WUM Projekt: Congressional Voting

Autorzy: Hubert Ruczyński, Bartosz Sawicki

Opis Problemu

Za zadanie mieliśmy przewidywać przynależność danego kongresmena ze stanów zjednoczonych do jednego z dwóch ugrupowań politycznych, tj. demokratów i republikanów, na podstawie oddanych przez niego głosów w 16 kluczowych głosowaniach w 1986 roku.

Użyte Dane

Źródło Danych

Dane używane w projekcie są dostępne pod następującym linkiem:
https://www.apispreadsheets.com/datasets/121

Opis zbioru

Ten zestaw danych obejmuje głosy oddane przez każdego z kongresmenów Izby Reprezentantów Stanów Zjednoczonych w 16 kluczowych głosach zidentyfikowanych przez CQA w 1986 r. CQA wymienia dziewięć różnych typów głosów: za uproszczone do y (yes), przeciw n (no), nie głosował ani w inny sposób nie przedstawił stanowiska ?.

EDA - Eksploracja Danych

Na początku projektu postanowiliśmy dokonać eksploracji danych, aby jak najlepiej zrozumieć przedstawiony nam problem i wyłuskać informacje, które mogą okazać się dla nas przydatne podczas tworzenia modelu.

Charakterystyka zbioru

Na początek chcieliśmy sprawdzić kompletność danych oraz sposób w jaki są one przechowywane

	handicapped_infants	water_project_cost_sharing	adoption_of_the_budget_resolution	physician_fee_freeze	el_salvador_aid	religious_groups_in_schools
33	n	y	n	y	y	y
321	y	y	y	n	n	n
165	n	y	y	n	n	y
262	y	n	y	n	n	n
39	y	n	y	n	n	n

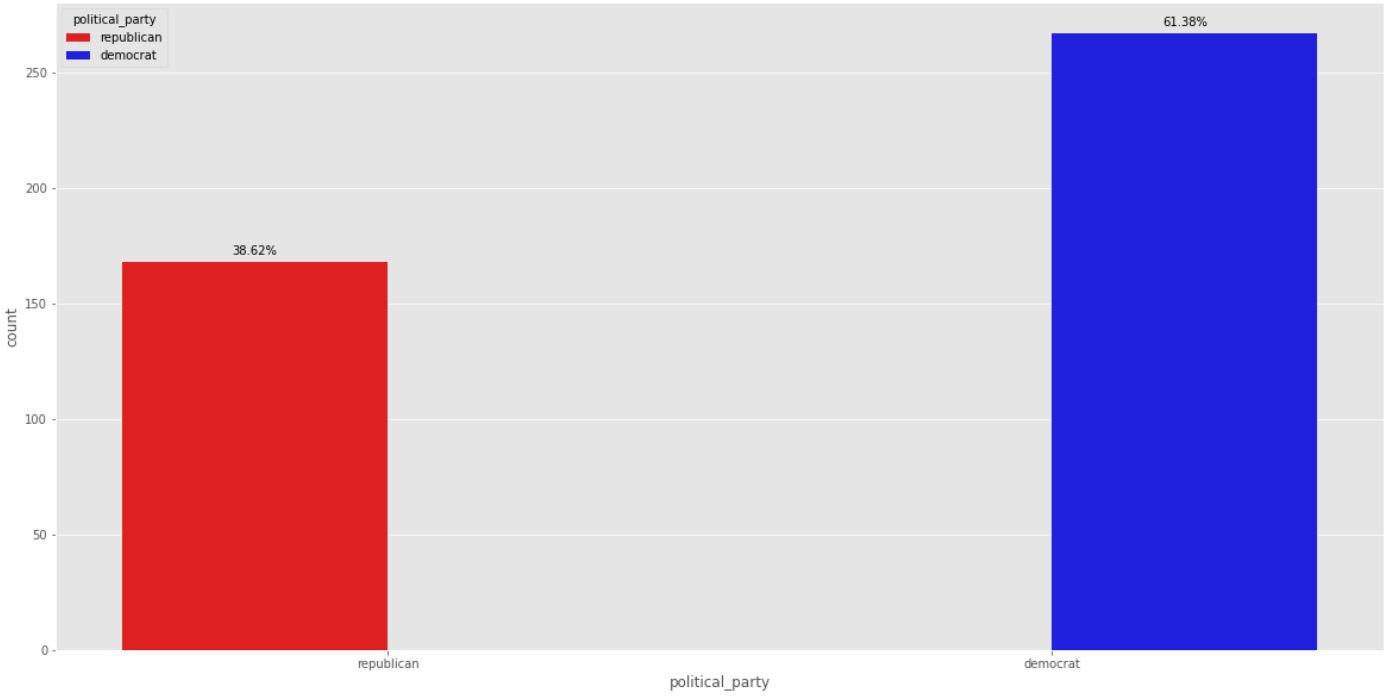
Opis danych na stronie zajmującej się nimi był adekwatny do ramki z której korzystaliśmy. Jako, że w późniejszym etapie chcieliśmy sprawdzić także korelację między poszczególnymi zmiennymi, dokonaliśmy autorskiego encodingu ramki danych. Encoding ten przekształcał dane y, n, ?, republican, democrat odpowiednio na 1,-1,0,-2,2.

	handicapped_infants	water_project_cost_sharing	adoption_of_the_budget_resolution	physician_fee_freeze	el_salvador_aid	religious_groups_in_schools
0	-1	1	-1	1	1	1
1	-1	1	-1	1	1	1
2	0	1	1	0	1	1
3	-1	1	1	-1	0	1
4	1	1	1	-1	1	1

Ponadto sprawdziliśmy typy danych oryginalnej ramki, aby poznać jej pełny kształt. Przy okazji dowiedzieliśmy się, że dane te są w pełni kompletne i nie brakuje żadnych rekordów.

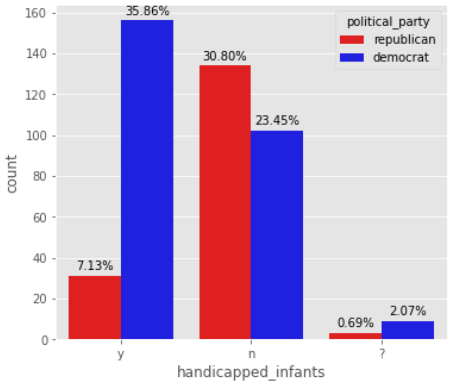
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 435 entries, 0 to 434
Data columns (total 17 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   handicapped_infants                   435 non-null    object
 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)
```

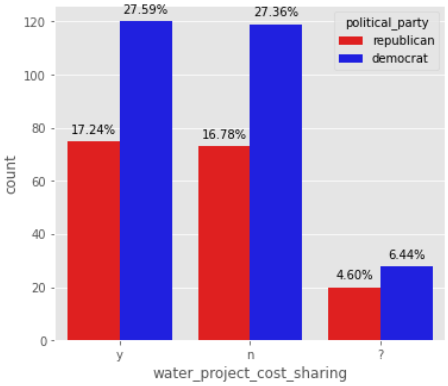
Ponadto sprawdziliśmy też stosunek liczności demokratów i republikanów w zadanym zbiorze i okazało się, że tych pierwszych jest prawie dwukrotnie więcej.

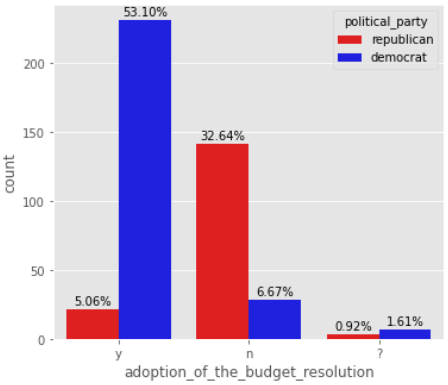


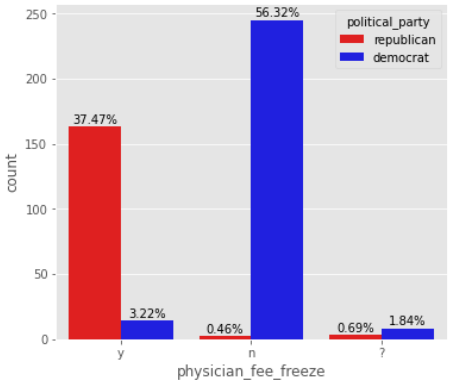
Charakterystyka głosowań

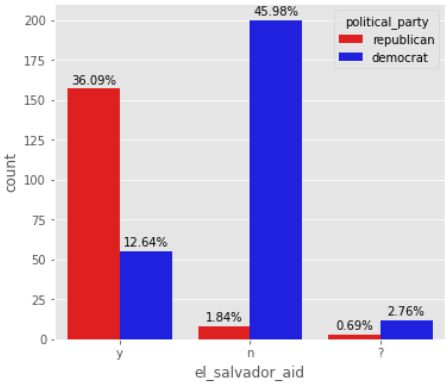
Najbardziej interesowała nas jednak charakterystyka samych głosowań, gdyż to na jej podstawie mieliśmy przewidywać do którego z ugrupowań należy dany kongresmen.

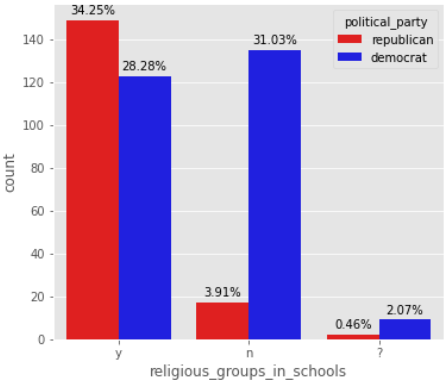

political_party
republican
democrat
count
handicapped_infants
y
n
?
7.13%
35.86%
30.80%
23.45%
0.69%
2.07%

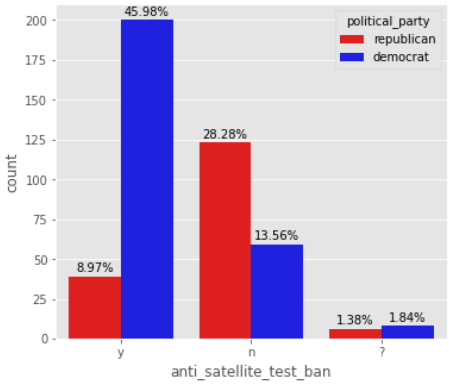

political_party
republican
democrat
count
water_project_cost_sharing
y
n
?
17.24%
27.59%
16.78%
27.36%
4.60%
6.44%

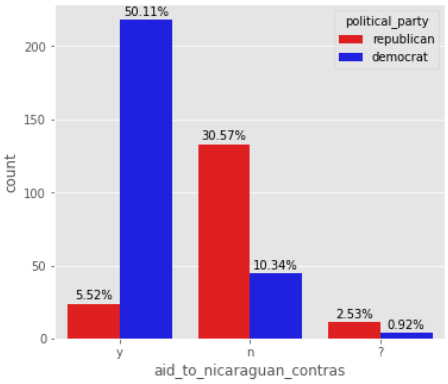

political_party
republican
democrat
count
adoption_of_the_budget_resolution
y
n
?
5.06%
53.10%
32.64%
6.67%
0.92%
1.61%

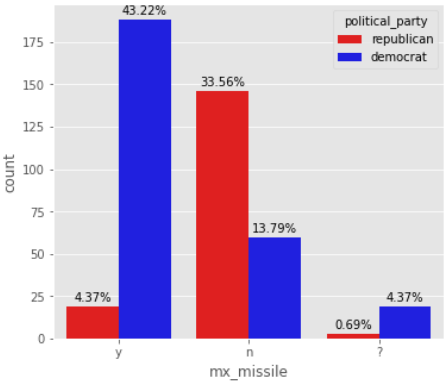

political_party
republican
democrat
count
physician_fee_freeze
y
n
?
37.47%
3.22%
0.46%
56.32%
0.69%
1.84%

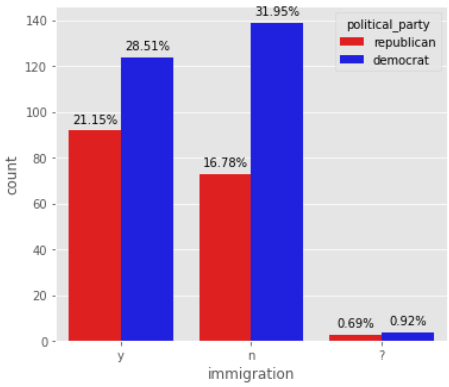

political_party
republican
democrat
count
el_salvador_aid
y
n
?
36.09%
12.64%
1.84%
45.98%
0.69%
2.76%

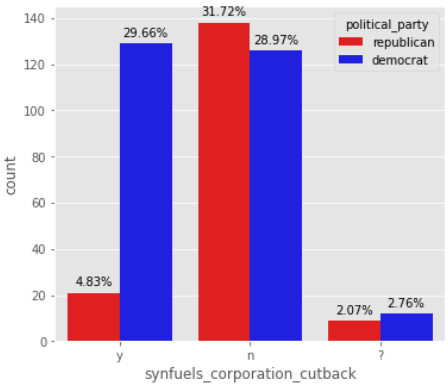

political_party
republican
democrat
count
religious_groups_in_schools
y
n
?
34.25%
28.28%
3.91%
31.03%
0.46%
2.07%

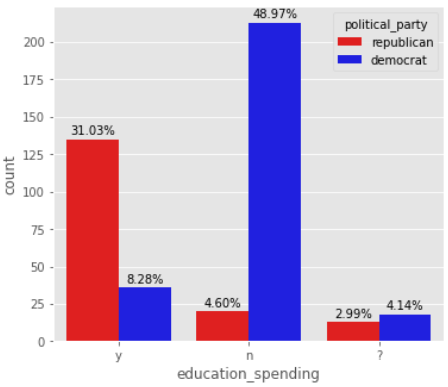

political_party
republican
democrat
count
anti_satellite_test_ban
y
n
?
8.97%
45.98%
28.28%
13.56%
1.38%
1.84%

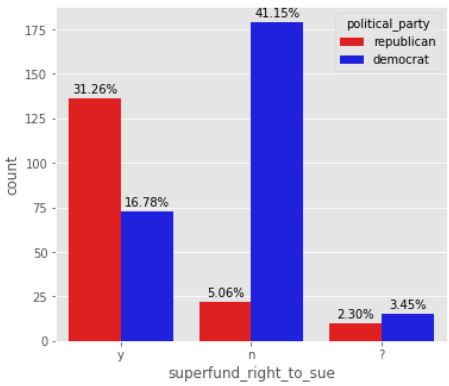

political_party
republican
democrat
count
aid_to_nicaraguan_contras
y
n
?
5.52%
50.11%
30.57%
10.34%
2.53%
0.92%

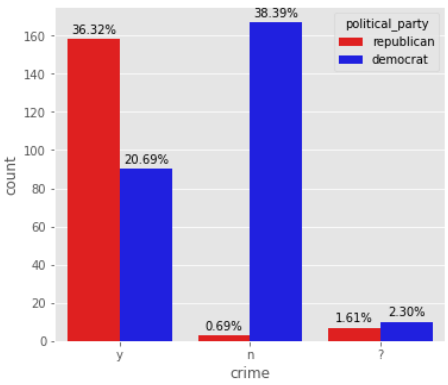

political_party
republican
democrat
count
mx_missile
y
n
?
4.37%
43.22%
33.56%
13.79%
0.69%
4.37%

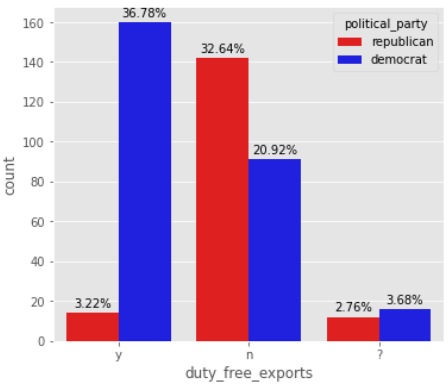

political_party
republican
democrat
count
immigration
y
n
?
21.15%
28.51%
16.78%
31.95%
0.69%
0.92%


political_party
republican
democrat
count
synfuels_corporation_cutback
y
n
?
4.83%
29.66%
31.72%
28.97%
2.07%
2.76%


political_party
republican
democrat
count
education_spending
y
n
?
31.03%
8.28%
4.60%
48.97%
2.99%
4.14%


political_party
republican
democrat
count
superfund_right_to_sue
y
n
?
31.26%
16.78%
5.06%
41.15%
2.30%
3.45%


political_party
republican
democrat
count
crime
y
n
?
36.32%
20.69%
0.69%
38.39%
1.61%
2.30%


political_party
republican
democrat
count
duty_free_exports
y
n
?
3.22%
36.78%
32.64%
20.92%
2.76%
3.68%

Z powyższej analizy wynikło, że najbardziej rozdzielającą ugrupowania kwestią jest zamrożenie opłat lekarskich (physician_fee_freeze), zaś tą niosącą za sobą najmniej informacji ze względu na swój regularny rozkład był projekt wodny (water_project_cost_sharing).

Aby utwierdzić się w poprawności tego podziału oraz wysnuć kolejne wnioski postanowiliśmy stworzyć heatmapę korelacji zmiennych przekształconym za pomocą naszego autorskiego encodingu.

Nasze przypuszczenia potwierdziły się oraz byliśmy w stanie wynuć kolejne wnioski, takie jak fakt, że głosowania adoption_of_the_budget_resolution, el_salvador_aid oraz education_spending także są mocno związane z przynależnością do ugrupowania natomiast immigration już nie jest.

Wnioski

- Struktura zbioru danych faworyzuje modele oparte o drzewa decyzyjne.
- Dobra jakość zbioru pozwoliła nam na uniknięcie imputacji oraz usuwania outlierów.
- Pewne kolumny danych nadają się do usunięcia, ponieważ nie niosą informacji o przynależności partyjnej kongresmena.

Tworzenie i testowanie modeli

Drugi z etapów projektu był dla nas tym z którego wyciągnęliśmy najwięcej doświadczeń i przydatnych umiejętności. Już wewnątrz tego pojedynczego etapu widać jak duży progres zrobiliśmy poprzez porównanie pierwszego modelu, który nieporadnie tworzyliśmy do kolejnych, które okazały się lepsze i kod który stworzyliśmy jest zdecydowanie czytelniejszy.

Pierwsze modele

Pierwsze modele uczone były na danych przetworzonych przez nasz autorski encoding co później okazało się pogarszać jakość predykcji, o czym na ten moment nie wiedzieliśmy.

Jako pierwszy model stworzyliśmy bardzo prosty SVM. O ile samo używanie modelu nie było dla nas niczym zbyt trudnym to sprawdzenie jego skuteczności okazało się problematyczne. W efekcie ręcznie liczyliśmy accuracy oraz, aby nasz wynik był bardziej rzetelny to całość potem zapętlaliśmy 1000 krotnie z losowym stanem początkowym, aby otrzymać możliwie bliski prawdy wynik (liczyliśmy średnie accuracy z 1000 iteracji). Ponadto metodą prób i błędów wybieraliśmy, które ze zmiennych odrzucić, aby uzyskać możliwie najlepsze rezultaty.

```
y:      142   -2
396     2
174     2
243     2
69      2
82     -2
193     2
290     2
98      2
419     2
Name: political_party, dtype: int64
y_pred: [-2  2  2  2  2 -2  2  2  2  2]
validation : 0.944954128440367
train : 0.9754601226993865
```

Rysunek 1 Pojedyncza iteracja SVM z nieporadnym sprawdzaniem jakości

```
accumulated accuracy validation : 0.9554403669724738
accumulated accuracy train : 0.9793773006134908
```

Rysunek 2 Najlepszy wynik dla SVM w pętli

Z metodyki prób i błędów przy wyborze kolumn udało nam się wybrać zestaw kolumn, które powinny być odrzucone, aby zapewnić jak najlepszą wydajność modelu. Wynik tego modelu zaprezentowany jest na powyższym rysunku.

One Hot Encoding

Po zagłębieniu się w fora o uczeniu maszynowym poprawiliśmy swoje podejście do zadania poprzez zautomatyzowanie wielu procesów, przy tym poprawiając ich wydajność. Przede wszystkim zamieniliśmy nasz encoding na One Hot, gdyż jest to po prostu lepsza wersja tego co sami zrobiliśmy. Zamiana ta przyczyniła się do poprawy modelu.

Ponadto zastosowaliśmy także selekcję zmiennych którą otrzymaliśmy z poprzedniego modelu, w efekcie testując nowe modele na dwóch zbiorach, zwykłym i dropowanym (bez zmiennych: 'water_project_cost_sharing','immigration','export_administration_act_south_africa').

Zaawansowane modele z kroswalidacją:

Następnie testowaliśmy różne bardziej zaawansowane modele klasyfikacyjne, z których zostawiliśmy trzy o najlepszej skuteczności. Skuteczność mierzyliśmy za pomocą kroswalidacji zamiast iterowania w pętli, co jest

znaczną poprawą od pierwszego modelu. Accuracy tych modeli przedstawione zostało na poniższych obrazkach (najpierw dla zbioru całego, a następnie dropowanego):

```
0.9539044289044288
0.9468531468531469
```

Rysunek 3 Gradient Boosting

```
0.9515734265734267
0.9561188811188811
```

Rysunek 4 XGB

```
0.951649476995099
0.9562577719259747
```

Rysunek 5 Ada Boost

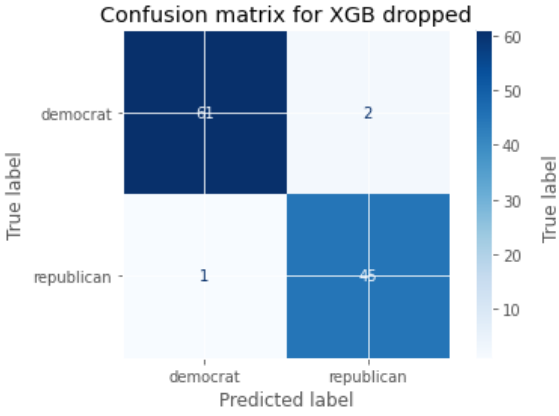
Wnioski

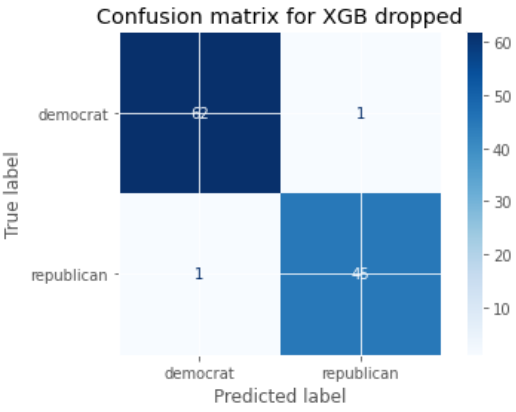
Dzięki temu etapowi nauczyliśmy się sprawnie korzystać z różnych modeli oraz poznaliśmy ich słabe i mocne strony. Ponadto zauważyliśmy, że selekcja zmiennych nie zawsze prowadzi do poprawienia jakości modelu. Co więcej nauczyliśmy się w lepszy sposób oceniać jakość modelu, przeskok z ręcznego i żmudnego oceniania jakości przerzuciliśmy na skuteczne i sprawne narzędzia jak kroswalidację, czy też classification report. Głównym wnioskiem jest jednak to, że zanim będziemy chcieli sami coś zrobić (jak w przypadku kodowania) to warto sprawdzić czy ktoś nie zrobił już tego lepiej od nas.

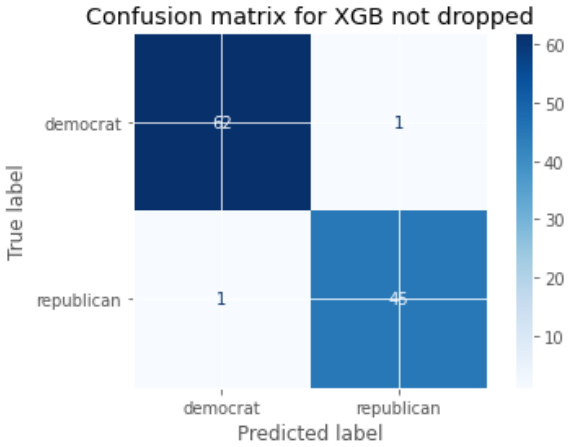
Finalny wybór modelu i strojenie hiper parametrów

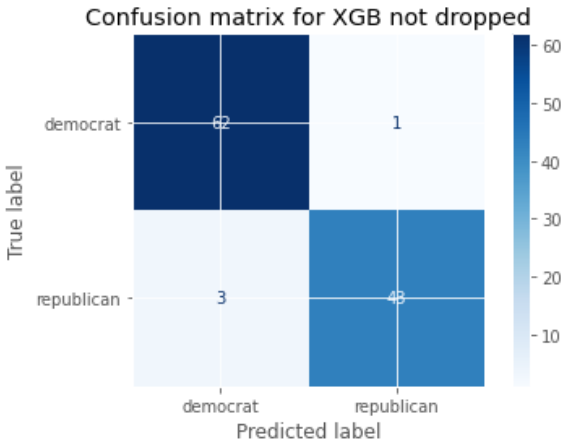
Ręczne tunowanie parametrów

Na początku postanowiliśmy ręcznie dobierać parametry tak aby jakość naszych modeli (mierzonych poprzez accuracy z kroswalidacji, confusion matrixes oraz tabele zawierające precision, recall, f1-score i accuracy) uległa poprawie. Działania te przeprowadziliśmy dla wszystkich trzech modeli, jednakże jako przykład przytoczymy tylko XGBClassifier (aby szczegółowo zagłębić się w temat zalecamy przeczytać Congressional Voting Tuning). Po lewej stronie pokazany jest confusion matrix dla modelu sprzed strojenia zaś po prawej po strojeniu.









Zauważyć można, że w tym przypadku strojenie hiper parametrów zamieniło niejako faworyzowanie zbioru względem dropowania danych. Teorię tę potwierdzają również miary widoczne poniżej (u góry mamy dane nie dropowane, na dole dropowane, strony jak wyżej).

	precision	recall	f1-score	support
democrat	0.95	0.98	0.97	63
republican	0.98	0.93	0.96	46
accuracy			0.96	109
macro avg	0.97	0.96	0.96	109
weighted avg	0.96	0.96	0.96	109

	precision	recall	f1-score	support
democrat	0.98	0.98	0.98	63
republican	0.98	0.98	0.98	46
accuracy			0.98	109
macro avg	0.98	0.98	0.98	109
weighted avg	0.98	0.98	0.98	109

Rysunek 6 Model przed strojeniem

	precision	recall	f1-score	support
democrat	0.98	0.98	0.98	63
republican	0.98	0.98	0.98	46
accuracy			0.98	109
macro avg	0.98	0.98	0.98	109
weighted avg	0.98	0.98	0.98	109

	precision	recall	f1-score	support
democrat	0.98	0.97	0.98	63
republican	0.96	0.98	0.97	46
accuracy			0.97	109
macro avg	0.97	0.97	0.97	109
weighted avg	0.97	0.97	0.97	109

Rysunek 7 Model po strojeniu

Ponadto wynik kroswalidacji przedstawiony jest poniżej (w taki sam sposób jak poprzednie):

```
0.954020979020979   0.9539044289044291
0.9699883449883451  0.9561188811188811
```

GridSearchCV

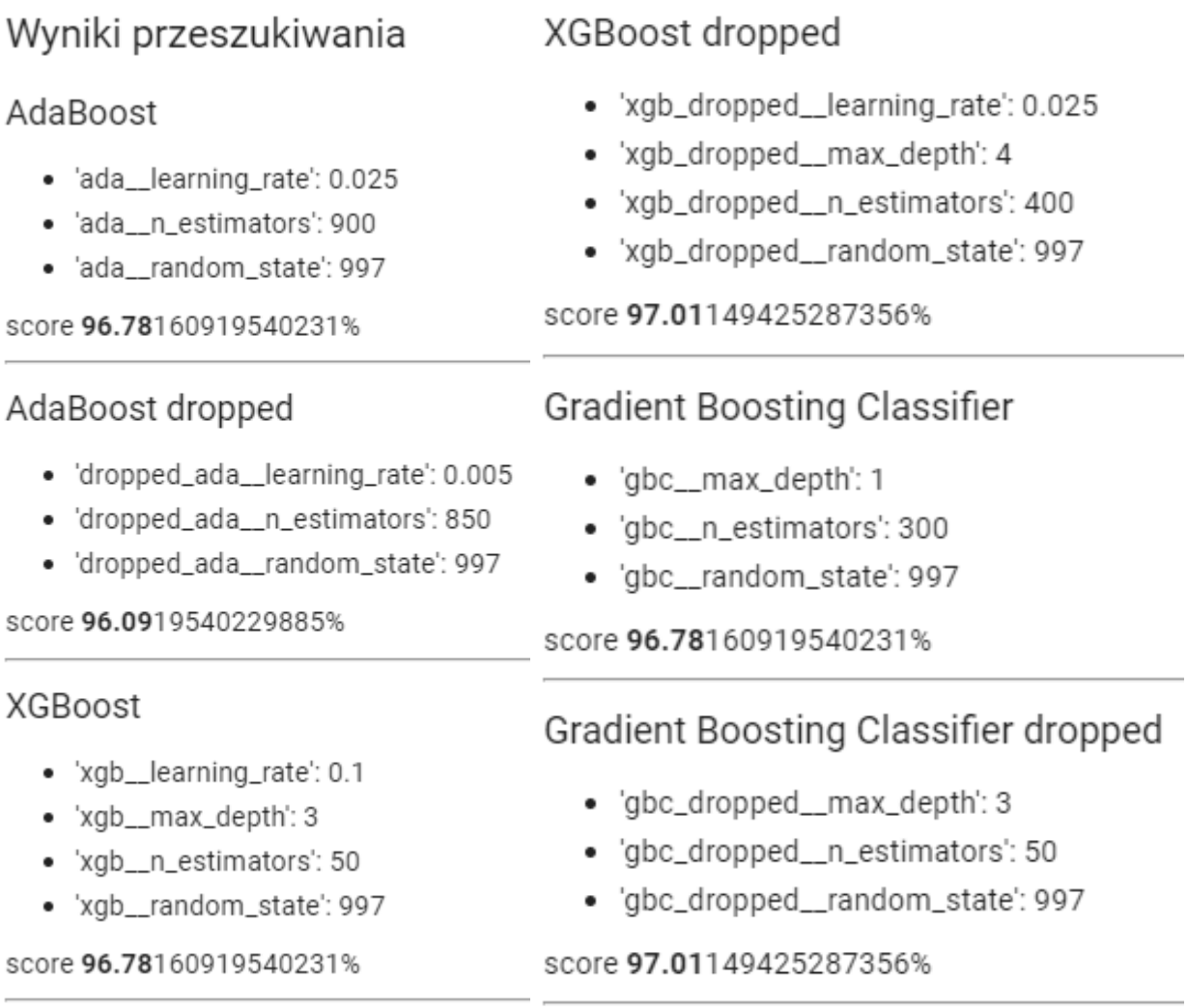
Aby zautomatyzować proces szukania najlepszych hiperparametrów użyliśmy funkcji sklearn.model_seletion.GridSearchCV. Umożliwia ona zdefiniowanie siatki parametrów, a następnie

stworzenie modeli o parametrach będących wszystkimi możliwymi kombinacjami parametrów z siatki. Funkcja dokonuje ewaluacji modeli poprzez korswalidację. Po wykonaniu przeszukiwania możemy odczytać wynik najlepszego modelu oraz jego hiperparametry.

Podjęliśmy próby przeszukiwania przestrzeni parametrów. Jednak zauważyliśmy, że tuning hiperparametrów wymaga dużej mocy obliczeniowej, a w konsekwencji spotrzeba sporo czasu do jego wykonania. Aby obliczenia kończyły się w rozsądnym czasie, definiowane siatki parametrów mogły zawierać co najwyżej kilka elementów. Podjęliśmy zatem decyzję o uruchomieniu obliczeń wieczorem i pozostawieniu pracującego komputera na noc. Tworząc ten notebook pracowaliśmy w Google Colab, więc postanowiliśmy wykorzystać możliwość podpięcia się do Google Drive w celu zapisania wyników, na wypadek, gdyby sesja w Colab się zakończyła.

Tym razem siatki parametrów były gęstsze i pokrywały większy zakres przestrzeni parametrów niż w poprzednim etapie. Każdy z trzech modeli był tuningowany w dwóch wariantach:

- opartym na wszystkich kolumnach,
- opartym na wybranych kolumnach (dropped)



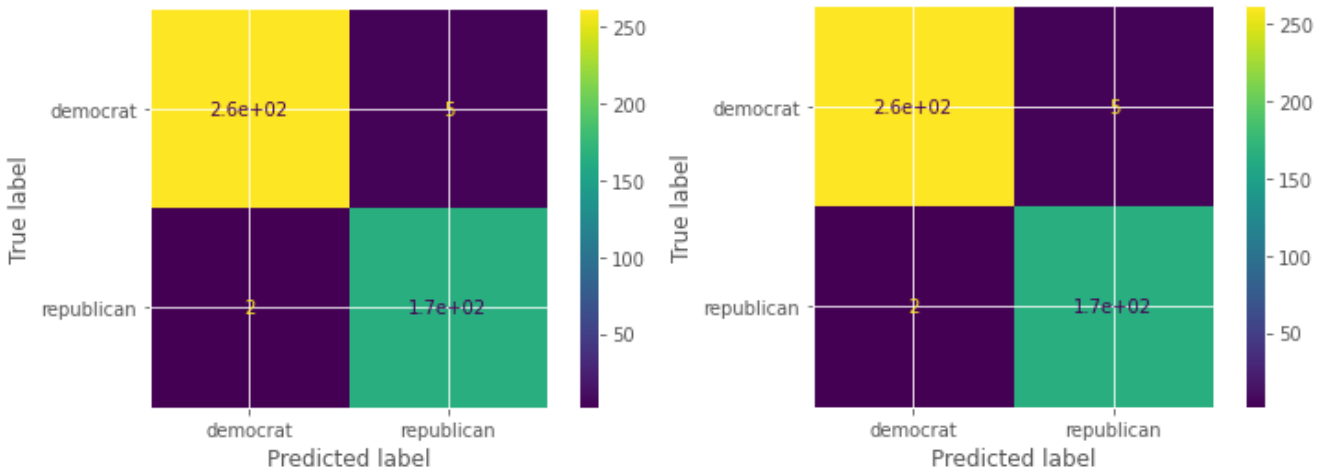
Zauważmy, że najlepszy wynik został ex-equo osiągnięty przez dwa modele:

- XGBoost oparty o dane z wybranych kolumn

- GBC oparty o dane z wybranych kolumn

Ewaluacja modeli

Poniżej stworzyliśmy modele o najlepszych parametrach w celu głębszego ich zbadania. Ponadto zbadaliśmy ich macierze pomyłek obliczone na całym zbiorze danych.



Wyniki w macierzy pomyłek były takie same dla obu klasyfikatorów. Sprawdziliśmy, czy pomyliły się w tych samych obserwacjach.

103	republican	168	republican
242	democrat	242	democrat
315	democrat	267	democrat
375	republican	375	republican
382	republican	382	republican
388	republican	388	republican
407	republican	407	republican

W 5 z 7 przypadków modele popełniły błędy w tych samych obserwacjach. To oznacza, że w zbiorze istnieją obserwacje trudne do zaklasyfikowania przez różne modele. Dokonaliśmy przeglądu niektórych z tych wierszy i zauważyliśmy, że większość z nich w kolumnie `physician_fee_freeze` ma wartość przeciwną do tego co mogłoby wynikać z przynależności partyjnej kongresmena. Inaczej mówiąc, modele mają problem z poprawną klasyfikacją demokratów głosujących na tak oraz republikanów głosujących na nie w tym głosowaniu.

Wnioski

- Model, który w pierwszej fazie planowaliśmy odrzucić (Gradient Boosting), po tuningu hiperparametrów zyskał najlepszą skuteczność. Stąd wnioskujemy, że warto przeprowadzić dostrajanie dla większej liczby modeli i dopiero wtedy dokonać ostatecznego wyboru.
- GridSearchCV długo wykonuje się. Gdy chcemy przeszukać większą przestrzeń hiperparametrów, warto rozważyć inny sposób optymalizacji.

- Korzystanie z Google Colab jest wygodne, umożliwia wygodną współpracę online przy pisaniu notatników ipynb. Dostajemy też przydział zasobów obliczeniowych, co zwiększa szybkość obliczeń jednocześnie zmniejszając obciążenie lokalnego komputera. Irytująca może być sesja, która wygasa po kilkudziesięciu minutach nieaktywności.

Wnioski i doświadczenia z projektu

Główną nauką którą wyciągnęliśmy z tego projektu było używanie dobrych narzędzi automatyzujących naszą pracę. Przede wszystkim mowa tu o:

- Automatycznych raportach wskaźników zamiast ręcznego ich liczenia
- One hot encodingu zamiast ręcznego kodowania zmiennych
- GridSearcha zamiast ręcznego dobierania parametrów
- Korzystanie z heatmapy korelacji zamiast wyłapywania okiem możliwych istotnych cech

Ponadto przekonaliśmy się o wysokiej jakości modelów boostingowych.