1 zadanie

1. Teoretická časť

Cieľom zadanie je navrhnúť model pre predikciu/klasifikáciu dať. Je možné využiť modely pre klasifikáciu/regresiu dostupné v rámci modulu scikit-learn.

Máme k dispozícii údaje X_public obsahujúce N riadkov, pričom každá vzorka obsahuje F znakov. Aby sme mohli začať, musíme načítať údaje zo súborov, použil som funkciu np.load(). Prvým krokom bolo klasické rozdelenie public na trénovanie a testovacie data. Druhým krokom bolo prejsť array a nahradiť triedy, ktoré boli String. Použil som fit_transform pre X_train a transform pre X-test.

```
X_train[:, 180:] = enc.fit_transform(X_train [:, 180:])
X_test[:, 180:] = enc.transform(X_test [:, 180:])
```

Po dlhých kontrolách som sa rozhodol ponechať OrdinalEncoder().

```
enc = preprocessing.OrdinalEncoder(unknown_value=np.nan,
                                   handle_unknown = 'use_encoded_value')
```

Použil som rovnaký klasifikátor ako na cvikach SVC().

```
clf = SVC(kernel='rbf', probability=True)

param_range = [0.1, 1, 10]

parameters_grid = [{'C': param_range,
                    'kernel': ['linear']},
                   {'C': param_range,
                    'gamma': param_range,
                    'kernel': ['rbf']},
                   {'C': param_range,
                    'gamma': param_range,
                    'degree': [1,2,3],
                    'kernel': ['poly']}]
```

Ten istý klasifikátor sa použil na strojové učenie.

```
clf = SVC(kernel='poly',
          decision_function_shape='ovr',
          degree=2,
          C=1,
          gamma='auto')
```

Na nahradenie chýbajúcich hodnôt som použil SimpleImputer(). Na predikciu, vyhodnotenie a fit som použil funkciu GridSearchCV.

```
grid_search = GridSearchCV(estimator=clf,
                           param_grid=parameters_grid,
                           scoring='roc_auc',
                           cv=5,
                           n_jobs=-1,
                           verbose=1)
```

Posledným krokom bolo nahradiť triedy typu String triedami X_eval a použiť pre ne SimpleImputer() a StandardScaler(), ako to bolo urobené predtým pre X_public a y_public. A zápis konečného výsledku do súboru y_predikcia.npy.

```
X_eval[:, 180:] = enc.transform(X_eval [:, 180:])
X_eval = input.transform(X_eval)
X_eval = scaler.transform(X_eval)

result = grid_search.predict(X_eval)
np.save('y_predikcia.npy', result)
```

2. Metóda, diskusia

Spočiatku som použil for z try a except, ktorý ručne načítala dátový súbor, našla triedy String a nahradila ich. Na svoju prácu som použil knižnicu pandas a pomocou DataFrame som zhromaždil všetky údaje zo súboru X_public. Vyzeralo to tak:

```
df = pd.DataFrame(X_public)
for col in df:
    try:
        df[col] = df[col].astype(float)
    except:
        l = df[col].unique()
        for i in range(len(l)):
            df[col] = df[col].replace(l[i], i)
```

Ale neskôr som našiel lepšiu možnosť:

```
X_train[:, 180:] = enc.fit_transform(X_train [:, 180:])
X_test[:, 180:] = enc.transform(X_test [:, 180:])
```

Ako vidíte, je tu oveľa menej riadkov kódu a lepší výkon. Zmizol aj problém s nahradením všetkých údajov, kvôli ktorému som napísal prvý for cykl. Dosiahol som s ním aj najvyššiu známku, a to 94 bodov.

Počas celého programovania tejto úlohy som neustále menil encodery v snahe nájsť ten najlepší. Podarilo sa mi to pomocou funkcie OrdinalEncoder(). Najprv som použil OneHotEncoder(), skóre bolo dobré, ale chcel som viac, začal som testovať LabelEncoder(), ale mal problémy s objektmi v súboroch, tento kodér nemohol náhodne prejsť všetky údaje, tak som sa zastavil na OrdinalEncoder(), s jeho pomocou som získal vyššie skóre a všetky údaje boli úspešne, nastavenie nasledujúcich parametrov – (unknown_value=np.nan, handle_unknown = 'use_encoded_value'). Podľa hodnotenia došlo k výraznému nárastu, a to o +5 bodov.

```
enc = preprocessing.OrdinalEncoder(unknown_value=np.nan,
                                   handle_unknown = 'use_encoded_value')
```

Ďalšie rozhodnutie, nad ktorým som dlho váhal, sa týkalo klasifikátora. Mohol som si vybrať medzi SVC a PipeLine. Ku koncu sa mi zdalo, že spoločnosť PipeLine poskytne. Ale SVC sa mi zdalo najlepšie. Ten istý klasifikátor sa použil na strojové učenie. Od učiteľa som si tiež zobral poznámku, že v tejto práci je lepšie nepoužívať funkciu PipeLine().

Hodnotenie. S hodnotením som nemal žiadne problémy, spočiatku som používal rovnaké ako na prvých hodinách a prednáškach. Bolo to pomocou cross_val_score(). Ďalej použil som GridSearchCV(). A pre mňa to bolo najlepšie riešenie z hodnoteniam.