DSC_80_Project_5_2

June 8, 2022

1 Predicting Party Affiliation from Stock Data

Aritra Das, Akshat Muir

1.1 Summary of Findings

1.1.1 Introduction

For Project 5 our group is trying to classify the party affiliations of a representative based on their stock trades. This is going to be binary classification(Democrat, and Republican). We are going to be using accuracy to evaluate our models. The reason why we were interested in finding party affiliation of the congressmember was to see if there is a correleation in a certain parties tendencies in trading. 'disclosure_year', 'disclosure_date', 'transaction_date', 'owner', 'ticker', 'asset_description', 'type', 'amount', 'representative', 'district', 'ptr_link', and 'cap_gains_over_200_usd' will all be know at time of prediction

1.1.2 Baseline Model

For our baseline model we applied ordinal and a one hot encoding on certain columns to give us the ability to use those unique features to do further analysis. For the ordinal model we applied it to the 'amount' column as it had certain type of variable that would be better represented as ordinal data. For the other categorical columns ['owner','type', 'cap_gains_over_200_usd'] we applied a one hot encoding as their variables would be better represented using that form of classification. We think our model will work for unseen data because we have chosen to drop the 'representative' column because we discovered earlier that it would more likely overfit to training data and with that removed we can have a more generalized prediction. We also choose to exclude 'district', 'ptr_link', 'disclosure_year', 'ticker', and 'asset-description' because we didn't think those would have much to do with our prediction in a general sense, as it may overfit to our data. We also excluded 'exclosure_date', and 'transaction_date' as we wanted to include these in our final model and didn't think they needed to be put in our baseline model because they only give us information regarding when something was bought and may not give us information specific to predicting political party. Our best score with the baseline model is 71.8%.

1.1.3 Final Model

We created two parameters in the form of columns and one of them was the number of days between the transaction date and disclosed date. Another column we made was whether the difference between transaction date and disclosure date was over a month of time, specifically 31 days. After creating these two columns we performed a GridSearchCV on the new feature matrix and got an improved accuracy of around 7% giving us a total accuracy of 78.5%. This improvement was due

to our new features and one of the features was the difference between the transaction date and disclosure date , and a feature stating if they took over a month to disclose their transaction. These two would find disparities between both party members tendencies regarding reporting their differences in disclosure and transaction date. We also tried to use DecisionTreeClasifier but the prediction accuracy was worse at 75% compared to 78.5%. The Hyperparameters that we got were leaf_size= 40, n_neighbors= 5.

1.1.4 Fairness Analysis

Null Hypothesis : Our model is fair. It's accuracy of prediction for making a cap gain over 200 and not making a cap gain over 200 are roughly the same.

Alternate Hypothesis : Our model is not fair. It's accuracy of prediction for making a cap gain over 200 is greater than the accuracy for not making a cap gain over 200.

P-Value is 0.00 and Alpha = 0.05.

This means that we reject the Null Hypothesis as the P-Value < Alpha. This means that our model is not fair and that the accuracy of prediction for making a cap gain over 200 is greater than the accuracy for not making a cap gain over 200.

2 Code

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
import seaborn as sns
from sklearn.utils import shuffle
%config InlineBackend.figure_format = 'retina'  # Higher resolution figures
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import OrdinalEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
```

2.1 Cleaning Data and Creating Features

[ ]:
```
stocks_df = pd.read_csv('/Users/aritradas/Documents/College/Code/Spring_2022/
 ↪dsc80-2022-sp/projects/05-prediction/data/all_transactions.csv')
house_affiliation = pd.read_csv('/Users/aritradas/Documents/College/Code/
 ↪Spring_2022/dsc80-2022-sp/projects/05-prediction/data/house_affiliation.csv')
stocks_df = stocks_df.merge(house_affiliation, how= 'left', on='representative')
```

[ ]:
```
# Changing disclosure_date and transaction_date to datetime object
stocks_df['disclosure_date'] = pd.to_datetime(stocks_df['disclosure_date'])
stocks_df['transaction_date'] = pd.to_datetime(stocks_df['transaction_date'],␣
 ↪errors = 'coerce')
# Cleaning all '--' entries to np.NaN
stocks_df = stocks_df.replace('--', np.NaN)
# Canonizing amount column
stocks_df['amount'] = stocks_df['amount'].replace(['$1,000 - $15,000', '$15,000␣
 ↪- $50,000', '$1,000,000 - $5,000,000', '$1,001 -'], ['$1,001 - $15,000',␣
 ↪'$15,001 - $50,000', '$1,000,001 - $5,000,000', '$1,001 +'])
# Canonizing representative columns
stocks_df['representative'] = stocks_df['representative'].str.lower().str.
 ↪replace('hon. ', '', regex=True)

stocks_df['days_since_transaction_disclosure'] =␣
 ↪(stocks_df['disclosure_date']-stocks_df['transaction_date']).
 ↪astype('timedelta64[D]')
stocks_df['days_since_transaction_disclosure'].
 ↪fillna(stocks_df['days_since_transaction_disclosure'].mean(), inplace=True)

stocks_df['over_a_month'] = stocks_df['days_since_transaction_disclosure'] > 31

stocks_df;
```

2.2 Baseline Model

[ ]:
```
amount_category = ['$1,001 - $15,000', '$1,001 +', '$15,001 - $50,000',␣
 ↪'$50,001 - $100,000',
       '$100,001 - $250,000',  '$250,001 - $500,000',
       '$500,001 - $1,000,000','$1,000,001 - $5,000,000',
       '$1,000,000 +', '$5,000,001 - $25,000,000','$50,000,000 +']

pre_process = ColumnTransformer(
    transformers=[
        ('ordinal', OrdinalEncoder(categories = [amount_category]), ['amount']),
        ('one-hot', OneHotEncoder(drop='first'), ['owner','type',␣
 ↪'cap_gains_over_200_usd'])
    ]
)
pipe = Pipeline([
    ('pre_process', pre_process),
    ('neighbors', KNeighborsClassifier(leaf_size= 20, n_neighbors= 4))
])
```

[ ]:
```
hyper_parameters = {
    'n_neighbors': range(1, 5),
    'leaf_size' : range(10, 100, 10)
```

```
}

X_train, X_test, y_train, y_test = train_test_split(pre_process.
 ↪fit_transform(stocks_df), stocks_df['party'])

gs = GridSearchCV(KNeighborsClassifier(), hyper_parameters, cv=5)
gs.fit(X_train, y_train)

print(f'Best GridSearchCV Score: {gs.best_score_}, Best Parameters: {gs.
 ↪best_params_}')
```

```
Best GridSearchCV Score: 0.7080988685777212, Best Parameters: {'leaf_size': 20,
'n_neighbors': 4}
```

2.3 Final Model

[ ]:
```
final_pre_process = ColumnTransformer(
    transformers=[
        ('ordinal', OrdinalEncoder(categories = [amount_category]), ['amount']),
        ('one-hot', OneHotEncoder(drop='first'), ['owner','type',␣
 ↪'cap_gains_over_200_usd', 'over_a_month']),
        ('standard', StandardScaler(), ['days_since_transaction_disclosure'])
    ]
)
final_pipe = Pipeline([
    ('pre_process', final_pre_process),
    ('neighbors', KNeighborsClassifier(leaf_size= 40, n_neighbors= 5))
])
```

[ ]:
```
hyper_parameters = {
    'n_neighbors': range(1, 10),
    'leaf_size' : range(10, 100, 10)
}

X_train, X_test, y_train, y_test = train_test_split(final_pre_process.
 ↪fit_transform(stocks_df), stocks_df['party'])

gs = GridSearchCV(KNeighborsClassifier(), hyper_parameters, cv=5)
gs.fit(X_train, y_train)

print(f'Best GridSearchCV Score: {gs.best_score_}, Best Parameters: {gs.
 ↪best_params_}')
```

```
Best GridSearchCV Score: 0.7873171896504447, Best Parameters: {'leaf_size': 10,
'n_neighbors': 5}
```

2.4 Fairness Analysis

Null Hypothesis : Our model is fair. It's accuracy of prediction for making a cap gain over 200 and not making a cap gain over 200 are roughly the same.

Alternate Hypothesis : Our model is not fair. It's accuracy of prediction for making a cap gain over 200 is greater than the accuracy for not making a cap gain over 200.

[ ]:
```
def calc_dif_accuracy(X,y,predictor):
    cap_gains_over_200_usd = X['cap_gains_over_200_usd']

    X_cap_gain = X[cap_gains_over_200_usd]
    X_no_cap_gain = X[~cap_gains_over_200_usd]


    accuracy_cap_gain = (predictor.predict(X_cap_gain) ==␣
 ↪y[cap_gains_over_200_usd]).mean()
    accuracy_no_cap_gain = (predictor.predict(X_no_cap_gain) ==␣
 ↪y[~cap_gains_over_200_usd]).mean()

    difference_of_accuracy = accuracy_cap_gain - accuracy_no_cap_gain
    return difference_of_accuracy
```

[ ]:
```
X, y = stocks_df.drop('party', axis=1), stocks_df['party']
predictor = final_pipe.fit(X, y)

observed_precision = calc_dif_accuracy(X, y, predictor)
observed_precision
```

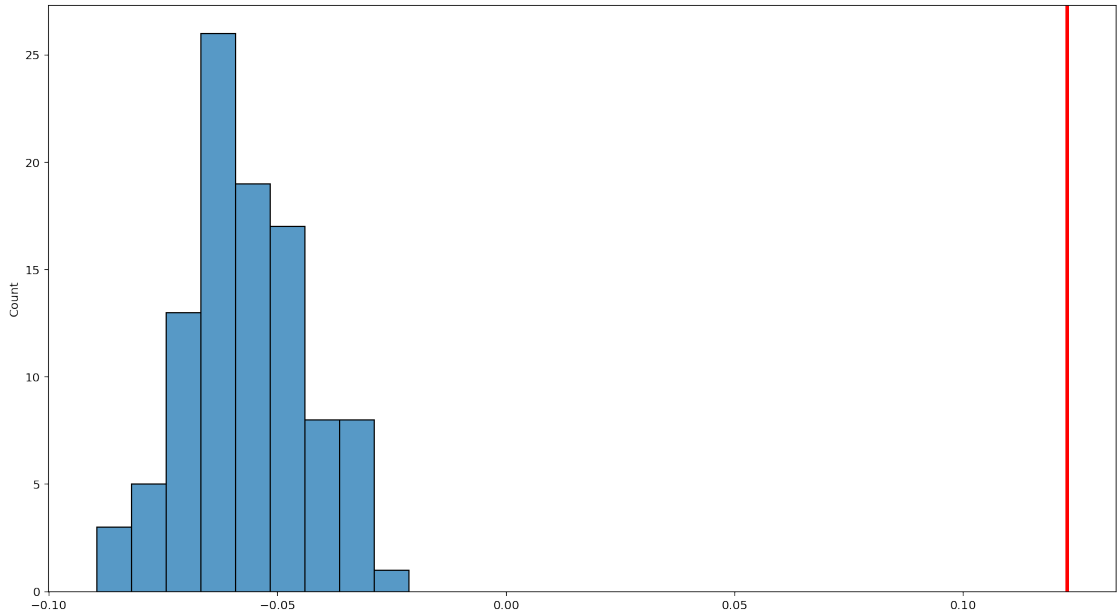
[ ]:
```
0.12274399248598378
```

[ ]:
```
n_reps = 100
differences = []

shuffled_X = X.copy()
cap_gains_over_200_usd = stocks_df['cap_gains_over_200_usd']

for _ in range(n_reps):
    shuffled_X['cap_gains_over_200_usd'] = np.random.
 ↪permutation(cap_gains_over_200_usd)
    differences.append(calc_dif_accuracy(shuffled_X, y, predictor))
```

[ ]:
```
plt.figure(figsize=(16,9))
sns.histplot(data=differences)
plt.axvline(x = observed_precision, color='red', linewidth=3);
```



[ ]:
```
p_val = (differences > observed_precision).mean()
print(f'The P-Value is {p_val}')
```

```
The P-Value is 0.0
```