Prosím doplňať dokumentáciu

Predspracovanie

Spracovanie jedného oleja

súbor: Predprocessing/convert_functions.py
volanie: convert_oil(file_name, output, head):

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)

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

príklad:
```
convert_oil("data/Grupo C-LOO/200428_071902/Spectra.POSITIVE.txt",
            "converted.csv",
            "LOO,200428_071902")
```

Spracovanie viacerých olejov

súbor: Predprocessing/convert.py
volanie: convert_multiple_oils(path, limit_sample, oil_type, output)

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

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.

Príklad:
```
out = open(SAVE_PATH, "w")
print_header(out)

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)
```

Overenie chýbajúcich spektier

súbor: Predprocessing/check_missing_spectra.py
volanie: add_missing_spectra(df)

df: dataframe súboru, v ktorom chceme overiť, či má všetky spektrá
output: súbor, ktorý obsahuje všetky spektrá pre každý olej

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.

Vytvorenie hlavičky súboru

súbor: Predprocessing/convert_functions.py
volanie: print_header(output)

output: otvorený súbor csv pre výpis

Vypíše hlavičku spracovaných dát do poskytnutého súboru. Hlavička obsahuje názvy stĺpcov:
"class,oil_id,spectrum,drift_field_intensity,pressure,temperature,drift_tube_length,analysis_time" a následne t_0-t_448 v_0-v_448

Normalizácia

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

Spracovanie normalizácie (fit a transform)

súbor: Predprocessing/normalization.py
volanie: df = normalize(df):

df: dataframe s dátami

Funkcia z priloženého dataframu odseparuje stĺpce v_xy, nad ktorými vykoná fit a transform normalizácie. Upravené stĺpce nahradí v pôvodnom dataframe a odstráni stĺpec v_448. Funkcia vracia takto upravený dataframe

príklad:
```
df = normalize(df)
```

Spracovanie iba transformácie

súbor: Predprocessing/normalization.py
volanie: df = transform_only(df):

df: dataframe s dátami

Funkcia z priloženého dataframu odseparuje stĺpce v_xy, nad ktorými vykoná transformáciu. Upravené stĺpce nahradí v pôvodnom dataframe a odstráni stĺpec v_448. Funkcia vracia takto upravený dataframe v prípade, že bolo možné transformáciu vykonať. V opačnom prípade (nebol nájdený súbor s fit-om) funkcia vráti None.

príklad:
```
df = transform_only(df)
```

Fit normalizácie

súbor: Predprocessing/normalization.py
volanie:normalize_fit(df1):

df: dataframe zložený len z dátových stĺpcov v_xy

Funkcia z priloženého dataframu identifikuje minimálnu a maximálnu hodnotu. Na základe týchto hodnôt sa vykoná fit pre škálovanie stĺpcov na hodnoty v rozmedzí [0,1]. Nafitovaný model sa uloží.

príklad:
```
normalize_fit(df1)
```

Transformácia

súbor: Predprocessing/normalization.py
volanie: df1 = normalize_transform(df1):

df: dataframe zložený len z dátových stĺpcov v_xy

Funkcia nad stĺpcami dataframu vykoná transformáciu na hodnoty v rozmedzí [0,1]. ďalej zavolá funkciu clip_values, ktorá zabezpečí, aby boli všetky hodnoty v rozmedzí [0,1]. Návratom funkcie je takto upravený dataframe. V prípade, že ne nebol nájdený súbor s fit-om funkcia vráti None.

príklad:

```
df1 = normalize_transform(df1)
```

Odstránanie stĺpca v_448

súbor: Predprocessing/normalization.py
volanie: df = delete_v_448_column(df):

df: dataframe s dátami

Funkcia odstráni stĺpec v_448 ak sa v df taký stĺpec nachádza. Návratom funkcie je upravený alebo pôvodný dataframe na vstupe v závislosti od toho či df obsahoval stĺpec v_448.

príklad:
```
df = delete_v_448_column(df)
```

Clip hodnôt

súbor: Predprocessing/normalization.py
volanie: df1 = clip_values(df1):

df1: dataframe zložený len z dátových stĺpcov v_xy

Funkcia zabezpečí orezanie hodnôt na hodnoty v rozmedzí [0,1].
Návratom hodnoty je upravený dataframe

príklad:
```
df1 = clip_values(df1)
```

Získanie v_xy stĺpcov

súbor: Predprocessing/normalization.py
volanie: df = cut_v_columns(df):

df1: dataframe s dátami

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

príklad:
```
df1 = cut_v_columns(df)
```

Získanie minimálnej a maximálnej hodnoty

súbor: Predprocessing/normalization.py
volanie: df = get_min_max_values(df1):

df1: dataframe zložený len z dátových stĺpcov v_xy

Funkcia z v_xy získa celkovú najmenšiu a najväčšiu hodnotu, ktoré vráti ako návratovú hodnotu funkcie.

príklad:
```
minimum, maximum = get_min_max_values(df1)
```

Práca s črtami

Črty

Priemer

súbor: FeatureSelection/features.py
`value = def mean(df_values, column):`
`df_values`: Dataframe obsahujúci surové predspracované dáta pre práve jeden olej
`column`: Názov stĺpca pre ktorý sa vytvorí priemer
`value`: Výstup funkcie. Vypočítaný priemer

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
```

Maximum

súbor: FeatureSelection/features.py
volanie: max(df_values, column)

df_values: dataframe, tabuľka s hodnotami
column: názov stĺpca (v_0-v_448)

Funkcia na vypočítanie maxima v danom stĺpci.

Minimum

súbor: FeatureSelection/features.py
volanie: min(df_values, column)

df_values: dataframe, tabuľka s hodnotami
column: názov stĺpca (v_0-v_448)

Funkcia na vypočítanie minima v danom stĺpci.

Medián

súbor: FeatureSelection/features.py
volanie: median(df_values, column)

df_values: dataframe, tabuľka s hodnotami
column: názov stĺpca (v_0-v_448)

Funkcia na vypočítanie mediánu v danom stĺpci.

Smerodajná odchýlka

súbor: FeatureSelection/features.py
volanie: std(df_values, column)

df_values: dataframe, tabuľka s hodnotami
column: názov stĺpca (v_0-v_448)

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

Rozdiel medzi maximom a minimom

súbor: FeatureSelection/features.py
volanie: delta(df_values, column)

df_values: dataframe, tabuľka s hodnotami
column: názov stĺpca (v_0-v_448)

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

Súčet

súbor: FeatureSelection/features.py
volanie: sum(df_values, column)

df_values: dataframe, tabuľka s hodnotami
column: názov stĺpca (v_0-v_448)

Funkcia na vypočítanie súčtu hodnôt v danom stĺpci.

Point

súbor: FeatureSelection/features.py
volanie: point(df, attribute)

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

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")

```
[ ]: 1 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)

Diff - odchýlka od šablóny

súbor: FeatureSelection/features.py
volanie: diff(df, oil_type)

d: 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

Funkcia vypočíta celkovú odchýlku od jednotlivých tried.

Postup:

1. Načíta sa šablóna pre olej, prípadne maska
    a. šablóna(DataFrame) bola vytvorená pomocou priemeru hodnôt oleju danej triedy (prvých 10 olejov)
    b. 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
2. Vytvorí sa Diff mapa odčítaním šablóny a aktuálneho oleja
    a. predstavuje rozdiel/odchylku
3. Odstránia sa riadky obsahujúce nan hodnoty (spôsobené rôznym počtom spektier v olejoch)
4. Z hodnôt sa spraví absolútna hodnota, čiže vzdialenosť hodnôt
5. Rozdiely menšie ako určená hranica (threshold=2) sa zamenia za hodnotu 0
6. Ak je nastavené použitie masky, aplikuje sa vynásobením aktuálnej mapy maskou.
7. Funkcia vráti súčet všetkých hodnôt

Príklad:
features.diff(df,”diff_EVOO-mask”)

Vytvorenie tabuľky s črtami

súbor: create_features_df.py
`df_features = def create_features_df(df):`
`df`: Dataframe obsahujúci predspracované dáta
`df_features`: Výstup funkcie. Dataframe obsahujúci údaje o triede k jedinečným id olejov

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
```

Zistenie chýbajúcich čŕt

súbor: FeatureSelection/add_features.py
Súbor obsahuje funkciu na zistenie chýbajúcich hodnôt (features) vo výslednom dataframe.

`def check_missing(df, features_to_chceck, df_features):`
`df` - dataframe obsahujúci predspracované dáta
`features_to_chceck` - pole features, ktoré majú svoje funckie pre ich výpočet
`df_features-` dataframe obsahujúci údaje o triede k jedinečným id olejov a môže už obsahovať aj niektoré features

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

Vytvorenie obrázka z dát

Vytvorenie obrázka

súbor: create_pictures.py
`def create_image(oil_df, directory, oil_id):`
`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

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 prečinka, ktorý označuje triedu oleja, pod menom zadefinovaným ako ID oleja

Lineárna interpolácia farby

súbor: create_pictures.py
`red, blue = def linear_interpolation_color(value):`
`value`: Hodnota konkrétnej bunky z dataframe

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.

Vytvorenie diff mapy

Funkcia na vytvorenie diff mapy

súbor: FeatureSelection/features.py
volanie: diff_map(df1, df2, start_column, end_column)

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

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")

Vygenerovanie diff máp pre jednu triedu

súbor: FeatureSelection/create_diff_maps.py
volanie: same_class_diff(source_file, output_path, size)

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

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)

Vygenerovanie diff máp pre rôzne triedy

súbor: FeatureSelection/create_diff_maps.py
volanie: different_class_diff(source_file1, source_file2, output_path, size)

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

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)

Vygenerovanie diff máp medzi olejom a jeho šablónou

súbor: FeatureSelection/create_diff_maps.py
volanie:template_diff(source_file, template, output_path, size)

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

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

Vygenerovanie šablóny

súbor: FeatureSelection/create_diff_maps.py
volanie: create_template(source_file, output_file)

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

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')

Vygenerovanie masky

súbor: FeatureSelection/create_diff_maps.py
volanie: create_mask(source_file, template, output_path)

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

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

1. Zistia sa oblasti v šablóne, kde sa nachádzajú variabilné hodnoty v rámci olejov jednej triedy.
2. Vytvorí sa základ masky, DataFrame rovnako veľký ako šablóna, obsahujúci len hodnoty = 1
3. 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.

Zistenie hodnôt pre črtu point (hľadanie anomálií)

Aplikovanie hraničnej hodnoty (threshold)

súbor: FeatureSelection/select_point_feature.py
volanie: apply_threshold(data, threshold)

data: DataFrame, na ktorý chceme aplikovať threshold
threshold: hraničná hodnota (odporúčaná hodnota 2)

Funkcia zmení hodnoty menšie ako hraničná hodnota na 0.

Nájdenie oblastí, kde sú hodnoty (threshold)

súbor: FeatureSelection/select_point_feature.py
volanie:get_points(data)

data: DataFrame, pre ktorý chceme nájsť oblasti

Funkcia je používaná na identifikovanie oblastí, kde sú hodnoty. Je používaná pre črtu point na identifikovanie hraníc anomálií.
Ohranienie je vykonané oddelením častí DataFrame pomocou odstránenia nulových riadkov/stĺpcov. Kvôli tomu nemusí oddeliť anomálie blízko seba alebo v niektorých špecifických pozíciách.
Príklad výstupu je na obrázku nižšie

	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

Spriemerovanie diff máp

súbor: FeatureSelection/select_point_feature.py
volanie:average_diff_maps(maps_path)

maps_path: cesta k mapám, ktoré chceme spriemerovať

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.

Získanie hodnôt pre črtu point

súbor: FeatureSelection/select_point_feature.py
volanie:get_diff_points(path, show, threshold)

path: cesta k mapám, na ktorých chceme vykonať analýzu
(napr. "../data/Feature/DiffMaps/templates/EVOO")
show: hodnota True/False podľa toho či chceme ukázať anomálie pomocou heatmapy
threshold: hraničná hodnota, určuje ako veľké/malé anomálie hľadáme, odporúčaná hodnota je 2 pre nenormalizované dáta

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)

Tradičné klasifikátory

Vytvorenie modelu random forest

súbor: StandardClasification/random_forest.py
Súbor obsahuje funkciu na trénovanie klasifikátora random forest.
`def classify_random_forest(df):`
`df` - dataframe s vypočítanými features

Vo funkcii sa delia vzorky na testovacie a trénovacie. Klasifikátor sa trénuje, následne sa uloží model a vyhodnotí klasifikátor. Ak už model existuje použije sa pri trénovaní.

Pri prvom spustení je výstupom natrénovaný model klasifikátoru random forest a zobrazená úspešnosť klasifikátora pomocou metík accuracy, F1 micro a F1 macro skóre. Taktiež sa vypíše matica zámen. Pri ďalších spusteniach je výstupom iba úspešosť.

Vytvorenie modelu decision tree

súbor: StandardClasification/decision_tree.py
Súbor obsahuje funkciu na trénovanie klasifikátora decision tree.
`def classify_decision_tree(df):`
`df` - dataframe s vypočítanými features

Vo funkcii sa delia vzorky na testovacie a trénovacie. Klasifikátor sa trénuje, následne sa uloží model a vyhodnotí klasifikátor. Ak už model existuje použije sa pri trénovaní.

Pri prvom spustení je výstupom natrénovaný model klasifikátoru decision tree a zobrazená úspešnosť klasifikátora pomocou metík accuracy, F1 micro a F1 macro skóre. Taktiež sa vypíše matica zámen. Pri ďalších spusteniach je výstupom iba úspešosť.

Vytvorenie modelu SVM

súbor: StandardClasification/svm.py
Súbor obsahuje funkciu na trénovanie klasifikátora
`def classify_svm(df):`

Vo funkcii sa delia vzorky na testovacie a trénovacie. Klasifikátor sa trénuje, následne sa uloží model a vyhodnotí klasifikátor. Ak už model existuje použije sa pri trénovaní.

Pri prvom spustení je výstupom natrénovaný model klasifikátoru SVM a zobrazená úspešnosť klasifikátora pomocou metík accuracy, F1 micro a F1 macro skóre. Taktiež sa vypíše matica zámen. Pri ďalších spusteniach je výstupom iba úspešosť.

Vytvorenie modelu KNN

súbor: StandardClasification/knn.py
Súbor obsahuje funkciu na trénovanie klasifikátora
`def classify_knn(df):`

Vo funkcii sa delia vzorky na testovacie a trénovacie. Klasifikátor sa trénuje, následne sa uloží model a vyhodnotí klasifikátor. Ak už model existuje použije sa pri trénovaní.

Pri prvom spustení je výstupom natrénovaný model klasifikátoru KNN a zobrazená úspešnosť klasifikátora pomocou metík accuracy, F1 micro a F1 macro skóre. Taktiež sa vypíše matica zámen. Pri ďalších spusteniach je výstupom iba úspešosť.

Vytvorenie modelu naive bayes (gaussian)

súbor: StandardClasification/naive_bayes.py
Súbor obsahuje funkciu na trénovanie klasifikátora
`def classify_naive_bayes(df):`
Vo funkcii sa delia vzorky na testovacie a trénovacie. Klasifikátor sa trénuje, následne sa uloží model a vyhodnotí klasifikátor. Ak už model existuje použije sa pri trénovaní.

Pri prvom spustení je výstupom natrénovaný model klasifikátoru naivný bayes a zobrazená úspešnosť klasifikátora pomocou metík accuracy, F1 micro a F1 macro skóre. Taktiež sa vypíše matica zámen. Pri ďalších spusteniach je výstupom iba úspešosť.

Neurónky a neurónové siete

Multilayer Perceptron

súbor: Neurons/train_mlp.py
`def classify_mlp(df):`
`df` - dataframe obsahujúci nájdené črty
Súbor obsahuje funkciu na trénovanie klasifikátora Multilayer Perceptron (MLP).
Funkcia na vstupe získa dataframe čŕt pre všetky oleje. Z dataframu si uloží pole tried olejov, aby sme na konci mohli vyhodnotiť úspešnosť klasifikátora. Načítané dáta sa náhodne rozdelia na trénovacie a testovacie. Na trénovacích dátach as natrénuje model MLP a následne sa vyhodnotí jeho úspešnosť na testovacích dátach.

VGG

súbor: Neurons/Vgg16Model.py
```
class Vgg16(nn.Module):
      def __init__(self, num_classes)
```
`num_classes`:počet tried, ktoré klasifikujeme

Trieda Vgg16 reprezentuje architektúru konvolučnej neurónovej siete.

súbor: Neurons/train_vgg_net.py
`def classify_vgg16()`
Súbor obsahuje funkciu na trénovanie neurónovej siete Vgg16. Funkcia si načíta obrázky zo súbora 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.

Evaluácia a grafy

Evaluácia klasifikátora

súbor: StandardClasification/evaluation.py
Súbor obsahuje funkciu na vyhodnotenie klasifikátora

```
def evaluate_clf(test_labels, pred_labels, name):
test_labels - skutočné lable z klasifikátora
pred_labels - predikované lable z klasifikátora
name - meno vstupujucého klasifikátora
```

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 a uloží ako obrázok do StandardClasification/charts pod názvom daného klasifikátora vo formáte názov-klasifikátora_matrix.png

Vytvorenie grafu výsledkov pre všetky tradičné klasifikátory

súbor: StandardClasification/create_chart.py
Súbor obsahuje funkciu na vytvorenie grafu výsledkov pre všetky tradičné klasifikátory.

```
def scores_charts(score, file_name=”score”):
scores - výsledky klasifikátora z funckie def evaluate_clf(test_labels,
pred_labels, name):
file_name - názov grafu pod ktorým bude uložený obrázok
```

Funkcia vytvorí graf výsledkov všetkých tradičných klasifikátor pre metriky accuracy, F1 micro a F1 macro.