1986 US congressmen voting

EDUCATIONAL DATA EXPLORATION AND MACHINE LEARNING PROJECT SUMMARY

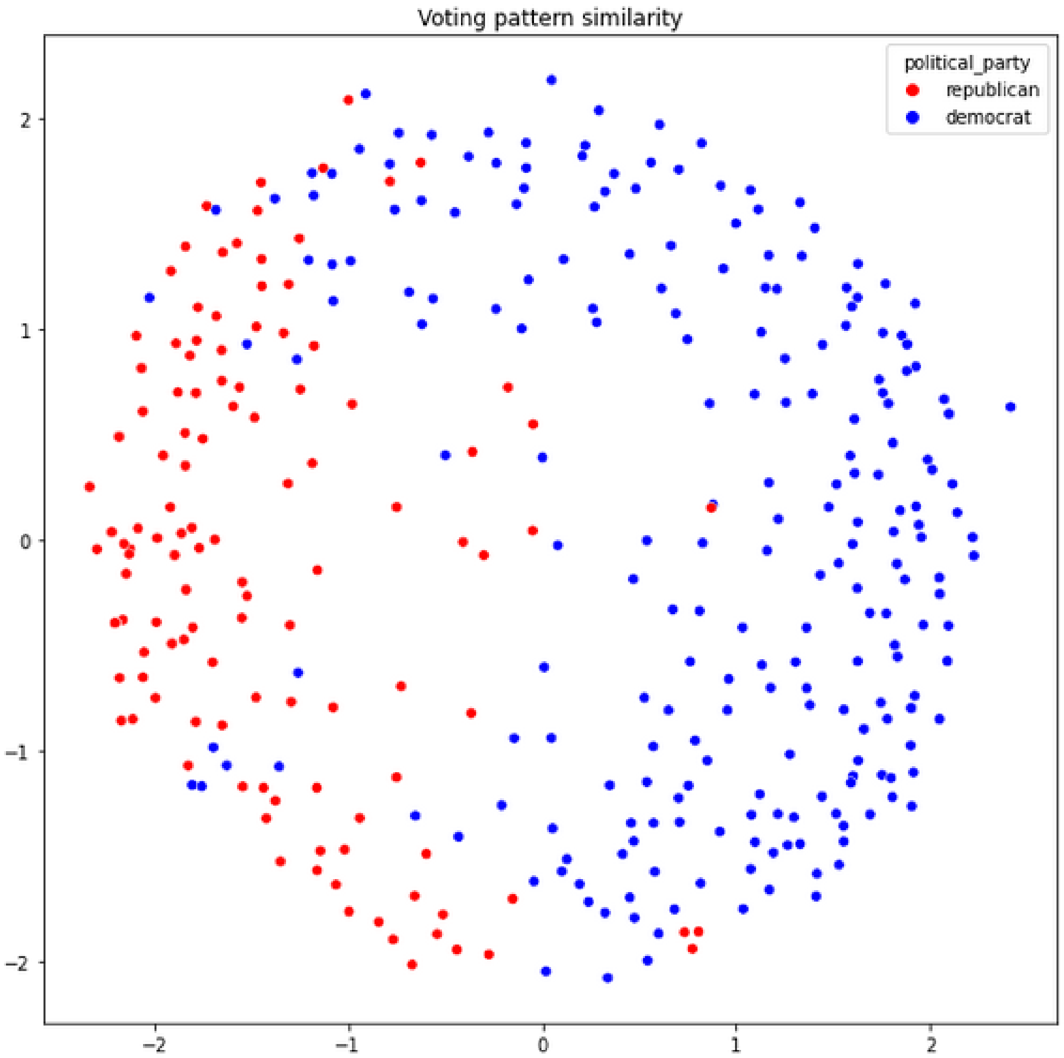
export_administration_act_south_africa	political_party
y	republican
?	republican
n	democrat
y	democrat

```
<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
```

Data info

Dataset used in this project was data containing votes of individual members of congress in selected votings in 1986 paired with their political affiliation (Source: https://www.apispreadsheets.com/datasets/121)

In this project we explore the data and then apply machine learning to distinct between republicans and democrats based on their voting choices.

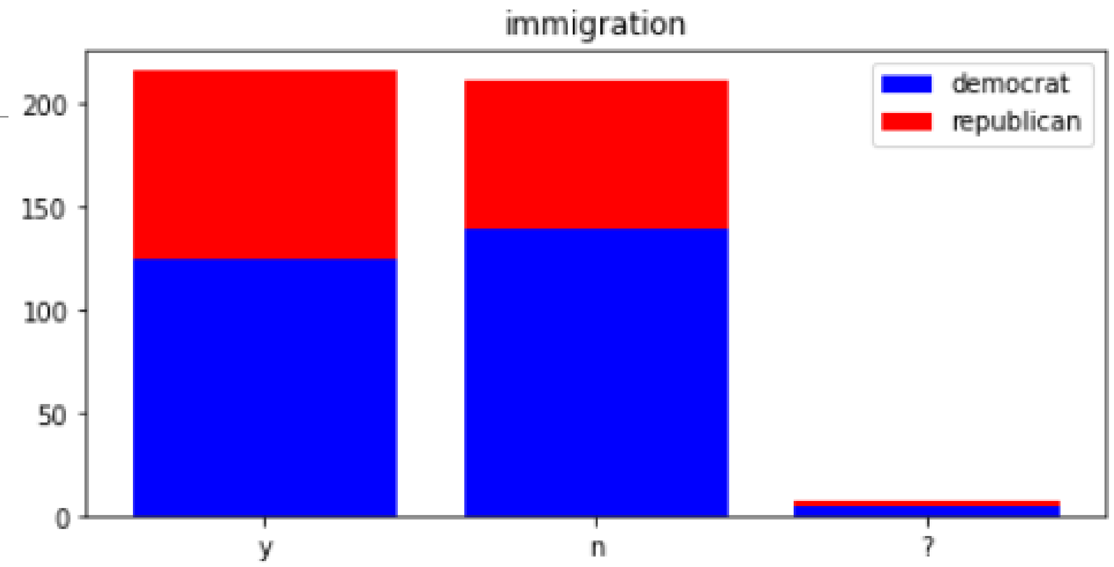


Voting patterns similarity

After encoding the data, I created a matrix of euclidean distance between each pair of congressmen and then used embedding function to project this data in 2D space. Each dot represents a member of the congress. Members voting in a similar manner are closer to each other.

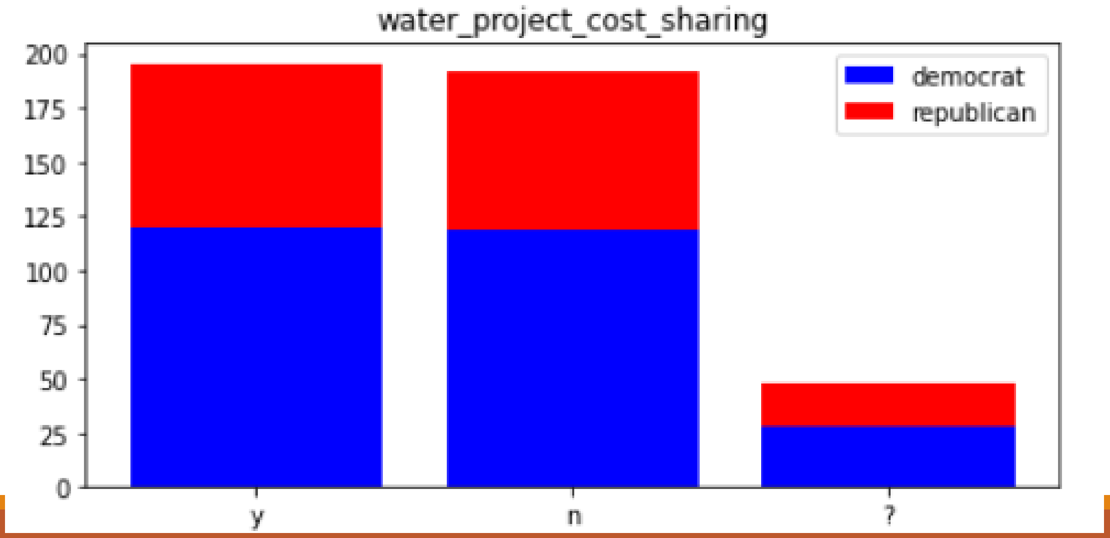
Obviosly, it's impossible to translate multidimensional dependencies into 2D, but the visualisation can give us some insight into nature of the task.

Least differentiating votings



Both parties voted pretty much the same :
- **water_project_cost_sharing**,
- **imigration**,

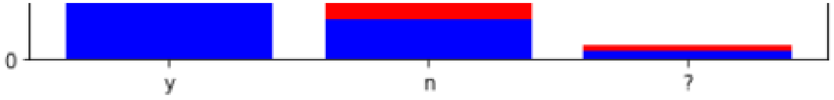
These variables may not be useful in predicting political affiliattion.

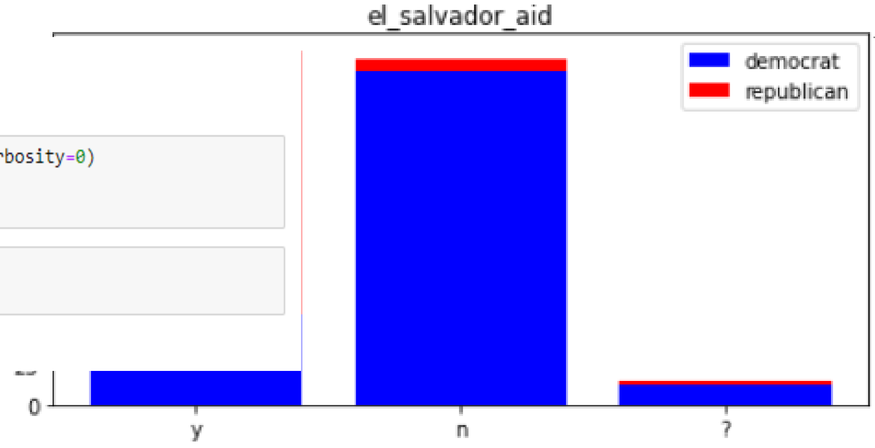


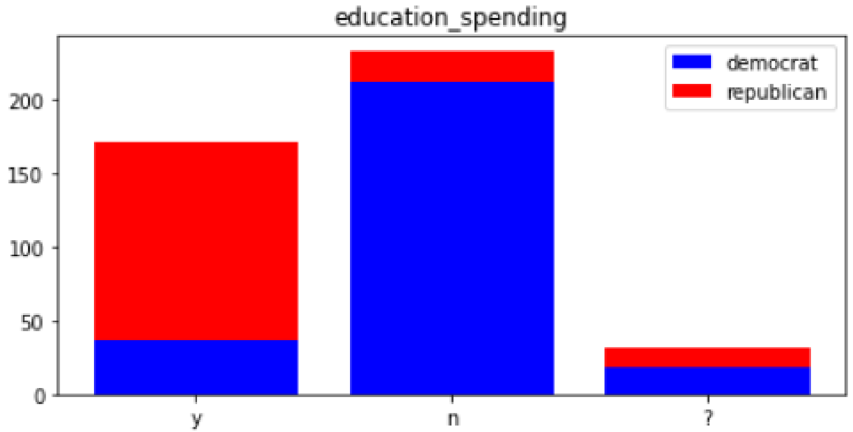
Most differentiating votings

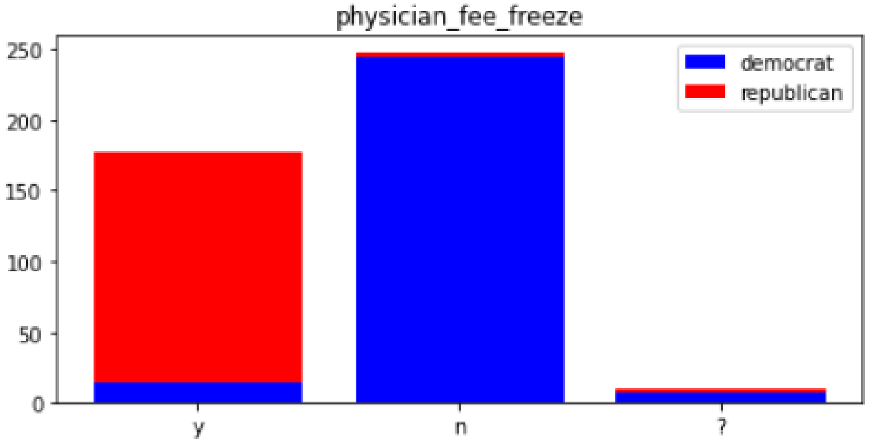
XGBoost

```
In [8]: xgb_base = xgb.XGBClassifier(objective = "binary:logistic", seed = 1613, use_label_encoder=False, verbosity=0)
        xgb_base.fit(X_train, y_train)
        preds=xgb_base.predict(X_test)

In [9]: xgb_base_acc=accuracy_score(preds,y_test)
        print("Accuracy XGB z domyślnymi hiperparametrami: " + str(xgb_base_acc))

        Accuracy XGB z domyślnymi hiperparametrami: 0.9724770642201835
```









Base Models

XGBoost

```
In [8]: xgb_base = xgb.XGBClassifier(objective = "binary:logistic", seed = 1613, use_label_encoder=False, verbosity=0)
        xgb_base.fit(X_train, y_train)
        preds=xgb_base.predict(X_test)
```

```
In [9]: xgb_base_acc=accuracy_score(preds,y_test)
        print("Accuracy XGB z domyślnymi hiperparametrami: " + str(xgb_base_acc))

        Accuracy XGB z domyślnymi hiperparametrami: 0.9724770642201835
```

SVM

```
In [4]: svm_base=SVC(random_state=42)
        svm_base.fit(X_train, y_train)
        preds=svm_base.predict(X_test)
```

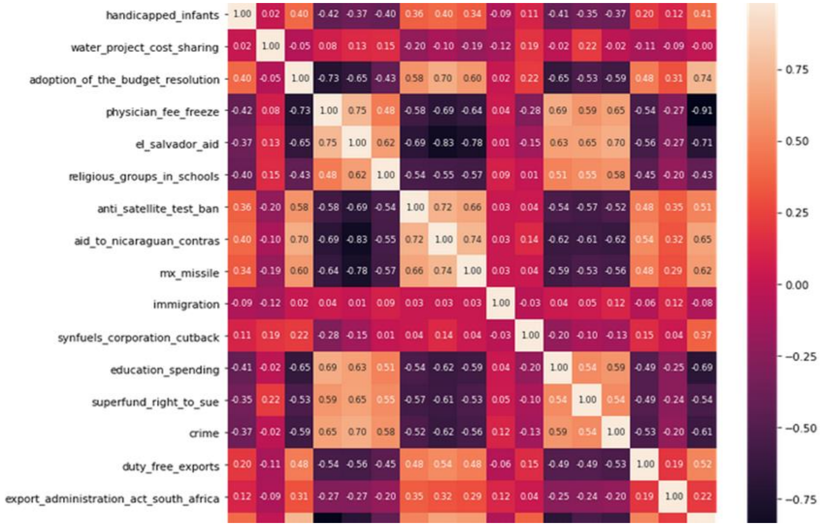
```
In [5]: svm_base_acc=accuracy_score(preds,y_test)
        print("Accuracy SVM z domyślnymi hiperparametrami: " + str(svm_base_acc))

        Accuracy SVM z domyślnymi hiperparametrami: 0.963302752293578
```

Random Forest

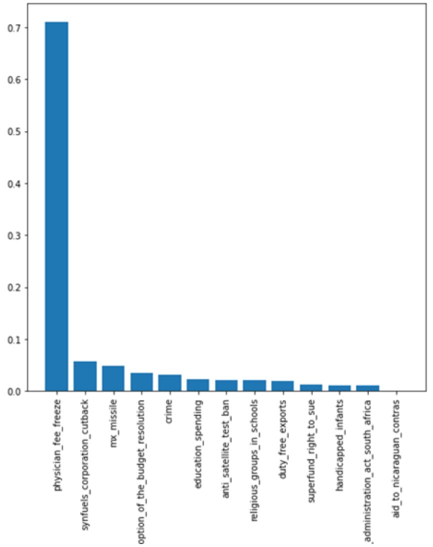
```
In [12]: rfc_base = RandomForestClassifier(random_state=16)
         rfc_base.fit(X_train, y_train)
         preds=rfc_base.predict(X_test)
```

```
In [13]: rfc_base_acc=accuracy_score(preds,y_test)
         print("Accuracy RFC z domyślnymi hiperparametrami: " + str(rfc_base_acc))

         Accuracy RFC z domyślnymi hiperparametrami: 0.9724770642201835
```

As a base, I used XGBoost, SVM and Random Forest models. As we can see, the base models applied on all features achieve a very good result of accuracy close ~97%. However, for educational purposes I applied some feature selection and tuning methods.



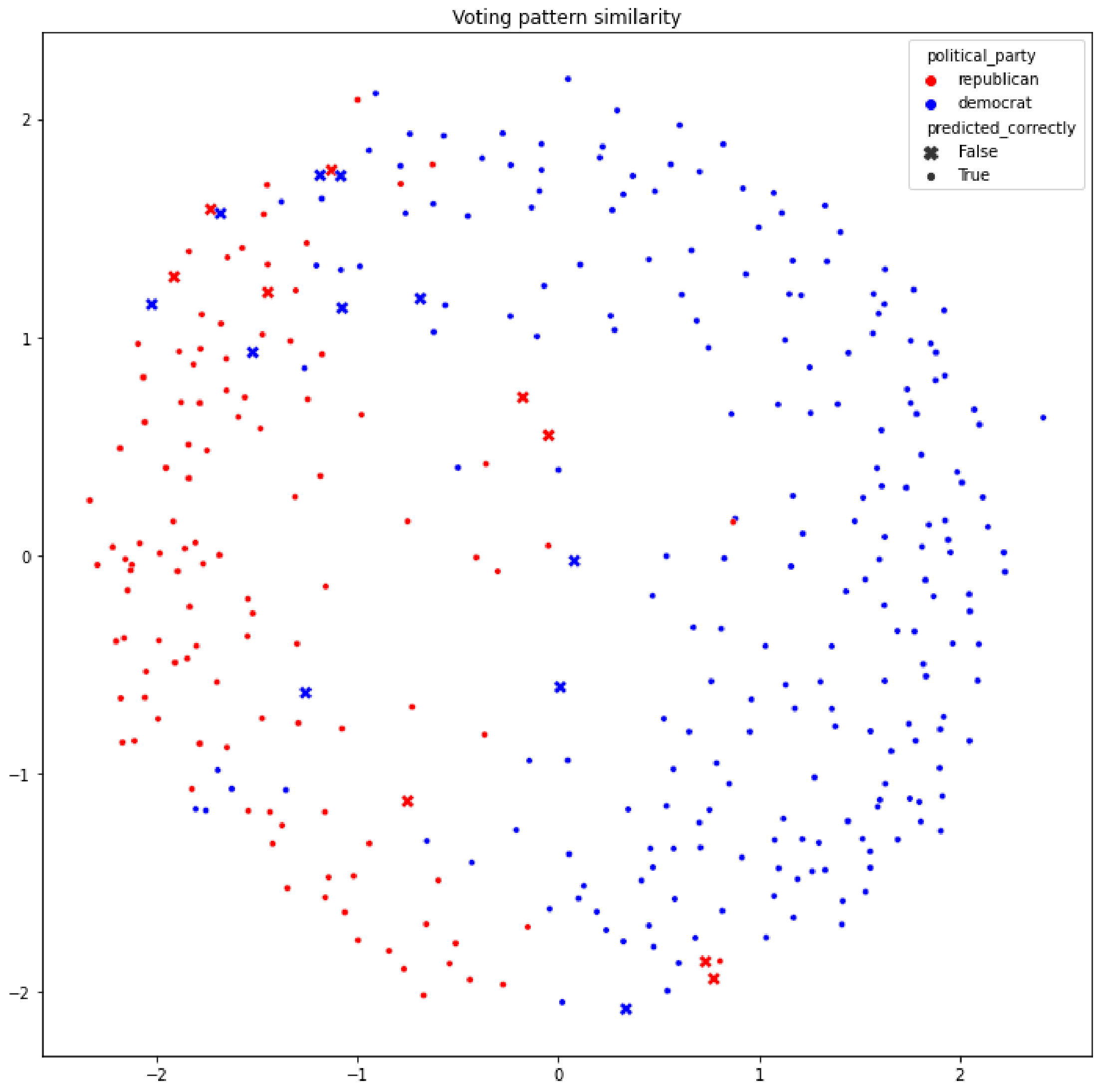

```
pyplot.bar(df1.col, df1.y)
plt.xticks(rotation=90)
pyplot.show()
```



Feature Selection and hyperparameter tuning

Using various techniques such as **Recursive Feature Elimination**, **SelectKBest** or XGBoost **feature_importances** I was able to greatly limit the numer of used variables without a significant of accuracy of the models. However, due to relatively small dataset I decided to stick to the whole dataset.

I then applied hyperparameter tuning via GridSearch to tune parameters of the models and gained a slight increase in accuracy.



Final classification by XGBoost model