CongerssionalVotingFirstModel

March 30, 2021

1 Congerssional Voting First Model Atempt

1.1 *Bartosz Sawicki, Hubert Ruczyński*

[1]:
```
import numpy as np
import pandas as pd
import requests
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

from matplotlib import pyplot as plt
plt.style.use('ggplot')

from sklearn import metrics
from sklearn import tree
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import cross_validate
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.pipeline import Pipeline

from xgboost import XGBClassifier # Inna paczka niż sklearn!
```

2 Wczytanie Danych

[2]:
```
url = 'https://api.apispreadsheets.com/api/dataset/congressional-voting/'

r = requests.get(url)
data = r.json()
df = pd.DataFrame.from_dict(data['data'], orient='columns')
df.sample(5)
```

[2]:
```
    handicapped_infants water_project_cost_sharing  \
119                   n                          n
342                   n                          y
```

```
24                    y                          n
405                   n                          n
290                   y                          n

    adoption_of_the_budget_resolution physician_fee_freeze el_salvador_aid  \
119                                 n                    y               y
342                                 y                    n               y
24                                  y                    n               n
405                                 n                    y               y
290                                 y                    n               ?

    religious_groups_in_schools anti_satellite_test_ban  \
119                           y                       n
342                           ?                       y
24                            n                       y
405                           y                       n
290                           y                       ?

    aid_to_nicaraguan_contras mx_missile immigration  \
119                         n          n           n
342                         n          n           y
24                          y          y           n
405                         n          n           n
290                         y          y           y

    synfuels_corporation_cutback education_spending superfund_right_to_sue  \
119                            n                  y                      y
342                            y                  n                      y
24                             n                  n                      n
405                            n                  y                      y
290                            n                  n                      y

    crime duty_free_exports export_administration_act_south_africa  \
119     y                 n                                      n
342     n                 y                                      y
24      n                 y                                      ?
405     y                 n                                      y
290     y                 n                                      y

    political_party
119      republican
342        democrat
24         democrat
405      republican
290        democrat
```

3 Autorski Encoding

[3]:
```
val = {'n':-1, '?':0, 'y':1}
party = {'democrat':2, 'republican':-2}

df_num=df.copy()
df_party = df_num['political_party'].copy()
df_num.drop('political_party', axis=1, inplace=True)

for column in df_num.columns:
  df_num[column] = df_num[column].map(val)

df_party = df_party.map(party)

df_num = pd.concat([df_num,df_party], axis=1)
df_num.head()
```

[3]:
```
   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               -2
1               -2
2                2
3                2
4                2
```

4 Informacje o Danych

[4]:
```python
x = df_num.drop('political_party', axis=1)
y = df_num.iloc[:,16]
x.info()
y
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 435 entries, 0 to 434
Data columns (total 16 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   handicapped_infants                     435 non-null    int64
 1   water_project_cost_sharing              435 non-null    int64
 2   adoption_of_the_budget_resolution       435 non-null    int64
 3   physician_fee_freeze                    435 non-null    int64
 4   el_salvador_aid                         435 non-null    int64
 5   religious_groups_in_schools             435 non-null    int64
 6   anti_satellite_test_ban                 435 non-null    int64
 7   aid_to_nicaraguan_contras               435 non-null    int64
 8   mx_missile                              435 non-null    int64
 9   immigration                             435 non-null    int64
 10  synfuels_corporation_cutback            435 non-null    int64
 11  education_spending                      435 non-null    int64
 12  superfund_right_to_sue                  435 non-null    int64
 13  crime                                   435 non-null    int64
 14  duty_free_exports                       435 non-null    int64
```

```
 15  export_administration_act_south_africa  435 non-null    int64
dtypes: int64(16)
memory usage: 54.5 KB
```

[4]:
```
0     -2
1     -2
2      2
3      2
4      2
      ..
430   -2
431    2
432   -2
433   -2
434   -2
Name: political_party, Length: 435, dtype: int64
```

5 Perwszy model: SVM

Model: https://scikit-learn.org/stable/modules/svm.html#classification

Za pierwszym razem nie dokonaliśmy selekcji zmiennych i z tego powodu otrzymaliśmy dość nieskuteczny model.

Acccuracy powyżej 0,9 może być mylące gdyż na pierwszy rzut wydaje się, że jest to wynik bardzo dobry, jednak po analizie której dokonywaliśmy ostatnio, wiemy, że pojedyncze głosowanie (physicians_fee) jest w stanie przewidywać z dokładnością prawie 94%.

[5]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y)
```

[6]:
```
from sklearn import svm
clf = svm.SVC()
clf.fit(x_train, y_train)
y_train_pred=clf.predict(x_train)
```

[7]:
```
y_pred=clf.predict(x_test)
```

[8]:
```
print('\ny:     ' + str(y_test[0:10]) + '\ny_pred: ' + str(y_pred[0:10]))
sum=0
y_tezt=y_test.to_numpy()
for i in range(len(y_pred)):
    if y_pred[i]==y_tezt[i]:
        sum+=1
accuracy=sum/len(y_pred)
print("validation : " + str(accuracy))
sum=0
y_trained=clf.predict(x_train)
y_trains=y_train.to_numpy()
```

```
for i in range(len(y_trained)):
    if y_trained[i]==y_trains[i]:
        sum+=1
accuracy_train=sum/len(y_trained)
print("train : " + str(accuracy_train))
```

```
y:     59    -2
72      2
429     2
153     2
282    -2
143     2
402    -2
198     2
103     2
23      2
Name: political_party, dtype: int64
y_pred: [-2  2  2  2 -2  2 -2  2  2  2]
validation : 0.944954128440367
train : 0.9846625766871165
```

Aby sprawdzić faktyczne accuracy, postanowiliśmy zapętlić wybór zbioru testowego, tworzenia i trenowania modelu 1000 razy, aby otrzymać średnie accuracy dla tych prób.

Ponadto wykonaliśmy selekcji zmiennych przy tym testowaniu w dwóch wariantach i okazało się, że lepiej wypadł ten przy mniejszej liczbie wyrzucanych kolumn.

5.1 Uwaga!

Ilość testowanych modeli i zbiory wyrzucanych kolumn były znacznie większe w fazie pisania raportu, jednak postanowiliśmy zostawić tylko te które coś wnoszą.

[9]:
```
accumulated_accuracy=0
accumulated_accuracy_train=0
for i in range(1000):
    x = df_num.drop(['political_party',␣
 ↪'water_project_cost_sharing','immigration','export_administration_act_south_africa'],␣
 ↪axis=1)
    y = df_num.iloc[:,16]
    x_train, x_test, y_train, y_test = train_test_split(x, y)
    clf = svm.SVC()
    clf.fit(x_train, y_train)
    y_train_pred=clf.predict(x_train)
    y_pred=clf.predict(x_test)
    sum=0

    y_tezt=y_test.to_numpy()
    for i in range(len(y_pred)):
```

```
        if y_pred[i]==y_tezt[i]:
            sum+=1
    accuracy=sum/len(y_pred)
    accumulated_accuracy=accumulated_accuracy+accuracy
    sum=0
    y_trained=clf.predict(x_train)
    y_trains=y_train.to_numpy()
    for i in range(len(y_trained)):
        if y_trained[i]==y_trains[i]:
            sum+=1
    accuracy_train=sum/len(y_trained)
    accumulated_accuracy_train=accumulated_accuracy_train+accuracy_train

print("accumulated accuracy validation : " + str(accumulated_accuracy/1000))
print("accumulated accuracy train : " + str(accumulated_accuracy_train/1000))
```

```
accumulated accuracy validation : 0.9565229357798124
accumulated accuracy train : 0.9793343558282152
```

[10]:

```
accumulated_accuracy=0
accumulated_accuracy_train=0
for i in range(1000):
    x = df_num.drop(['political_party',␣
↪'water_project_cost_sharing','immigration','export_administration_act_south_africa',
                    'education_spending',␣
↪'synfuels_corporation_cutback','religious_groups_in_schools'], axis=1)
    y = df_num.iloc[:,16]
    x_train, x_test, y_train, y_test = train_test_split(x, y)
    clf = svm.SVC()
    clf.fit(x_train, y_train)
    y_train_pred=clf.predict(x_train)
    y_pred=clf.predict(x_test)
    sum=0

    y_tezt=y_test.to_numpy()
    for i in range(len(y_pred)):
        if y_pred[i]==y_tezt[i]:
            sum+=1
    accuracy=sum/len(y_pred)
    accumulated_accuracy=accumulated_accuracy+accuracy
    sum=0
    y_trained=clf.predict(x_train)
    y_trains=y_train.to_numpy()
    for i in range(len(y_trained)):
        if y_trained[i]==y_trains[i]:
            sum+=1
    accuracy_train=sum/len(y_trained)
```

```
    accumulated_accuracy_train=accumulated_accuracy_train+accuracy_train

print("accumulated accuracy validation : " + str(accumulated_accuracy/1000))
print("accumulated accuracy train : " + str(accumulated_accuracy_train/1000))
```

```
accumulated accuracy validation : 0.9484403669724762
accumulated accuracy train : 0.9693190184049109
```

Na autorskim kodowaniu postanowiliśmy też wykorzystać bardziej złożony model w postaci XGB-Classifier'a, który zapewnił nam o wiele wyższe accuracy od poprzedników.

[11]:
```
x = df_num.drop(['political_party',␣
 ↪'water_project_cost_sharing','immigration','export_administration_act_south_africa'],␣
 ↪axis=1)
y = df_num.iloc[:,16]
```

[12]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=14)
```

[13]:
```
model=XGBClassifier(random_state=2,
                    learning_rate=0.01, # Szybkość "uczenia" się
                    booster='gbtree', # Jaki model wykorzystujemy (drzewo -␣
 ↪gbtree, liniowe - gblinear)
                    nround = 1000, # Ilość itereacji boosingowych
                    max_depth=3, # Maksymalna głębokość drzewa
                    verbosity = 0 # Nie chcemy warningów
                    )
model.fit(x_train, y_train)
prediction_test=model.predict(x_test)
print(model.score(x_test,y_test))
print(metrics.classification_report(y_test, prediction_test))
```

```
0.963302752293578
              precision    recall  f1-score   support

          -2       0.94      0.98      0.96        46
           2       0.98      0.95      0.97        63

    accuracy                           0.96       109
   macro avg       0.96      0.97      0.96       109
weighted avg       0.96      0.96      0.96       109
```

6 Kodowanie one hot

Aby uzyskać jak najlepszy model użyliśmy jednak one-hot encodingu, ze względu na to, że jest to lepsza wersja tego co sami robimy w naszym kodowaniu.

[14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 435 entries, 0 to 434
Data columns (total 17 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   handicapped_infants                     435 non-null    object
 1   water_project_cost_sharing              435 non-null    object
 2   adoption_of_the_budget_resolution       435 non-null    object
 3   physician_fee_freeze                    435 non-null    object
 4   el_salvador_aid                         435 non-null    object
 5   religious_groups_in_schools             435 non-null    object
 6   anti_satellite_test_ban                 435 non-null    object
 7   aid_to_nicaraguan_contras               435 non-null    object
 8   mx_missile                              435 non-null    object
 9   immigration                             435 non-null    object
 10  synfuels_corporation_cutback            435 non-null    object
 11  education_spending                      435 non-null    object
 12  superfund_right_to_sue                  435 non-null    object
 13  crime                                   435 non-null    object
 14  duty_free_exports                       435 non-null    object
 15  export_administration_act_south_africa  435 non-null    object
 16  political_party                         435 non-null    object
dtypes: object(17)
memory usage: 57.9+ KB
```

Modele działać, będą na dwóch ramkach, jedna cała, bez selekcji zmiennych i druga z odrzuceniem tych które są w bardzo niskim stopniu skorelowane z targetem.

[15]:
```
df_dropped = df.
 ↪drop(['water_project_cost_sharing','immigration','export_administration_act_south_africa'],
 ↪axis=1)
X_dropped = df_dropped.iloc[:,:12]
y_dropped = df_dropped.iloc[:,13]

X = df.iloc[:,:15]
y = df.iloc[:,16]
```

[16]:
```
x_train, x_test, y_train, y_test = train_test_split(df_dropped.iloc[:,:12],␣
 ↪df_dropped.iloc[:,13], random_state=43)
```

[17]:
```
import category_encoders as ce
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.pipeline import Pipeline

one_hot_encoder = ce.OneHotEncoder()

one_hot = one_hot_encoder.fit_transform(X,y)
```

```
one_hot_encoder_dropped = ce.OneHotEncoder()
one_hot_dropped = one_hot_encoder_dropped.fit_transform(X_dropped,y_dropped)
```

7 GradientBoostingClassifier

[18]:
```
clf = GradientBoostingClassifier(n_estimators=1000, learning_rate=.05,
    max_depth=3, random_state=997)

pipe_one_hot = Pipeline(
[
    ('transformer_one_hot', one_hot_encoder),
    ('classifier', clf)
])

pipe_one_hot_dropped = Pipeline(
[
    ('transformer_one_hot', one_hot_encoder_dropped),
    ('classifier', clf)
])
```

[19]:
```
print(np.mean(cross_validate(pipe_one_hot, X, y, cv=11, scoring='accuracy').
 ↪get('test_score')))
print(np.mean(cross_validate(pipe_one_hot_dropped, X_dropped, y_dropped, cv=11,␣
 ↪scoring='accuracy').get('test_score')))
```

```
0.9493589743589744
0.9491841491841492
```

8 XGBClassifier

[20]:
```
model=XGBClassifier(random_state=1,
                    learning_rate=0.01, # Szybkość "uczenia" się
                    booster='gbtree', # Jaki model wykorzystujemy (drzewo -␣
 ↪gbtree, liniowe - gblinear)
                    nround = 1000, # Ilość itereacji boosingowych
                    max_depth=3, # Maksymalna głębokość drzewa
                    verbosity = 0
                    )
XGB_one_hot = Pipeline(
[
    ('transformer_one_hot', one_hot_encoder),
    ('classifier', model)
])

XGB_one_hot_dropped = Pipeline(
```

```
[
    ('transformer_one_hot', one_hot_encoder_dropped),
    ('classifier', model)
])
```

[21]:
```
print(np.mean(cross_validate(XGB_one_hot, X, y, cv=11, scoring='accuracy').
 ↪get('test_score')))
print(np.mean(cross_validate(XGB_one_hot_dropped, X_dropped, y_dropped, cv=11,␣
 ↪scoring='accuracy').get('test_score')))
```

```
0.9515734265734267
0.9561188811188811
```

9 AdaBoostClassifier

[22]:
```
model = AdaBoostClassifier(random_state=1,learning_rate=0.1,algorithm ='SAMME.
 ↪R')
Ada_one_hot = Pipeline(
[
    ('transformer_one_hot', one_hot_encoder),
    ('classifier', model)
])

Ada_one_hot_dropped = Pipeline(
[
    ('transformer_one_hot', one_hot_encoder_dropped),
    ('classifier', model)
])
```

[23]:
```
print(np.mean(cross_validate(Ada_one_hot, X, y, cv=7, scoring='accuracy').
 ↪get('test_score')))
print(np.mean(cross_validate(Ada_one_hot_dropped, X_dropped, y_dropped, cv=7,␣
 ↪scoring='accuracy').get('test_score')))
```

```
0.951649476995099
0.9562577719259747
```

9.1 *Uwaga!*

Ze względu na istotę problemu, czyli wybór między dwoma ugrupowaniami politycznymi, jedyną interesującą nas miarą jest accuracy, gdyż dla nas jest to bez różnicy czy pomylimy się mówiąc, że republikanin jest demokratą, czy też na odwrót.

10 TO DO:

Zamierzamy teraz skupić się na dwóch najlepiej zapowiadających się modelach, czyli AdaBoostClassifier i XGBClassifier. Chcemy poprawić ich parametry w taki sposób, aby otrzymać jak najlepsze wyniki.