. Marta Šoltésová Prnová, PhD.
Číslo tímu: 05
Kontakt: mikasa.fiit@gmail.com

Obsah

1 Úvod

V našom projekte sa venujeme automatickému rozpoznávaniu spektier olejov zo vzoriek získaných z AIMS (Advanced Ion Mobility Spectrometer). V tomto dokumente opisujeme priebeh projektu. Definujeme globálne ciele, ale aj ciele pre jednotlivé semestre, celkový pohľad na systém, organizáciu v podobe modifikovaných WBS diagramov a diagramu monolitickej architektúry nášho systému. V krátkosti sú opísané funkcionality, ktoré aplikácia poskytuje.

2 Globálne ciele

Hlavným cieľom nášho projektu je vytvoriť systém na automatické rozpoznávanie spektier, ktorý využíva metódy strojového učenia a neurónových sietí na klasifikáciu olivových olejov. Následne chceme tieto výsledky prezentovať v aplikácii.

2.1 Ciele zimného semestra

Počas prvého semestra sme sa na systém pozreli z vyššej perspektívy. Oboznámili sme sa s dátami a možnými smermi, akými by sme mohli ísť.

Za zimný semester sme pripravili viacero prototypov rôznych klasifikátorov. Dáta sme nie len spracovali do podoby .csv súborov, ale vytvorili sme aj obrázky, ktoré sme následne analyzovali.

Hlavné ciele, ktoré chceme splniť do konca zimného semestra sú:

- Predspracovanie dát a tvorba čŕt na vylepšenie pôvodných atribútov.
- Vytvorenie prvotných tradičných klasifikátorov (decision tree, knn, naive bayes, svm a random forest).
- Vytvorenie prvotných neurónových sieti (mlp a Vgg16).
- Klasifikácia obrázkov vytvorených z dát.
- Spísanie požiadaviek a návrhu obrazoviek pre aplikáciu.

2.2 Ciele letného semestra

V letnom semestri sa zameriame na dokončovanie klasifikácie a aplikácie. V tomto období sa bude konať súťaž TP cup, ktorej sme účastníkmi.

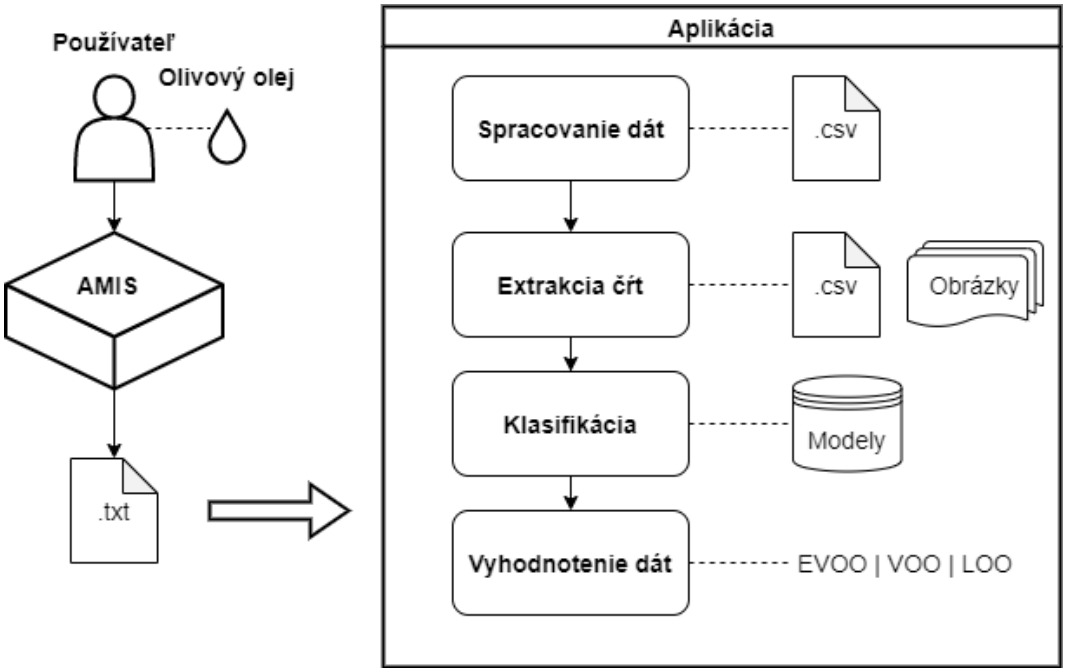
Ciele letného semestra sú:

- Vylepšovanie existujúcich klasifikátorov.
- Finalizácia tvorby klasifikačných modelov.
- Vyhodnotenie úspešnosti klasifikátorov.
- Dokončenie implementácie aplikácie pre klasifikáciu olivových olejov.
- Výhra na súťaži TP cup.

3 Celkový pohľad na systém

Na obrázku 7 vidieť monolitickú architektúru vytvorenej aplikácie. Aplikácia má 4 hlavné funkcionality:

- spracovanie dát
- extrakcia čŕt
- vyhodnotenie dát
- zobrazenie dát



Obr. 1: Pohľad na zjednodušenú architektúru

Vstupom pre aplikáciu sú .txt súbory vygenerované spektrometrom.

Hlavnou funkcionalitou aplikácie je klasifikácia olejov do troch tried EVOO, VOO a LOO. Na klasifikáciu sú použité tradičné klasifikátory KNN, SVM, Naive Bayes, Random forest a Decision tree. Z neurónových sietí je použitý MLP. Ďalšou funkcionalitou je vizualizácia jednotlivých olejov zo vstupných dát a príprava dát na klasifikáciu. V rámci prípravy dát na klasifikáciu sú dáta spracované do .csv súborov, taktiež sú vytvorené nové črty, z ktorých sú následne vybrané len tie podstatné.

Súčasťou aplikácie sú natrénované modely vyššie spomenutých klasifikátorov, ale aj modely pripravené pre trénovanie na vlastných dátach.

Aplikácia obsahuje aj šablóny, masky, ktoré sú generované pre každú triedu oleja a reprezentujú jej vlastnosti. Šablóna predstavuje očakávané hodnoty a maska zabezpečuje odstránenie variabilných hodnôt v rámci triedy, čiže hodnôt, ktoré sú rôzne v rámci olejov danej triedy. Tieto súbory boli vytvorené na základe existujúcich olejov a sú potrebné pre vypočítanie črty diff. Aplikácia taktiež ponúka možnosť vizualizácie vzoriek pomocou heatmapy.

3.1 Dosiahnuté výsledky

V rámci súčasného stavu riešenia boli dosiahnuté výsledky až na úrovni presnosti 89.29%. Tieto výsledky boli dosiahnuté pomocou klasifikátora Random Forest a pomocou kombinovaného rozhodovania na základe všetkých vytvorených klasifikátorov. Kompletné výsledky klasifikácie pre jednotlivé klasifikátory je možné vidieť v tabuľke 1.

Klasifikátor	**Presnosť klasifikácie (%)**
Decision Tree	73.81
k-NN	70.24
MLP	66.67
Naive Bayes	70.24
Random Forest	89.29
SVM	84.52
Total	89.29

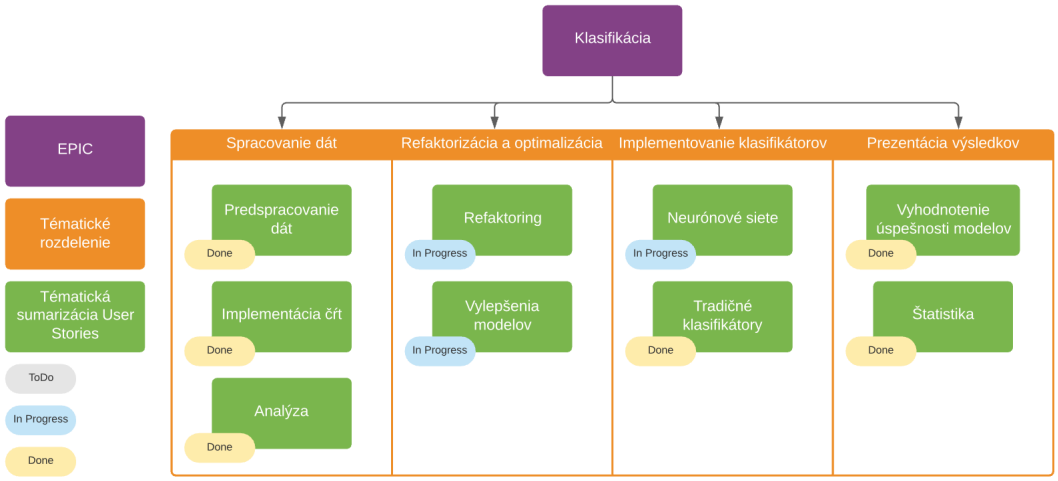
Tabuľka 1: Výsledky dosiahnuté pomocou jednotlivých klasifikátorov.

3.2 Modifikované WBS diagramy

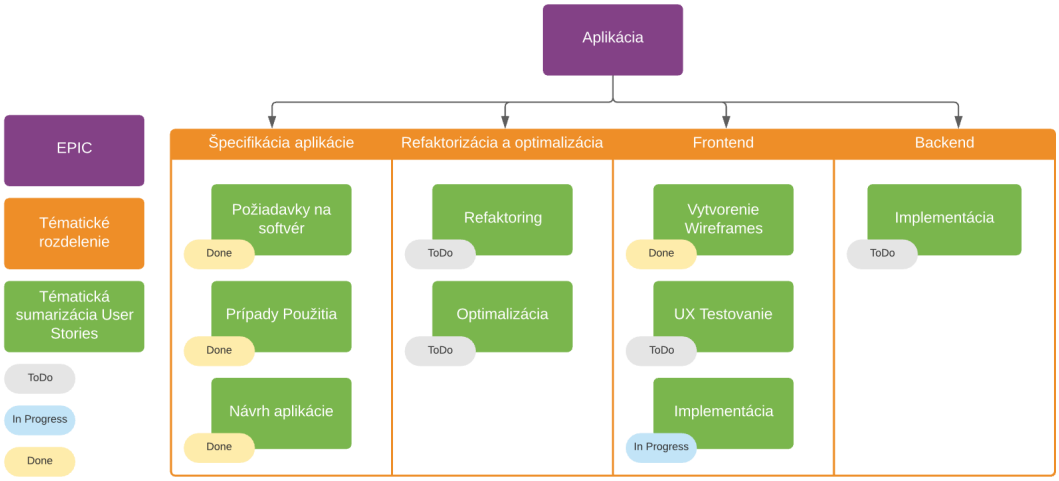
WBS diagramy znázorňujú menšie komponenty projektu. Názvy komponentov sú na diagrame znázornené fialovou farbou. Tieto komponenty sú v Jire reprezentované ako Epic. Každý Epic má pridelené Story. Tie sme tematicky rozdelili. Tematické rozdelenie je na obrázkoch znázornené oranžovou farbou. Sumarizácia stories je znázornená zelenou farbou.

Stavy jednotlivých sumarizácií stories sú v ľavom dolnom rohu. Nezačatá Story je označená ako 'ToDo'. Začatá Story je označená ako 'In Progress' a dokončená ako 'Done'.

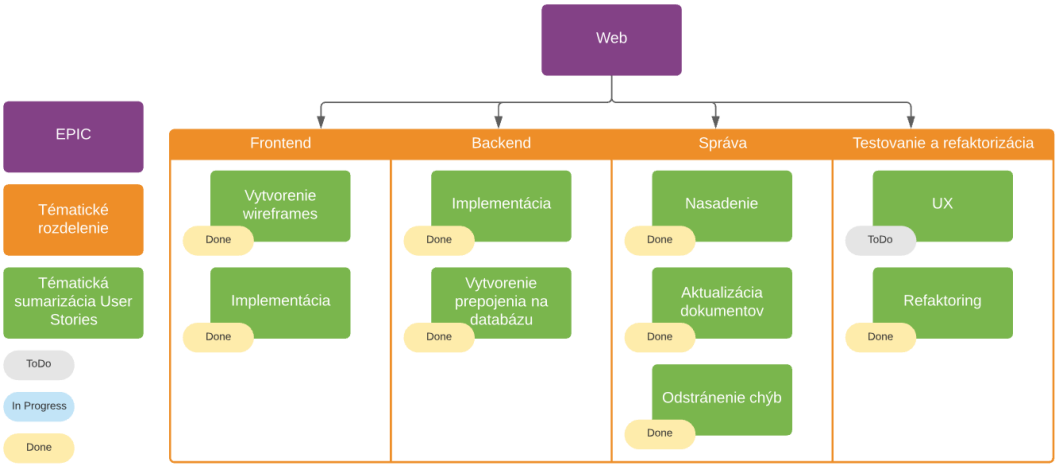
Zimný semester



Obr. 2: WBS diagram pre implementáciu klasifikátorov, na konci obdobia zimného semestra.

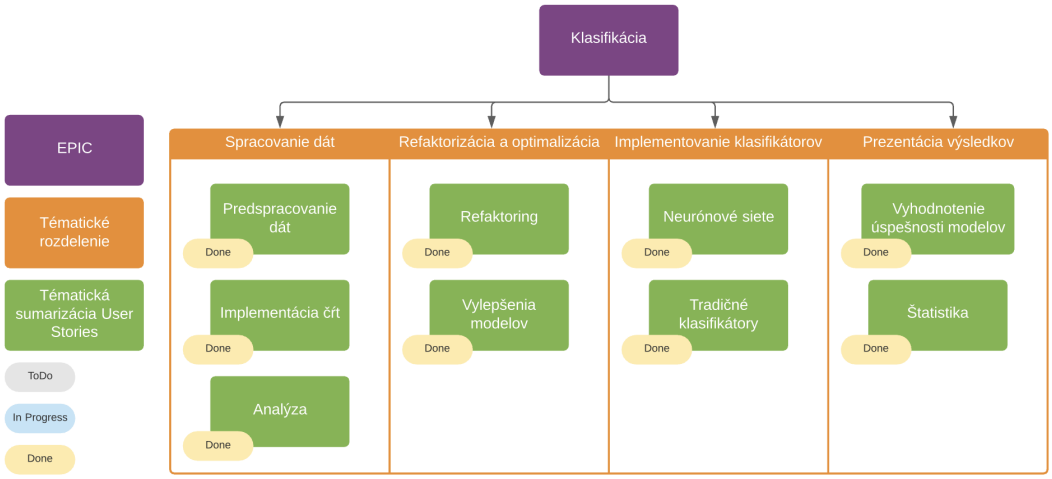


Obr. 3: WBS diagram pre implementáciu aplikácie, na konci obdobia zimného semestra.

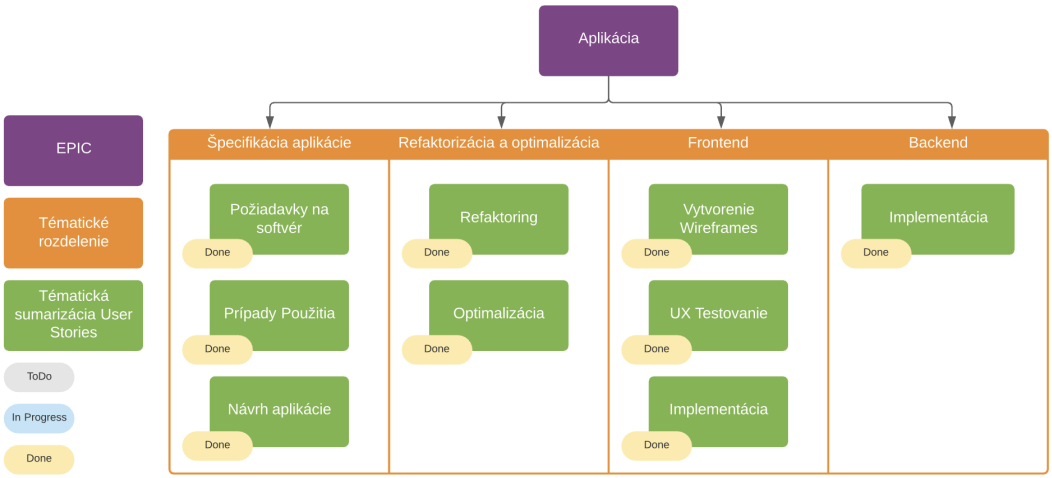


Obr. 4: WBS diagram pre implementáciu webu, na konci obdobia zimného semestra.

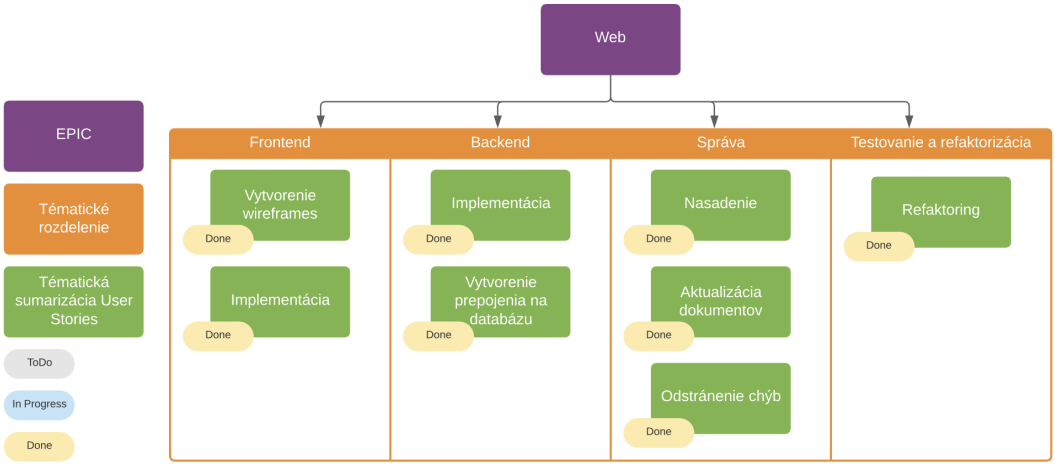
Letný semester



Obr. 5: WBS diagram pre implementáciu klasifikátorov, na konci obdobia letného semestra.



Obr. 6: WBS diagram pre implementáciu aplikácie, na konci obdobia letného semestra.



Obr. 7: WBS diagram pre implementáciu webu, na konci obdobia letného semestra.

A Posudok prototypu

Aplikácia bola nasadená u zadávateľa projektu, firmy MaSa Tech s.r.o., ktorej zakladateľ bol zároveň product owner-om nášho projektu. Vzhľadom na to, že sme nemohli zorganizovať osobné nasadenie priamo vo firme, product owner-ovi sme poskytli exe súbor našej aplikácie aj s inštalačnou a používateľskou príručkou. V tejto podkapitole sme zhrnuli spätnú väzbu na zvolený postup, kvalitu odvedenej práce a dosiahnuté výsledky, ktorú nám poskytol Mgr. Martin Sabo, PhD. po otestovaní aplikácie.

A.1 Hodnotenie prototypu

Tím sa zaoberal projektom, ktorý je zameraný na automatické rozpoznávanie spektier. Ich cieľom bolo klasifikovať 3 triedy olivového oleja z nameraných vzoriek, ktoré boli poskytnuté zadávateľom projektu. Výstupom práce bola aplikácia, ktorá je rozdelená do 3 väčších celkov, klasifikácia olejov, vizualizácia a natrénovanie nových modelov. Samotnému vyhodnocovaniu olejov predchádza ešte načítanie vzoriek olejov, ktoré je z časového hľadiska optimalizované. Oceňujem, že od posledného UX testovania bolo pridané načítavanie viacerých vzoriek naraz. Klasifikácia je implementovaná na veľmi dobrej úrovni. Čas vyhodnocovania vzoriek je dostatočne krátky. V porovnaní s panelovým testom je vyhodnotenie niekoľkonásobne kratšie, pri zachovaní, resp. zvýšení úspešnosti klasifikácie. Za dobrý krok považujem implementáciu viacerých klasifikačných modelov a možné spojenie ich predikcií pri vyhodnocovaní. Zobrazenie výsledkov v tabuľke a následné zobrazenie detailu vzorky je prehľadné. Dobrým nápadom je zapracovanie aj zobrazenie miery istoty modelu. Implementovanie natrénovania nových modelov hodnotím pozitívne, pretože otvára priestor vyhodnocovaniu iných vzoriek, ako sú napríklad vína. Myšlienku vizualizácie vzoriek považujem za správny krok, ktorý môže pomôcť používateľovi aplikácie predstaviť si ako približne vyzerajú namerané dáta.

Potenciálne rozšírenie

V budúcnosti by bolo možné aplikáciu rozšíriť o:

- Nastaviteľný koeficient pri vizualizácií obrázkov.
- Informáciu o obrázku, jeho parametre, triedu oleja.

- Výber hyperparametrov, ako napríklad počet stromov v Random Foreste.
- Filtrovanie vzoriek a modelov.

A.2 Hodnotenie práce tímu

Tím pracoval počas oboch semestrov samostatne. Dokázali sa vysporiadať s rôznymi problémami a rýchlo nájsť riešenie. Všetky úlohy splnili načas a v požadovanej kvalite. Komunikácia s tímom prebiehala na dobrej úrovni.

A.3 Celkový dojem

S navrhnutou aplikáciou som spokojný, je príjemná pre používateľa. Veľmi jednoducho sa s ňou pracuje. Po oboznámení sa s aplikáciou je ľahké sa v nej orientovať. Taktiež sa mi páčila zmena jazyka a zaujímavý dizajn. Rýchlosť načítavania a vyhodnocovania vzoriek je dostatočná, aby aplikácia mohla byť využiteľná aj v praxi. Pri klasifikácií sa dokázali vysporiadať aj z dátami, ktoré boli namerané dvoma rôznymi spektrometrami bez výraznej straty úspešnosti modelov. Výsledné úspešnosti jednotlivých klasifikátorov sú veľmi sľubné a niektoré ako Random Forest alebo SVM prekonali svojou presnosťou aj panelové testy. Dosiahnuté výsledky svedčia o tom, že navrhnutá aplikácia môže byť využitá aj pri iných problémoch z oblasti automatického rozpoznávania spektier.

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Dokumentácia k tímovému projektu

Moduly systému

Vedúci tímu: Mgr. Martin Sabo, PhD., Ing. Marta Šoltésová Prnová, PhD.
Číslo tímu: 05
Kontakt: mikasa.fiit@gmail.com

Obsah

1 Analýza

Táto kapitola je zameraná na analýzu problematiky. V prvej časti je vysvetlený problém klasifikácie olivových olejov, následne je opísaný prehľad existujúcich riešení. V ďalšej časti tejto kapitoly uvádzame funkcionálne a nefunkcionálne požiadavky na vyvíjaný softvér. V poslednej časti sú opísané a analyzované dáta pre trénovanie klasifikátorov poskytnuté vedúcim projektu (PO).

1.1 Úvod do problematiky

Olivové oleje sú posudzovaná primárne na základe chuti, vzhľadu a vône. Kvalita olivového oleja nezávisí len od využitia technologických postupov pri spracovaní olív. Na jeho kvalitu vplývajú aj prírodné faktory.

Významným faktorom, ktorý ovplyvňuje chuť oleja je druh pôdy. Pôda obsahuje rôzne zložky, ktorých výskyt a koncentrácia sa často líšia v závislosti od oblasti a nadmorskej výšky. Zloženie pôdy vo veľkej miere ovplyvňuje výslednú chuť a celkovú kvalitu olivového oleja.

Ďalším faktorom, ktorý ovplyvňuje kvalitu oleja je typ podnebia a s ním úzko súvisiace zrážky. Podnebie indukuje syntézu sekundárnych metabolitov, ktoré sa v rámci spektrometra analyzujú. Kvalita oleja závisí od koncentrácie látok. Napríklad, keď sa v pôde nachádza väčšia koncentrácia vápnika a podnebie je suchšie, budú sa produkovať látky s polyfenolyckými štruktúrami. Naopak, v prípade vlhkého podnebia a hornatého kraja, budú sa produkovať lipidické štruktúry.

V súčasnosti je hodnotenie kvality oleja regulované normami Európskej únie. Tieto nariadenia definujú 3 triedy olivového oleja, „extra panenský“ (EVOO), „panenský“ (VOO) a „lampantový“ (LOO).

Každá vzorka oleja je klasifikovaná do jednej z týchto tried pomocou panelového testu. Experti v danej oblasti posudzujú oleje v rôznych kategóriách a prideľujú mu určité skóre. Aplikovaním štatistickej analýzy na skóre, ktoré bolo udelené účastníkmi testu sa vyhodnotí trieda oleja. Tento proces býva časovo náročný a drahý. Z tohto dôvodu je potrebné sa v súčasnosti zamerať na modernejšiu a lacnejšiu metódu zisťovania kvality oleja. Preto sa kvalita zisťuje napríklad pomocou spektrometra.

Spektrometer je druh vedeckého prístroja, ktorý umožňuje skúmať prvkové chemické

zloženie látky či objektu na báze merania odrazeného svetla respektíve odrazenej vlnovej dĺžky svetla a jeho absorpcie alebo na základe merania vzniknutého svetla.

1.2 Existujúce riešenia

V článku [3] sa autori zaoberali falšovaním olivových olejov a rozpoznávaním pravosti oleja. Teda zisťovali, či sa jedná o čistý a kvalitný olej alebo bol zamenený napríklad za slnečnicový olej.

Autori navrhli dve metódy kontroly kvality oleja pomocou klasifikačných metód:

- 32 získaných vstupov je aplikovaných celých bez úprav priamo do klasifikátora
- 32 získaných vstupov je zredukovaných na 8 a tie sú aplikované do klasifikátora

Dataset

Surové dáta z olejov boli zozbierané a zdigitalizované pomocou zariadenia e-nose (Cyranose 320), čo je imitačný nástroj čuchu, ktorý má 32 senzorov a z nich boli vygenerované trénovacie a testovacie dáta.

Bolo získaných 12 rôznych druhov olejov (z rôznych regiónov v Balikesir v Turecku) a pre každý druh bolo zozbieraných 10 rôznych vzoriek. Testovacie a trénovacie dáta týchto dvanástich druhov sa nemiešali.

Metódy mali rôzne vstupné dáta (Obrázok 1), pretože pri druhej metóde bol použitý algoritmus PCA, ktorý redukuje dimenzionalitu (počet atribútov).

For the first method				For the second method			
S1	⋯	S32	Class	svd_1	⋯	svd_8	Class
0,001309	⋯	0,00054	(1)	0,035625215	⋯	0,046224595	(1)
0,001093	⋯	0,000626	(2)	0,039502122	⋯	−0,007833031	(2)
0,001858	⋯	0,000817	(3)	0,063373384	⋯	0,09547233	(3)
0,001043	⋯	0,000465	(4)	0,033603573	⋯	0,07006505	(4)
0,000819	⋯	0,00032	(5)	0,026595428	⋯	−0,01141755	(5)
0,000673	⋯	0,000177	(6)	0,020821755	⋯	−0,019554513	(6)
0,001115	⋯	0,000454	(7)	0,03409242	⋯	−0,146848992	(7)
0,001757	⋯	0,000852	(8)	0,050890223	⋯	−0,00503703	(8)
0,007326	⋯	0,003501	(9)	0,248933465	⋯	0,008284441	(9)
0,002671	⋯	0,001396	(10)	0,087664315	⋯	0,043067494	(10)
0,00404	⋯	0,002006	(11)	0,131178594	⋯	0,00053172	(11)
0,003203	⋯	0,001444	(12)	0,105479212	⋯	−0,034223378	(12)
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯

Obr. 1: Ukážka dát pre prvú metódu (vľavo) a pre druhú metódu (vpravo) [3]

Zoznam použitých klasifikátorov:

- *k*-NN
- Naive bayes
- Decision Tree
- Support Vector Machine (SVM)
- Artificial Neural Networks (ANN)
- Linear Discriminate Analysis (LDA)

Autori v článku [3] uviedli aj nastavenia hyperparamterov daných klasifikátorov. Tieto nastavenia môžeme vyskúšať aj v našom projekte, keďže autori pomocou nich dosiahli výborné výsledky.

Nastavenia hyperparametrov *k*-NN:

- Measure type = Numerical Measures
- Numerical measure = Kernel Euclidean Distance
- Kernel type = Anova
- Kernel gamma = 0.5
- Kernel degree = 0.5

Nastavenia hyperparametrov Decision Tree:

- Criterion = gain ratio
- Maximal depth = 20
- Confidence = 0.25
- Minimal gain = 0.1
- Minimal leaf size = 1
- Minimal size for split = 4
- Number of prepruning = 3

Nastavenia hyperparametrov *k*-NN:

- SVM type = C-SVC
- Kernel type = linear
- C = 2.0
- Cache size = 80
- Epsilon = 0.001

Výsledky

Na vyhodnocovanie úspešnosti klasifikátorov sa používala metrika accuracy.

Výsledky pre prvú metódu (Obrázok 2) sú v rozpätí 45% - 65%. Najlepší výsledok dosiahla neurónová sieť, ktorá má náskok takmer 10% nad najlepším tradičným klasifikátorom SVM. Ďalšie klasifikátory mali približne rovnakú úspešnosť.

	Train	Test
Naïve Bayes	65%	45,83%
K-NN	66,67%	48,33%
LDA	94,17%	52,50%
Decision Tree	98,33%	52,50%
ANN	71,67%	65,83%
SVM	74,17%	56,67%

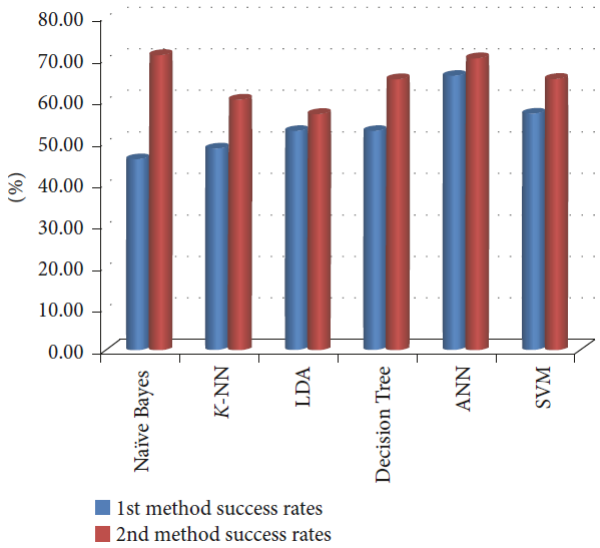
Obr. 2: Vyhodnotenie prvej metódy pomocou metriky accuracy. [3]

Výsledky pre druhú metódu (Obrázok 3) sú v rozpätí 56% - 70%. Najlepší výsledok dosiahol klasifikátor naivný bayes a hneď za ním bola neurónová sieť. Aj v tomto prípade je rozdiel medzi najlepším a najhorším klasifikátorom takmer 15%. Ďalšie klasifikátory mali približne rovnakú úspešnosť v danom rozpätí.

	Train	Test
Naïve Bayes	85,83%	70,83%
K-NN	70%	60%
LDA	75,83%	56,67%
Decision Tree	98,33%	65%
ANN	95%	70%
SVM	90%	65%

Obr. 3: Vyhodnotenie druhej metódy pomocou metriky accuracy. [3]

Pri porovnaní metód (Obrázok 4) môžeme vidieť, že druhá metóda bola oveľa úspešnejšia ako prvá. Teda aj pri zredukovanom počte dát klasifikátor dokázal správne určiť triedu. Najväčší rozdiel je vidieť pri klasikátory naivný bayes, kde sa úspešnosť zväčšila až o 25%.



Obr. 4: Porovanie výsledkov metód. [3]

V článku [2] sa autori zaoberali viacerými možnosťami zaradenia oleja pomocou klasifikačných metód. Teda vznikli tri hlavné body, ktorým sa venovali. Tie sú nasledovné:

- Zaradenie oleja do troch tried podľa kvality oleja.
- Zaradenie oleja do piatich tried podľa odrodu olív.
- Zaradenie oleja do troch tried podľa regiónu.

Dataset

Bolo zozbieraných 193 vzoriek (5 rôznych druhov olív) z rôznych oblastí Grécka. Boli získavané spektrálne dáta vo forme 2001 vlnových dĺžok v rozmedzí od 20 do 700 nm. Taktiež boli zaznamenané charakteristiky oleja ako kyslosť a oxidačné ukazovatele – hodnota (PV), rovnako aj hodnota absorpcie K232 a K270, ktoré sú tradične spojené s kvalitou oleja.

Pri testovacích a trénovacích dátach bola použitá krížová validácia, konkrétne metóda *leave one out*.

Použité klasifikátory

Zoznam použitých klasifikátorov:

- k-NN
- Linear discriminant analysis (LDA)
- Logistic regression (LR)
- Quadratic discriminant analysis (QDA)
- Artificial Neural Networks (ANN)

Výsledky

Pri vyhodnotení (Obrázok 5) bola použitá metrika accuracy a vyhodnocovali sa dva body. Variety – odroda olív a Sens – kvalita oleja. V prípade odrody olív mali klasifikátory väčšiu úspešnosť ako pri kvalite. Avšak pri hodnotení kvality sú tieto výsledky dobré, nakoľko niekedy ani ľudia kontrolujúci kvalitu nemávajú úspešnosť 100%

Method	Success	Criterion: Variety		Criterion: Sens		Software
		Training set	Leave-one-out	Training set	Leave-one-out	
LDA	Number	186	177	87	81	SPSS
	[%]	99.5	94.7	95.6	89.0	
QDA	Number	187	131	91	60	SAS
	[%]	100.0	70.1	100.0	65.9	
KNN	Number	186	185	85	80	SAS
	[%]	99.5	98.7	93.4	87.9	
LR	Number	187	–	91	–	SPSS
	[%]	100.0	–	100.0	–	
ANN	Number	–	–	91	–	JMP
	[%]	–	–	100.0	–	

Obr. 5: Porovnanie výsledkov pre odrody a kvalitu. [2]

Zhodnotenie

Vďaka analýze článkov sme zistili, že pre získanie vysokej úspešnosti postačujú aj tradičné klasifikátory. Avšak neurónová sieť sa ukázala ako vhodný klasifikátor, keďže vždy patrila k skupine najlepších klasifikátorov. Taktiež sme zistili, že náš nápad vložiť všetky získané dáta do klasifikátora bez úprav a výberu čŕt nemá veľký zmysel. Zredukované dáta mali oveľa väčšiu úspešnosť ako neupravované dáta. Takisto môžeme vyskúšať nastavenie hyperparametrov a krížovú validáciu.

1.3 Požiadavky na softwér

V tejto časti analýzy sme opísali identifikované požiadavky na softvér. V časti Funkcionálne požiadavky sme identifikovali funkcionality, ktoré má systém vykonávať. V časti Nefunkcionálne požiadavky sme definovali vlastnosti, ktorú musí aplikácia spĺňať.

Funkcionálne požiadavky

Používateľ môže do aplikácie vložiť 1 alebo viacero vzoriek oleja v jednom behu načítania súborov z priečinkov, ktoré si zvolil. Tieto vzorky musia byť vo formáte .txt. Používateľovi sa po načítaní zobrazí zoznam súborov, ktoré načítal. V prípade, že sa rozhodne nejakú vzorku nepoužiť, môže tento súbor zo zoznamu vymazať. Ak sa rozhodne nevyhodnotiť načítané vzorky, môže ich všetky naraz vymazať. Pri načítavaní nových vzoriek musí používateľ zvoliť priečinok, v ktorom sa daná vzorka v txt formáte nachádza.

Používateľ môže v aplikácií zobraziť vzorku oleja ako obrázok. Obrázok je vygenerovaný aplikáciou a reprezentuje hodnoty v spektrách namerané v čase.

Používateľ môže vyhodnotiť načítaný súbor, teda vzorku oleja, zvoleným klasifikátorom. Klasifikátor si používateľ môže zvoliť z ponuky. Ponuka musí obsahovať predtrénované klasifikátory a možnosť vybrať vlastný natrénovaný model.

Používateľ môže natrénovať nový model na nových dátach. Pri trénovaní modelu nie je možné meniť hyperparametre jednotlivých klasifikátorov.

Používateľ môže uložiť obrázok vo formáte .png do priečinka, ktorý si zvolí.

Používateľ si vie zvoliť klasifikátor, ktorý vie identifikovať tri druhy oleja, ktoré sú EVOO, VOO a LOO.

Nefunkcionálne požiadavky

Používateľ vie do aplikácie načítať naraz minimálne 100 súborov.

V prípade, že používateľ chce vložiť súbor, ktorý je v inom formáte ako .txt, tak systém nezlyhá. Ak táto situácia nastane, systém upozorní používateľa, že sa snaží vložiť súbor s nepodporovaným formátom.

Aplikáciu je možné používať na počítačoch s operačným systémom Windows 7 a vyššie.

Frekvencia zlyhania musí byť menšia ako 0.5%. To znamená, že napríklad z 1000

načítaní nových súborov sa súbor nenačíta maximálne 5-krát, resp. načítavanie súboru zlyhá v 5 prípadoch.

Aplikácia nekomunikuje alebo nespolupracuje s inými softvérmi. Systém spracováva dáta namerané pomocou spektrometra, ktoré sú uložené v súbore. To znamená, že spektrometer uloží dáta do súboru v priečinku a aplikácia tieto dáta načíta, ale priamo s ním nekomunikuje.

Používateľ musí vidieť možnosť vkladania nových dát a päť načítaných súborov na jednej obrazovke. Pri výbere klasifikátora sa všetky možnosti musia nachádzať na jednej obrazovke.

1.4 Opis dát

Od zadávateľa projektu sme dostali dve dátové sady, z dvoch rôznych spektrometrov. V oboch prípadoch sú dáta uložené v 3 priečinkoch podľa triedy oleja (EVOO, VOO, LOO), ktorá bola priradená danej vzorke. Každý priečinok obsahuje priečinky zo vzorkami, z ktorých každý obsahuje súbor vo formáte .txt, reprezentujúci 1 vzorku olivového oleja. V nich sú uložené namerané hodnoty pre daný olej. Takýto súbor je pomenovaný Spektra.POSITIVE.txt. Každá vzorka oleja obsahuje záznamy o spektrách a hodnotách, ktoré boli namerané v týchto spektrách. Počet spektier je v jednotlivých súboroch variabilný, od 9149 do 10607 spektier pre prvú dátovú sadu a od 4557 do 5993 pre druhú. Dátové sady sa líšia aj v počte nameraných hodnôt, intenzít v danom čase. Prvá sada obsahovala 448-449 hodnôt a druhá 569. V jednotlivých spektrách sa zaznamenávajú:

- hodnoty intenzita driftového poľa (ang. drift_field_intensity),
- tlak (ang. pressure),
- teplota (ang. temperature),
- dĺžka driftovej trubice (ang. drift_tube_length),
- čas analýzy (ang. analysis_time),
- čas, kedy bola nameraná intenzita,
- intenzity namerané v danom čase.

2 Návrh

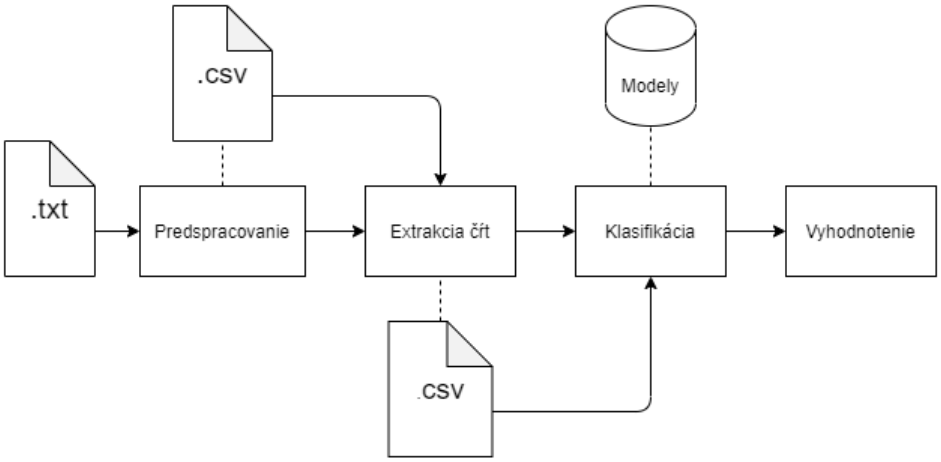
V tejto kapitole je opísané navrhované riešenie. Riešenie je rozdelené do dvoch častí.

V prvej časti Klasifikácia opisujeme navrhované moduly systému, spôsoby predspracovania dát, navrhované črty, trénovanie neurónových a tradičných klasifikátorov a spôsoby vyhodnocovania .

V druhej časti Aplikácia popisujeme návrh aplikácie, ako napríklad prípady použitia navrhovaného softvéru.

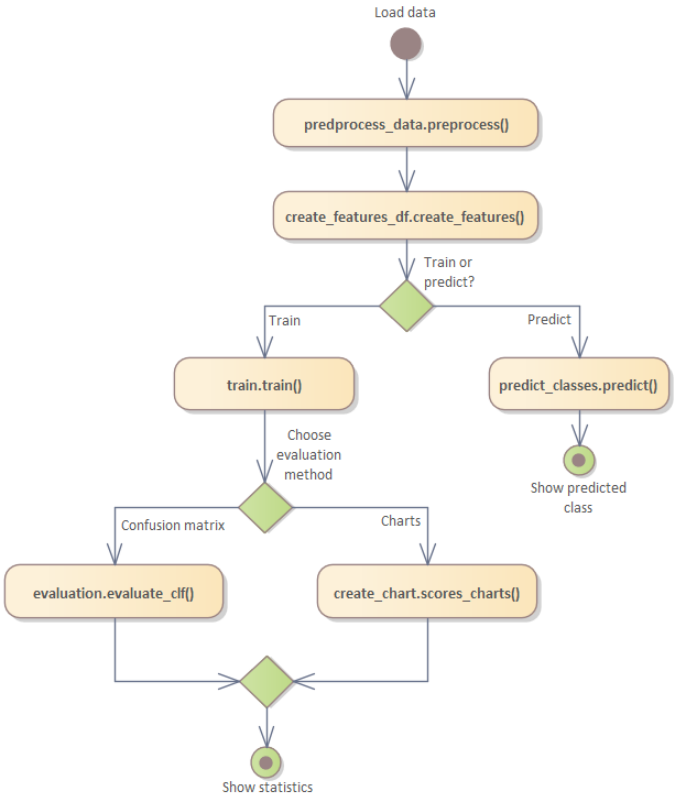
2.1 Návrh metódy

Náš systém identifikácie tried oleja je založený na 4 väčších moduloch (Predspracovanie, Extrakcia čŕt, Klasifikácia, Vyhodnotenie), ktoré sú znázornené na Obrázku 6. Vstupom sú surové dáta z .txt súboru. Tieto dáta vstupujú do modulu predspracovanie, čoho výstupom je .csv súbor. Tento súbor následne vstupuje do modulu extrakcia čŕt, kde sú vytvorené jednotlivé črty a tie sú uložené do nového súboru .csv. Novovzniknuté dáta sú popísané v časti Opis dát. Takto upravené dáta už vstupujú do klasifikácie, kde sa trénujú klasifikátory a vytvárajú modely. Následne sú modely vyhodnotené.



Obr. 6: Návrh metódy pre klasifikáciu kvality oleja.

Obrázok 7 reprezentuje priebeh predspracovania, trénovania, predikcie a vyhodnotenia dát.



Obr. 7: Návrh predspracovania, trénovania, predikcie a vyhodnotenia.

2.2 Predspracovanie dát

V procese predspracovania dát sa bude konvertovať .txt súbor do .csv súboru, pre lepšiu manipuláciu s dátami. Vo formáte .csv bude spektrum reprezentované ako jeden riadok s príslušnými hodnotami.

V tejto časti budeme identifikovať chýbajúce hodnoty alebo chybné hodnoty a upravovať nesúlad v dátach. Taktiež sa vykoná normalizácia dát. Normalizácia transformuje dáta do normálneho rozdelenia, aby sme zmenšili rozptyl dát. Napríklad všetky hodnoty budú z rozsahu 0 – 1. Takéto rozdelenie požadujú napríklad klasifikátory *k*-NN a SVM, keďže počítajú so vzdialenosťou a podobnosťou medzi dátami.

2.2.1 Extrakcia čŕt

Po predspracovaní dát a dôkladnej analýze vzťahov medzi črtami v zozbieraných dátach, vyberieme črty, ktoré budú pre nás dôležité a pomôžu nám v identifikácii tried. Najprv budeme vyberať črty ako priemer, maximum, minimum, medián, štandardná odchýlka pre všetky stĺpce a neskôr sa doimplementujú aj zložitejšie črty.

Pri extrakcii sa najskôr vytvorí samostatná tabuľka so základnými údajmi a do nej sa budú postupne pridávať nové črty. Teda po vytvorení základnej tabuľky a pridaní čŕt sa nebudú musieť počítať všetky črty od začiatku, ak budeme chcieť pridať novú črtu.

2.2.2 Klasifikácia

Klasifikácia sa bude vykonávať pomocou piatich tradičných klasifikačných algoritmov a pomocou neurónových sietí. Tieto klasifikátory boli vybrané aj na základe podobných prác, kde dosiahli výborné výsledky v danej oblasti. Tieto práce sú popísané v časti Existujúce riešenia. Vybrané klasifikačné algoritmy sú:

- k-NN
- SVM
- Decision Tree
- Random Forest
- Naive Bayes
- Artificial Neural Networks

Pred trénovaním budeme deliť dáta na testovacie a trénovacie v určitých pomeroch a z nich vyberieme jeden najefektívnejší. Taktiež skúsime aplikovať krížovú validáciu.

Pri trénovaní klasifikátorov budeme ladiť hyperparametre a snažiť sa dosiahnuť čo najlepšie výsledky pomocou rôznych nastavení.

2.2.3 Vyhodnotenie

Pre každý klasifikátor vyhodnotíme ich skóre úspešnosti pomocou metrík na vyhodnocovanie úspešnosti kvalifikátorov. Pre každý klasifikátor vytvoríme obrázok matice zámen a taktiež vytvoríme jeden graf, na ktorom budú porovnané výsledky všetkých klasifikátorov.

V prípade, ak sú triedy nevyvážené, samotná správnosť (angl. *accuracy*) môže byť zavádzajúca. Preto sa používa viacero metrík, ktoré rôzne opisujú úspešnosť klasifikátora. Z matice zámen (Tabuľka 1), možno odvodiť viacero metrík (Tabuľka 2).

Tabuľka 1: Matica zámen pre dve triedy. [1]

	Predikovaná pozitívna trieda	Predikovaná negatívna trieda
Pozitívna trieda	TP	FN
Negatívna trieda	FP	TN

- TP – pravdivé pozitíva (angl. *true positive*), predstavujú počet správne označených vzoriek, ktoré do triedy patria [1].
- TN – pravdivé negatíva (angl. *true negative*), predstavujú počet správne označených vzoriek, ktoré do triedy nepatria [1].
- FP – falošné pozitíva (angl. *false positive*), predstavujú počet vzoriek, ktoré boli nesprávne klasifikované do triedy, do ktorej nepatria [1].
- FN – falošné negatíva (angl. *false negative*), predstavujú počet vzoriek, ktoré boli označené, že do triedy nepatria, ale do triedy patrili [1].

Tabuľka 2: Metriky hodnotiace úspešnosť klasifikátorov odvodené z matice zámen. [1]

Metrika	**Vzorec**
Správnosť (acc)	$\frac{TP+TN}{TP+TN+FP+FN}$
Presnosť (pre)	$\frac{TP}{TP+FP}$
Úplnosť (rec)	$\frac{TP}{TP+FN}$
F1 skóre	$\frac{2.pre.rec}{pre+rec}$

V našom prípade máme tri triedy (EVOO, LOO, VOO), čiže sa jedná o klasifikáciu do viacerých tried. V tomto prípade použijeme metriky pre vyhodnotenie úspešnosti klasifikátora pri viacerých triedach (počítame mikro- a makro-priemer metrík)

$$B_{macro}\frac{1}{q}\sum_{\lambda}^{q} B(TP_\lambda, FP_\lambda, TN_\lambda, FN_\lambda) \tag{1}$$

$$B_{micro} = B(\sum_{\lambda}^{q} TP_{\lambda} \sum_{\lambda}^{q} FP_{\lambda} \sum_{\lambda}^{q} TN_{\lambda} \sum_{\lambda}^{q} FN_{\lambda}) \tag{2}$$

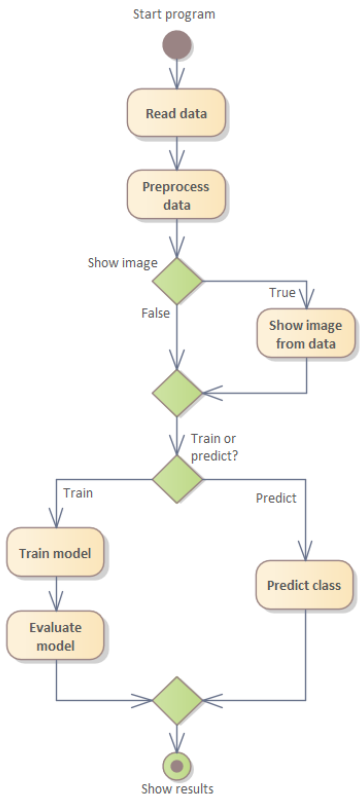
- B – binárna vyhodnocovacia metrika ako napr. presnosť, úplnosť, F_1 skóre
- q – počet tried
- λ – trieda

B_{macro} vypočítava metriku zvlášť pre každú triedu a potom určí ich priemer. B_{micro} agreguje výsledky všetkých tried na výpočet priemernej metriky [4].

2.3 Aplikácia

V tejto kapitole si zadefinujeme prípady použitia, ktoré boli vytvorené po konzultácii s PO, ale rovnako tak s celým tímom.

Obrázok 8 reprezentuje návrh fungovania aplikácie.



Obr. 8: Návrh fungovania aplikácie.

2.3.1 Prípady použitia

UC01 Načítanie dát

Opis

V tomto prípade použitia je potrebné vložiť dáta do aplikácie a tie budú postupne spracované do požadovaného .csv formátu. Tento prípad použitia sa môže opakovať, nakoľko vieme pridať viacero súborov.

Predpoklady
Používateľ má nainštalovanú a spustenú aplikáciu.

Dôsledky
Dáta sú pripravené na klasifikáciu a ďalšie operácie s nimi spojené.

Hlavný scenár

1. Používateľ v aplikácií zvolí možnosť načítať dáta.
2. Aplikácia zobrazí okno výberu súboru.
3. Používateľ vyberie 1 alebo viacero adresárov, ktoré obsahujú .txt súbor, ktorý chce načítať do aplikácia. Zároveň používateľ vyberie triedy oleja, ktorú načítava (EVOO, VOO, LOO, Nonlabeled).
4. Aplikácia načíta .txt súbory z vybraných adresárov a odošle ich na spracovanie. Aplikácia zablokuje počas načítavania dát všetky tlačidlá. Zároveň informuje používateľa o tom, že načítavanie dát prebieha.
5. Aplikácia odblokuje ďalšie tlačidlá. [Vymazanie vzorky]
6. Prípad použitia končí.

UC02 Klasifikácia dát

Opis
Prípad použitia je zameraný a vyhodnotenie predspracovaných dát.

Predpoklady
Aplikácia má predspracované dáta pomocou UC01 Načítanie dát a máme natrénované modely.

Dôsledky
Používateľ vie do akej triedy patria jeho vzorky.

Hlavný scenár

1. Používateľ zvolí možnosť klasifikácie dát.
2. Aplikácia zobrazí zoznam natrénovaných klasifikátorov a zoznam vzoriek, ktoré nemajú označenie.
3. Používateľ vyberie 1 alebo viacero klasifikátorov. Taktiež vyberie 1 alebo viaceré vzorky, ktoré sa majú vyhodnotiť.
4. Aplikácia vyhodnotí vybrané vzorky dát zvolenými klasifikátormi.
5. Aplikácia zobrazí klikateľný zoznam vyhodnotených vzoriek z dát, ktorý obsahuje informáciu o názve vzorky a predikovanej triede. [Zobrazenie detailu vzorky] [Uloženie výsledku vyhodnotenia]
6. Prípad použitia končí.

UC03 Zobrazenie dát z obrázka

Opis
Prípad použitia má za úlohu transformovať textové dáta do podoby obrázkov.

Predpoklady
Aplikácia má predspracované dáta pomocou UC01 Načítanie dát.

Dôsledky
Vizualizácia dát pomocou obrázkov.

Hlavný scenár

1. Používateľ zvolí možnosť vizualizácie obrázka buď pri detaile vyhodnotenej vzorky alebo priamo vo vizualizácií vzoriek.
2. Aplikácia odošle súbor na transformáciu na obrázok a zobrazí správu o tom, že spracovanie dát prebieha.
3. Po transformácii aplikácia zobrazí klikateľný zoznam obrázkov z dát.
4. Aplikácia zobrazí detail vzorky. [Zobrazenie detailu obrázka][Uloženie obrázka]
5. Prípad použitia končí.

UC04 Trénovanie modelu

Opis
Prípad použitia má na starosť vytvoriť vlastné modely z predspracovaných dát.

Predpoklady
Aplikácia má predspracované dáta pomocou UC01 Načítanie dát.

Dôsledky
Používateľ má vytvorený vlastný model, na ktorom bude môcť klasifikovať nové údaje.

Hlavný scenár

1. Používateľ zvolí možnosť natrénovania vlastného modelu.
2. Aplikácia zobrazí zoznam modelov, ktoré vieme natrénovať a zoznam vzoriek, ktoré majú označenie (EVOO, VOO, LOO).
3. Používateľ zvolí modely, ktoré chce trénovať a vzorky, na ktorých chce natrénovať vybrané modely.
4. Aplikácia natrénuje modely a doplní ich do tabuľky modelov.
5. Prípad použitia končí.

UC05 Pridanie modelu

Opis
Prípad použitia má na starosť pridanie existujúceho modelu do aplikácie.

Predpoklady
Používateľ má na vlastnom úložisku natrénovaný model, ktorý vie predikovať olivové oleje.

Dôsledky
Aplikácia pridá existujúci model k ostatným modelom. Vložené dáta v aplikácií je možné vyhodnotiť týmto modelom.

Hlavný scenár

1. Používateľ zvolí možnosť pridania existujúceho modelu.
2. Aplikácia otvorí okno s možnosťou výberu adresára, kde je súbor uložený.
3. Používateľ zvolí model, ktorý chce vložiť.
4. Aplikácia načíta existujúci súbor a doplní ho do tabuľky modelov.
5. Prípad použitia končí.

UC06 Ukladanie súboru

Opis
Prípad použitia má na starosť ukladanie súboru.

Predpoklady
Máme na výber súbor, ktorý vieme uložiť.

Dôsledky
Súbor je uložený na lokálnom počítači.

Body rozšírenia
Uloženie modelu.
Uloženie obrázka.
Uloženie detailu.
Uloženie výsledku vyhodnotenia.
Hlavný scenár

1. Používateľ zvolí možnosť uloženia súboru.
2. Aplikácia otvorí okno s možnosťou výberu adresára, kde bude súbor uložený.
3. Používateľ vyberie adresár, kam majú byť uložené dáta a zvolí názov uloženého súboru.
4. Aplikácia súbor uloží.
5. Prípad použitia končí.

UC07 Vymazanie súborov

Opis
Používateľ viem vymazať súbory, ktoré pridal do aplikácie.

Predpoklady
Máme načítané súbory pomocou UC01 Načítanie dát, ktoré chceme odstrániť.

Dôsledky
Súbory nie sú súčasťou aplikácie.

Body rozšírenia
Vymazanie vzorky.

Hlavný scenár

1. Používateľ vyberie súbory, ktoré chce vymazať. Následne zvolí možnosť vymazať súbor.
2. Aplikácia sa opýta na potvrdenie rozhodnutia.
3. Používateľ potvrdí vymazanie.
4. Aplikácia vymaže súbory.
5. Prípad použitia končí.

UC08 Zobrazenie detailu

Opis
Prípad použitia zahŕňa zobrazenie detailu po vybratí položky zo zoznamu.

Predpoklady
Zobrazená zoznam prvkov, pre ktoré vieme zobraziť detail.

Dôsledky
Zobrazenie detailu prvku.

Body rozšírenia

Zobrazenie detailu vzorky.

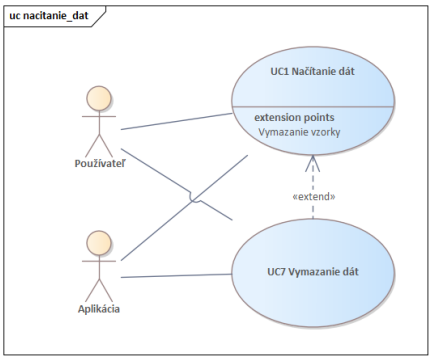
Zobrazenie detailu obrázka.

Hlavný scenár

1. Používateľ zvolí možnosť zobraziť detail.
2. Aplikácia zobrazí detail. [Uloženie detailu]
3. Prípad použitia končí.

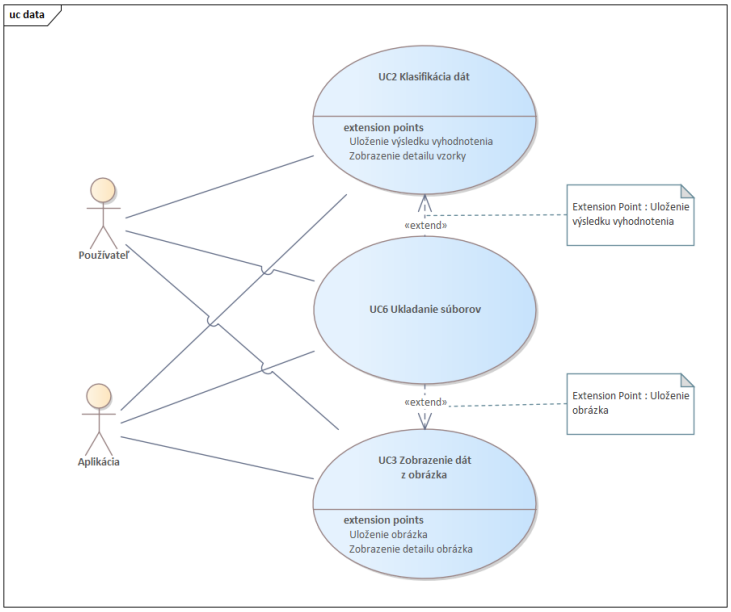
2.3.2 Závislosti medzi prípadmi použitia

Obrázok 9 reprezentuje načítavanie údajov do aplikácie. Súbory, ktoré sú načítané, môžu byť z aplikácie aj odstránené.



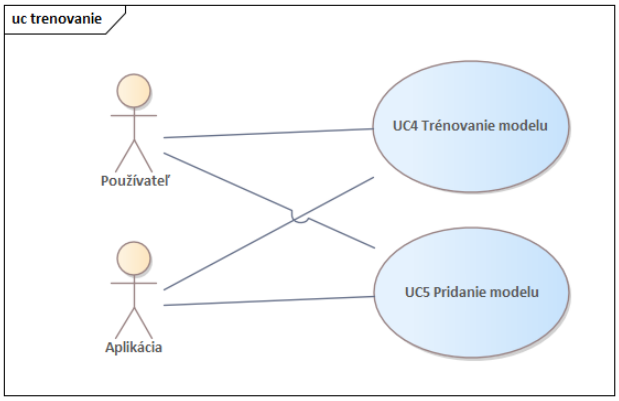
Obr. 9: Diagram, ktorý ukazuje závislosť medzi prípadmi použitia pri načítavaní dát do aplikácie.

Obrázok 10 reprezentuje klasifikovanie dát do jednotlivých tried. Taktiež je načítané dáta možné zobrazovať vo forme obrázka, ktorý je možné uložiť.



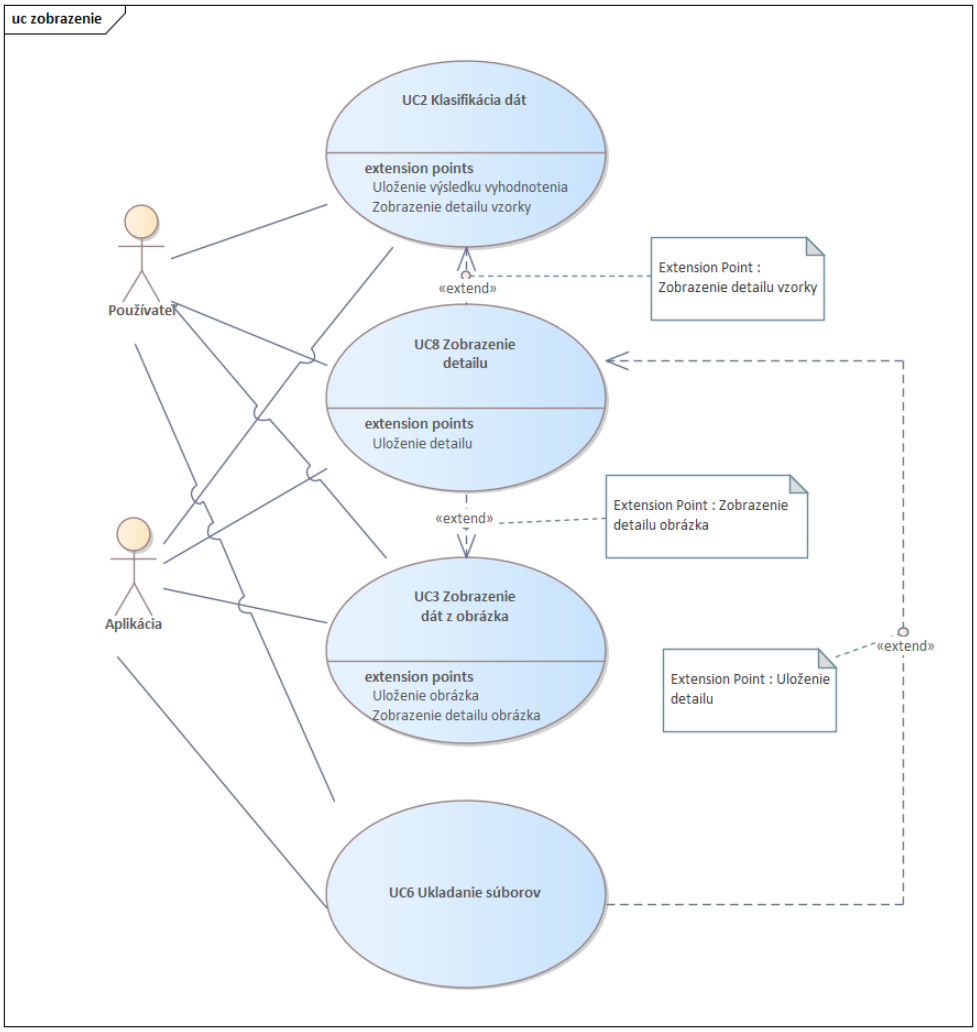
Obr. 10: Diagram, ktorý ukazuje závislosť medzi prípadmi použitia, ktoré zodpovedajú za klasifikovanie dát a zobrazovanie obrázka z dát.

Obrázok 11 reprezentuje možnosť vytvorenia nových modelov, ktoré vzniknú natrénovaním modelov na nových dátach.



Obr. 11: Diagram, ktorý ukazuje závislosť medzi prípadmi použitia pri trénovaní klasifikačných metód.

Obrázok 12 ukazuje, čo všetko je možné zobrazovať pomocou aplikácie.



Obr. 12: Diagram, ktorý ukazuje závislosť medzi prípadmi použitia pri ukladaní dát v aplikácii.

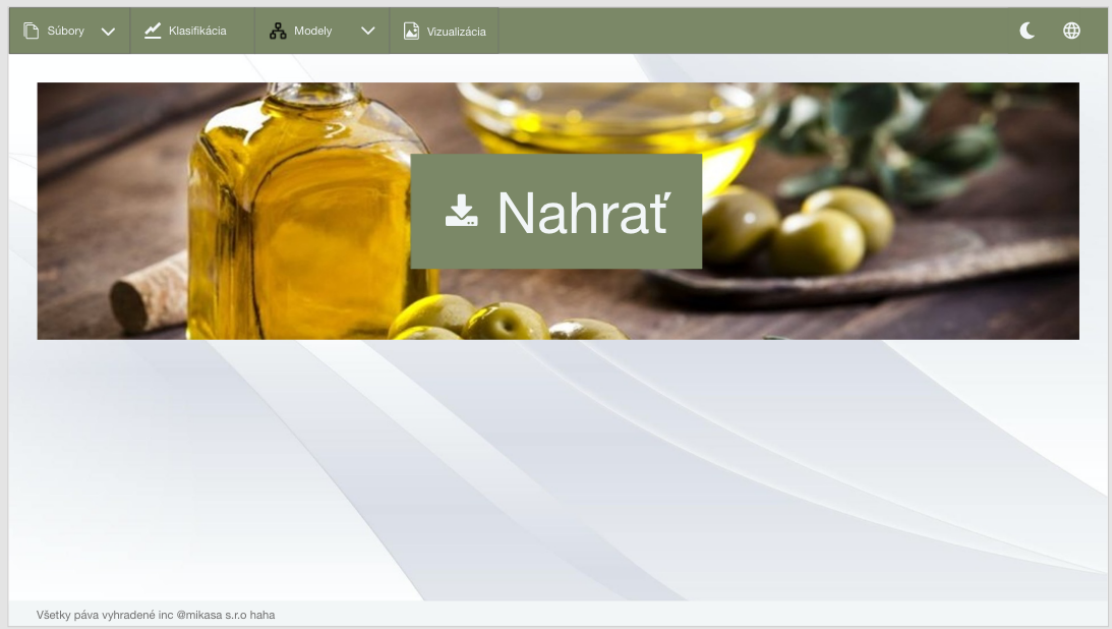
2.3.3 Návrh grafického rozhrania aplikácie

Úvodná obrazovka

V úvodnej obrazovke budú všetky možnosti zablokované, okrem možnosti nahratia dát. Všetky operácie v aplikácii sa vykonávajú nad dátami, resp. bez dát nie je možné vykonať žiadnu operáciu. Návrh úvodnej obrazovky je znázornený na obrázku 13.

Po zvolení možnosti nahratia dát sa zobrazí vyskakovacie okno, v ktorom používateľ zvolí druh dát, ktoré ide nahrať. Môže sa jednať o nespracované dáta, spracované dáta, vyhodnotené dáta a modely. Po nahratí dát sa odblokujú jednotlivé operácie nad dátami

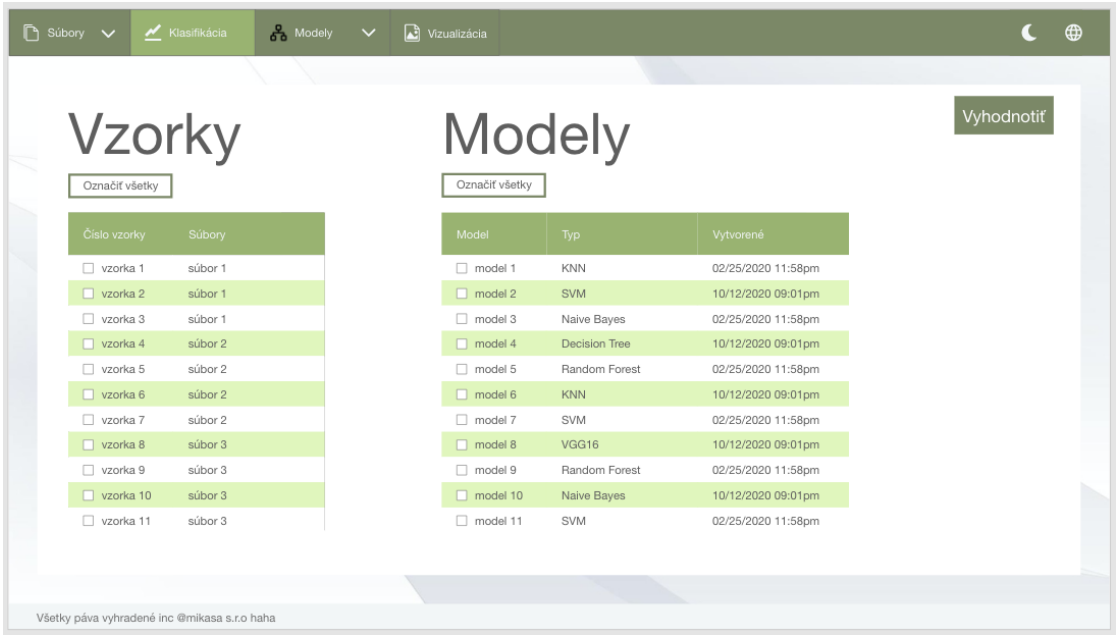
v hornom menu.



Obr. 13: Úvodná obrazovka

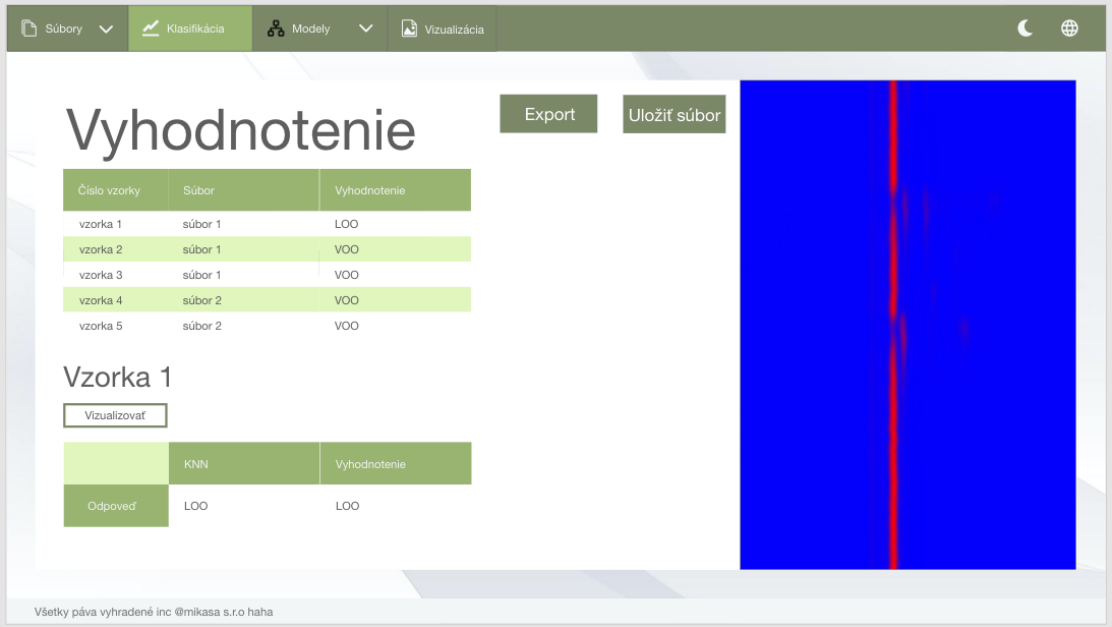
Obrazovky Klasifikácie

Zvolením možnosti klasifikácie sa obrazovke zobrazí výber spracovaných vzoriek, ktoré chceme klasifikovať a zoznam modelov, ktoré chceme na klasifikáciu použiť. Návrh klasifikačnej obrazovky je na obrázku 14.



Obr. 14: Úvodná obrazovka klasifikácie modelov

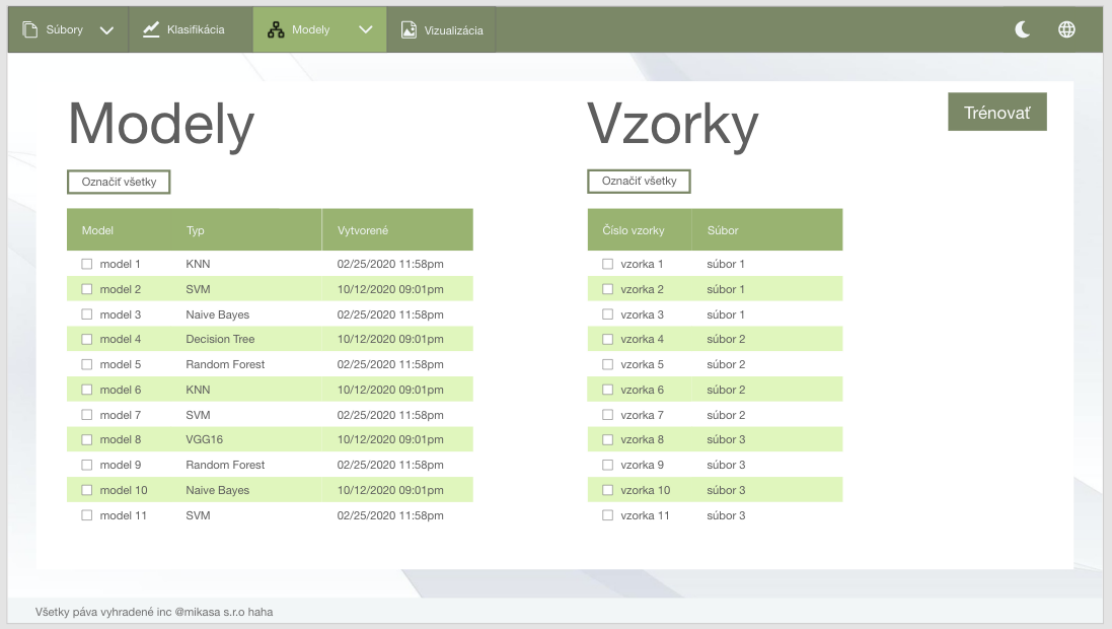
Stlačením tlačidla ”Vyhodnotiť”sa spustí vyhodnocovanie vzoriek a začnú sa klasifikovať modely. Po ukončení procesu sa zobrazí tabuľka výsledkov. Detailnú správu o výsledkov pre jednotlivé vzorky je možné stiahnuť vo formáte pdf. Taktiež je možné vidieť jednotlivé detaily vzoriek zvolením jednej z vyhodnotených vzoriek. Ako možno vidieť na obrázku 15 každú vzorku je možné vizualizovať a obrázok vizualizácie uložiť do pracovného počítača.



Obr. 15: Úvodná obrazovka klasifikácie modelov

Obrazovky pre správu modelov

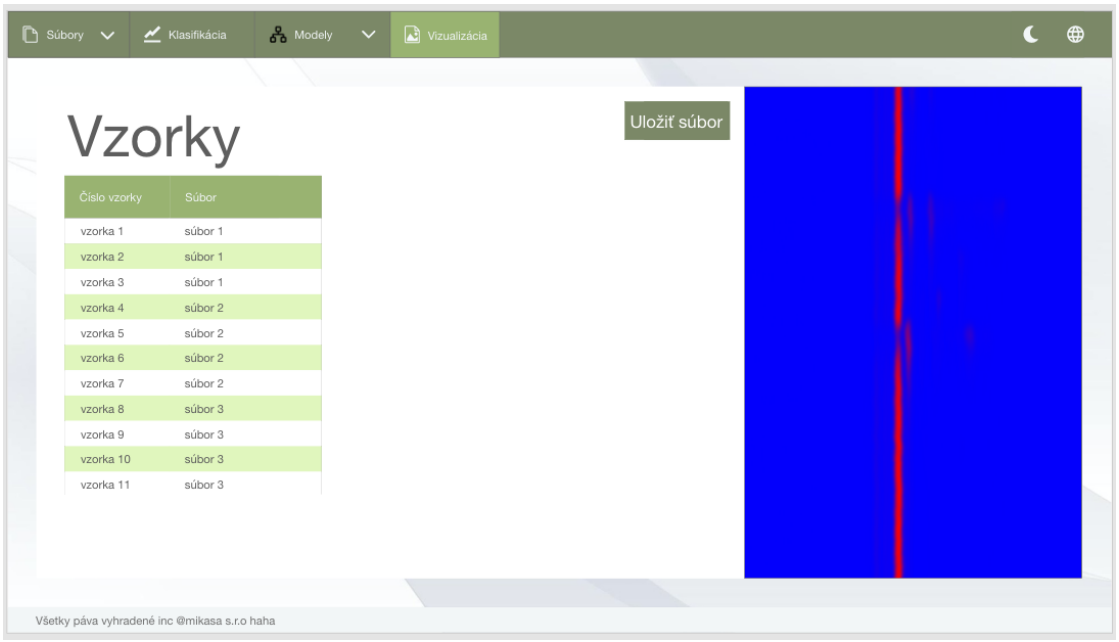
V správe modelov je možné trénovať nové modely. Na rozdiel od prvej obrazovky klasifikácie si v modeloch používateľ najprv zvolí modely a až potom vyberie vzorky, nad ktorými chce vykonať operáciu. Obrazovka pre trénovanie modelu sa nachádza na obrázku 16.



Obr. 16: Výsledná obrazovka klasifikácie modelov

Obrazovka vizualizácie vzoriek

Pre samostatnú vizualizáciu vzorky slúži obrazovka na obrázku 17. Používateľ si zo zoznamu vzoriek vyberie vzorku, ktorú chce zobraziť a vzorka sa mu vizualizuje na pravej strane obrazovky.



Obr. 17: Obrazovka pre vizualizáciu jednotlivých vzoriek

3 Implementácia

V tejto kapitole sú opísané postupy implementácie predspracovania, jednotlivých klasifikátorov a vyhodnocovania úspešnosti nami navrhnutých klasifikátorov. Následne je opísaná implementácia aplikácie. Nakoniec sú v tejto kapitole opísané aj všetky súbory, ktoré sme vytvorili pri spracovávaní dát.

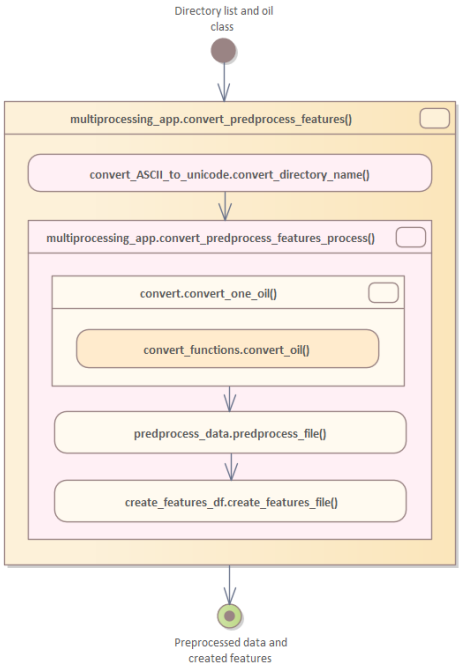
3.1 Opis implementácie klasifikácie

V tejto kapitole sa nachádzajú diagramy, ktoré zobrazujú jednotlivé kroky implementácie. Funkcie z diagramov sú bližšie popísané v spomínanej dokumentácii zverejnenej v prílohe D.

3.1.1 Spracovanie dát

Obrázok 19 zobrazuje priebeh volania funkcií pri spracovaní dát. Na vstup vie používateľ zadať cestu k adresáru so vzorkou alebo k viacerým adresárom zároveň, pre spracovanie väčšieho množstva vzoriek. Nakoľko názov samotnej vzorky je u všetkých rovnaký, vzorky sa rozlišujú pomocou názvu adresára. Aby sme dokázali tieto názvy efektívne používať, pomocou funkcie convert_directory_name zmeníme kódovanie na Unicode.

Naša aplikácia podporuje spracovanie viacerých nezávislých vzoriek naraz, pomocou spracovania v samostatných procesoch. Počet procesov si vie používateľ nastaviť v konfiguračnom súbore, odporúčané je použiť paralelných 8 procesov. V každom procese sa pre danú vzorku vykonajú tri kroky: konvertovanie, predspracovanie a výpočet čŕt.



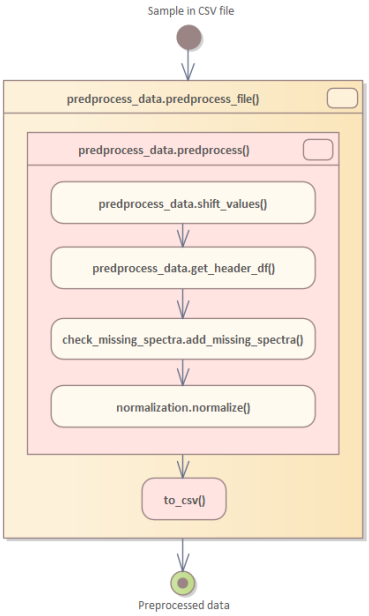
Obr. 18: Ukážka priebehu spracovania dát.

Konvertovanie dát

Konvertovanie vzoriek prebieha ako prvé pri spracovaní, kedy je potrebné zmeniť formát z textového súboru na tabuľku, neskôr ukladanú ako csv súbor. Aby bol textový súbor správne spracovaný musí mať špecifický formát, predpísaný výstupom z iónového spektrometra.

Predspracovanie dát

Takto konvertovaný súbor vstupuje do predspracovania. Ako prvé sa vykonáva posun hodnôt vo funkcií shift_values, pretože vstupné dáta môžu mať rozličný počet hodnôt na spektrum. Na základe stĺpca s maximálnou hodnotou (čas, kedy bola nameraná najväčšia hodnota) sa vyberie istý počet hodnôt pred a za týmto stĺpcom, pričom počet je vybraný na základne dátovej analýzy. V prípade, že vzorka obsahuje menší počet hodnôt, hodnoty budú doplnené nulovými hodnotami. Po tomto kroku majú všetky vzorky rovnaký počet stĺpcov a je aj pridaná hlavička do tabuľky.



Obr. 19: Ukážka priebehu predspracovania dát.

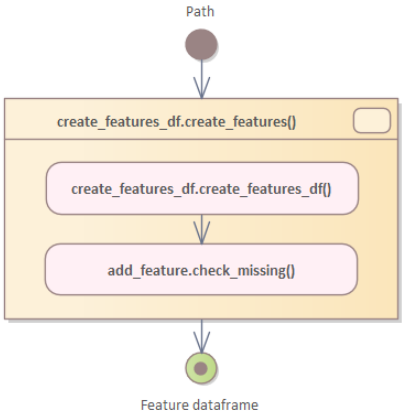
V ďalšom kroku funkcia add_missing_spectra zabezpečuje doplnenie chýbajúcich spektier na základe analysis_time, ktorý určuje čas, kedy bola vzorka nameraná od začiatku merania. Ak niektoré spektrá neboli zaznamenané, doplnia sa priemerov okolitých spektier.

Posledným krokom spracovania je normalizácia dát, kedy prepočítavame namerané hodnoty do rozsahu 0-1 pomocou minimálnej a maximálnej hodnoty. Nakoniec tieto dáta ukladáme vo formáte csv a môžu byť použité pre výpočet čŕt a pre vizualizáciu.

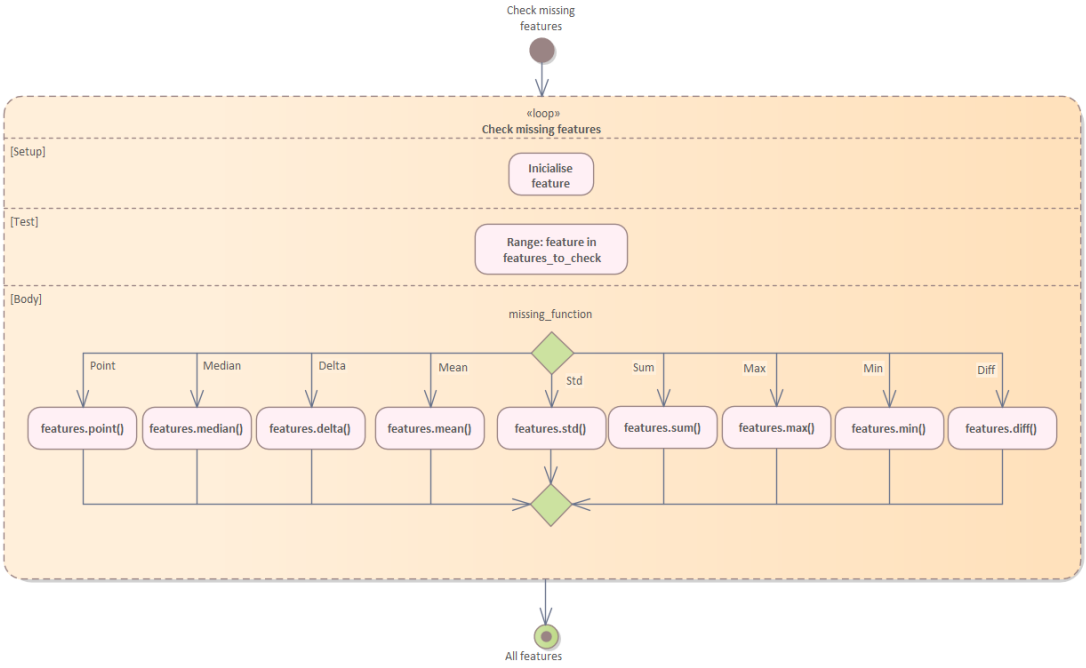
3.1.2 Črty

Výpočet čŕt

V rovnakom procese ako konvertovanie a predspracovanie prebieha aj výpočet čŕt znázornený na obrázku 20. Proces prebieha vytvorením tabuľky s názvom vzorky a triedy oleja. Následne sa procesom znázorneným na obrázku 21 dopočítavajú jednotlivé štatistické črty: median, delta, mean, std, sum, max, min. Tiež používame naše vlastné črty point a diff.



Obr. 20: Ukážka priebehu vytvárania čŕt.

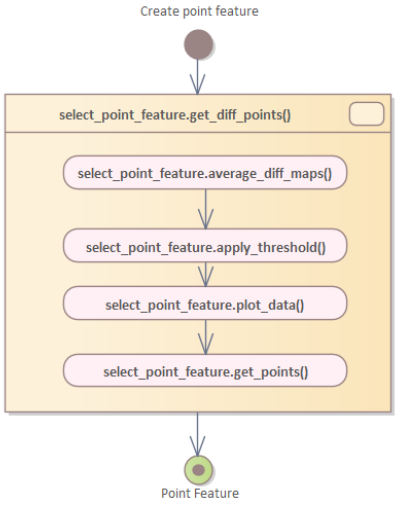


Obr. 21: Ukážka vytvorenia chýbajúcich čŕt.

Črty point (bod) a diff (skratka od diferenciálnej mapy) sme vymysleli na základne analýzy zaujímavých oblastí, kde sme pozorovali výkyvy medzi hodnotami vo viacerých olejoch. Tieto zaujímavé oblasti sme získali rozdielov dvojíc olejov rôznej triedy. Tieto rozdiely sme spriemerovali a vynulovali hodnoty menšie ako určená hranica. Získané

oblasti, ktoré neboli vynulované, tvoria črtu point, ktorá zaznamenáva maximum a priemer hodnôt v danej oblasti. Tento proces je znázornený na obrázku 22 .

Črta diff predstavuje rozdiel od priemerného oleja danej triedy, takzvanej šablóny. Vypočítaný je ako rozdiel hodnôt na pozíciach medi olejom a šablónou, prenásobený maskou, aby sme vynechali oblasti s veľkou variabilitou nezávislou od triedy. Zachované hodnoty sa následne sčítavajú, čo predstavuje celkovú vzdialenosť od danej triedy.

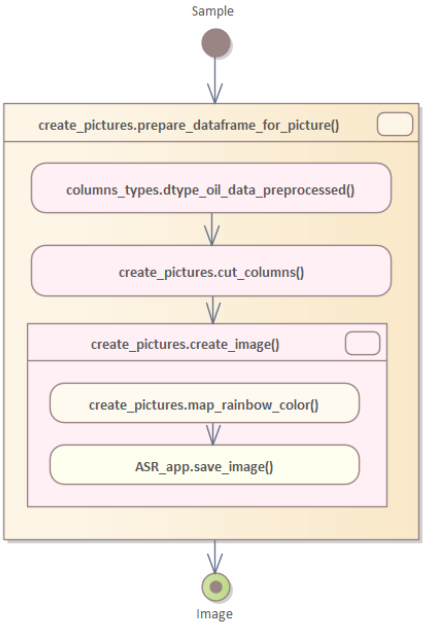


Obr. 22: Ukážka vytvorenia črty Point.

Jednotlivé črty boli vybrané na základe výberu čŕt. Uvažovali sme dva spôsoby výberu. Prvý spôsob analyzuje korelácie medzi triedou a jednotlivými črtami. Druhý spôsob, ktorého črty aj sú aktuálne používané, je výber prvých 500 čŕt, ktoré klasifikátor Random forest vyhodnotil ako najdôležitejšie pre svoje rozhodnutie. Klasifikátor bol zvolený na základe najvyššej úspešnosti, vyhodnotenej na všetkých vytvorených črtách.

Vizualizácia

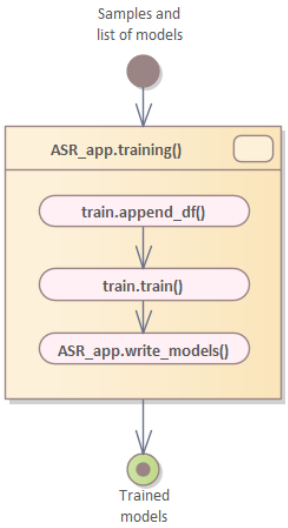
Jednou z funkcií aplikácie je vizualizácia poskytnutých dát po spracovaní. Dokážeme generovať obrázky z hodnôt, kde hodnoty na x osi sú namerané hodnoty intenzít a na y osi sú jednotlivé spektrá. Modrá farba predstavuje najnižšie, zelená stredné a červená najvyššie namerané hodnoty. Vytváranie obrázkov je opísané na obrázku 23.



Obr. 23: Ukážka generovania obrázkov z predspracovaných dát.

3.1.3 Trénovanie modelov

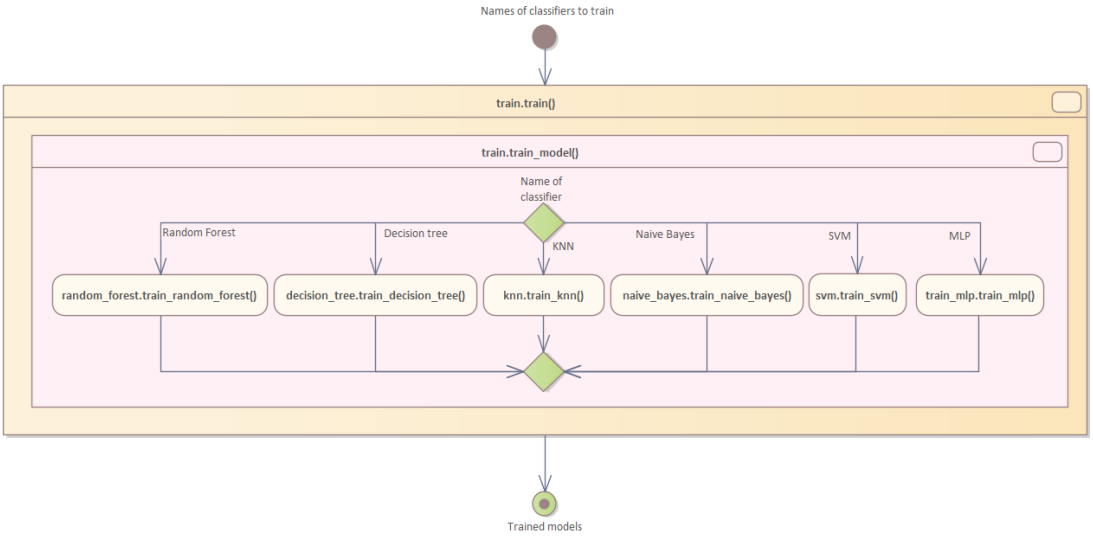
Pre vyhodnotenie triedy oleja sú používané modely. Ich trénovanie prebieha na dátovej sade vytvorenej spojením jednotlivých vzoriek. Takto natrénovaný model je uložený ako joblib súbor a môže byť opakovane použitý na vyhodnotenie. Obrázok 24 zobrazuje tento proces.



Obr. 24: Ukážka trénovania modelu.

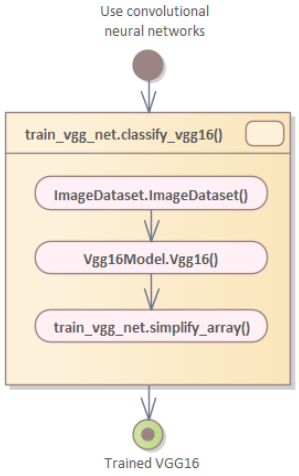
3.1.4 Trénovanie klasifikátorov

Vytváranie a trénovanie klasifikátorov prebieha pomocou vlastných metód pre každý zvolený typ klasifikátora, znázornených na obrázku 25. Podporujeme random forest, decision tree, knn, naive bayes, svm a mlp. Jednotlivé hyperparametre klasifikátorov boli vybrané na základe algoritmu gridsearch.



Obr. 25: Ukážka trénovania klasifikátorov.

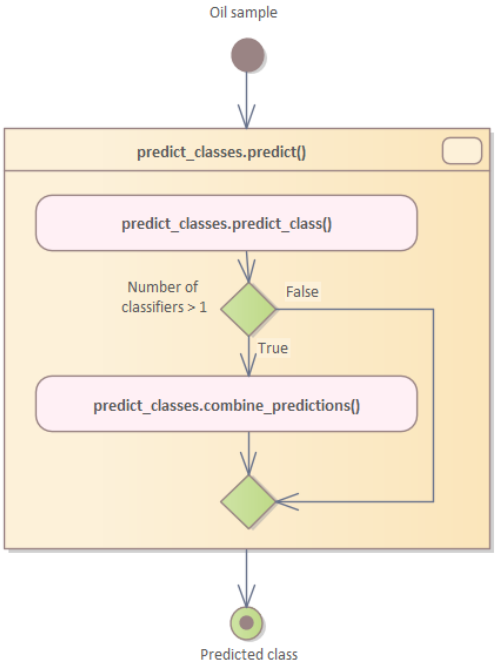
V projekte sme sa rozhodli implementovať aj konvolučnú neurónovú sieť VGG-16. Sieť má na vstupe obrázky, ktoré boli vygenerované pomocou vizualizácie dát. Následne boli tieto obrázky zmenšené na veľkosť 224x224px. Pri trénovaní sme použili Adam optimizer a CrossEntropyLoss. Obrázok 26 zobrazuje ako priebeh trénovania konvolučnej neurónovej siete.



Obr. 26: Ukážka trénovania konvolučnej neurónovej siete.

3.1.5 Predikovanie

Predikcia triedy oleja prebieha u každého klasifikátora zvlášť pomocou funkcie predict_class, ako je znázornené na obrázku 27. Pokiaľ je zvolený viac ako jeden klasifikátor na vyhodnotenie, prebieha kombinovanie predikcií na základe úspešnosti jednotlivých typov klasifikátorov, v snahe dosiahnuť čo najpresnejší výsledok.



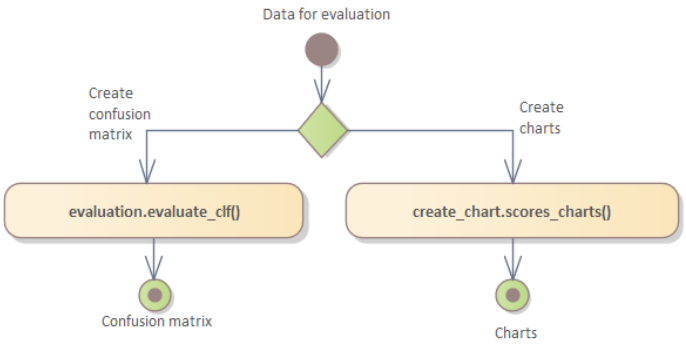
Obr. 27: Ukážka predikovania triedy.

Kombinovanie predikcií je založené na úspešnosti jednotlivých modelov. Aby sme zdôraznili tieto úspešnosti je od nich odčítaná hodnota 0.5 a sú umocnené na štvrtú. Pre každý druh oleja sa tieto hodnoty sčítajú a výsledok s najväčším číslom považujeme za celkový. Na výpočet bol použitý vzorec:

$$hodnotadruhuoleja = \sum_{m}^{modely} (presnost[typ(m)] - 0.5)^4 \tag{3}$$

3.1.6 Vyhodnocovanie

Pre vyhodnotenie našich modelov používame funkciu evaluate_clf, prípadne funkciu scores_charts na vykreslenie stĺpcového grafu s úspešnosťami. Obrázok 28 zobrazuje ako prebieha vyhodnocovanie.



Obr. 28: Ukážka vyhodnocovania úspešnosti.

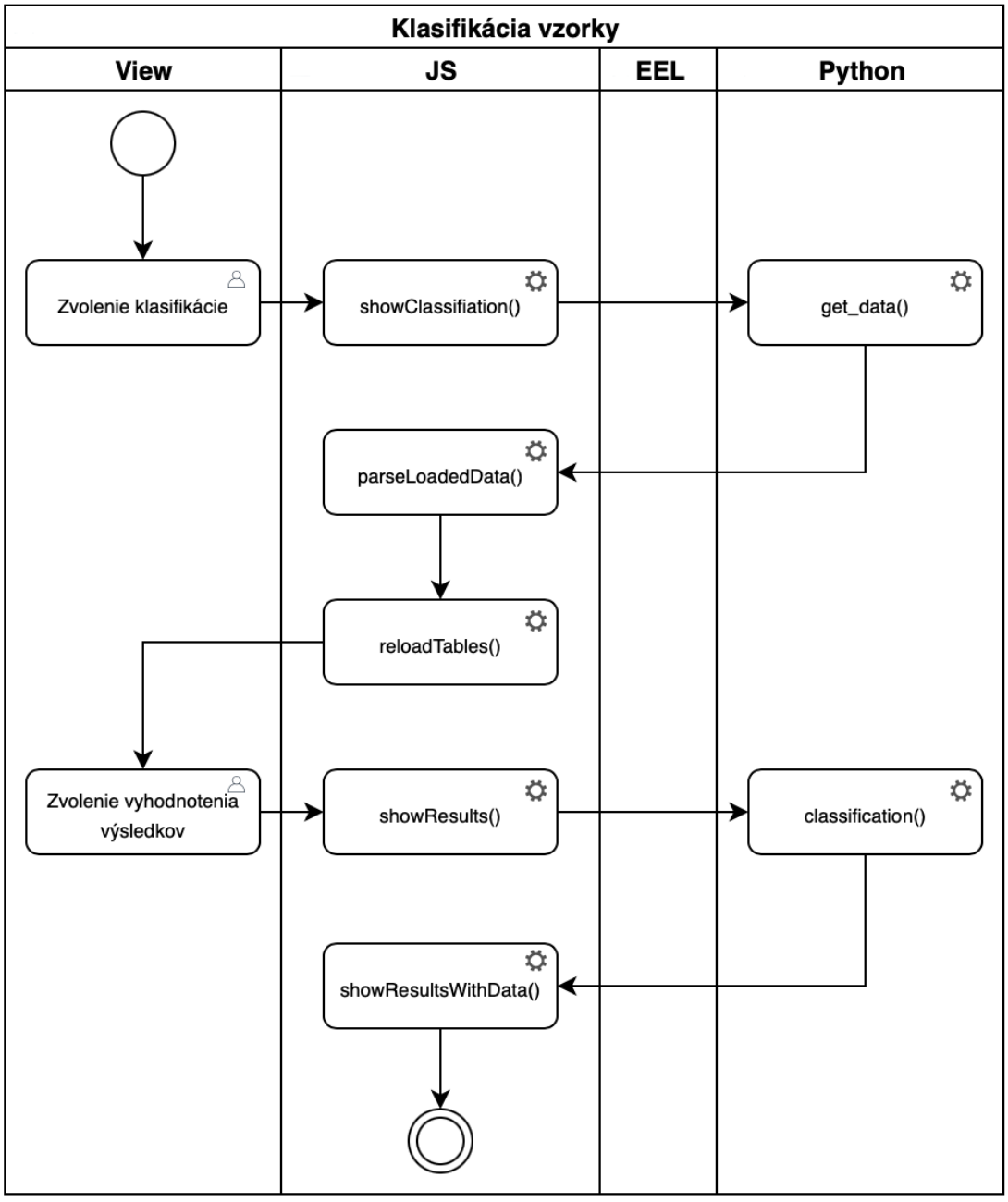
3.2 Opis implementácie aplikácie

Pre sprístupnenie možnosti klasifikácie vzoriek aj neodbornej verejnosti bola navrhnutá aplikácia. Tá umožňuje klasifikovať nahraté vzorky, vytváranie nových klasifikačných modelov a vizualizáciu vzorky.

Grafické rozhranie aplikácie je vytvorené za použitia HTML a JavaScriptu. Frontend bol napojený na aplikačnú logiku za použitia rozhrania Eel. Dáta medzi aplikačnou logikou a grafickým rozhraním sú sprostredkované cez formát JSON.

Klasifikácia vzoriek

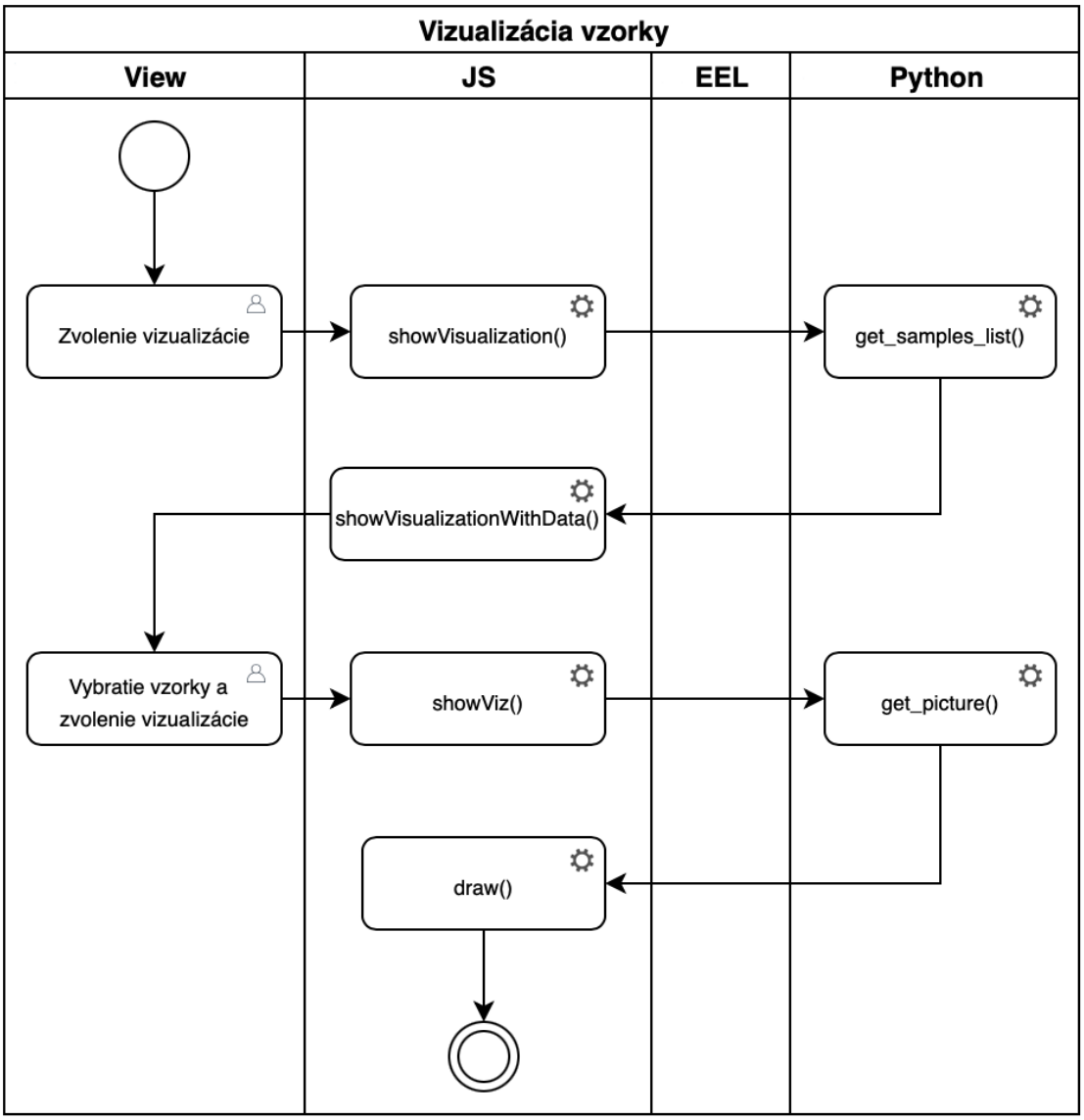
Po zvolení možnosti klasifikácie sa zobrazia nahrané vzorky a vytvorené modely, ako možno vidieť na obrázku 29. Tie sú získavané prostredníctvom aplikačnej logiky. Po zobrazení vzoriek a modelov si požívateľ vyberie vzorky, ktoré chce klasifikovať a modely, ktoré majú určiť triedu kvality zvolenej vzorky. Aplikačná logika klasifikuje vzorky. Zobrazia sa klasifikované vzorky.



Obr. 29: Diagram toku volania funkcií počas klasifikovania vzoriek.

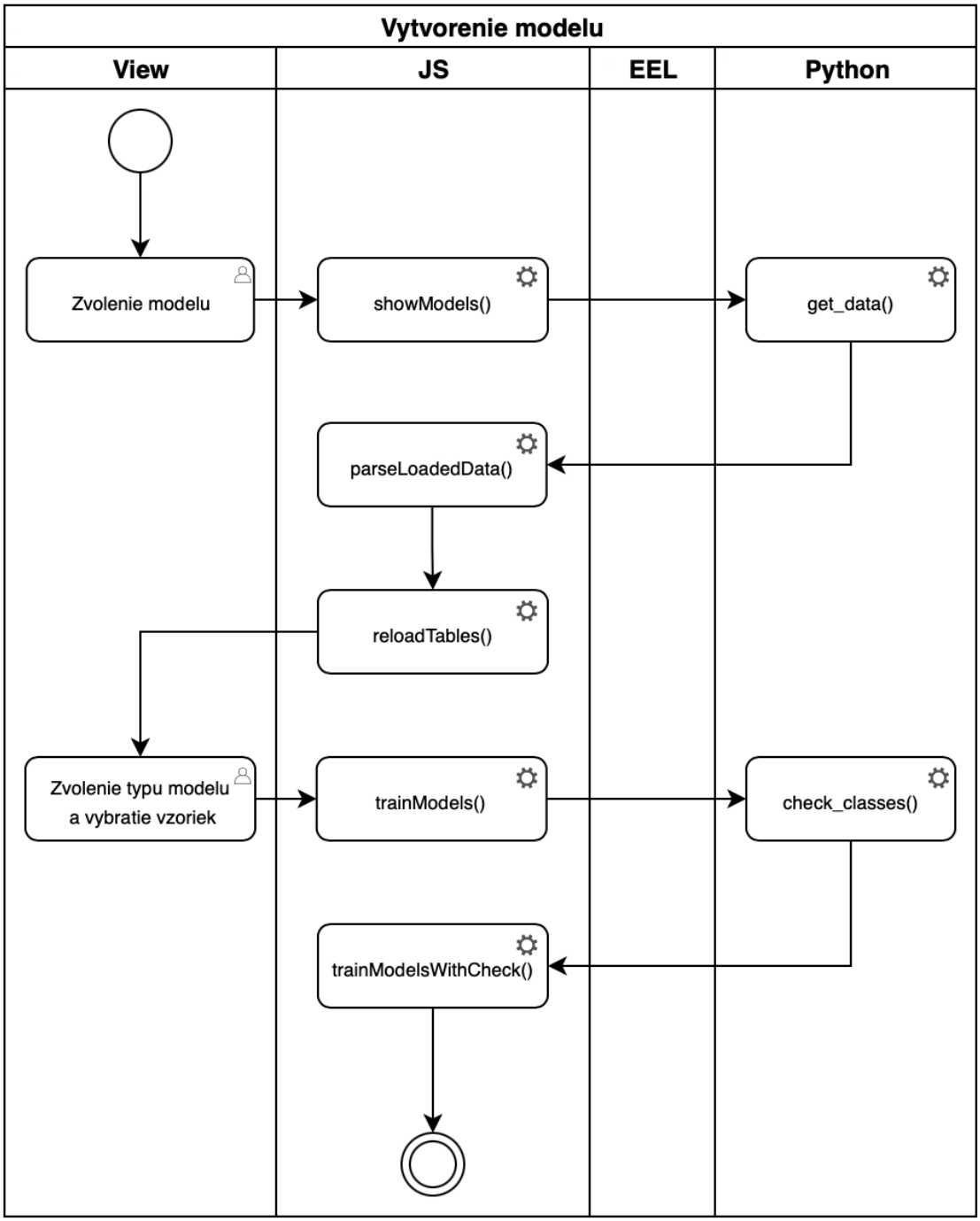
Vizualizácia vzorky

Po zvolení možnosti vizualizácie sa získajú z aplikačnej logiky všetky nahraté vzorky. Používateľ si následne vyberie vzorku, ktorú chce vizualizovať. Aplikačné rozhranie sprostredkuje dáta na vykreslenie obrázka. Grafické rozhranie následne vykreslí heatmap vzorky v canvas. Celý tok je zobrazený na obrázku 30.



Obr. 30: Diagram toku volania funkcií počas vizualizácie vzoriek.

Vytvorenie modelu



Obr. 31: Diagram toku volania funkcií počas vytvorenia modelu.

Po zvolení možnosti vytvorenia modelu sa získajú z aplikačnej logiky všetky vzorky a typy modelov, ktoré aplikácia podporuje. Následne si používateľ vyberie typy modelov, ktoré chce vytvoriť a vzorky, na ktorých sa majú modely natrénovať. Používateľ musí zvoliť aspoň 20 vzoriek a každá trieda (EVOO, VOO, LOO) musí byť zastúpená aspoň raz. Následne aplikačná logika zabezpečí kontrolu tried a trénovanie modelov. Nakoniec sa opätovne vykreslí obrazovka pre vytvorenie modelov s už vytvorenými modelmi.

3.3 Opis dát

Spracované dáta

Súbory, ktoré sme dostali na začiatku sme spracovali do 3 súborov – dataEVOO.csv, dataVOO.csv, dataLOO.csv. Tieto súbory obsahujú záznamy o všetkých olejoch z jednotlivých tried. Súbory obsahujú stĺpce class, oil_id, spectrum, drift_field_intensity, pressure, temperature, drift_tube_length, analysis_time, v_0, v_151 – v_448:

- stĺpec class definuje triedu oleja,
- oil_id reprezentuje vzorku oleja,
- spectrum identifikuje číslo spektra, ktorého namerané hodnoty sú uložené v riadku,
- drift_field_intensity, pressure, temperature, drift_tube_length, analysis_time sú stĺpce, v ktorých sú zaznamenané hodnoty namerané pre jednotlivé spektrá olejov,
- v_0, v_151 – v_448 sú vybrané stĺpce z pôvodných 449 stĺpcov, v ktorých sú uložené namerané intenzity v spektrách.

	class	oil_id	spectrum	drift_field_intensity	pressure	temperature	drift_tube_length	analysis_time	v_0	v_151	...
0	EVOO	200427_190222	1	558.521	994.471	110.683	11.51	0.13	0.0050	0.0046	...
1	EVOO	200427_190222	2	558.521	994.471	110.683	11.51	0.26	0.0050	0.0044	...
2	EVOO	200427_190222	3	558.521	994.471	110.605	11.51	0.39	0.0050	0.0044	...
3	EVOO	200427_190222	4	558.521	994.471	110.526	11.51	0.52	0.0049	0.0045	...
4	EVOO	200427_190222	5	558.521	994.471	110.605	11.51	0.65	0.0049	0.0045	...

Obr. 32: Ukážka súboru dataEVOO.csv.

Riadok v každom súbore obsahuje namerané dáta z jedného spektra, preto sme pridali do súboru stĺpec oil_id, ktorý identifikuje o aký olej ide, pretože 1 vzorka oleja obsahuje

dáta o viacerých spektrách. V súboroch sa nenachádzajú chýbajúce spektrá, teda každá vzorka oleja obsahuje všetky spektrá.

Súbory dataEVOO.csv, dataVOO.csv a dataLOO.csv sme spojili do 1 súboru a vytvorili sme dataAll.csv, ktorý má rovnaké stĺpce ako predošlé súbory.

V rámci testovania funkcionality softvéru sme vytvorili 3 testovacie súbory – data1.csv, data 3x1.csv, data3x5.csv. Všetky 3 súbory majú rovnaké stĺpce ako predošlé súbory. Súbor data1.csv obsahuje namerané hodnoty pre 1 olej, data3x1.csv obsahuje hodnoty pre 1 olej z každej triedy a data3x5.csv obsahuje hodnoty pre 5 olejov z každej triedy.

Dáta používané v aplikácií sa ukladajú po vzorkách, aby sa urýchlil výber a umožnilo sa spracovanie viacerých vzoriek paralelne, vo vlastných procesoch.

Črty

Súbor featuresAll.csv obsahuje niekoľko čŕt, ktoré sme vytvorili počas prvých 2 šprintov. Črty boli vypočítané zo stĺpcov, ktoré sme identifikovali ako podstatné pre klasifikátor. Z daných stĺpcov sme postupne vypočítali:

- priemer pre každý olej,
- priemernú hodnotu pre každý stĺpec v danej vzorke oleja,
- maximum pre každý stĺpec v danej vzorke oleja,
- minimum pre každý stĺpec v danej vzorke oleja,
- medián pre každý stĺpec v danej vzorke oleja,
- štandardnú odchýlku pre každý stĺpec v danej vzorke oleja,
- deltu pre každý stĺpec v danej vzorke oleja (maximum - minimum),
- súčet hodnôt pre každý stĺpec v danej vzorke oleja.

V rámci čŕt sme vytvárali aj diferenčné mapy, ktoré reprezentujú rozdiel medzi 2 porovnávanými vzorkami oleja. V jednotlivých olejoch porovnávame rozdiel hodnôt vo vybraných stĺpcoch. Porovnávali sme 2 oleje z rovnakej triedy, to znamená, vytvorili sme 5 súborov pre každú triedu, ktoré porovnávajú 2 rôzne oleje. Zároveň sme porovnávali aj oleje z 2 rôznych tried (EVOO-VOO, EVOO-LOO, LOO-VOO). Opäť sme vytvorili 5 súborov pre porovnanie z každej dvojice.

Aby sme mohli vyskúšať konvolučné neurónové siete, vytvorili sme obrázky z vybraných stĺpcov. Tieto obrázky sú vo formáte .png a boli vytvorené pre všetky oleje z každej triedy. Obrázky majú veľkosť počet_spektrierX449 pixlov. Hodnoty sú reprezentované odtieňmi modrej a červene farby, podľa jej veľkosti. Modrá reprezentuje nízke hodnoty, červená vysoké, na prechod medzi farbami sme využili lineárnu interpoláciu.

4 Testovanie

V tejto kapitole sú stručne opísané testy. Na validáciu navrhnutého softvéru používame akceptačné testy a pre zamedzenie chýb v programe sú používané jednotkové testy (ďalej ako unit testy).

4.1 Unit testy

Hlavným účelom unit testov je odhaľovanie chýb vo vytvorených funkciách. Pri testovaní unit testami sa porovnáva očakávaný výsledok funkcie s vráteným výsledkom.

V rámci klasifikácie sú týmto spôsobom testované funkcie na spracovanie dát. Cieľom je odhaliť chyby a spraviť systém robustným voči neočakávaným vstupom.

4.2 UX testy

V prílohe UX test je celý report z UX testov aplikácie. Report obsahuje vybavenie a parametre vybavenia, zoznam účastníkov, zoznam úloh, výsledky, vyjadrenie účastníkov na aplikáciu a na úroveň zložitosti zadaní.

Taktiež sú tam spísané postrehy počas testovania, čo sa darilo účastníkom a čo sa naopak nedarilo a čo to spôsobilo. Report obsahuje zoznam otázok viažúcich sa k aplikácii a jej funkcionalite.

Na základe tohto UX testu sme zapracovali niektoré dôležité pripomienky ako napríklad vylepšenie systému vkladania vzoriek, rozmiestnenie tlačidiel, vizualizáciu vzoriek, pridali sme úvodnú domovskú stránku a upravili hornú lištu.

4.3 Akceptačné testy

Aby sme aplikáciu mohli odovzdať product owner-ovi, potrebuje mu ukázať, že sme vytvorili požadovanú aplikáciu. Na túto prezentáciu použijeme akceptačné testy, ktoré sme si rozdelili do 2 kategórií.

Prvá kategória popisuje najjednoduchšie testy, ktoré je možné vykonať pozorovaním.

- Aplikácia nesmie obsahovať gramatické chyby.
- Dizajn obrazoviek vyzerá ako v návrhu, upravené o dotazy, ktoré vyšli z UX testov.

V druhej kategórií sú zadefinované testy, ktoré overujú funkčnosť aplikácie.

- Používateľ vie vložiť adresár, v ktorom je súbor vo formáte .txt.
- Používateľ nevie vložiť iný adresár ako adresár, ktorý obsahuje súbor .txt.
- Používateľ dokáže načítať 100 súborov do aplikácie.
- Používateľ vie vymazať načítaný súbor.
- Používateľ vie zobraziť obrázok z načítaného súboru.
- Používateľ vie klasifikovať vzorku oleja so zvoleným klasifikátorom.
- Používateľ vie natrénovať nový model.
- Používateľ vie uložiť obrázok.
- Používateľ vie načítať vlastný model na trénovanie.
- Používateľ vie vyhodnotiť triedu EVOO, LOO a VOO.
- Používateľ vie uložiť vyhodnotené dáta.

Kroky a očakávaný výstup akceptačných testov

Vloženie súboru .txt.

1. Otvorte si aplikáciu.
2. Vložte adresár so súborom Spectra.POSITIVE.txt do aplikácie.
3. **Očakávaný výstup** – Súbor bude vložený do aplikácie.

Vloženie súboru s inou príponou ako .txt.

1. Otvorte si aplikáciu.
2. Vložte adresár so súborom Spectra.POSITIVE.pdf do aplikácie.
3. **Očakávaný výstup** – Súbor Spectra.POSITIVE.pdf nebude vložený do aplikácie. Aplikácia upozorní používateľa, že sa snaží vložiť súbor v nepodporovanom formáte.

Používateľ dokáže načítať 100 súborov do aplikácie.

1. Otvorte si aplikáciu.

2. Vložte všetky adresáre so súbormi z adresára test100 do aplikácie.
3. **Očakávaný výstup** – Súbory z adresárov budú načítané do aplikácie.

Používateľ vie vymazať načítaný súbor.

1. Otvorte si aplikáciu.
2. Vložte adresár s názvom ”200427_174846”so súborom Spectra.POSITIVE.txt do aplikácie.
3. Vymažte pridanú vzorku s názvom ”200427_174846”.
4. **Očakávaný výstup** – Vzorka ”200427_174846”bude odstránená z aplikácie.

Používateľ vie zobraziť obrázok z načítaného súboru.

1. Otvorte si aplikáciu.
2. Vložte adresár so súborom Spectra.POSITIVE.txt do aplikácie.
3. Zobrazte obrázok z dát načítaného súboru.
4. **Očakávaný výstup** – Zobrazí sa obrázok, ktorý reprezentuje načítané dáta.

Používateľ vie klasifikovať vzorku oleja so zvoleným klasifikátorom.

1. Otvorte si aplikáciu.
2. Vložte adresár so súborom Spectra.POSITIVE.txt do aplikácie.
3. Vyberte možnosť klasifikácie dát.
4. Vyberte si 1 z ponúkaných klasifikátorov.
5. Vyhodnoťte triedu oleja.
6. **Očakávaný výstup** – Používateľovi sa zobrazí predpokladaná trieda oleja aj s percentuálnou pravdepodobnosťou.

Používateľ vie natrénovať nový model.

1. Otvorte si aplikáciu.
2. Vložte všetky adresáre so súbormi z adresára test100 do aplikácie.
3. Vyberte možnosť natrénovania modelu.
4. Vyberte si 1 z ponúkaných klasifikátorov.

5. Natrénujte model.
6. **Očakávaný výstup** – Používateľovi sa zobrazí natrénovaný model v časti klasifikácia v tabuľke modely.

Používateľ vie uložiť obrázok.

1. Otvorte si aplikáciu.
2. Vložte adresár so súborom Spectra.POSITIVE.txt do aplikácie.
3. Zobrazte obrázok z dát načítaného súboru.
4. Uložte načítaný obrázok ako test_obrazok.png do adresára test.
5. **Očakávaný výstup** – Uložený súbor test_obrazok.png v adresári test.

Vyhodnotenie triedy olivového oleja a uloženie výsledkov.

1. Otvorte si aplikáciu.
2. Vložte všetky adresáre so súbormi z adresára test100 do aplikácie.
3. Vyberte možnosť klasifikácie dát.
4. Vyberte si 1 z ponúkaných klasifikátorov.
5. Vyhodnoťte triedy oleja.
6. Uložte vyhodnotené vzorky do adresára test.
7. **Očakávaný výstup** – Dáta budú v súboroch uložené vo forme „názov_adresára_vzorky.txt“.

4.4 Report z testovania produktu ASR product ownerom

Vzhľadom na to, že testovanie s product ownerom nemohlo byť zorganizované osobne, pre testovanie sme zvolili online stretnutie cez platformu Google Meet. Testovanie aplikácie prebehlo vo forme otestovania funkcionality aplikácie a prejdenia akceptačných testov product ownerom.

Priebeh stretnutia

- Krátky úvod pre product ownera čo ideme vykonávať
- Testovanie produktu
- Diskusia a spätná väzba k produktu

Úvod

Na začiatku sme predstavili zámer testovania a vykonania akceptačných testov. Zaslali sme .exe súbor aplikácie a .zip súbor so vzorkami olejov. Následne bola aplikácia bez problémov nainštalovaná a prešlo sa k testovaniu.

Testovanie aplikácie

V rámci testovania aplikácie bola otestovaná všetka funkcionalita:

- Pridávanie vzoriek
- Vymazanie vzoriek
- Klasifikácia (vyhodnotenie) vzoriek
- Uloženie výsledku klasifikácie
- Vizualizácia vzoriek
- Uloženie vizualizácie vzorky
- Natrénovanie vlastného modelu
- Vloženie existujúceho modelu do aplikácie
- Prepnutie jazyka
- Prepnutie svetlého/tmavého módu aplikácie

Pri testovaní aplikácie sme postupne prešli všetky akceptačné testy spomenuté v časti Akceptačné testy. Všetky testy prebehli podľa očakávaní, až na test uloženia obrázku, pri ktorom najskôr aplikácia nezobrazila výber pre miesto uloženia obrázku. Pri opakovaní tohto testu sa táto chyba už neskôr nezopakovala.

Diskusia a spätná väzba k produktu

Produkt bol ohodnotení veľmi pozitívne či už z pohľadu použiteľnosti alebo z pohľadu dosiahnutých výsledkov pri klasifikácii vzoriek. Product owner nemal žiadne výhrady ani iné pripomienky k aplikácii.

Taktiež sme preberali potenciál využitia našej aplikácie aj v iných oblastiach, než je klasifikácia olivového oleja. Boli spomenuté napríklad klasifikácia tried kvality vína, testovanie prítomnosti coronavírusu z dychu.

Zhrnutie

Poskytli sme .exe súbor aplikácie a vzorky olejov. Funkcionalita aplikácie bola otestovaná product ownerom a aplikácia prešla akceptačnými testami. Product owner nemal žiadne výhrady a bol spokojný s finálnym výsledkom našej práce. Taktiež boli navrhnuté ďalšie oblasti, kde by sme náš produkt vedeli ďalej využiť.

Literatúra

[1] S. Agarwal. Data mining: Data mining concepts and techniques. In *2013 International Conference on Machine Intelligence and Research Advancement*, pages 203–207, 2013.

[2] Dasa Kruzlcova, Jan Mocak, Evagelos Katsoyannos, and Ernst Lankmayr. Classification and characterization of olive oils by uv-vis absorption spectrometry and sensorial analysis. *Journal of food and nutrition research*, 47:181–188, 01 2008.

[3] Emre Ordukaya and Bekir Karlik. Quality control of olive oils using machine learning and electronic nose. *Journal of Food Quality*, 2017:1–7, 10 2017.

[4] S. Wang and W. Shi. *Data Mining and Knowledge Discovery*, pages 49–58. 01 2011.

A UX test

UX Test Report

ASR application

Dátum vypracovania reportu: 10.3.2021

1. Zhrnutie

Testovanie ukázalo niekoľko nedostatkov našej aplikácie, avšak každý z testerov splnil všetky zadané úlohy. Vo väčšine prípadov testeri neprekročili maximálny odhadovaný čas na dokončenie úlohy. Testeri našu aplikáciu označili ako jednoduchú na používanie. Celkovo mali z aplikácie dobrý pocit a našu aplikáciu by odporučili aj ďalej.

Testeri mali väčšinou podobné problémy. Najťažšie pre každého bolo pridávanie vzoriek, kde bolo potrebné poradiť ako táto časť funguje, keďže každý bol zmätený prečo sa im stále zobrazuje obrazovka vkladania súborov.

Druhým väčším problémov bola vizualizácia vzoriek, keďže iba jeden tester si všimol, že obrázok sa dá scrollovať. Všetci testeri očakávali viac informácií pri obrázku ohľadom vizualizovanej vzorky.

2. Opis postupu testovania

Údaje o používateľoch.

Našu aplikáciu by mali vedieť používať ľudia, ktorí pracujú v oblasti dátovej analýzy. To však nevylučuje, že aplikáciu budú používať aj ľudia, ktorí sa dátovej analýze nevenujú.

Testeri

Testeri	TP1	TP2	TP3	TP4	TP5
Dátum testovania	5.3.2021	5.3.2021	5.3.2021	8.3.2021	10.3.2021
Čas testovania	11:20	19:15	23:30	9:00	14:00
Jazyk testovania	slovenský	slovenský	anglický	slovenský	slovenský
Všeobecné informácie					
Pohlavie	žena	muž	muž	žena	muž
Vek	22	25	27	-	-
Vzdelanie v odbore	informatika	informatika	informatika	biotechnológia	informatika
Zrakové choroby					
Okuliare	nie	nie	nie	nie	nie
Farbosleposť	nie	nie	nie	nie	nie
Znalosti spojené s témou aplikácie					

Znalosti dátovej analýzy	áno	áno	áno	áno	áno

Tabuľka 1: Prehľad základných informácií o testovaných používateľoch.

Testovacie prostredie

Test bol uskutočnení online formou, kde každý používateľ mal u seba stiahnutú našu aplikáciu a sadu vzoriek.

Vybavenie	
Zariadenie na ktorom bol test vykonávaný	Laptop, PC
Operačný systém	Windows 10
Pripojenie k internetu	Áno (kvôli pripojeniu počas testovania)
Webový prehliadač	Google Chrome, kvôli pripojeniu počas testovania

Tabuľka 2: Prostredie použité pri testovaní.

Úlohy

Max. predpokladaný čas bol počítaný ako trojnásobok nášho bežného času zvládnutia úlohy.

Číslo úlohy	**Opis**	**Predpoklady**	**Kritériá dokončenia úlohy**	**Max. predpokladaný čas**
1	Motivačná úloha, ktorú by mal zvládnuť každý používateľ. V aplikácii používateľ zmení dark mode aplikácie na light mode a zmení jazyk aplikácie z angličtiny na slovenčinu.	Aplikácia je zapnutá na úvodnej obrazovke.	1. Používateľ klikne na ikonu pre zmenu dark módu na light mód. 2. Používateľ klikne na ikonu jazyka a vyberie možnosť slovenčina.	30 sekúnd
2	Používateľ nahrá jednu vzorku, ktorú má vopred pripravenú v zadanom adresári. Po pridaní vzorky ju klasifikuje pomocou klasifikátora random forest.	Aplikácia je zapnutá na úvodnej obrazovke.	1. Používateľ klikne na možnosť vložiť súbory. 2. Vyberie zadaný adresár so vzorkou. 3. Vyberie možnosť klasifikácia. 4. V časti klasifikácia zaškrtne vloženú vzorku a klasifikátor random forest. 5. Vyberie možnosť klasifikovať.	1 minúta 30 sekúnd
3	Používateľ nahrá ďalšie tri vybrané vzorky, ktoré má vopred pripravené v zadanom adresári.	Aplikácia je zapnutá na obrazovke klasifikácia.	1. Používateľ vyberie možnosť nahrať súbory. 2. Vyberie zadaný adresár so vzorkou. (Túto časť zopakuje ešte 2x)	1 minúta 10 sekúnd

	Po pridaní vzoriek ich klasifikuje pomocou všetkých existujúcich klasifikátorov.		3. Vyberie možnosť klasifikácia. 4. V časti klasifikácia zaškrtne pole pre výber všetkých vzoriek a pole pre výber všetkých klasifikátorov. 5. Vyberie možnosť klasifikovať.	
4	Po vyhodnotení vzoriek si používateľ pozrie detail vzorky 200427_212926 a vizualizuje ju. Následne si si stiahne export vzoriek a obrázok vzorky.	Aplikácia je zapnutá na obrazovke klasifikácia.	1. Používateľ klikne na vzorku 200427_212926. 2. Používateľ vyberie možnosť vizualizovať. 3. Používateľ vyberie možnosť exportovať obrázok.	1 minúta 20 sekúnd
5	Používateľ vloží klasifikačný model decision_tree_UX-2021-02-08 21-19-20, ktorý je uložený v zipku Data. Dotrénuje model knn a model decision_tree_UX na všetkých vložených vzorkách.	Aplikácia je zapnutá na obrazovke klasifikácia.	1. Používateľ vyberie možnosť pridať modely. 2. Vloží model decision_tree_UX-2021-02-08 21-19-20 3. Vyberie možnosť dotrénovať. 4. Zaklikne modely knn a decision_tree_UX-2021-02-08 21-19-20. 5. Zaklikne všetky vzorky. 6. Vyberie možnosť dotrénovať.	2 minúty

Tabuľka 3: Interný zoznam úloh.

Zadanie úlohy:

Predstavte si, že máte doma olivový olej a chcete zistiť jeho kvalitu. Tento olej ste zaslali na test a z prístroja Vám prišli zdigitalizované vzorky. Pomocou týchto vzoriek teraz chcete zistiť kvalitu Vášho oleja.

Číslo úlohy	Opis úlohy
1	V aplikácii zmeňte light mode aplikácie na darkmode a zmeňte jazyk aplikácie z angličtiny na slovenčinu.
2	Nahrajte adresár 200427_212926, kde je uložený súbor so vzorkou Spectra.POSITIVE. Tento adresár máte v stiahnutom zipku Data. Následne vloženú vzorku klasifikujte a pri klasifikácii použite model random_forest.
3	Pridajte ďalšie tri adresáre. Adresár 200428_105943, 200428_121317 a 200428_155447, kde sa nachádzajú ďalšie vzorky. Tieto adresáre nájdete v stiahnutom zipku Data. Následne klasifikujte všetky vzorky, ktoré boli doteraz vložené a pri klasifikácii zvoľte všetky vytvorené modely.
4	Po vyhodnotení vzoriek si pozrite detail vzorky 200427_212926 a vizualizujte ju.

	Následne si stiahnite export vzoriek a obrázok vzorky.
5	Vložte klasifikačný model decision_tree_UX-2021-02-08 21-19-20, ktorý je uložený v zipku Data. Dotrénujte model knn a model decision_tree_UX na všetkých vložených vzorkách.

Tabuľka 4: Úlohy zadané testerom.

3. Výsledky

V tejto časti sú výsledky z testovania a taktiež aj problémy počas vykonávania úloh.

Miera dokončenia úloh

	Task 1	Task 2	Task 3	Task 4	Task 5
TP1	1	1*	1	1	1
TP2	1	1*	1	1	1
TP3	1	1*	1	1	1
TP4	1*	1*	1	1	1
TP5	1	1*	1	1	1*
%	100	100	100	100	100

Tabuľka 5: Miera úspešnosti dokončenia úloh(0- nedokončená, 1- dokončená, 1*- dokončená s pomocou)

Čas dokončenia úloh

	Task 1	Task 2	Task 3	Task 4	Task 5	Spolu	Spolu bez tasku 2
TP1	8	120	102	40	55	325	205
TP2	7	40	78	60	56	241	201
TP3	8	60	53	35	47	203	143
TP4	36	115	28	25	35	239	124
TP5	12	72	80	47	85	296	224
Priemer	14	81	68	41	56	261	179

Tabuľka 6: Čas dokončenia úloh v sekundách.

Zachytené problémy počas vykonávania jednotlivých úloh.

TP1	Dropdown nezmizol po zakliknutí jazyka. Zmena jazyka na pozadí aplikácie nebola dostatočne viditeľná – opakované klikanie na jazyk.

TP2	Úloha prebehla bez problémov.
TP3	Úloha prebehla bez problémov.
TP4	Tester klikal na rôzne časti aplikácie, nevedel prečo sú ostatné časti neaktívne, dlho pozeral, kde je chyba a prečo sa nedá klikať. Po čase zistil, že sa dá kliknúť na ikony zmeny jazyka alebo módu.
TP5	Úloha prebehla bez problémov.

Tabuľka 7a: Problémy testerov počas vykonávania úlohy č. 1

TP1	Nahrávanie vzoriek bolo veľmi mätúce, hľadanie či je tam vôbec daný txt súbor uložený (stratenie ďalšieho času navyše). Tester nevedel, že má vkladať adresár (tester si nevšimol dole časť SELECT FOLDER). Po poradení ako vkladať vzorky ďalšia časť úlohy prebehla bez problémov.
TP2	Tester sa preklikal až do vnútra adresára, ale si všimol, že mu file explorer hovorí, že má vybrať adresár. Vybral ho z vnútra, kde sa mal nachádzať súbor SpectraPositive. Bol zmätený keď sa mu znova načítalo okno s vložením. Začal zmätočne klikať, až vypol celé okno X-kom a pokračoval ďalej v úlohe bez problémov.
TP3	Tester si nevšimol, že sa majú vkladať foldre, neskôr si všimol, že ďalej vo foldri nič nie je, no aj napriek tomu stlačil možnosť vybrať. Čo mu znova ponúklo nahratie foldru a tentokrát si už všimol, že sa vkladajú foldre. Keď videl, že sa nič nepridáva vypol nahrávanie a ďalej pokračoval v úlohe bez problémov.
TP4	Tester správne vložil adresár, avšak znova sa načítalo vloženie ďalších adresárov a tester viackrát zopakoval nahratie. Nakoniec sám zistil, že má vypnúť nahrávanie a ďalej úloha prebehla bez ďalších problémov.
TP5	Tester bol zmätený po prvom správnom vložení adresára, keďže sa mu znova zobrazila obrazovka nahratia ďalšieho. Skúšal znova vložiť adresár, avšak ostal ešte viac zmätený. Po dlhšom čase mu bolo poradené, že má iba vypnúť vkladanie. Ďalej úloha prebehla bez problémov.

Tabuľka 7b: Problémy testerov počas vykonávania úlohy č. 2

TP1	Nefungovalo tlačidlo označiť, bolo použité v prvej úlohe na označenie, čiže v ďalšej úlohe bolo všetko označené a tlačidlo sa volalo stále označiť.
TP2	Tester vložil nové adresáre už bez problémov. Následne na vzorkách nevyužil tlačidlo označiť, ale klikal to po jednom. Najskôr klikal hocikam na riadok ale zistil, že treba kliknúť do checkboxu. Pri modeloch si už všimol tlačidlo označiť.

TP3	Tester klikal po jednom všetky vzorky, následne si všimol tlačidlo select. (Dôvod bol, že si myslel, že sú to secondary buttony a majú väčšiu funkcionalitu, tak si ani neprečítal čo vykonávajú.)
TP4	Tester si taktiež nevšimol možnosti označiť všetko až po upozornení na takúto možnosť.
TP5	Tester ani pri jednom označovaní nevyužil možnosť označiť všetko.

Tabuľka 7c: Problémy testerov počas vykonávania úlohy č. 3

TP1	Tester stiahol najskôr export iba jednej vzorky, až po pripomenutí, že treba stiahnuť všetky, stiahol správny súbor.
TP2	Tester najskôr stiahol iba export jednej vzorky, potom si všimol, že je tam button exportovať všetky.
TP3	Úloha prebehla bez problémov.
TP4	Tester stiahol najskôr export iba jednej vzorky, až po pripomenutí, že treba stiahnuť všetky, stiahol správny súbor.
TP5	Úloha prebehla bez problémov.

Tabuľka 7d: Problémy testerov počas vykonávania úlohy č. 4

TP1	Vkladanie prebehlo bez problémov, tester vedel ako vložiť nový model. Nebolo úplne jasné, ktorý model bol práve vložený. Tester sa to snažil zistiť podľa času, ale čas vloženia klasifikátora sa nezobrazuje, len čas vytvorenia.
TP2	Nevedel, kde má dotrénovať model. Hľadal button späť do obrazovky trénovania. Klikal všade vo vyhodnotení. Po dlhšom hľadaní sám našiel, kde vložiť model a ostatok úlohy prebehol bez problémov.
TP3	Vkladanie prebehlo bez problémov, len nebol na obrazovke dotrénovania, ale vyhodnotenia, neskôr prešiel na správnu obrazovku.
TP4	Úloha prebehla bez problémov.
TP5	Tester sa snažil dostať späť na obrazovku klasifikácie. Hľadal vkladanie v hornej lište pri trénovaní a dotrénovaní modelov. Keď nenašiel vkladanie na týchto obrazovkách prechádzal klasifikáciu. Po dlhšom čase hľadania mu bolo poradené, aby si prešiel ešte raz hornú lištu. Následne už našiel vkladanie ale si myslel, že už je na obrazovke dotrénovania, až neskôr sa preklikal na správnu obrazovku.

Tabuľka 7e: Problémy testerov počas vykonávania úlohy č. 5

Feedback k úlohám

Číslo úlohy	Miera náročnosti			TP1	TP2	TP3	TP4	TP5	Priemer
1.	Jednoduchá	1 2 3 4 5 7 8 9 10	Náročná	2	1	1	1	1	1.2
2.	Jednoduchá	1 2 3 4 5 7 8 9 10	Náročná	10	8	7	1	8	6,8
3.	Jednoduchá	1 2 3 4 5 7 8 9 10	Náročná	2	3	1	1	1	1,6
4.	Jednoduchá	1 2 3 4 5 7 8 9 10	Náročná	1	1	1	1	1	1
5.	Jednoduchá	1 2 3 4 5 7 8 9 10	Náročná	2	3	1	1	1	1,6

Tabuľka 8: Súhrn hodnotení používateľov z dotazníka spätnej väzby k jednotlivým úlohám.

4. Zoznam zistených problémov

1. **Nahrávanie vzoriek/modelov.**
    - Pridať návod na pridávanie vzoriek/modelov na úvodnú obrazovku.
2. **Vizualizácia obrázkov**
    - Zlá interpretácia.
    - Iba ⅛ testerov si všimol, že sa dá scrollovať, aj to iba keď na to už dlho pozeral.
    - Chýba informácia o obrázku/vzorke.
3. **Funkcionalita označiť všetko**
    - Málokto si to hneď všimol.
    - Pridať túto funkcionalitu priamo do tabuľky naľavo od modelu.
    - Nečakali, že v secondary buttone je takáto malá funkcionalita.
4. **Primary/secondary buttony**
    - Umiestnenie primary buttonov - testeri mali problém s tým, že boli na spodku obrazovky a primary button bol hore.
    - Všetko treba čítať. Kým testeri zistili, ktorý button je ten, ktorý ich posunie v úlohe ďalej, museli prečítať všetky informácie na stránke.
    - Nedávať malé funkcionality do buttonov.
5. **Chýba úvodná home page**
6. **Krok späť**
    - Každý tester sa pokúsil hľadať back button, najmä pri klasifikácii, keď si chceli spätne niečo pozrieť alebo keď boli na výsledkoch a znova chceli ísť trénovať.
7. **Chybové hlášky**
    - Hlášky sa nevymazávali.
    - Bolo by vhodné nastaviť časový limit.
8. **Výsledky**
    - Nie je možné znova zobraziť už natrénované vzorky a ich výsledky.
    - Testerom chýbali ďalšie informácie pri vyhodnotení.
9. **Export sample/ export all**
    - Takmer všetci dali namiesto exportu všetkých vzoriek iba export jednej vzorky.
    - Nájsť lepšie miesto pre tieto buttony.
10. **Filtrovanie vzoriek/modelov**
    - Každý mal problém zistiť, ktorý model práve pridali.
11. **Natrénovať/dotrénovať**
    - Inak pomenovať?

    - Dať na jednu obrazovku a len pridať button dotrénovať.
12. **Veľkosť písma/rozbitá obrazovka**
    - Uviesť, že sa dá zoomovať.
13. **Označenie, kde sa práve nachádzam**
    - Zvýrazniť na hornej lište, kde sa práve nachádzam
    - Napísať do obrazovky jej názov
14. **Premenovanie "Files" na "Add files"**
    - Súbory sa dajú iba pridávať, nevieme ich prezerať.
    - Alebo pridať obrazovku, kde je nejaká súborová štruktúra.

5. Interview

1. **Aký pocit ste mali z používania aplikácie?**

   TP1: Celkom dobrý, až na to pridávanie vzoriek, to by určite chcelo zmenu. Ostatné úlohy boli jednoduché.

   TP2: Zmiešaný, aplikácia niekedy nespravila to čo mala.

   TP3: Dobrý, veľmi sa mi páči dizajn, jednoduchosť. Všetko bolo jednoduché, až na vkladanie vzoriek.

   TP4: Dobrý pocit. Máte dobre urobenú aplikáciu, user friendly - jednoduché na používanie.

   TP5: Super! Pekné prostredie.

2. **Všimli ste si niečo čo sa Vám veľmi páčilo?**

   TP1: Páči sa mi výber farieb (sivo-zelená), neťahá to veľmi oči.

   TP2: Layout aplikácie je cool.

   TP3: Jednoduchosť a pekný dizajn.

   TP4: Najviac sa mi páčila jednoduchosť aplikácie, všetko bolo dosť jasné, dá sa celkom jednoducho orientovať na stránke. Myslím si, že keď si človek na to zvykne a párkrát prekliká, že je to jednoduché na používanie.

   TP5: Zmena jazyku, pekný dizajn.

3. **Všimli ste si niečo čo sa Vám veľmi nepáčilo?**

   TP1:

   - Nahrávanie vzoriek a modelov by mohlo prejsť zmenou. Napríklad by mohol fungovať štýl drag and drop, alebo by tam mohla byť druhá štruktúra adresára.
   - Nedá sa vrátiť na home page.

- Písmená sú moc veľké.
- Niektoré veci nefungujú úplne tak, ako by mali. Výber jazyka(dropdown nezmizne).
- Select nefungoval tak ako mal. Taktiež by som ho zaradila do tabuľky, pretože button vo mne vyvoláva veľkú funkcionalitu alebo nejaký ďalší veľký krok.
- Súbory by som premenovala na “Add files”, keďže ich viem len pridávať a neviem si ich prezerať.
- Chybové hlášky sa nevymazávali.

TP2:

- Niekedy ma nehodí na výsledok, keď vyberám vzorky. Bolo by dobré, keby sa dalo vrátiť k výsledok minulých klasifikácií, že po kliknutí na vzorku, ktorá už bola klasifikovaná, by sa zobrazili jej výsledky.
- Nahrávanie vzoriek a modelov by mohlo byť jednoduchšie a mohlo by sa dať vkladať viac vzoriek naraz.
- Často som hľadal kde sa nachádzam, teda by sa hodilo, keby bolo zvýraznené hore, že som napr. v klasifikácii.
- Žiadny back button.

TP3:

- Určite nahrávanie vzoriek. Keby mám klikať už len 10 vzoriek, tak je to celkom zdĺhavé.
- Chybová hláška nezmizla sama.
- Niekedy neviem, kde sa nachádzam. Pridať zvýraznenie na taby hore, že kde práve som alebo dať aspoň nadpis do obrazovky.
- Nevšimol som si, že obrázok sa dá prechádzať. Dať k tomu nejaký popis by bolo dobré.

TP4:

- Pridávanie viacerých vzoriek naraz
    - Možno to rozdeliť, že dať dve ikony jednu na pridávanie iba jednej vzorky a druhú na pridávanie viacerých.
- Vizualizácia obrázkov – treba pridať o čo sa jedná, nejaké parametre, lebo pri vizualizácii nie je nič napísané ani o aký olej sa jedná.

TP5:

- Vizualizácie – treba viac farieb, žltá biela. Dať celú mapu a dať tam farby, nastaviť tam rozsah a intenzitu.

4. **Ako sa Vám páčila časť klasifikácia, modely, vizualizácia, ...**

TP1: Všade by som doplnila možnosť editovať názvy. Treba čítať názov každého buttonu čo nie je úplne najlepšie. Button evaluate by som dala dole, keďže to predstavuje posunutie sa niekam ďalej a takéto veci by som očakávala na spodku obrazovky.

TP2: Všetky tabuľky sú OK. Select by mohol fungovať kliknutím na riadok. Prerobil by som zobrazovanie výsledkov. A dal zoradenie aspoň na modely. Inak sa mi všetko ostatné páčilo.

TP3: Pridal by som vlastnú stranu aj pre files. Do časti vizualizácie by som pridal aj nejaké iné veci ako len vykreslenie obrázka. Nejaké dodatočné info, ktoré mi aj niečo k tomu obrázku a vzorke povie.

TP4: Všetky sekcie boli intuitívne**.**

TP5: Vsetko bolo ok až na vizualizáciu. Dať lepší obrázok.

5. **Chýba Vám v aplikácii nejaká funkcionalita?**

TP1: Vlastné trénovanie modelov (vlastné klasifikátory s vlastnými nastaveniami). Vybrať viac vzoriek naraz. Vedieť sa vrátiť späť k detailom vyhodnotenia – podporujte len opätovnú vizualizáciu vzorky. Takže by bolo super, keby sa viem vrátiť k výsledkom, aby som nemusela znova dlho čakať na vyhodnotenie. Urobiť zvýraznenie sekcie, kde sa nachádzam.

TP2: Vidieť minulé výsledky, teda mať nejakú historiu aj v aplikácie nielen v txt súboroch, možno celkovo história čo som robil.

TP3: Filtrovanie vzoriek a modelov podľa času pridania ASC, DESC. Návod ako nahrávať súbory. A vidieť detaily vzoriek, keď už boli raz klasifikované. Súborová štruktúra pri vzorkách a modeloch. Dať popis, že aplikácia sa vie aj zoomovať.

TP4: Zhodnotenie vzoriek, väčší detail o danej vzorke. Pri obrázku by mohlo byť napísané o aký olej ide a čo tam môžeme sledovať. Inak pre mňa nemá obrázok žiadnu výpovednú hodnotu, keď k tomu nič nemám napísané, nie sú tam ani osi popísané. Takže pridať k tomu pár jednoduchých parametrov. Taktiež by bolo dobré vidieť, kedy som pridala daný model, keďže aj v úlohe som musela viac rozmýšľať, ktorý bol pridaný.

TP5: Lepšia vizualizácia jednotlivých vzoriek. Treba popracovať na lepšom zobrazení. Pri vyhodnotení vzorky k detailu napísať aj percentá klasifikácie.

6. **Našli ste niečo čomu by ste nechápali?**

TP1: Dotrénovať a natrénovať. Asi by som nechala iba jednu obrazovku.

TP2: Všetko bolo zrozumiteľné.

TP3: Bolo to jasné.

TP4: Možno len názov dotrénovať a natrénovať by som zvolila inak.

TP5: Nie.

7. **Chápete názvom a ikonám, sú dostatočne zrozumiteľné?**

   TP1: Áno, možno ikonu pre klasifikáciu by som zmenila na niečo iné.

   TP2: Áno je to zrozumiteľné.

   TP3: Áno.

   TP4: Áno, sú zrozumiteľné a chválim, že ste si ich spravili sami.

   TP5: Všetko jasné.

8. **Aké jazyky by ste chceli, aby aplikácia podporovala? Stačí slovenčina a angličtina?**

   TP1: Kľudne len angličtina.

   TP2: Tieto stačia, aj tak každý vie anglicky.

   TP3: Angličtinu. (Hovoril anglicky.)

   TP4: Stačia tieto dva.

   TP5: Slovenčina a angličtina stačia.

9. **Používali by ste dark mód alebo Vám vyhovuje light mód aplikácie?**

   TP1: Využíval, ale stačil by mi aj light mode.

   TP2: Oba sú v pohode**.**

   TP3: Preferujem dark mód, avšak nevadí mi ani váš light mode.

   TP4: Oba sú super, no stačil by mi aj light.

   TP5: Je to super feature, ale vôbec nie je potrebná.

10. **Koľko vzoriek naraz zvyknete vkladať do aplikácie?**

    Do vizualizácie max. dve vzorky, do aplikácie okolo 200 vzoriek.

B Inštalačná príručka

Úvod

Naša aplikácia je pripravená na okamžité používanie vďaka predpripravenému EXE súboru. Je tiež možné aplikáciu vyvíjať alebo spustiť pomocou terminálu na sledovanie stavu. Aplikácia bola vyvíjaná primárne pre platformu Windows. Aplikácia je spúšťaná pomocou localhostu. Port je ale nastavený na auto, teda aplikácii sa pridelí prvý voľný, vďaka tomu by nemalo dôjsť ku konfliktom.

Požiadavky na systém

Pre správne fungovanie aplikácie je potrebné mať systém s:

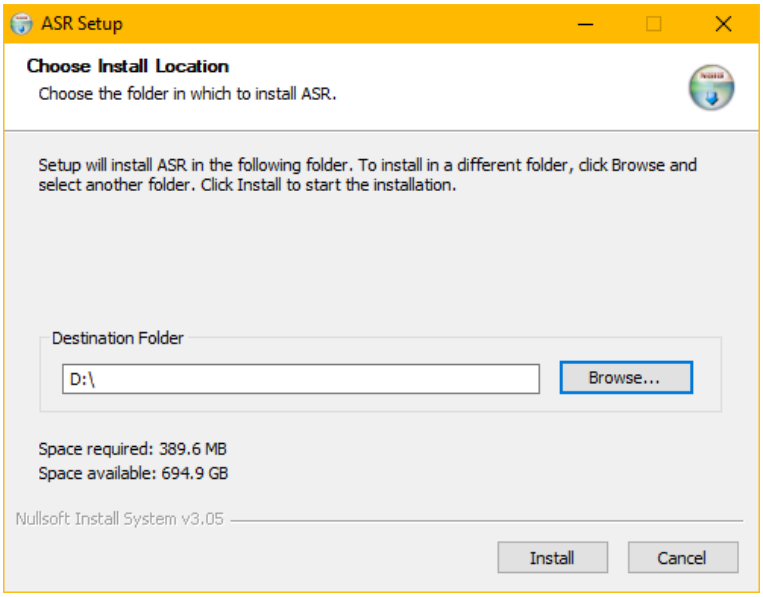
- RAM minimálne 8GB.
- operačný systém Windows 10.
- voľný priestor aspoň 600MB.

Inštalačná príručka pre používateľov

Pre používateľov je sprístupnená iba verzia pre platformu Windows. Samotné spustenie nie je nič náročné. Je potrebné riadiť sa touto príručkou.

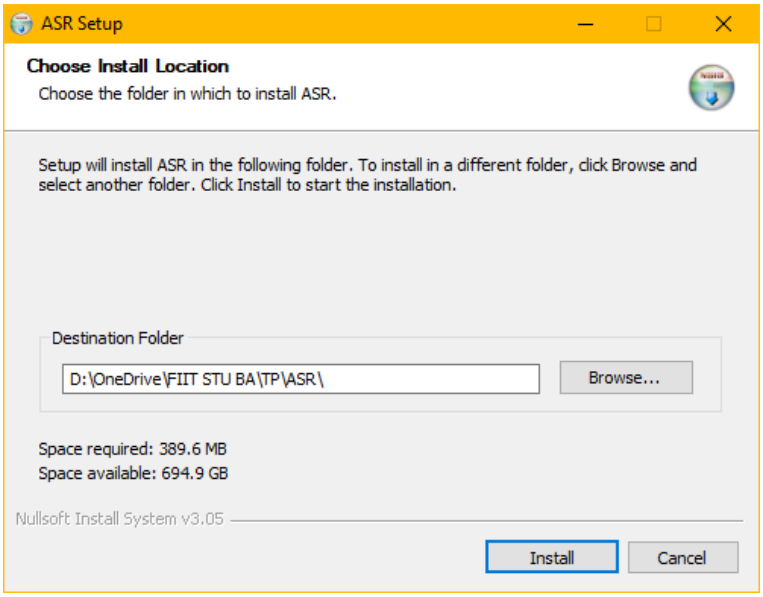
Postup

1. Aplikáciu je možné stiahnuť na odkaze. V prípade nefunkčného prepojenia nás prosím kontaktujte emailom: **mikasa.fiit@gmail.com**.
2. Po stiahnutí ASR-installer.exe súboru je potrebné súbor otvoriť a vybrať umiestnenie pomocou tlačidla **Browse**.



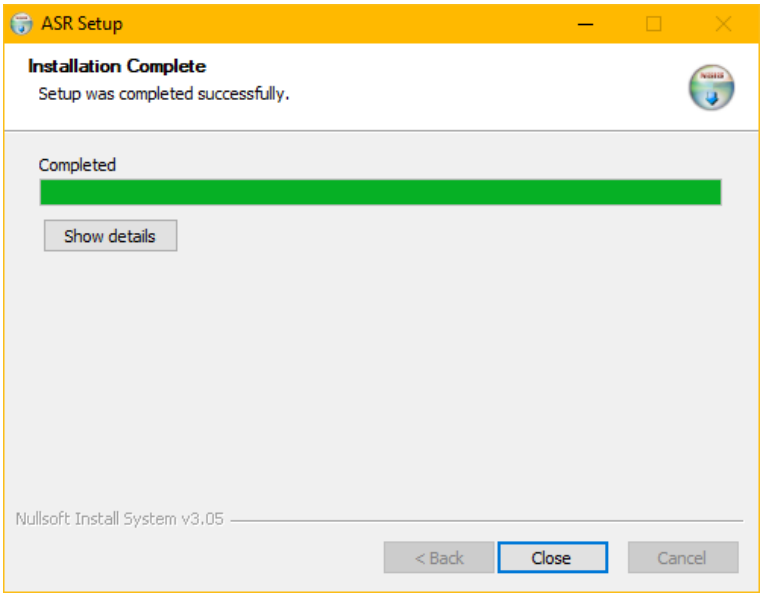
Obr. 33: Výber umiestnenia aplikácie

3. Výber umiestnenia a spustenie inštalácie je potrebné potvrdiť tlačidlom **Install**.



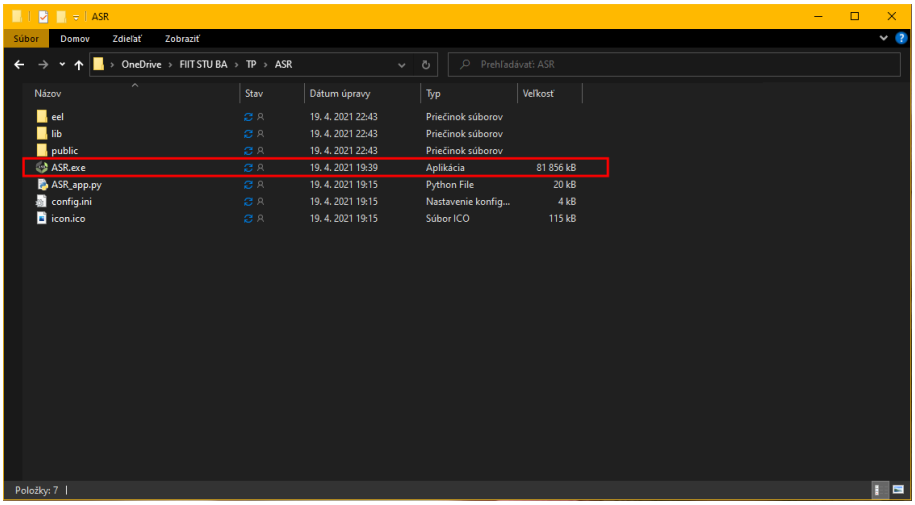
Obr. 34: Potvrdenie inštalácie

4. Súbor si nainštaluje všetky potrebné závislosti. Po dokončení inštalácie je potrebné ukončiť okno pomocou tlačidla **Close**.



Obr. 35: Ukončenie inštalácie

5. Prejdite do súboru kam ste aplikáciu nainštalovali. Otvorte súbor **ASR.exe**



Obr. 36: Spustenie aplikácie

6. Aplikácia je pripravená na používanie.

Inštalačná príručka pre vývojárov

Aplikácia je pre vývojárov pripravená pre platformu Windows. Pre vývoj aplikácie je potrebné mať nainštalovaný Python a potrebné závislosti.

Postup prípravy prostredia

1. Aplikáciu nájdete na GIT-e. V prípade nefunkčného prepojenia nás prosím kontaktujte emailom: **mikasa.fiit@gmail.com**.
2. Ak nemáte Python, stiahnite si ho z odkazu alebo stránky. Pri inštalácii je potrebné pridať Python do PATH ENVIRONMENTS.
3. Po inštalácii Pythonu je potrebné doinštalovať všetky potrebné závislosti. Prejdite do adresára, kde máte stiahnuté zdrojové kódy aplikácie. A spustite príkaz:

```
pip install -r requirements.txt
```

4. Po pridaní závislostí je možné aplikáciu spustiť pomocou príkazu:

```
python ASR_app.py
```

Príprava EXE súboru

Pre vytvorenie EXE súboru je potrebné nasledovné:

1. V prvom rade je potrebné doinštalovať Pyinstaller. Urobíme tak príkazom:

```
pip install pyinstaller
```

2. Prejdite do priečinka kde máte uloženú aplikáciu.
3. Spustenie buildu je možné príkazom:

```
python -m eel ASR_app.py public --onefile \
    --icon=icon.ico --windowed --name=ASR
```

- ***python*** spúšťa inštanciu python programu
- prepínač ***-m*** spúšťa inštanciu pyinstaller, príkaz *python -m* je možné nahradiť príkazom *pyinstaller*
- ***eel*** určuje umiestnenie závislosti pre framework eel
- ***ASR_app.py*** definuje python script s main funkciou
- ***public*** obsahuje webové závislosti, treda css, js a html súbory (poprípade aj iné potrebné)
- prepínač ***–onefile*** zabezpečí zabalenie výsledného exe do jedného súboru so všetkými závislosťami pridanými do Pythonu
- prepínač ***–icon*** definuje ikonu exe súboru
- prepínač ***–windowed*** zabezpečí spustenie aplikácie bez spustenia terminálu. Vo Windowse je možné tento príkaz vypustiť, ak používate pre python script príponu .pyw. Pri debug operáciach nad exe súborom je lepšie tento prepínač vynechať.
- prepínačom ***–name*** je možné definovať názov výsledného exe súboru.

4. Po dokončení buildovania je potrebné presunúť výsledný exe z adresára *dist* do koreňového. Môžeme tak urobiť príkazom:

```
mv dist/ASR.exe ./
```

5. Následne môžeme adresár *dist* vymazať. Napríklad príkazom:

```
rm -rf dist
```

Závislosti

Všetky závislosti Pythonu nájdeme aj v súbore *requirements.txt*. Informačne ich pridávame aj do dokumentu:

- psutil
- bottle==0.12.19
- bottle-websocket==0.2.9

- cffi==1.14.5
- gevent==21.1.2
- gevent-websocket==0.10.1
- greenlet==1.0.0
- pycparser==2.20
- pyparsing==2.4.7
- whichcraft==0.6.1
- zope.event==4.5.0
- zope.interface==5.2.0
- wxPython =4.1.1
- numpy =1.19.5
- Pillow =8.1.0
- scikit-learn==0.23.1
- pandas =1.1.5
- joblib =1.0.1
- matplotlib =3.3.4

C Používateľská príručka

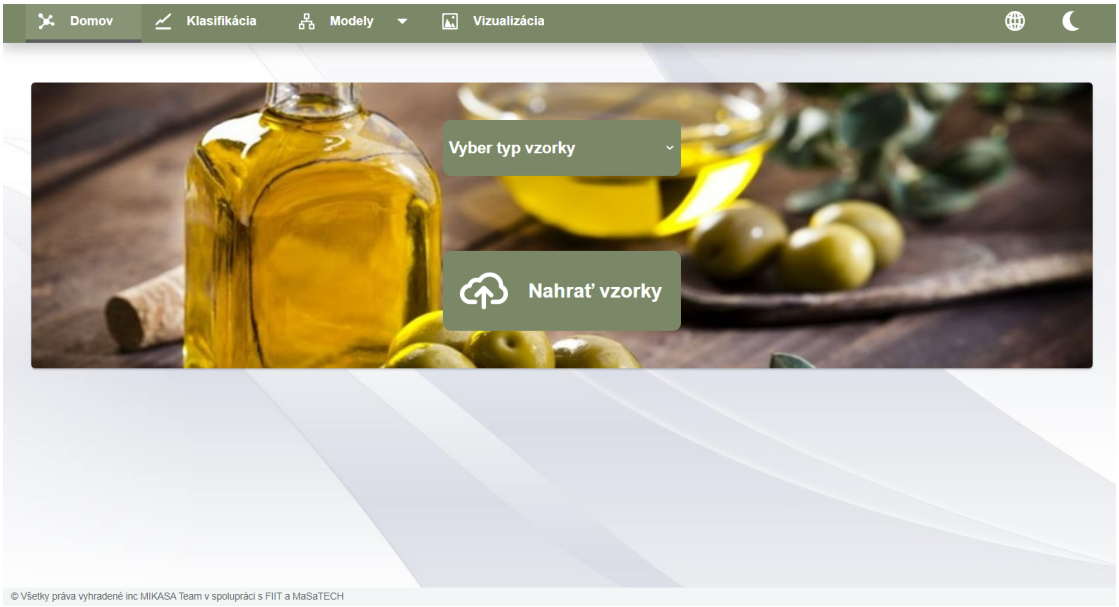
Úvod

Používateľská príručka aplikácie má za účel priblížiť jednotlivé funkcionality aplikácie na klasifikáciu druhov olivového oleja.

V aplikácii máme niekoľko funkcionalít ako napríklad klasifikácia oleja, vytvorenie nového modelu na klasifikáciu, vizualizáciu vložených vzoriek a iné ďalšie, ktoré sú spomenuté v jednotlivých sekciách.

Domovská obrazovka

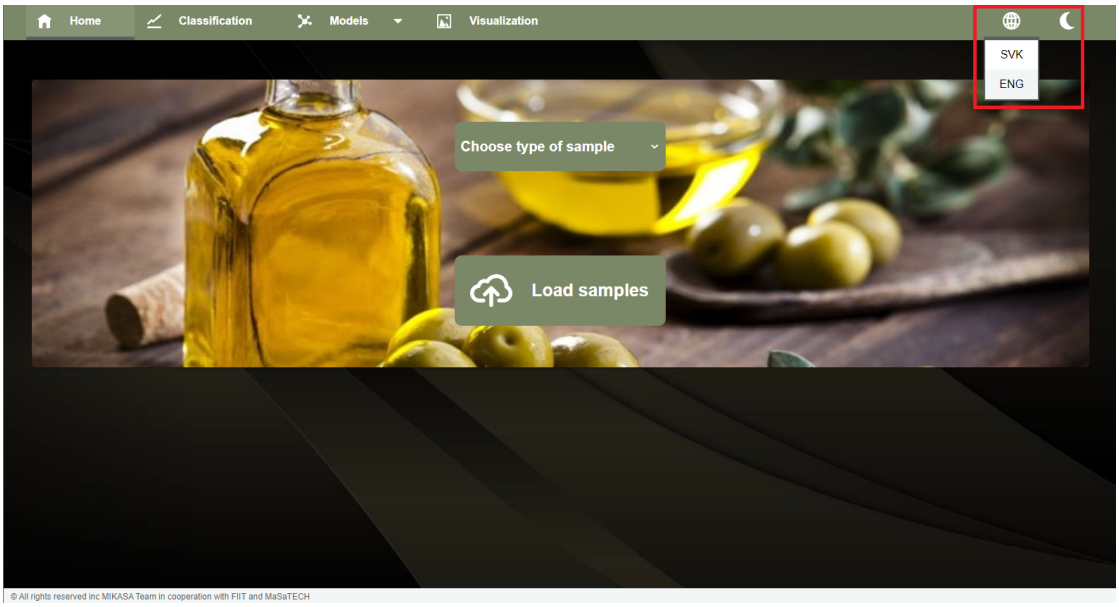
Používateľ otvorí aplikáciu ASR a zobrazí sa mu domovská stránka aplikácie (Obr.37). Na každej obrazovke je horná lišta slúžiaca ako navigátor aplikácie. Nachádzajú sa tu časti Domov, Klasifikácia, Modely a Vizualizácia. Časť Modely je rozdelená na dve časti: Pridať modely a Vytvoriť modely. Po kliknutí na jednotlivé možnosti je používateľ presmerovaný na novú obrazovku a zobrazí sa mu nový obsah podľa toho čo vybral.



Obr. 37: Obrazovka domovskej stránky aplikácie

Zmena jazyka a módu

Taktiež sa na každej obrazovke v pravej časti hornej lišty nachádzajú dve ikony, ktoré slúžia pre zmenu jazyka (ľavá ikona) a pre zmenu svetlého/tmavého módu aplikácie (pravá ikona) (Obr.38). Po kliknutí na ikonu jazyka sa zobrazia dve možnosti (slovenský a anglický jazyk), z ktorých si používateľ vie zvoliť jeden a následne sa jazyk zmení v celej aplikácii. Po kliknutí na ikonu zmeny svetelného módu aplikácie sa prepne aplikácia do tmavého módu a po opätovnom kliknutí sa zmení späť do svetlého módu.



Obr. 38: Zmena jazyka a svetlého/tmavého módu aplikácie

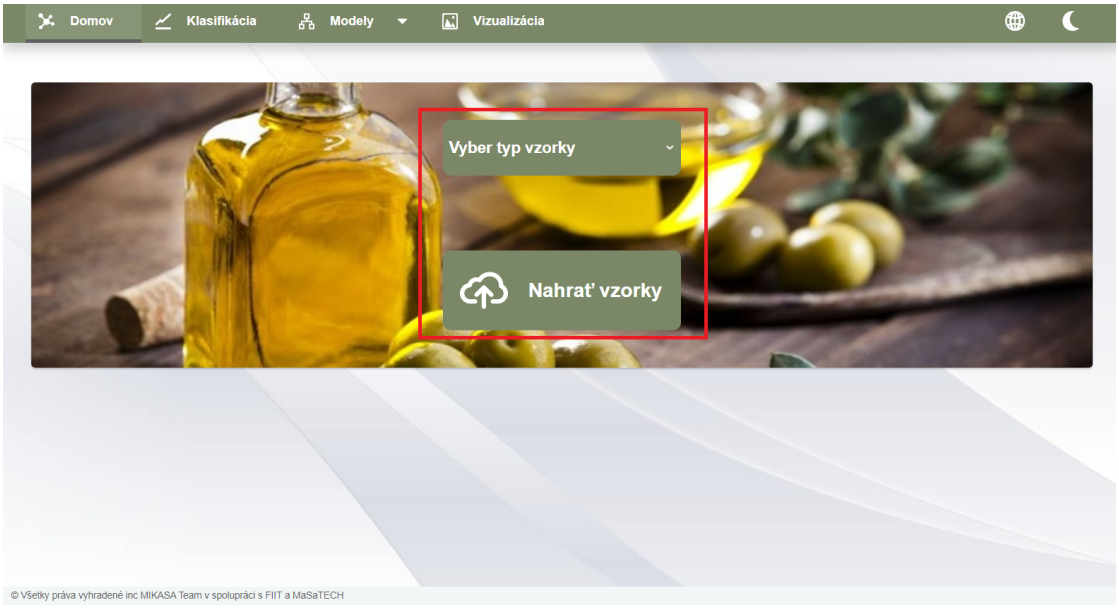
Nahrávanie vzoriek olejov

Na domovskej stránke je možnosť pridávania vzoriek olejov do aplikácie pomocou kliknutia na dropdown (Obr.39). Vzorky môžu byť štyroch druhov. Po výbere typu vzorky sa musia nahrať všetky vzorky vybraného typu, teda by sa nemalo stať, že jedna zo 100 vložených vzoriek je iného typu. Za správnosť vložených vzoriek si zodpovedá sám používateľ. Vzorky sa nahrávajú tak, že sa nahrá adresár, v ktorom je daná vzorka uložená.

- Neoznačené - nepoznáme typy vzoriek a vzorky sú použité pri klasifikácii.

- EVOO - vkladané vzorky patria oleju typu EVOO a vzorky sú použité pri trénovaní nového modelu.
- VOO - vkladané vzorky patria oleju typu VOO a vzorky sú použité pri trénovaní nového modelu.
- LOO - vkladané vzorky patria oleju typu LOO a vzorky sú použité pri trénovaní nového modelu.

Po vybratí typu vzorky a kliknutí na tlačidlo nahrať vzorky, sú vzorky nahraté do aplikácie a je možné s nimi ďalej pracovať v iných častiach aplikácie.



Obr. 39: Nahrávanie vzoriek olejov.

Klasifikácia

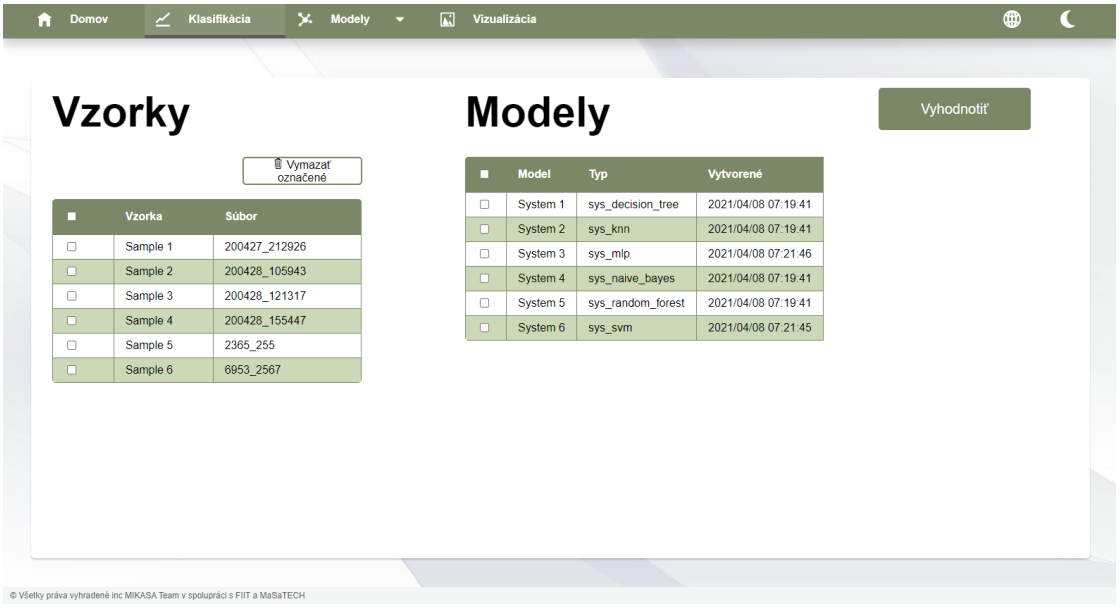
Po kliknutí na klasifikáciu v hornej lište sa zobrazí obrazovka klasifikácie (Obr.40). Nachádzajú sa tu dve tabuľky. V ľavej tabuľke sú uložené pridané vzorky olejov a v pravej modely klasifikátorov.

Tabuľka vzorky obsahuje tri stĺpce. V prvom stĺpci je možné zakliknúť jednotlivé vzorky alebo je možné použiť horný checkbox pre zakliknutie všetkých vzoriek. V druhom stĺpci je uložené poradové číslo vzorky v aplikácii a v poslednom stĺpci je uložený názov adresára, kde je vzorka uložená a zároveň je to aj názov vzorky v aplikácii.

Tabuľka modely obsahuje štyri stĺpce. V prvom stĺpci je možné zakliknúť jednotlivé modely alebo je možné použiť horný checkbox pre zakliknutie všetkých modelov. V druhom stĺpci je uložené poradové číslo modelu v aplikácii. V treťom stĺpci je uložený typ klasifikačného modelu a v poslednom stĺpci je uložená časová pečiatka, kedy bol daný model natrénovaný.

Pri tabuľke vzorky je tlačidlo vymazať označené, ktoré vymaže všetky označené vzorky.

V pravej časti sa nachádza tlačidlo vyhodnotiť, ktoré vyhodnotí označené vzorky pomocou označených modelov.



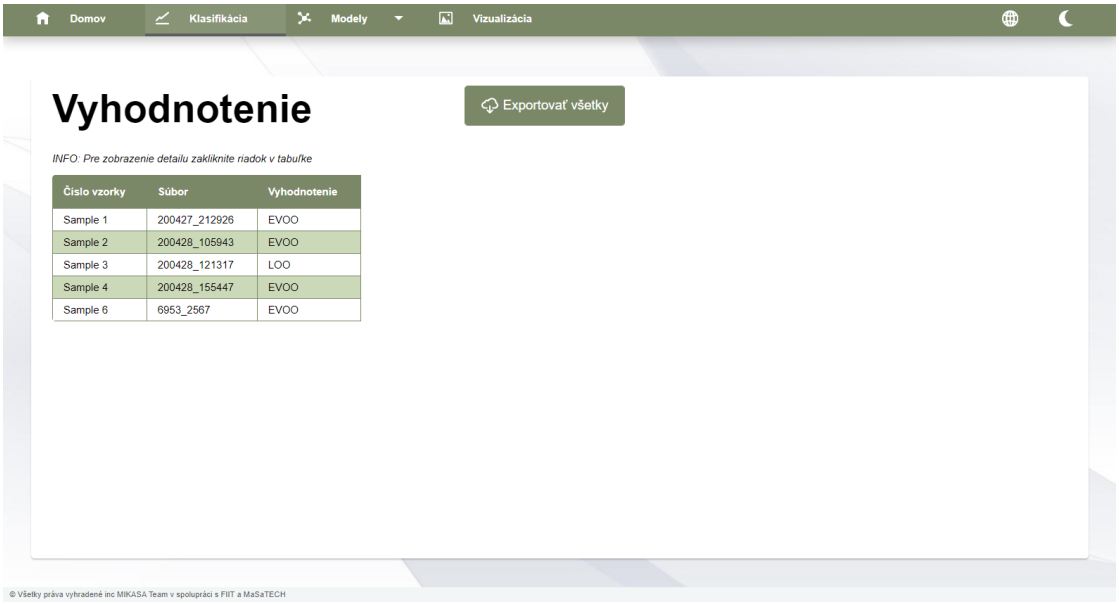
Obr. 40: Základná obrazovka klasifikácie

Vyhodnotenie

Po kliknutí na tlačidlo vyhodnotiť v úvodnej obrazovke klasifikácie sa dostávame na obrazovku vyhodnotenia klasifikácie (Obr.41).

Nachádza sa tu tabuľka Výsledky, ktorá obsahuje tri stĺpce. V prvom stĺpci je ID vzorky, v druhom sa nachádza názov vzorky a v treťom je výsledok predikcie klasifikátora o aký druh oleja sa jedná. Každý riadok vzorky je klikateľný.

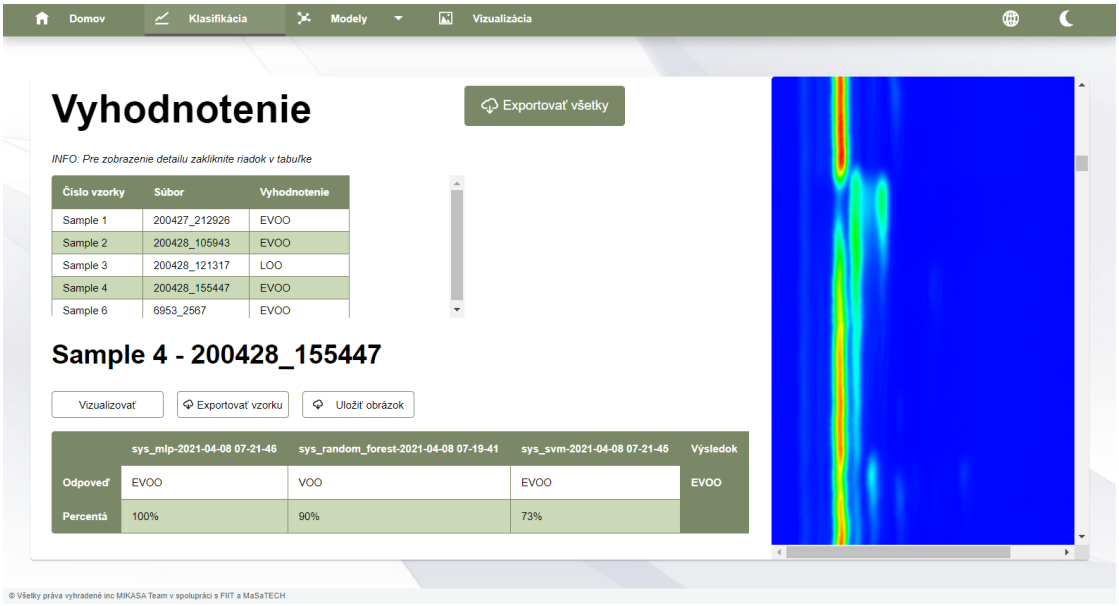
V hornej časti sa nachádza tlačidlo exportovať všetko, ktoré vyexportuje všetky vzorky do .TXT súboru.



Obr. 41: Obrazovka vyhodnotenia klasifikácie

Po kliknutí na vybranú vzorku sa zobrazí tabuľka detailov pre zvolenú vzorku (Obr.42). V tabuľke sa nachádzajú informácie o tom, aký klasifikátor predikoval aký druh oleja daná vzorka je a na koľko % si je týmto výsledkom istý. Taktiež sa tu nachádza celkový výsledok vzorky.

Nad tabuľkou sú tri tlačidlá. Tlačidlo vizualizovať zobrazí danú vzorku v pravej časti aplikácie. Obrázok je možné posúvať vertikálne alebo horizontálne pomocou scrollbaru. Tlačidlo exportovať vzorku uloží zvolenú vzorku do .TXT s jej dátami. Tlačidlo uložiť obrázok uloží obrázok zvolenej vzorky.



Obr. 42: Obrazovka vyhodnotenia s detailom vzoriek a vizualizáciou.

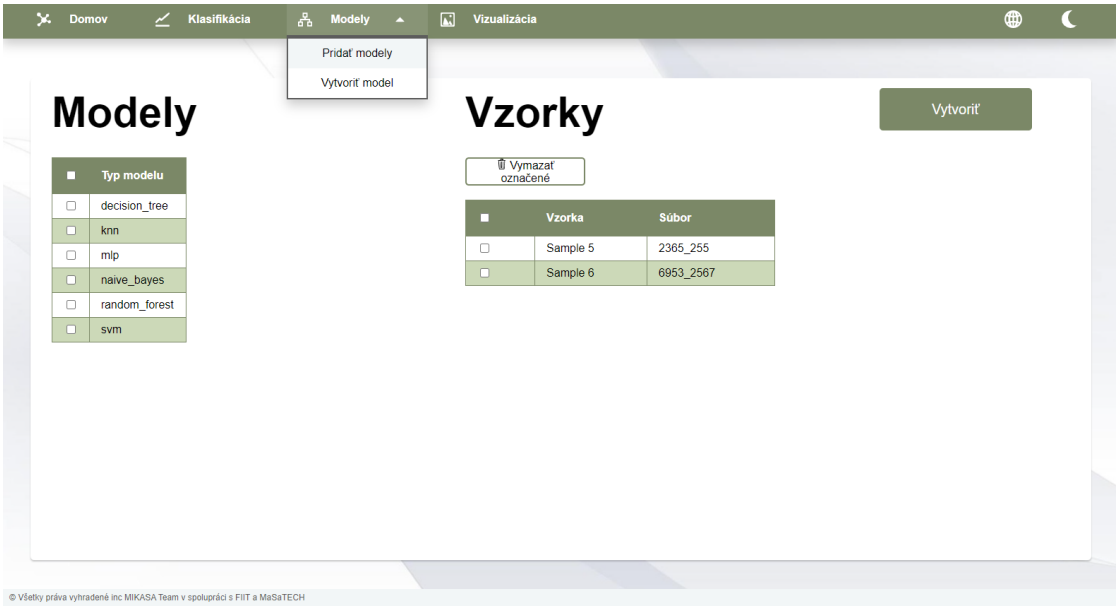
Modely

Obrazovka modely obsahuje dve tabuľky (Obr.43). Tabuľka modely obsahuje zoznam modelov, ktoré je možné pomocou našej aplikácie natrénovať. V prvom stĺpci je možné zakliknúť jednotlivé vzorky alebo je možné použiť horný checkbox pre zakliknutie všetkých vzoriek. V druhom stĺpci sú názvy modelov.

V tabuľke vzorky sa nachádzajú označené vzorky, pomocou ktorých je možné trénovať modely. V prvom stĺpci je možné zakliknúť jednotlivé vzorky alebo je možné použiť horný checkbox pre zakliknutie všetkých vzoriek. V druhom stĺpci sú názvy adresárov, v ktorých sú dané vzorky uložené. Názvy adresárov zároveň reprezentujú názvy vzoriek v aplikácií. Tlačidlo vytvoriť natrénuje zvolené modely na zvolených vzorkách a uloží ich do tabuľky modely v časti klasifikácia.

Taktiež sa tu nachádza tlačidlo Vymazať označené, ktoré po označení vzoriek dané vzorky vymaže z aplikácie.

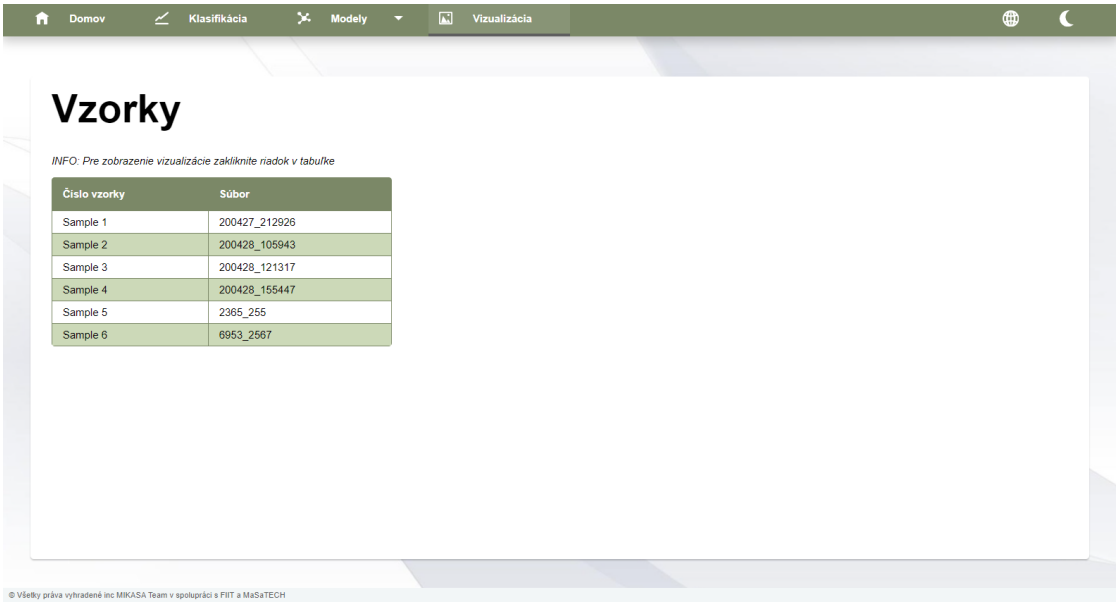
V hornej lište je možnosť zvoliť pridať modely. Pomocou tejto možnosti môže používateľ nahrať vlastné modely s príponou .joblib.



Obr. 43: Obrazovka trénovania nových modelov

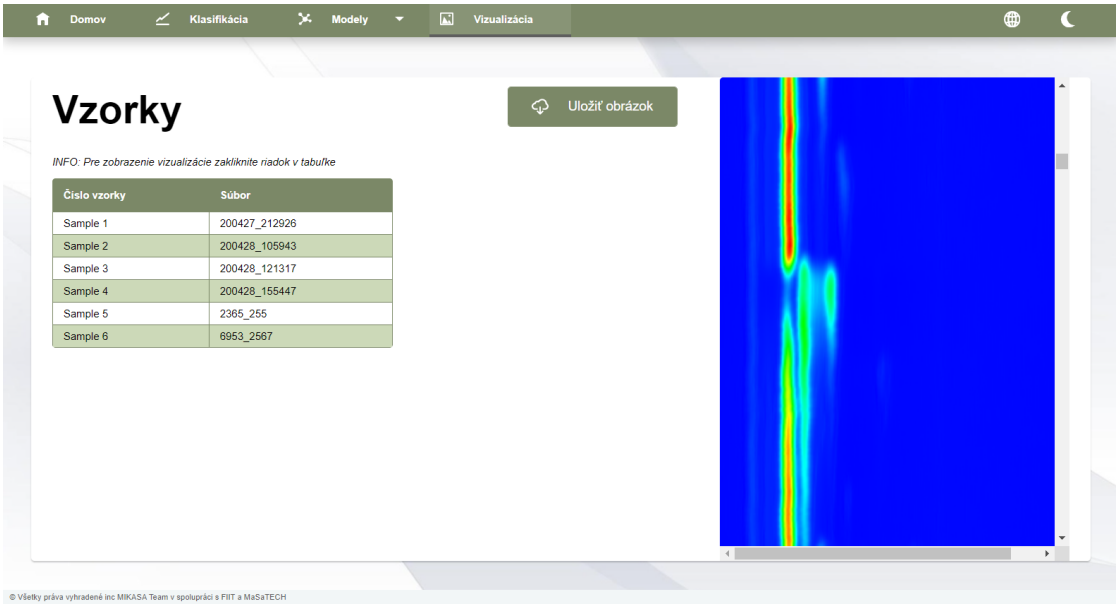
Vizualizácia

Na obrazovke vizualizácie sa nachádza tabuľka s vloženými vzorkami olejov (Obr.44). Tabuľka obsahuje dva stĺpce: poradové číslo vzorky a názov adresára, kde je vzorka uložená a zároveň to je aj názov vzorky v aplikácii. Každý riadok je klikateľný.



Obr. 44: Základná obrazovka vizualizácie dát

Po kliknutí na zvolený riadok sa vykreslí obrázok vzorky (Obr.45). Obrázok je možné posúvať vertikálne alebo horizontálne pomocou scrollbaru. Po vykreslení obrázka sa zobrazí aj tlačidlo uložiť obrázok, pomocou ktorého je vykreslená vzorka uložená.



Obr. 45: Obrazovka s vizualizáciou dát

D Dokumentácia implementácie

Dokumentácia systému automatického rozpoznávania spektier.

1. Predspracovanie

1.1. Spracovanie jedného oleja

súbor:

Predprocessing/convert_functions.py

volanie:

convert_oil(file_name, output, head)

vstupy:

file_name:

cesta k vstupnému súboru.

output:

otvorený súbor pre výpis použitím funkcie open.

head:

hlavička obsahujúca typ oleja a id oleja (v našom prípade názov súboru).

výstupy:

žiaden (zmeny sa zapisujú priamo na output).

opis:

Funkcia načíta údaje z textového súboru a prerobí ich na formát csv. Tieto údaje následne uloží do výstupu. Funkcia môže byť spustená opakovane, aby spracovala viacero súborov, v tom prípade sa údaje z nich zapíšu na samostatné riadky.

príklad:

```
convert_oil("data/GrupoC
-LOO/200428_071902/Spectra.POSITIVE.txt",
            "converted.csv",
            "LOO,200428_071902")
```

1.2. Spracovanie viacerých olejov

súbor:

Predprocessing/convert.py

volanie:

convert_multiple_oils(path, limit_sample, oil_type, output)

vstupy:

path:

cesta k priečinku s priečinkami vzoriek.

limit_sample:

číslo, koľko vzoriek chceme (-1 ak všetky).

oil_type:

názov triedy.

output:

otvorený súbor s výstupom.

výstupy:

žiaden (výstup je zapísaný priamo na output).

opis:

Funkcia vykoná spracovanie meraní s poskytnutých. Spracuje niekoľko text súborov a zapíše ich do poskytnutého cvs súboru. Tento súbor na zápis ostáva otvorený, aby sa do neho mohli zapísať vzorky z iných tried. V aktuálnej verzií sa hlavička nevypisuje, prida sa až v predspracovaní.

príklad:

```
out = open(SAVE_PATH, "w")

convert_multiple_oils(LOAD_PATH_EVOO, 1, "EVOO", out)
convert_multiple_oils(LOAD_PATH_VOO, 1, "VOO", out)
convert_multiple_oils(LOAD_PATH_LOO, 1, "LOO", out)
```

1.3. Odstránenie nepotrebných stĺpcov

1.3.1. Shift values

súbor:

Predprocessing/filter_values.py

volanie:

shift_values(data,
left_range=int(CONFIG['filter_values']['left_range']),
right_range=int(CONFIG['filter_values']['right_range'])):

vstupy:

data:
DataFrame, pre ktorý chceme odstrániť stĺpce.
left_range:
počet stĺpcov vľavo od maxima
right_range:
počet stĺpcov vpravo od maxima

výstupy:

data:
DataFrame s odstránenými stĺpcami.

opis:

Funkcia zisti a napraví posunutie daných dát a to nasledovne:

1. Vypočíta priemer pre každý stĺpec
2. Vyberie pozíciu maximuma z priemerov, čo považujeme za orientačnú hodnotu
3. Vráti dataframe o veľkosti *left_range* stĺpcov naľavo a *right_range* stĺpcov napravo. Tieto hodnoty sa môžu upravovať podla podtreby, aktuálne sa používajú hodnoty z analýzy prvých dát: 205 stred, 55 vľavo, 243 vpravo.

1.3.2. Overenie chýbajúcich spektier

súbor:

Predprocessing/check_missing_spectra.py

volanie:

add_missing_spectra(df)

vstupy:

df:

DataFrame súboru, v ktorom chceme overiť, či má všetky spektrá.

output:

súbor, ktorý obsahuje všetky spektrá pre každý olej.

výstupy:

df:

DataFrame súboru s chýbajúcimi riadkami.

opis:

Funkcia overí, či súbor obsahuje záznam o všetkých spektrách olejov, ktoré sa v súbore nachádzajú. V prípade, že chýba nejaké spektrum pre konkrétny olej, na chýbajúce miesto sa doplní riadok, ktorý obsahuje hodnoty oil_id, spectrum a analysis_time, ostatné hodnoty sú NaN.

1.4. Celkové predspracovanie

súbor:

Predprocessing/predprocess_data.py

volanie:

predprocess(data, train)

vstupy:

data:

DataFrame, ktorý treba predspracovať.

train:

True/False podľa toho, či sú dáta trénovacie alebo validačné.

výstupy:

data:

predspracovaný DataFrame.

opis:

Funkcia skontroluje chýbajúce spektrá a odstráni vybrané stĺpce. V prípade, že sa jedná o trénovacie dáta, funkcia normalizuje dáta. Ak máme testovacie dáta, funkcia ich len transformuje.

1.5. Spracovanie a predspracovanie v procese

súbor:

multiprocessing_app.py

volanie:

convert_predprocess_features(directories, sample_type)

vstupy:

directories:

zoznam adresárov na spracovanie

sample_type:

string obsahujúci typ vzorky: EVOO/LOO/VOO/OTHER

výstupy:

žiadne

opis:

Funkcia volá funkcie convert, predprocess a create_features_file. Vytvára pritom procesy podľa aktuálnej hodnoty max_chunk_size v configu. Každý proces vykoná convert a predprocess a po max_chunk_size sa procesy počkajú aby nenastalo preťaženie procesora príliš veľkým počtom procesov.

2. Normalizácia

Normalizovanie stĺpcov v_xy na hodnoty v rozmedzí [0,1].

2.1. Spracovanie normalizácie

súbor:

Predprocessing/normalization.py

volanie:

normalize(df)

vstupy:

df:

DataFrame s dátami.

výstupy:

df:

DataFrame s normalizovanými dátami.

opis:

Funkcia zavolá orezanie stĺpcov s hodnotami zo spektrometra, nad ktorými vykoná škálovanie hodnôt. Ďalej zavolá funkciu clip_values, ktorá zabezpečí, aby boli všetky hodnoty v rozmedzí [0,1]. Upravené hodnoty sa následne vrátia do pôvodného df. Návratom funkcie je takto upravený DataFrame.

2.2. Clip hodnôt

súbor:

Predprocessing/normalization.py

volanie:

clip_values(df1)

vstupy:

df1:

DataFrame zložený len z dátových stĺpcov v_xy.

výstupy:

df1:

DataFrame s orezanými hodnotami.

opis:

Funkcia zabezpečí orezanie hodnôt na hodnoty v rozmedzí [0,1]. Získanie v_xy stĺpcov. Návratom hodnoty je upravený DataFrame.

2.3. Orezanie df

súbor:

Predprocessing/normalization.py

volanie:

cut_v_columns(df)

vstupy:

df1:

DataFrame s dátami.

výstupy:

df1:

DataFrame obsahujúci iba v stĺpce.

opis:

Funkcia vytvorí nový DataFrame obsahujúci iba v_xy stĺpce vstupného df. Tento nový df je návratovou hodnotou funkcie.

3. Práca s črtami

3.1. Črty

3.1.1. Priemer

súbor:

FeatureSelection/features.py

volanie:

def mean(df_values, column)

vstupy:

df_values:

DataFrame obsahujúci surové predspracované dáta pre práve jeden olej.

column:

názov stĺpca pre ktorý sa vytvorí priemer.

výstupy:

mean_value:

vypočítaný priemer.

opis:

Funkcia na vstupe získa DataFrame dát pre práve jeden olej a názov stĺpca z tohto DataFramu (alebo slova ,,ALL"). V prípade názvu stĺpca sa pre daný stĺpec vypočíta priemer. V prípade slova ,,ALL" v argumente sa vypočíta priemerná hodnota pre všetky bunky v DataFrame. Vypočítaná hodnota je výstupom funkcie.

V prípade využitia pre jeden konkrétny stĺpec sa využíva zápis v tvare "v_n" čiže napríklad mean(df,"v_0") vypočíta priemer zo stĺpca v_0.

príklad:

```
df = pd.DataFrame(data = {'v_0':[1, 4], 'v_1':[4, 7]})
value = mean(df, 'ALL')
assert value == 4
```

3.1.2. Maximum

súbor:

FeatureSelection/features.py

volanie:

max(df_values, column)

vstupy:

df_values:

DataFrame, tabuľka s hodnotami.

column:

názov stĺpca (v_0-v_448).

výstupy:

max_value:

vypočítané maximum.

opis:

Funkcia na vypočítanie maxima v danom stĺpci.

3.1.3. Minimum

súbor:

FeatureSelection/features.py

volanie:

min(df_values, column)

vstupy:

df_values:

DataFrame, tabuľka s hodnotami.

column:

názov stĺpca (v_0-v_448).

výstupy:

mean_value:

vypočítané minimum.

opis:

Funkcia na vypočítanie minima v danom stĺpci.

3.1.4. Medián

súbor:

FeatureSelection/features.py

volanie:

median(df_values, column)

vstupy:

df_values:

DataFrame, tabuľka s hodnotami.

column:

názov stĺpca (v_0-v_448).

výstupy:

median_value:

vypočítaný medián.

opis:

Funkcia na vypočítanie mediánu v danom stĺpci.

3.1.5. Smerodajná odchýlka

súbor:

FeatureSelection/features.py

volanie:

std(df_values, column)

vstupy:

df_values:

DataFrame, tabuľka s hodnotami.

column:

názov stĺpca (v_0-v_448).

výstupy:

std_value:

vypočítaná smerodajná odchýlka.

opis:

Funkcia na vypočítanie smerodajnej odchýlky (Standard deviation) v danom stĺpci.

3.1.6. Rozdiel medzi maximom a minimom

súbor:

FeatureSelection/features.py

volanie:

delta(df_values, column)

vstupy:

df_values:

DataFrame, tabuľka s hodnotami.

column:

názov stĺpca (v_0-v_448).

výstupy:

delta_value:

vypočítaný rozdiel maxima a minima.

opis:

Funkcia na vypočítanie rozdielu medzi maximom a minimom v danom stĺpci.

3.1.7. Súčet

súbor:

FeatureSelection/features.py

volanie:

sum(df_values, column)

vstupy:

df_values:

DataFrame, tabuľka s hodnotami.

column:

názov stĺpca (v_0-v_448).

výstupy:

sum_value:

vypočítaný súčet.

opis:

Funkcia na vypočítanie súčtu hodnôt v danom stĺpci.

3.1.8. Point

súbor:

FeatureSelection/features.py

volanie:

point(df, attribute)

vstupy:

df:

DataFrame, vstup je tabuľka, musí obsahovať stĺpec v_0.

attribute:

[function]_[value]-[value_len]-[spectrum]-[spectrum_len]:

function:

žiadaná funkcia max alebo mean.

value:

číslo, ktoré určuje začiatočný stĺpec.

value_len:

počet hodnôt (stĺpcov).

spectrum:

číslo spektra od ktorého začať.

spectrum_len:

počet spektier.

výstupy:

point:

vypočítaná hodnota črty point.

opis:

Výpočet črty point, čo je maximum/priemer z vyrezanej oblasti. Používa sa najmä na odhalené anomálie medzi triedami oleja. Získanie hodnôt pre attribute je popísané v Zistenie hodnôt pre črtu point (hľadanie anomálií).

príklad:

```
point(df, "mean_0-3-1-2")

      v_0  v_1   v_2   v_3
0     1    1     2     -1
1     9    5     4     -1
2     14   224   22    0

23.333333333333332

(zobrazený index nieje spektrum, nakoľko spektrum začína od 1)
```

3.1.9. Diff - odchýlka od šablóny

súbor:

FeatureSelection/features.py

volanie:

diff(df, oil_type)

vstupy:

df:

DataFrame, tabuľka s hodnotami.

oil_type:

typ oleja, na základe ktorého sa vyberie šablóna. v prípade, ak je k typu pripojené “-mask” použije sa aj maska.

výstupy:

diff:

vypočítaná hodnota črty diff.

opis:

Funkcia vypočíta celkovú odchýlku od jednotlivých tried.

postup:

- Načíta sa šablóna pre olej, prípadne maska.
  - šablóna(DataFrame) bola vytvorená pomocou priemeru hodnôt oleju danej triedy (prvých 10 olejov).
  - maska (DataFrame) je veľkosti šablóny a obsahuje na všetkých miestach hodnotu 1, okrem miest, ktoré boli vyhodnotené ako variabilné v rámci triedy na základe analýzy rovnakých 10-tich olejov.
- Vytvorí sa Diff mapa odčítaním šablóny a aktuálneho oleja.
  - predstavuje rozdiel/odchylku.

- Odstránia sa riadky obsahujúce nan hodnoty (spôsobené rôznym počtom spektier v olejoch).
- Z hodnôt sa spraví absolútna hodnota, čiže vzdialenosť hodnôt.
- Rozdiely menšie ako určená hranica (threshold=2) sa zamenia za hodnotu 0.
- Ak je nastavené použitie masky, aplikuje sa vynásobením aktuálnej mapy maskou.
- Funkcia vráti súčet všetkých hodnôt.

príklad:

```
features.diff(df,”diff_EVOO-mask”)
```

3.2. Vytvorenie tabuľky s črtami

súbor:
create_features_df.py
volanie:
def create_features_df(df)

vstupy:
df:
DataFrame obsahujúci predspracované dáta.
výstupy:
df_features:
výstup funkcie. DataFrame obsahujúci údaje o triede k jedinečným id olejov.

opis:
Funkcia z DataFramu surových predspracovaných dát vytvorí DataFrame obsahujúci len informácie o jedinečných id olejov s informáciou a kategórií oleja.

príklad:

```
df = pd.DataFrame(data = {'class':['EVOO', 'EVOO', 'EVOO'],
'oil_id':['1', '1', '2'], 'data' :['1234', '2345', '3456']})

df_control = pd.DataFrame(data = {'class':['EVOO', 'EVOO'],
'oil_id':['1', '2']})

df_features = create_features_df(df)

assert df_features.equals(df_control) == True
```

3.3. Vytvorenie tabuľky so všetkými črtami

súbor:

create_features_df.py

volanie:

def create_features(df)

vstupy:

df:

DataFrame obsahujúci predspracované dáta.

výstupy:

df_features:

výstup funkcie. DataFrame obsahujúci vytvorené črty pre všetky unikátne oleje.

opis:

Funkcia z DataFramu surových predspracovaných dát vytvorí DataFrame obsahujúci vypočítané všetky črty pre každý olej z daného DataFramu.

3.4. Zistenie chýbajúcich čŕt

súbor:

FeatureSelection/add_features.py - súbor obsahuje funkciu na zistenie chýbajúcich hodnôt (features) vo výslednom DataFrame.

volanie:

check_missing(df, features_to_check, df_features)

vstupy:

df:

DataFrame obsahujúci predspracované dáta.

features_to_check:

pole features, ktoré majú svoje funkcie pre ich výpočet.

df_features:

DataFrame obsahujúci údaje o triede k jedinečným id olejov a môže už obsahovať aj niektoré features.

výstupy:

žiadny (črty sa pridajú priamo do DataFramu).

opis:

Funkcia zistí chýbajúce črty a pre ne zavolá príslušné funkcie na výpočet. Následne sú do DataFramu pridané chýbajúce črty.

3.5. Vytvorenie obrázka z dát

3.5.1. Vytvorenie obrázka

súbor:

FeatureSelection/create_pictures.py

volanie:

create_image(oil_df, directory, oil_id)

vstupy:

oil_df:

DataFrame obsahujúci dáta (v_x stĺpce) jedného oleja. Index DataFrame(u) je stĺpec Spectrum.

directory:

trieda do ktorej je olej zaradený (EVOO, VOO, LOO).

oil_id:

ID oleja.

výstupy:

žiadne (obrázok sa uloží do súboru)

opis:

Funkcia z vytvorí obrázok z DataFramu dát pre jeden olej. Atribúty obrázka (výška a šírka) sú nastavované ako globálne premenné. Vytvorený obrázok sa uloží do priečinka, ktorý označuje triedu oleja, pod menom zadefinovaným ako ID oleja.

3.5.2. Lineárna interpolácia farby

súbor:

FeatureSelection/create_pictures.py

volanie:

linear_interpolation_color(value)

vstupy:

value:

hodnota konkrétnej bunky z DataFrame.

výstupy:

red:

hodnota červenej farby, ktorá bude zapísaná do obrázka. Hodnota je v rozmedzí 0 - 255.

blue:

hodnota modrej farby, ktorá bude zapísaná do obrázka. Hodnota je v rozmedzí 0 - 255.

opis:

Funkcia robí lineárnu interpoláciu medzi modrou a červenou farbou. Vstupom funkcie je hodnota, ktorá je interpolovaná. Maximálna a minimálna hodnota, medzi ktorými sa interpoluje sú nastavené ako globálne premenné.

3.6. Vytvorenie diff mapy

3.6.1. Funkcia na vytvorenie diff mapy

súbor:

FeatureSelection/features.py

volanie:

diff_map(df1, df2, start_column, end_column)

vstupy:

df1,df2:

DataFrame, jednotlivé tabuľky medzi ktorými chceme vytvoriť diffmapu.

start_column, end_column:

názov začiatočného a koncového stĺpca, odkiaľ a pokiaľ chceme zobrať hodnoty pre diff mapy.

výstupy:

result:

DataFrame, ktorý predstavuje diff mapu.

opis:

Funkcia na vytvorenie takzvanej diff mapy, čiže odčítanie hodnôt v dvoch tabuľkách údajov. Používa sa na zistenie rozdielov medzi hodnotami rôznych tried alebo v rámci jednej triedy.

príklad:

```
diff_map(df1, df2, "v_0", "v_447")
```

3.6.2. Vygenerovanie diff máp pre jednu triedu

súbor:

FeatureSelection/create_diff_maps.py

volanie:

same_class_diff(source_file, output_path, size)

vstupy:

source_file:

súbor, ktorý obsahuje oleje rovnakej triedy, po predspracovaní (napr.dataEVOO.csv v Data/Raw).

output_path:

cesta, kde chceme aby sa vytvorili súbory s diff mapami. Každá mapa je samostatný súbor. cesta končí znakom “/”.

size:

veľkosť vzorky, počet súborov, ktoré chceme vytvoriť, “-1” ak chceme všetky možné.

výstupy:

žiadne (výstup sa ukladá priamo do súboru).

opis:

Táto funkcia vytvorí diff mapu pre za sebou idúce dvojice olejov. Pre súbor s 10-timi olejmi tak vytvorí 5 diff máp.

príklad:

```
same_class_diff('../Data/Raw/dataEVOO-10.csv',
'../Data/Feature/DiffMaps/EVOO/', 5)
```

3.6.3. Vygenerovanie diff máp pre rôzne triedy

súbor:

FeatureSelection/create_diff_maps.py

volanie:

different_class_diff(source_file1, source_file2, output_path, size)

vstupy:

source_file1,source_file1:

súbory, kde každý obsahuje oleje rovnakej triedy, po predspracovaní (napr.dataEVOO.csv v Data/Raw).

output_path:

cesta, kde chceme aby sa vytvorili súbory s diff mapami. Každá mapa je samostatný súbor. cesta končí znakom "/".

size:

veľkosť vzorky, počet súborov, ktoré chceme vytvoriť, "-1" ak chceme všetky možné.

výstupy:

žiadne (výstup sa ukladá priamo do súboru).

opis:

Táto funkcia vytvorí diff mapu pre dvojice olejov pričom sa zoberie vždy i-ty olej z každej triedy. Pre súbory s 10-timi olejmi tak vytvorí 10 diff máp.

príklad:

```
different_class_diff('../Data/Raw/dataEVOO-10.csv',
'../Data/Raw/dataLOO-10.csv','../Data/Feature/DiffMaps/EVOO-LOO/'
, 5)
```

3.6.4. Vygenerovanie diff máp medzi olejom a jeho šablónou

súbor:

FeatureSelection/create_diff_maps.py

volanie:

template_diff(source_file, template, output_path, size)

vstupy:

source_file:

súbor obsahuje oleje rovnakej triedy, po predspracovaní (napr.dataEVOO.csv v Data/Raw)

template:

cesta k šablóne, ktorá bude použitá.

output_path:

cesta, kde chceme aby sa ukladali vzniknuté diff mapy.

size:

veľkosť vzorky, počet súborov, ktoré chceme vytvoriť, “-1” ak chceme všetky možné.

výstupy:

žiadne (výstup sa ukladá priamo do súboru).

opis:

Táto funkcia vytvorí diff mapu pre oleje zo vstupu vzhľadom na zadanú šablónu. Tieto mapy ďalej slúžia pre vytvorenie masky oleja.

3.6.5. Vygenerovanie šablóny

súbor:

FeatureSelection/create_diff_maps.py

volanie:

create_template(source_file, output_file)

vstupy:

source_file:

súbor obsahuje oleje rovnakej triedy, po predspracovaní (napr.dataEVOO.csv v Data/Raw).

output_file:

cesta aj s názvom súboru, kam sa uloží šablóna.

výstupy:

žiadne (výstup sa ukladá priamo do súboru).

opis:

Táto funkcia vytvorí šablónu danej triedy oleja s poskytnutých olejov, a to priemerom hodnôt v olejoch.

príklad:

```
create_template('../Data/Raw/dataEVOO-10.csv',
'../Data/Feature/DiffMaps/templates/EVOO.csv')
```

3.6.6. Vygenerovanie masky

súbor:

FeatureSelection/create_diff_maps.py

volanie:

create_mask(source_file, template, output_path)

vstupy:

source_file:

cesta k súborom ktoré vznikli funkciou template_diff. Sú to diff mapy medzi olejmi a šablónou ich triedy.

template:

cesta k použitej šablóne, pre ktorú chceme vytvoriť masku, je potrebná aby maska bola rovnako veľká.

output_file:

cesta aj s názvom súboru, kam sa uloží maska.

výstupy:

žiadne (výstup sa ukladá priamo do súboru).

opis:

Táto funkcia vytvára masku danej triedy oleja s poskytnutých olejov, a to nasledovným postupom:

- Zistia sa oblasti v šablóne, kde sa nachádzajú variabilné hodnoty v rámci olejov jednej triedy.
- Vytvorí sa základ masky, DataFrame rovnako veľký ako šablóna, obsahujúci len hodnoty = 1.
- Na miesta zistené v kroku 1 sa doplnia hodnoty 0.

Maska sa používa tak, že ňou vynásobíme hodnoty, čiže miesta kde je maska rovná 1 sa zachovajú a miesta, kde je hodnota 0 sa vynulujú.

3.7. Zistenie hodnôt pre črtu point (hľadanie anomálií)

3.7.1. Aplikovanie hraničnej hodnoty (threshold)

súbor:

FeatureSelection/select_point_feature.py

volanie:

apply_threshold(data, threshold)

vstupy:

data:

DataFrame, na ktorý chceme aplikovať threshold.

threshold:

hraničná hodnota (odporúčaná hodnota 2).

výstupy:

data:

DataFrame po aplikovaní threshold.

opis:

Funkcia zmení hodnoty menšie ako hraničná hodnota na 0.

3.7.2. Nájdenie oblastí, kde sú hodnoty (threshold)

súbor:

FeatureSelection/select_point_feature.py

volanie:

get_points(data)

vstupy:

data:

DataFrame, pre ktorý chceme nájsť oblasti.

výstupy:

points:

Dataframe obsahujúci nájdené oblasti.

opis:

Funkcia je používaná na identifikovanie oblastí, kde sú hodnoty. Je používaná pre črtu point na identifikovanie hraníc anomálií.

Ohraničenie je vykonané oddelením častí DataFrame pomocou odstránenia nulových riadkov/stĺpcov. Kvôli tomu nemusí oddeliť anomálie blízko seba alebo v niektorých špecifických pozíciách.

príklad:

	start index	end index	start v	end v	max	mean
0	1699.0	1700.0	v_205	v_205	2.0086	2.005700
6	5270.0	5280.0	v_204	v_206	2.1764	1.841061
3	4579.0	4595.0	v_204	v_208	2.7110	1.787266
2	3724.0	3741.0	v_202	v_208	2.6580	1.758643
5	5085.0	5105.0	v_204	v_208	2.4000	1.675175

3.7.3. Spriemerovanie diff máp

súbor:

FeatureSelection/select_point_feature.py

volanie:

average_diff_maps(maps_path)

vstupy:

maps_path:

cesta k mapám, ktoré chceme spriemerovať.

výstupy:

map_sum:

DataFrame tvorený priemernými hodnotami.

opis:

Vytvorí mapu pomocou priemeru hodnôt v poskytnutých mapách. Funkcia sa používa na zovšeobecnenie odchýlok pre dva oleje na odhalenie všeobecných odchýlok. Čiže ak vytvoríme 10 máp, ktoré reprezentujú rozdiely medzi 20-timi olejmi, tak spriemerovaním týchto máp dostaneme priemerné rozdiely v danej triede. Rozdiely boli používané v absolútnej hodnote.

3.7.4. Získanie hodnôt pre črtu point

súbor:

FeatureSelection/select_point_feature.py

volanie:

get_diff_points(path, threshold)

vstupy:

path:

cesta k mapám, na ktorých chceme vykonať analýzu, (napr. "../data/Feature/DiffMaps/templates/EVOO").

threshold:

hraničná hodnota, určuje ako veľké/malé anomálie hľadáme, odporúčaná hodnota je 2 pre normalizované dáta.

výstupy:

DataFrame obsahujúci hodnoty len na zaujímavých oblastiach

opis:

Zistí hodnoty pre črty point, čiže hranice anomálií pre zadané diff mapy olejov. Táto funkcia zahŕňa funkcie vyššie:

- average_diff_maps(path).
- apply_threshold(averaged, threshold).
- get_points(data_threshold).

3.8. Feature selection

3.8.1. Filtrácia

súbor:

FeatureSelection/select_features.py

volanie:

filter_method(df, threshold=0.5)

vstupy:

df:

celý DataFrame spracovaných dát s úpravou stĺpca class z typu string na int.

threshold:

hranica korelácie pre vybrané črty.

výstupy:

žiadne (výstup sa vypíše priamo do konzoly).

opis:

Metóda hľadá korelácie medzi črtami. Následne určí relevantné črty(x>threshold). Všetky črty s koreláciou väčšou ako threshold(0,5) vypíše.

3.8.2. RFE - Hľadanie vhodného počtu čŕt

súbor:

FeatureSelection/select_features.py

volanie:

recursive_feature_elimination(df,labels)

vstupy:

df:

spracované dáta načítané do DataFrame (bez stĺpca class).

label:

DataFrame stĺpca class.

výstupy:

žiadne (výstup sa vypíše priamo do konzoly).

opis:

Funkcia greedy prístupom vyhľadáva optimálny počet čŕt. Metóda vypíše optimálny počet features a skóre zvolených features s decision tree klasifikátorom

3.8.3. Výpis najvhodnejších čŕt s daným množstvom

súbor:

FeatureSelection/select_features.py

volanie:

print_rfe_features(df, labels, head)

vstupy:

df:

spracované dáta načítané do DataFrame (bez stĺpca class).

label:

DataFrame stĺpca class.

nof:

počet features, pre ktoré má nájsť optimálne features.

výstupy:

žiadne (výstup sa vypíše priamo do konzoly).

opis:

Funkcia zistí optimálne features pri danom množstve.

4. Tradičné klasifikátory

4.1. Vytvorenie modelu random forest

4.1.1. Train random forest

súbor:

StandardClasification/random_forest.py

volanie:

train_random_forest(train_features, train_labels, tuning=False)

vstupy:

train_features:

trénovacie features.

train_labels:

trénovacie labels.

tuning=False:

volanie funkcie na vylepšenie hyperparametrov je primárne vypnuté.

výstupy:

clf:

nastavený klasifikátor na random forest spolu s hyperparametrami

opis:

Funkcia, ktorá vytvorí a vráti klasifikátor Random Forest.

4.1.2. Tuning random forest

súbor:

StandardClasification/random_forest.py

volanie:

rf_tuning(train_features, train_labels)

vstupy:

train_features:

trénovacie features.

train_labels:

trénovacie labels.

výstupy:

clf.best_params_:

najlepšie parametre zvoleného klasifikátora.

opis:

Funkcia na nájdenie najlepších hyperparametrov zo zvolenej množiny.

4.2. Vytvorenie modelu decision tree

4.2.1. Train decision tree

súbor:

StandardClasification/decision_tree.py

volanie:

train_decision_tree(train_features, train_labels, tuning=False)

vstupy:

train_features:

trénovacie features.

train_labels:

trénovacie labels.

tuning=False:

volanie funkcie na vylepšenie hyperparametrov je primárne vypnuté.

výstupy:

clf:

nastavený klasifikátor na decision tree spolu s hyperparametrami

opis:

Funkcia, ktorá vytvorí a vráti klasifikátor Decision Tree.

4.2.2. Tuning decision tree

súbor:

StandardClasification/decision_tree.py

volanie:

decision_tree_tuning(train_features, train_labels)

vstupy:

train_features:

trénovacie features.

train_labels:

trénovacie labels.

výstupy:

clf.best_params_:

najlepšie parametre zvoleného klasifikátora.

opis:

Funkcia na nájdenie najlepších hyperparametrov zo zvolenej množiny.

4.3. Vytvorenie modelu SVM

4.3.1. Train SVM

súbor:

StandardClasification/svm.py

volanie:

train_svm(train_features, train_labels, tuning=False)

vstupy:

train_features:

trénovacie features.

train_labels:
trénovacie labels.
tuning=False:
volanie funkcie na vylepšenie hyperparametrov je primárne vypnuté.

výstupy:

clf:
nastavený klasifikátor na SVC spolu s hyperparametrami

opis:

Funkcia, ktorá vytvorí a vráti klasifikátor SVM.

4.3.2. Tuning SVM

súbor:

StandardClasification/svm.py

volanie:

svm_tuning(train_features, train_labels)

vstupy:

train_features:
trénovacie features.
train_labels:
trénovacie labels.

výstupy:

clf.best_params_:
najlepšie parametre zvoleného klasifikátora.

opis:

Funkcia na nájdenie najlepších hyperparametrov zo zvolenej množiny.

4.4. Vytvorenie modelu KNN

4.4.1. Train KNN

súbor:

StandardClasification/knn.py

volanie:

train_knn(train_features, train_labels, tuning=False)

vstupy:

train_features:
trénovacie features.
train_labels:
trénovacie labels.
tuning=False:
volanie funkcie na vylepšenie hyperparametrov je primárne vypnuté.

výstupy:

clf:

nastavený klasifikátor na knn spolu s hyperparametrami

opis:

Funkcia, ktorá vytvorí a vráti klasifikátor knn.

4.4.2. Tuning KNN

súbor:

StandardClasification/knn.py

volanie:

knn_tuning(train_features, train_labels)

vstupy:

train_features:

trénovacie features.

train_labels:

trénovacie labels.

výstupy:

clf.best_params_:

najlepšie parametre zvoleného klasifikátora.

opis:

Funkcia na nájdenie najlepších hyperparametrov zo zvolenej množiny

4.5. Vytvorenie modelu naive bayes (gaussian)

4.5.1. Train naive bayse

súbor:

StandardClasification/naive_bayes.py

volanie:

train_naive_bayes(train_features, train_labels, tuning=False)

vstupy:

train_features:

trénovacie features.

train_labels:

trénovacie labels.

tuning=False:

volanie funkcie na vylepšenie hyperparametrov je primárne vypnuté.

výstupy:

clf:

nastavený klasifikátor na naive bayes spolu s hyperparametrami

opis:

Funkcia, ktorá vytvorí a vráti klasifikátor Naive Bayes.

4.5.2. Tuning naive bayes

súbor:

StandardClasification/naive_bayes.py

volanie:

naive_bayes_tuning(train_features, train_labels)

vstupy:

train_features:

trénovacie features.

train_labels:

trénovacie labels.

výstupy:

clf.best_params_:

najlepšie parametre zvoleného klasifikátora.

opis:

Funkcia na nájdenie najlepších hyperparametrov zo zvolenej množiny.

5. Neurónové siete

5.1. Multilayer Perceptron

5.1.1. Train MLP

súbor:

StandardClasification/mlp.py

volanie:

train_mlp(train_features, train_labels, tuning=False)

vstupy:

train_features:

trénovacie fetures.

train_labels:

trénovacie labels.

tuning=False:

volanie funkcie na vylepšenie hyperparametrov je primárne vypnuté.

výstupy:

clf

nastavený klasifikátor MLP spolu s hyperparametrami

opis:

Funkcia, ktorá vytvorí a vráti klasifikátor Multilayer Perceptron.

5.1.2. Tuning MLP

súbor:

Neuronsn/train_mlp.py

volanie:

mlp_tuning(train_features, train_labels)

vstupy:

train_features:

trénovacie fetures.

train_labels:

trénovacie labels.

výstupy:

clf.best_params_

najlepšie hyperparametre MLP klasifikátora.

opis:

Funkcia na nájdenie najlepších hyperparametrov zo zvolenej množiny.

5.2. VGG

5.2.1. Init VGG

súbor:

Neurons/Vgg16Model.py

volanie:

def __init__(self, num_classes)

vstupy:

num_classes:

počet tried, ktoré klasifikujeme.

výstupy:

out:

posledná vrstva siete, ktorá obsahuje predikcie.

opis:

Trieda Vgg16 reprezentuje architektúru konvolučnej neurónovej siete.

5.2.2. Classify VGG

súbor:

Neurons/train_vgg_net.py

volanie:

classify_vgg16()

vstupy:

žiadne.

výstupy:

žiadne (výstup sa vypíše do konzoly).

opis:

Súbor obsahuje funkciu na trénovanie neurónovej siete Vgg16. Funkcia si načíta obrázky zo súboru pomocou datasetu. Dáta sú náhodne rozdelené na trénovacie a testovacie. Na trénovacích dátach as natrénuje model Vgg16 a následne sa vyhodnotí jeho úspešnosť na testovacích dátach.

6. Trénovanie

6.1. Spúšťanie trénovania 1 modelu

súbor:

StandardClasification/train.py

volanie:

append_df(df, path)

vstupy:

df:

existujúci DataFrame.

path:

cesta k súboru, v ktorom sú uložené vypočítané features.

výstupy:

df:

DataFrame, ktorý je doplnený o dáta z nového súboru.

opis:

Funkcia načíta nový DataFrame zo súboru ku ktorému sa poslala cesta. Následne sa tento DataFrame spojí s pôvodným DataFramom. Táto funkcia je potrebná v aplikácií, kedy každá vzorka má vlastný súbor s vypočítanými features.

6.2. Spúšťanie trénovania 1 modelu

súbor:

StandardClasification/train.py

volanie:

train_model(classifier, train_features, train_labels)

vstupy:

classifier:

klasifikátor, ktorý chceme natrénovať.

train_features:

načítané trénovacie dáta.

train_labels:

triedy, ktoré patria k jednotlivým dátam.

výstupy:

clf:

natrénovaný klasifikátor.

name:

názov natrénovaného klasifikátora.

opis:

Vo funkcii sa načíta klasifikátor, ktorý je následne natrénovaný na zvolených dátach. Natrénovaný klasifikátor je uložený. Funkcia vracia natrénovaný klasifikátor a jeho názov.

6.3. Dotrénovanie modelu

súbor:

StandardClasification/train.py

volanie:

retrain_model()

vstupy:

-

výstupy:

-

opis:

Prázdna funkcia, ktorá bude obsahovať kód na dotrénovanie vybraného modelu na nových dátach.

6.4. Spúšťanie trénovania

súbor:

StandardClasification/train.py

volanie:

train(classifiers, data)

vstupy:

classifiers:

pole klasifikátorov, ktoré chceme natrénovať.

data:

načítané trénovacie dáta vo forme DataFramu.

výstupy:

names:

pole názvov natrénovaných klasifikátorov.

opis:

Funkcia rozdelí trénovacie dáta na dáta a triedy. Následne natrénuje na dátach zvolené klasifikátory. Funkcia vráti názvy natrénovaných klasifikátorov, aby sme ich mohli neskôr identifikovať.

7. Klasifikovanie

7.1. Klasifikácia pomocou 1 modelu

súbor:

StandardClasification/predict_classes.py

volanie:

predict_class(classifier, test_features)

vstupy:

classifier:

klasifikátor, ktorý chceme vyhodnotiť.

test_features:

načítané testovacie dáta.

výstupy:

pred_labels:

pole predikovaných tried.

probability:

maximum z poľa pravdepodobností o akú triedu sa jedná

opis:

Funkcia načíta natrénovaný klasifikátor a predikuje triedu načítaných dát. Funkcia vráti pole predikovaných tried.

7.2. Klasifikácia modelov

súbor:

StandardClasification/predict_classes.py

volanie:

predict(classifiers, data)

vstupy:

classifiers:

pole klasifikátorov, ktoré chceme vyhodnotiť.

data:

načítané testovacie dáta vo forme DataFramu.

výstupy:

predictions:

pole predikovaných tried.

predictions_arr:

pole predikcií jednotlivých klasifikátorov pre každú predikciu.

probability_arr:

pole maximálnych pravdepodobností o akú triedu sa jedná

opis:

Funkcia rozdelí testovacie dáta na dáta a triedy. Klasifikátory natrénuje a vyhodnotí úspešnosť predikcie jednotlivých klasifikátorov. Klasifikátory sa zadávajú ako celý názov súboru napr.: ['decision_tree-2020-12-07 12-43-57.joblib', 'knn-2020-12-07 12-24-48.joblib']

7.3. Klasifikácia pomocou využitia kombinácie modelov

súbor:

StandardClasification/predict_classes.py

volanie:

combine_predictions(predictions_arr)

vstupy:

predictions_arr:

pole predikcií jednotlivých klasifikátorov.

výstupy:

final_arr:

pole predikcií.

opis:

Funkcia načíta pole predikcí samostatných klasifikátorov. Pre jednotlivé vzorky identifikuje triedu, ktorá sa vyskytuje najčastejšie ako odpoveď.

Funkcia vracia pole predikcií, ktoré boli vytvorené na základe jednotlivých predikcií klasifikátorov.

8. Evaluácia a grafy

8.1. Evaluácia klasifikátora

súbor:

Statistic/evaluation.py

volanie:

evaluate_clf(test_labels, pred_labels)

vstupy:

test_labels:

skutočné lable z klasifikátora.

pred_labels:

predikované lable z klasifikátora.

výstupy:

evaluation:

úspešnosť klasifikátora, pole accuracy, F1 micro a F1 macro.

opis:

Funkcia vypočíta metriky accuracy, F1 micro a F1 macro pre klasifikátor, tie vracia v percentách. Taktiež sa vytvára matica zámen, ktorá sa vypíše do konzoly.

8.2. Evaluácia viacerých klasifikátorov

súbor:

Statistic/evaluation.py

volanie:

evaluate_multiple_clf(classifiers, data, total=True)

vstupy:

classifiers:

zoznam klasifikátorov ako názov súboru joblib aj s príponou.

data:

vstupný DataFrame, na ktorom sa majú modely vyhodnotiť.

total = True:

logická premenná určujúca či sa vyhodnotí aj kombinácia všetkých modelov.

výstupy:

scores:

úspešnosť klasifikátorov, pole accuracy, F1 micro a F1 macro.

opis:

Funkcia vypočíta metriky accuracy, F1 micro a F1 macro pre všetky vybrané klasifikátory, tie vracia v percentách. Taktiež sa vytvára matica zámen, ktorá sa vypíše do konzoly.

8.3. Vytvorenie grafu výsledkov pre všetky tradičné klasifikátory

súbor:

Statistic/create_chart.py, Súbor obsahuje funkciu na vytvorenie grafu výsledkov pre všetky tradičné klasifikátory.

volanie:

scores_charts(score, labels, file_name=”score”)

vstupy:

scores:

výsledky klasifikátora z funkcie def evaluate_clf(test_labels, pred_labels, name).

labels:

názvy stĺpcov v grafe.

file_name:

názov grafu pod ktorým bude uložený obrázok.

výstupy:

žiadne (výstup sa priamo uloží do súboru).

opis:

Funkcia vytvorí graf výsledkov všetkých tradičných klasifikátor pre metriky accuracy, F1 micro a F1 macro. Názvy stĺpcov sú v tvare: ['KNN', 'RF', 'SVC', 'DT', 'NB', 'MLP', 'Total']

E Dokumentácia implementácie aplikácie

Dokumentácia aplikácie automatického rozpoznávania spektier.

1. JS

1.1. Front-end

1.1.1. initScene

súbor:

public/script.js

volanie:

initScene(lang)

vstupy:

lang:

JSON key values s textami pre jednotlivé komponenty

opis:

funkcia dostane zavolá funkciu na vyplnenie komponentov a nastaví počiatočnú veľkosť screenu a nastaví zavolá funkciu na nastavenie témy.

príklad:

```
# javascript
  eel.get_lang(lang)(initScene);
```

1.1.2. setLang

súbor:

public/script.js

volanie:

setLang(lang)

vstupy:

lang:

JSON key values s textami pre jednotlivé komponenty

opis:

funkcia dostane JSON, ktorý preiteruje podľa kľúčov a jednotlivým komponentom priradí hodnoty. Kľúče v JSONE sa musia zhodovať s id existujúcich komponentov.

1.1.3. setTheme

súbor:

public/script.js

volanie:

setTheme(theme)

vstupy:

theme:

light alebo dark

opis:

funkcia nastaví css súbor podľa toho akú tému máme zvolenú. Správne fungovanie danej funkcie zabezpečíme zavolaním funkcie changeTheme().

príklad:

```
# javascript
function changeTheme() {
    eel.get_conf("theme")(getThemeInHTML);
}

function getThemeInHTML(theme) {
    if (theme === "dark") setTheme("light");
    if (theme === "light") setTheme("dark");
}
```

1.1.4. clearContent

súbor:

public/script.js

volanie:

clearContent()

opis:

keďže aplikácia funguje pomocou komponentov v dive content, pri zobrazení nového komponentu je potrebné premazať to čo sa v ňom už nachádza. Funkcia teda prejde childy a zmení im display na none.

1.1.5. scrollFunction

súbor:

public/script.js

volanie:

table.parentElement.onscroll = function () {scrollFunction(table.parentElement, "samplesLabel");}

pozor:

pre naviazanie na novú tabuľku je potrebné pridať výpočet zmeny veľkosti do dictionary.

Každá tabuľka musí byť súčasťou div

vstupy:

tableClass:

referencia na div, v ktorom je tabuľka zaobalená

headingId:

ID nadpisu v danom stĺpci

opis:

Funkcia počíta posuny pri zmenšení nadpisov.

Objasnenie hodnôt premennej dict

95% - celková výška mínus 5% margin pre každý nadpis,

69px - výška nadpisu h2

34px - výška nadpisu h3

-16px-18px - odrátanie výšky elementu info a marginu tohto elementu

1.1.6. disableAllContent

súbor:

public/script.js

volanie:

disableAllContent()

opis:

vytvorí div, ktorý prekryje všetko okrem tlačidiel na zmenu témy, jazyka a načítanie vzoriek.

1.1.7. unDisableAllContent

súbor:

public/script.js

volanie:

unDisableAllContent()

opis:

opak k disableAllContent().

Table Selection Controller

1.1.8. addRowEventListener

súbor:

public/script.js

volanie:

tableSelectionController.addRowEventListener(tr, tableId);

vstupy:

tr:

riadkový element tabuľky s komplentnými dátami (už má aj všetky stĺpce)

tableId:

ID tabuľky, napríklad: "samplesTable"

opis:

funkcia priradí všetkým stĺpcom riadka listener. Po kliknutí sa zavolá funckcia selectRow()

1.1.9. selectRow

súbor:

public/script.js

volanie:

tableSelectionController.selectRow(row, tableId)

vstupy:

row:

riadkový element tabuľky (ten, na ktorý sa kliklo)

tableID:

ID tabuľky, napríklad: "samplesTable"

opis:

funkcia označí checkbox v tabuľke a zavolá funkciu markOne

1.1.10. markOne

súbor:

public/script.js

volanie:

tableSelectionController.markOne(tableId)

vstupy:

tableID:

ID tabuľky, napríklad: "samplesTable"

opis:

funkcia skontroluje či sú všetky checkboxy zaškrtnuté alebo nie a podľa toho, rozhodne o označení/odznačenní horného checkboxu.

1.1.11. markAll

súbor:

public/script.js

volanie:

tableSelectionController.markAll(tableId)

vstupy:

tableID:

ID tabuľky, napríklad: "samplesTable"

opis:

funkcia označí/odznačí všetky checkboxy podľa stavu horného checkboxu. Funkcia sa volá v index.html.

1.2. Back-end

1.2.1. saveImg

súbor:
public/script.js
volanie:
saveImg()

opis:
načíta sa cesta k obrázku a ten sa uloží.

1.2.2. saveExport

súbor:
public/script.js
volanie:
saveExport(all)

vstupy:
all:
boolean ci ukladáme všetko ale iba jeden súbor

opis:
ak je true hodnota all tak sa uložia všetky exporty, ak nie uloží sa konkrétny export.

1.2.3. getFiles

súbor:
public/script.js
volanie:
getFiles(files)

vstupy:
files:
JSON objekt so súbormi

opis:
funkcia iba prekonvertuje JSON z pythonu a pošle do funkcie na generovanie obsahu tabuliek.

1.2.4. loadTableData

súbor:

public/script.js

volanie:

loadTableData(obj)

vstupy:

obj:

prekonvertovaný JSON z pythonu do JSON objektu

opis:

funkcia vymaže obsah tabuliek, zobrazí tabuľky a zavolá dunkciu reloadTables na ich naplnenie.

1.2.5. reloadTables

súbor:

public/script.js

volanie:

reloadTables(obj)

vstupy:

obj:

prekonvertovaný JSON z pythonu do JSON objektu

opis:

funkcia vymaže obsah tabuliek, ktorý obsahuje a naplní ich novým obsahom.

1.2.6. getAllSelected

súbor:

public/script.js

volanie:

getAllSelected(table)

vstupy:

table:

element tabuľky, z ktorej chceme vybrať prvky.

výstupy:

data:

zoznam vzoriek, ktoré používateľ vybral

opis:

Funkcia prejde tabuľku, ktorú dostala na vstupe a vyberie vzorky, ktoré boli označené používateľom.

1.2.7. getAllSelectedModel

súbor:

public/script.js

volanie:

getAllSelectedModel(table)

vstupy:

table:

element tabuľky, z ktorej chceme vybrať prvky.

výstupy:

data:

zoznam modelov, ktoré používateľ vybral

opis:

Funkcia prejde tabuľku, ktorú dostala na vstupe a vyberie modely, ktoré boli označené používateľom.

1.2.8. parseLoadedData

súbor:

public/script.js

volanie:

parseLoadedData(data)

vstupy:

data:

JSON objekt, ktorý potrebuje spracovať

opis:

Funkcia rozbalí JSON dáta a následne ich opäť načíta do tabuľky.

1.2.9. draw

súbor:

public/script.js

volanie:

draw(data)

vstupy:

data:

zoznam informácii o obrázku

opis:

funkcia vyberie informácie o obrázku a vykreslí obrázok na plochu.

1.2.10. showResultsWithData

súbor:

public/script.js

volanie:

showResultsWithData(data)

vstupy:

data:

JSON objekt pre resultScreen

opis:

funkcia zobrazí potrebné komponenty a naplnia sa tabuľky so vzorkami.

1.2.11. showSampleDetail

súbor:

public/script.js

volanie:

showSampleDetail(element)

vstupy:

element:

riadok tabuľky, ktorý chceme zobraziť

opis:

podobné ako showResultsWithData akurát zobrazí detail vzorky.

1.2.12. showHomePage

súbor:

public/script.js

volanie:

showHomePage()

opis:

funkcia zobrazuje komponenty pre výber typu vzorky a nahranie zvolených vzoriek.

1.2.13. showVisualizationWithData

súbor:

public/script.js

volanie:

showVisualizationWithData(data)

vstupy:

data:

JSON objekt pre vizualizácia

opis:

podobné ako showResultsWithData akurát zobrazí vizualizáciu vzorky.

1.2.14. showModels

súbor:

public/script.js

volanie:

showModels()

opis:

funkcia zobrazuje komponenty pre vytvorenie nového modelov.

1.2.15. showClassification

súbor:

public/script.js

volanie:

showClassification(reload)

vstupy:

reload:

boolean, ktorý v sebe nesie informáciu, či je potrebné znovu načítať dáta v tabuľkách.

opis:

funkcia zobrazí komponenty pre klasifikáciu.

1.2.16. trainModels

súbor:

public/script.js

volanie:

trainModels()

opis:

funkcia ošetruje, či sú vybraté nejaké modely a vzorky, ak áno tak natrénuje nové vybrané modely na zvolených dátach.

1.2.17. deleteAllSelected

súbor:

public/script.js

volanie:

deleteAllSelected()

opis:

funkcia vymaže vybrané vzorky.

1.3. Aj pre python

1.3.1. showMessage

súbor:

public/script.js

volanie:

showMessage(data)

vstupy:

data:

súbor: {“message”: “”, “type”: 0}

type:

0 - error
1 - sucess
2 - warning
3 - hidden

opis:

funkcia vyparsuje z dát súbor a vloží do alertify podľa typu správy.

príklad:

```
# javascript
  eel.get_mess("noModelsOrSamples", 0)(showMessage);

# python
  eel.showMessage(get_mess("save", 1))
```

1.4. Alertify

Private object.

1.4.1. success, error, warning, hidden

súbor:

public/script.js

volanie:

alertify.success(message);
alertify.error(message);
alertify.warning(message);
alertify.hidden(message);

vstupy:

message:

String. Správa, ktorá sa zobrazí vo vyskakovacom okne.

wait:

Optional. Number. Počet milisekúnd, koľko má, čakať, kým sa okno zobrazí.

opis:

funkcia zobrazí alertify message

1.4.2. set

súbor:

public/script.js

volanie:

alertify.set({});

vstupy:

delay:

Počet milisekúnd, za koľko sa zobrazená správa skryje.

opis:

funkcia nastaví milisekundy, za koľko sa zobrazená správa skryje.

2. Python

2.1. get_lang

súbor:

ASR_app.py

volanie:

get_lang(language)

vstupy:

language:

zo languages.py sa podľa jazyka vyberu názvy k jednotlivim komponentom

výstupy:

súbor:

dict key value {'files': 'súbory'}

opis:

funkcia podľa jazyka vyberie JSON zo súboru lib/languages.py a pošle ho do aplikácie. Pri neznámom jazyku vrati eng a zaloguje chybu.

príklad:

```
# python
  get_lang(“sk”)

# javascript
  eel.get_lang(“it”);
```

2.2. get_mess

súbor:

ASR_app.py

volanie:

get_mess(code, type_mess)

vstupy:

code:

je key do súboru messge.py, odkiaľ sa vyberie hláška, ktorú chceme zobraziť

type_mess:

0 pre error a 1 pre succes

výstupy:

JSON:

{'message': ‘message’, 'type': type_mess}

opis:

funkcia podľa jazyka vyberie hlášku zo lib/message.py, tu pošle do js a k nej typ správy. Pri neznámom jazyku vráti eng a zaloguje chybu.

príklad:

```
# javascript
  eel.get_mess("noModelsOrSamples", 0);
```

2.3. get_data

súbor:

ASR_app.py

volanie:

get_data()

výstupy:

súbor:

dict key value {"files": samples_list, "models": models_list}

2.4. print_log

súbor:

ASR_app.py

volanie:

print_log(x)

vstupy:

x:

message log

príklad:

```
# python
  print_log(“messge”)
```

```
#javascript
  eel.print_log(“messge”);
```

2.5. select_files

súbor:

AS_app.py

volanie:

select_files(file_type, sample_type)

vstupy:

file_type:

typ súborov, ktoré chceme pridať, teda “models” alebo všeličo iné (pridajú sa vzorky)

sample_type:

typ vložených vzoriek (none, EVOO, LOO, VOO)

výstupy:

JSON:

{"files": samples_list, "models": models_list})

opis:

podľa typu sa spustí načítanie modelov alebo vzoriek.

2.6. save_file

súbor:

ASR_app.py

volanie:

save_file(url)

vstupy:

url:

cesta k obrázku, ktorý je zobrazený v aplikácii.

výstupy:

message o uložený súboru

2.7. save_export

súbor:

ASR_app.py

volanie:

save_export(sample)

vstupy:

sample:

názov vzorky, ktorú chceme exportovať, ak príde prázdny reťazec, uložia sa všetky.

výstupy:

message o úspešnom exporte

opis:

funkcia vytvorí txt súbor, s jednotlivými údajmi o klasifikovanej vzorke.

príklad:

```
# javascript
  save_export(“”);

  save_export(“sample.csv”);
```

2.8. classification

súbor:

ASR_app.py

volanie:

classification(JSON_data)

vstupy:

JSON_data:

models:

pole modelov

samples:

pole vzoriek

výstupy:

JSON:

pole vyhodnotení, jednotlivých vzoriek

[názov, názov súboru, výsledok, [názov modelu, výsledok modelu]]

opis:

funkcia vyberie vzorky a modely z JSON objektu, pošle ich do klasifikácie, ktorá vráti výsledky. Výsledky sa uložia do výstupného poľa.

príklad:

```
# javascript
  eel.classification({“samlpes”:[name, file_name], “models”:[model_file_name]});
```

2.9. training

súbor:

ASR_app.py

volanie:

training(JSON_data)

vstupy:

JSON_data:

models:

pole modelov

samples:

pole vzoriek

opis:

funkcia vyberie vzorky a modely z JSON objektu, natrénuje nové modely na vyraných dátach. Natrénovaný model sa uloží do priečinka k ostatným modelom.

príklad:

```
# javascript
  eel.training({“samlpes”:[name, file_name], “models”:[model_file_name]});
```

2.10. get_samples_list

súbor:

ASR_app.py

volanie:

get_samples_list( )

výstupy:

JSON:

pole vzoriek, ktoré sa globálne ukladajú počas behu

2.11. get_picture

súbor:

ASR_app.py

volanie:

get_picture(name, sample)

vstupy:

name:

názov vzorky, v podstate id

sample:

názov súboru

výstupy:

JSON:

name:

iba sa vráti name aké prišlo

sample:

názov súboru

path:

cesta k súboru od koreňového umiestnenia index.html

opis:

zavolá sa funkcia na vytvorenie obrázka, vrátia sa všetky potrebné informácie k zobrazeniu obrázka.

príklad:

```
# javascript
  eel.get_picture(name, sample);
```

2.12. get_conf

súbor:

ASR_app.py

volanie:

get_conf(key)

vstupy:

key:

hodnota kľúča, ktorú chceme vytiahnuť z conf

výstupy:

ak sa kľúč nájde v conf, vráti sa hodnota, ak nie tak sa vráti prázdny reťazec a zaloguje sa chyba.

príklad:

```
# python
  get_conf(“lang”)

# javascript
  eel.get_conf(“lang”);
```

2.13. set_conf

súbor:

ASR_app.py

volanie:

set_conf(key, value)

vstupy:

key:

kľúč k hodnote v conf

value:

hodnota ukrytá pod kľúčom

opis:

Ak potrebujeme prepísať conf, pošleme kľúč a hodnotu, funkcia si načíta celý conf a prepíše hodnotu, ktorú potrebuje, alebo pridá novú. Upravený conf sa opäť zapíše.

príklad:

```
# python
  set_conf(“lang”, “sk”)

# javascript
  eel.set_conf(“lang”, “sk”);
```

2.14. validate_and_get_class_from_csv

súbor:

ASR_app.py

volanie:

validate_and_get_class_from_csv(sample_path)

vstupy:

sample_path:

cesta k súboru

opis:

funkcia načíta scv do pandasu, pozrie či obsahuje hlavičku, ak áno vráti meno triedy, ak nie vrati -1.

2.15. write_samples(samples)

súbor:

ASR_app.py

volanie:

write_samples(sample)

vstupy:

sample:

kľúč k hodnote v conf

opis:

na vstupe dostane pole vzoriek, ktoré funkcia prejde a zapíše do sample_listu.

2.16. Loading screen

import:

from lib.loading_screen_controller import LoadingScreenPercentage

volanie:

lsp = LoadingScreenPercentage()

2.16.1. init_run

súbor:

lib/loading_screen_controller.py

volanie:

lsp.init_run(number_of_itterations)

vstupy:

number_of_itterations:

počet iterácií, ktoré má spraviť

opis:

inicializuje kruhový loading bar.

2.16.2. iterate_run

súbor:

lib/loading_screen_controller.py

volanie:

lsp.iterate_run()

opis:

funkcia zabezpečí iteráciu a včasné ukončenie runu. Beh ukončí po naplnení všetkých iterácií, ktoré boli na začiatku inicializované.

2.17. iter_lsp

súbor:

ASR_app.py

volanie:

iter_lsp

opis:

Zabezpečí volanie iterácie v súbore ASR_app.py. Keby sa funkcia volala tradičným spôsobom, vznikali by zacyklené importy.

3. Python lib

3.1. languages.py

súbor:

lib/languages.py

volanie:

svk_lang()
eng_lang()

výstupy:

JSON:

{key: value}

3.2. message.py

súbor:

lib/message.py

volanie:

svk_message()
eng_message()

výstupy:

dict:

{key: value}