WUM Projekt 1

Congressional voting - temat 3

Grupa 3 - Patryk Słowakiewicz, Kacper Kurowski

Opis zbioru danych

Opis zbioru danych

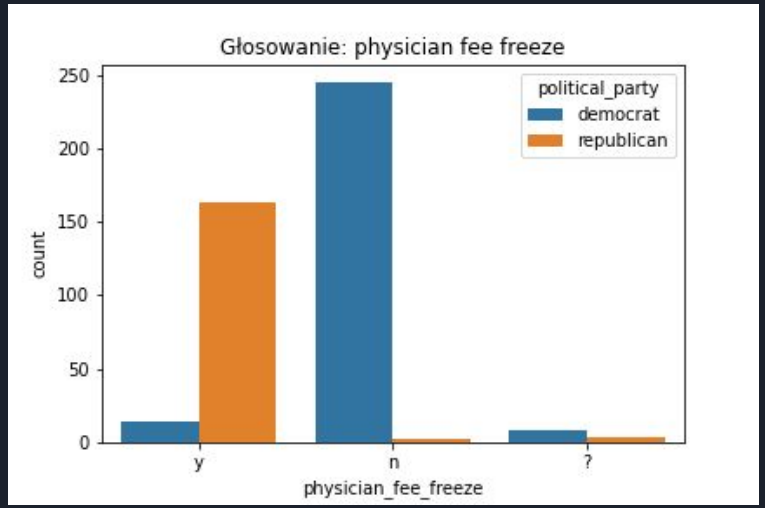
1. Obserwacjami w zbiorze danych są głosy kongresmenów w 16 głównych (według CQA) głosowaniach z 1986,
2. Każdy wiersz odpowiada głosom dokładnie jednego kongresmena, (i wierszy jest tyle ile kongresmenów),
3. CQA dopuszcza 9 sposobów zagłosowania - zbiór danych zamienia je na 3, z grubsza odpowiadające głosom: na tak (‘y’), na nie (‘n’), oraz wstrzymanie się (‘?’).
4. W ostatniej kolumnie jest informacja do jakiej partii dany kongresmen należy - celem jest stworzenie modelu przewidujacego właśnie tę informację.

Eksploracja danych

Zmienne dzielące

Podczas automatycznej eksploracji danych zauważyliśmy kilka głosowań, które miały silną korelację z przynależeniem do danej partii - są to:

- ‘Physician fee freeze’,
- ‘Education spending’,
- ‘El salvador aid’,
- ‘Crime’

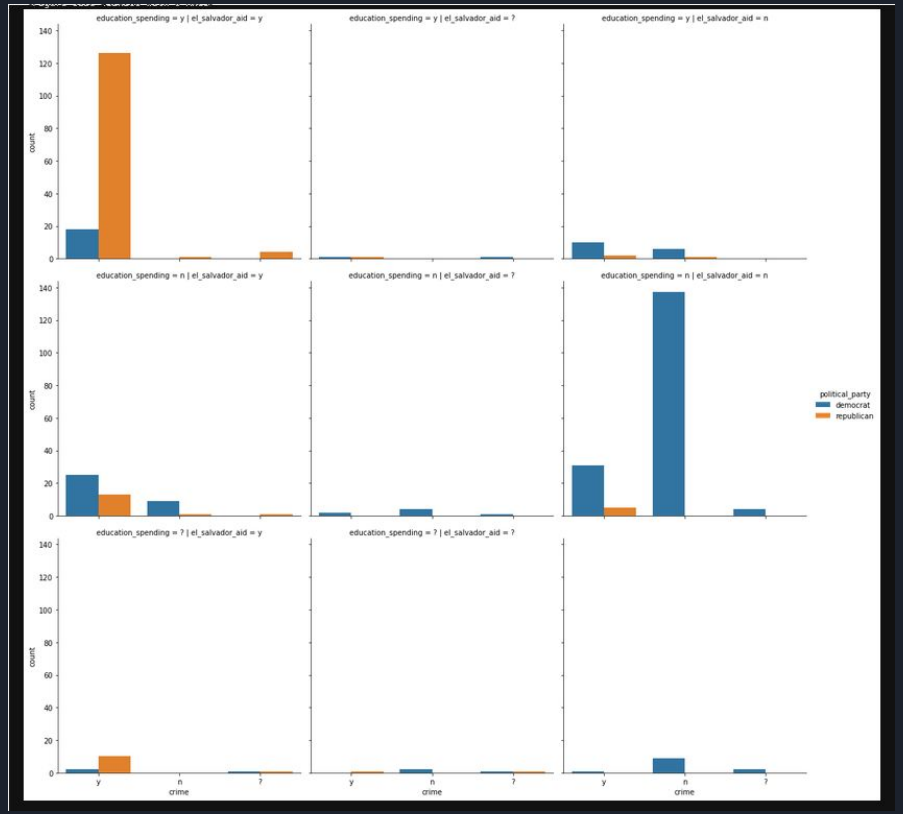


Kombinacje zmiennych

Nie tylko physician fee freeze dobrze rozdziela nasz zbiór ponieważ istnieją również znaczące zależności jeśli pod uwagę weźmiemy parę zmiennych na raz. Jak na wykresie po prawej widać że prawie wszyscy Republikanie głosują za we wszystkich trzech głosowaniach

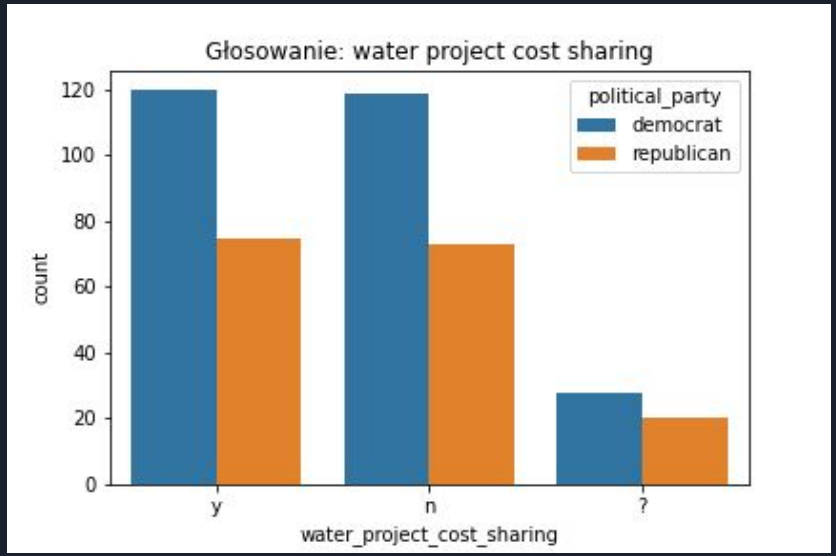
- el salvador aid (kolumny),
- education spending (wiersze),
- crime (kolumny w komórkach)

Prawie wszyscy demokraci za to głosują przeciw powyższych głosowaniach, gdzie *crime* jest głosowaniem w którym są najmniej jednomyślni.



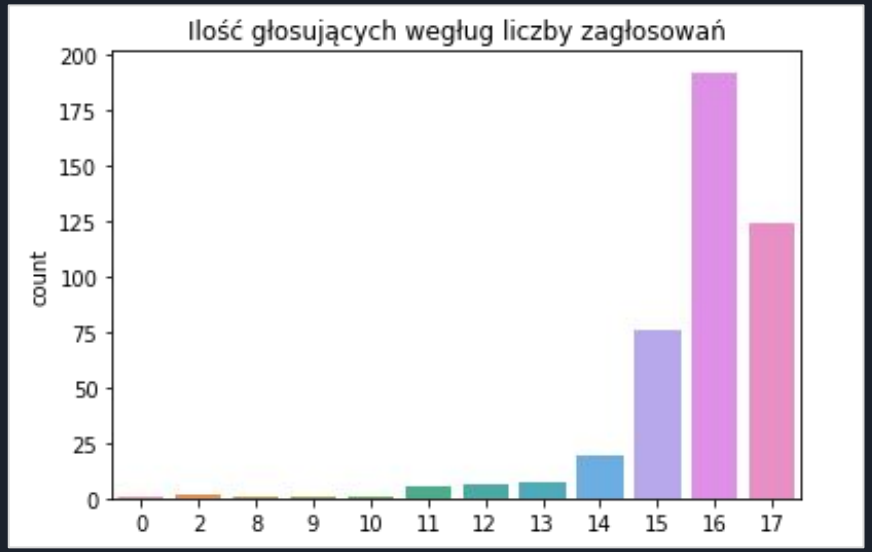
Eksploracja danych

Były również głosowania, które miały wręcz zerową korelację z przynależnością partyjną kongresmenów - przykładowo, ‘water project cost sharing’. Ale również: 'immigration' oraz 'export_administration_act_south_africa'. Można by je uznać za zbędne w naszym modelu ale okazuje się, że wyniki modelowania są lepsze jeśli pozostawimy te trzy kolumny. Może to być spowodowane zależnościami zmiennych których nie byliśmy wstanie wyłapać w czasie EDA.



Eksploracja danych

Znaleźliśmy również kilku kongresmenów, którzy oddawali swój głos niezwykle rzadko (większość obserwacji w ich wierszu to ‘?’). Można uznać ich za formę outlierów.



Przekształcanie danych

Przekształcanie danych

Jedynymi zmian, jakich dokonaliśmy w celu uzyskania finalnego modelu były:

1. Zakodowanie (“y”, “?”, “n”, “democrat”, “republican”) |-> (1, 0, -1, 1, -1)
2. Usunięcie duplikatów

Przy testowaniu innych zmian (patrz Dodatkowe Uwagi) nie uzyskaliśmy poprawy działania modeli - stąd nie zostały one włączone do finalnego modelu.

Stworzone modele

Baseline

Traktując zmienną physician_fee_freeze jako baseline do klasyfikacji uzyskaliśmy Accuracy:

- 85% na walidacji,
- 95% na teście.

```
5]: def encode_base(x):
        if x==-1:
            return 1
        if x== 1:
            return -1
        if x == 0:
            return 0

6]: y_val_base = X_val["physician_fee_free

7]: accuracy_score( y_val, y_val_base)

7]: 0.855072463768116

8]: accuracy_score( y_test, X_test["physic

8]: 0.9565217391304348
```

Stworzone modele

Wśród modeli, które testowaliśmy są:

- AdaBoostClassifier,
- GradientBoostingClassifier,
- HistGradientBoostingRegressor,
- Random Forest,
- Tree,
- Bayes,
- Log Reg

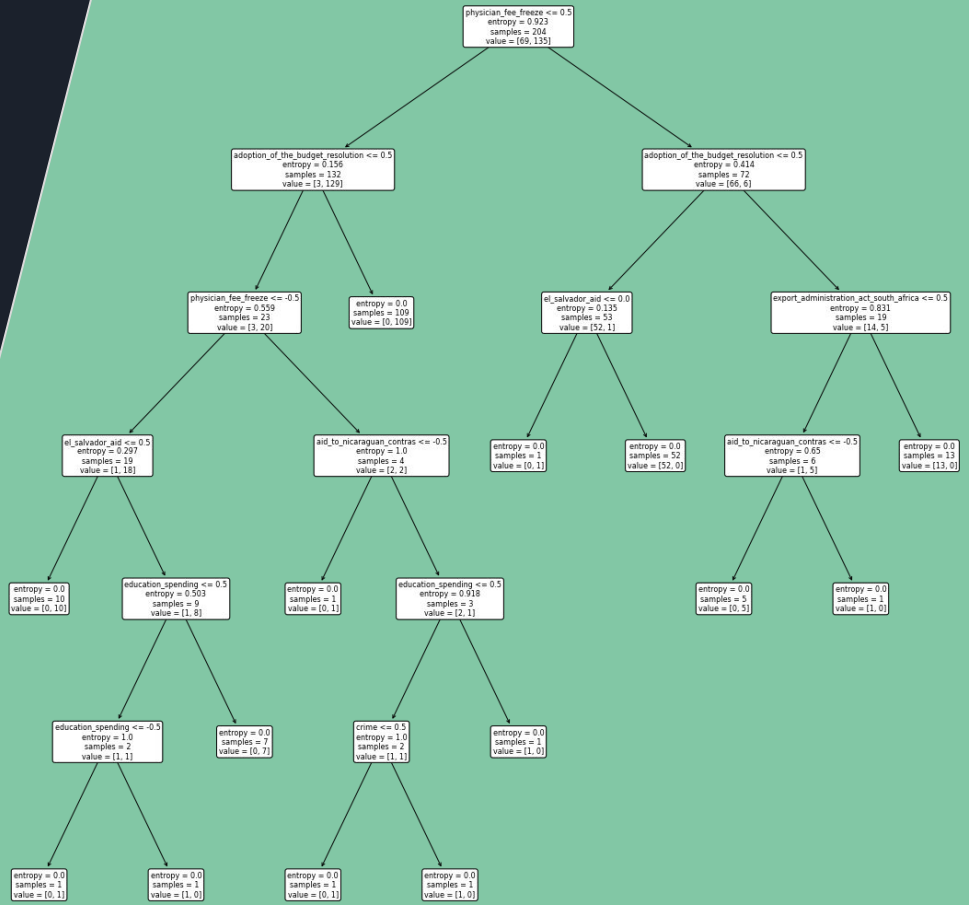
Miarą walidacji naszych modeli jest accuracy. Jest to dobra miara do naszego zbioru ponieważ obie wartości targetu są dla nas tak samo istotne, więc ta miara sprowadza się do procentowej ilości poprawnych trafień.

Model drzewa

Pierwszy stworzony model drzewa miał accuracy ok. 91% na zbiorze testowym co jest dla nas już dobrym wynikiem. Po tunowaniu przy pomocy GridSearch doszliśmy do accuracy na poziomie 95%. Jednak na zbiorze walidacyjnym było to tylko ok 87%.

Mimo wszystko model ten jest warty uwagi ponieważ jako glass box daje nam dobre zrozumienie jego działania, a na podstawie tego można wnioskować coś o naszych danych.

Ciekawe jest to, że model zdecydował się na pierwszy podział według ‘Physician fee freeze’ ponieważ my również ustaliliśmy tą zmienną jako najlepiej dzielącą.



Random Forest

Skoro nasz model drzewa był już dość dobry to może warto sprawdzić model random forest. Model ten też tuningowaliśmy ale nie mogliśmy użyć Grid Search ponieważ każdy z modeli liczy się pewną ilość czasu więc sprawdzenie wszystkich możliwości hiper parametrów w realnym czasie dlatego wykorzystaliśmy Random Search.

```
{'n_estimators': [200, 233, 266, 300, 333, 366, 400, 433, 466, 500],
 'max_features': ['auto', 'sqrt'],
 'max_depth': [10, 11, 12, 13, 14, 16, None],
 'min_samples_split': [2, 5, 10],
 'min_samples_leaf': [1, 2, 4, 8, 12],
 'bootstrap': [True, False]}
```

Dzięki temu modelowi udało nam się poprawić wyniki do:

```
Accuracy Random Forest test: 0.971
Accuracy Random Forest val: 0.899
```

O ile wynik na zbiorze testowym jest bardzo dobry to jednak zbiór walidacyjny jest nadal niezadowalający. Dlatego nadal szukamy modelu którego wyniki będą bardziej stabline.

Inne modele proste

Trenowaliśmy różne inne modele proste takie jak:

Naive Bayes - jedyny parametr po jakim tunowaliśmy ten model był '*var_smoothing*' ale po mimo to nie uzyskał tak dobrych wyników jak random forest.

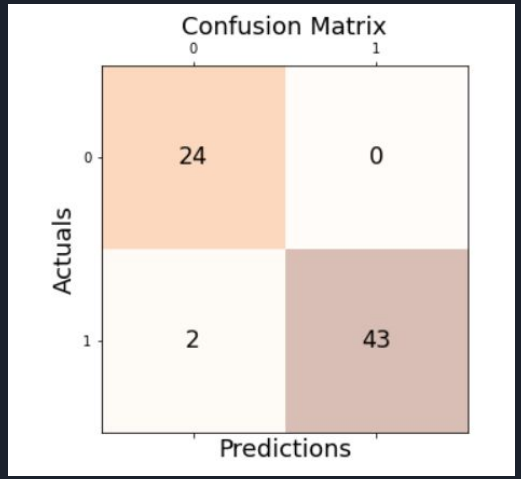
```
Accuracy Bayes test: 0.957
Accuracy Bayes val: 0.884
```

KNN - mogłoby się wydawać, że przy takich zmiennych które przyjmują tylko 3 wartości ten model będzie działał bardzo dobrze. Niestety jego wyniki są dokładnie takie same jak Naive Bayes.

```
Accuracy KNN test: 0.957
Accuracy KNN val: 0.884
```

Logistic Regression - ten model okazał się zaskakująco dobry dla naszych danych danych mimo utrzymującej się tendencji gdzie zbiór walidacyjny jest gorszy niż testowy to w obu wypadkach przekroczyliśmy 90%. Warto zauważyć, że wyniki są lepsze niż dla Random Forest. A te 3% brakujące procent to dwóch senatorów którzy zostali mylnie uznali za Republikanów.

```
Accuracy Logistic Regresion test: 0.971
Accuracy Logistic Regresion val: 0.913
```



Adaboost i Gradboost

Kolejnymi modelami, które stworzyliśmy były Adaboost i Gradboost. Pierwotnie modele miały accuracy minimalnie większe od 90%, ale po tuningu (tym razem wystarczyło wprowadzenie szczęśliwych parametrów z palca), uzyskaliśmy Accuracy 97% na valu i 95% na teście. Poniższe metryki są identyczne dla obydwu modeli.

```
print('F1 Score: %.3f' % f1_score(y_test, y_test_hat))
print('F1 Score: %.3f' % f1_score(y_val, y_val_hat))
print('Accuracy val: %.3f' % accuracy_score(y_val, y_val_hat))
print('Accuracy test: %.3f' % accuracy_score(y_test, y_test_hat))
```

```
F1 Score: 0.966
F1 Score: 0.978
Accuracy val: 0.971
Accuracy test: 0.957
```

HistGradboost

Inny stworzony przez nas model był oparty o HistGradBoost. W tym przypadku rezultaty nie były już tak zachęcające - porównywany był z wytunowanymi Adaboost i Gradboost i w porównaniu z nimi wypadł gorzej.

```
print( accuracy_score(y_val, y_val_hat))
print( accuracy_score(y_test, y_test_hat))
```

```
0.8695652173913043
0.9420289855072463
```

Wyniki modelowania

Dwoma najlepszymi są:

- AdaBoost,
- GradientBoost

```
print( accuracy_score(y_val, y_val_hat2))
print( accuracy_score(y_test, y_test_hat2))
```

```
0.9710144927536232
0.9565217391304348
```

Obydwa uzyskały accuracy 97% (2 błędy) na zbiorze walidacyjnym i 95% (3 błędy) na testowym. Zbiory walidacyjny i testowy są tego samego rozmiaru.

Trzecim dobrym jest Regresja logistyczna która uzyskała wyniki 91% na zbiorze walidacyjnymi 97% na zbiorze testowym. Mimo zalety jaką jest prostota tego modelu nie decydujemy się na wybranie go ponieważ wyniki nie są tak stabilnie dobre jak w powyższych modelach.

Dodatkowe uwagi

Dodatkowe uwagi

1. Sprawdziliśmy, czy usunięcie głosowań, które słabo rozdzielają zbiór poprawia dokładność modeli,
2. Sprawdziliśmy, czy usunięcie głosowań-outlierów wspiera dokładność modeli

W obydwu przypadkach nie uzyskaliśmy rezultaty negatywne.