WUM_inzynieria_cech

March 30, 2021

1 [WUM] Inżynieria cech

Wpierw załączamy paczki i dane

[ ]:
```
import pandas as pd
import numpy as np

from sklearn.datasets import load_boston
from sklearn.linear_model import LinearRegression, Lasso
from sklearn.metrics import mean_squared_error
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import FunctionTransformer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split

from matplotlib import pyplot as plt
import seaborn as sns
from scipy import stats
import matplotlib.image as mpimg
```

[3]:
```
df = pd.DataFrame(pd.read_json("https://api.apispreadsheets.com/api/dataset/
↪congressional-voting/",
                    orient = 'split'))
```

[57]:
```
df_encoded = pd.DataFrame( df)
```

[58]:
```
change_dict = {"y": 1, "n": -1, "?": 0, "republican": 0, "democrat": 1}
df_encoded.replace(change_dict, inplace=True)
```

Następnie wyłuskamy tych głosujących, którzy mieli najmniej głosów, niezależnie od tego, czy na tak, czy na nie.

[49]:
```
df_encoded['votes_given'] = df_encoded.drop(["political_party"], axis=1)[:].
↪apply(lambda x: np.sum(np.abs(x)), axis = 0)
```

[66]:
```
df_encoded.head()
```

[66]:

	handicapped_infants	water_project_cost_sharing	adoption_of_the_budget_resolution	physician_fee_freeze	el_salvador_aid	religious_groups_in_schools	anti_satellite_test_ban	aid_to_nicaraguan_contras	mx_missile	immigration	synfuels_corporation_cutback	education_spending	superfund_right_to_sue	crime	duty_free_exports	export_administration_act_south_africa	political_party	no_of_votes
0	-1	1	-1	1	1	1	-1	-1	-1	1	0	1	1	1	-1	1	0	15
1	-1	1	-1	1	1	1	-1	-1	-1	-1	-1	1	1	1	-1	0	0	15
2	0	1	1	0	1	1	-1	-1	-1	-1	1	-1	1	1	-1	-1	1	15
3	-1	1	1	-1	0	1	-1	-1	-1	-1	1	-1	1	-1	-1	1	1	16
4	1	1	1	-1	1	1	-1	-1	-1	-1	1	0	1	1	1	1	1	16

```
4                1           16
```

[65]:
```
df_encoded["no_of_votes"] = df_encoded.apply( lambda x: np.sum( np.abs(x)),␣
↪axis=1)
```

[67]:
```
df_removed = pd.DataFrame( df_encoded)
```

i ich usuniemy. W ten sposób usuwamy tych głosujących, którzy nie mieli zdecydowanego zdania, więc też nie głosowaliby (średnio) tak jak głosowałaby którakolwiek z partii. Nie dają przez to sugestii jak senatorzy z danej partii oddawaliby swoje głosy.

[69]:
```
df_removed = df_removed.drop( df_removed[ df_removed["no_of_votes"] < 6].index )
```

[72]:
```
df_removed.head()
```

[72]:
```
   handicapped_infants  water_project_cost_sharing  \
0                   -1                           1
1                   -1                           1
2                    0                           1
3                   -1                           1
4                    1                           1

   adoption_of_the_budget_resolution  physician_fee_freeze  el_salvador_aid  \
0                                 -1                     1                1
1                                 -1                     1                1
2                                  1                     0                1
3                                  1                    -1                0
4                                  1                    -1                1

   religious_groups_in_schools  anti_satellite_test_ban  \
0                            1                       -1
1                            1                       -1
2                            1                       -1
3                            1                       -1
4                            1                       -1

   aid_to_nicaraguan_contras  mx_missile  immigration  \
0                         -1          -1            1
1                         -1          -1           -1
2                         -1          -1           -1
3                         -1          -1           -1
4                         -1          -1           -1

   synfuels_corporation_cutback  education_spending  superfund_right_to_sue  \
0                             0                   1                       1
1                            -1                   1                       1
2                             1                  -1                       1
3                             1                  -1                       1
```

```
4                              1                    0                         1

   crime  duty_free_exports  export_administration_act_south_africa  \
0      1                 -1                                      1
1      1                 -1                                      0
2      1                 -1                                     -1
3     -1                 -1                                      1
4      1                  1                                      1

   political_party
0                0
1                0
2                1
3                1
4                1
```

[71]:
```
del df_removed["no_of_votes"]
```

[74]:
```
df_selected = pd.DataFrame( df_removed)
```

[76]:
```
y = np.array(df_selected['political_party'])
X = df_selected.drop(['political_party'],axis=1)
```

Na tak zmienionym zbiorze danych przetrenujmy trzy proste modele

[77]:
```
from sklearn.model_selection import train_test_split

X_train, X_val, y_train, y_val = train_test_split(
    X, y, stratify=y, test_size=0.3, random_state=42
)
X_val, X_test, y_val, y_test = train_test_split(
    X_val, y_val, stratify=y_val, test_size=0.3, random_state=42
)
```

[81]:
```
from sklearn.dummy import DummyClassifier
from sklearn.metrics import accuracy_score
```

[79]:
```
dc = DummyClassifier(strategy='uniform', random_state=42)
dc.fit(X_train,y_train)
y_proba = dc.predict_proba(X_val)
y_hat = dc.predict(X_val)
print("proba: " + str(y_proba[0:10,0]) + '\ny:      ' + str(y_hat[0:10]) +␣
↪'\ny_hat: ' + str(y_val[0:10]))
```

```
proba: [0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5]
y:      [0 1 0 0 0 1 0 0 0 1]
y_hat: [0 1 1 1 1 0 1 0 0 1]
```

[89]:
```
accuracy_score(y_val, y_hat)
```

[89]:
```
0.42857142857142855
```

[85]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(max_iter=1000)

lr.fit(X_train,y_train)
y_hat_lr = lr.predict(X_val)
print('y:     ' + str(y_hat_lr[0:10]) + '\ny_hat: ' + str(y_val[0:10]))
```

```
y:     [0 1 1 1 1 0 1 0 0 1]
y_hat: [0 1 1 1 1 0 1 0 0 1]
```

[87]:
```
accuracy_score(y_val, y_hat_lr)
```

[87]:
```
0.9560439560439561
```

[90]:
```
from sklearn.svm import SVC
svm = SVC()

svm.fit(X_train,y_train)
y_hat_svm = svm.predict(X_val)
print('y:     ' + str(y_hat_svm[0:10]) + '\ny_hat: ' + str(y_val[0:10]))
```

```
y:     [0 1 1 1 1 0 1 0 0 1]
y_hat: [0 1 1 1 1 0 1 0 0 1]
```

[91]:
```
accuracy_score(y_val, y_hat_svm)
```

[91]:
```
0.945054945054945
```

Jak widzimy, największe accuracy uzyskaliśmy dla regresji logistycznej, niedaleko potem jest svm, zaś najgorszy jest dummy classifier.