Party Affiliation Prediction

Michael Chen, Siyu Chen

Summary of Findings

Introduction

After some basic EDA with the stock trading data, we like to build some models that predict the party affiliation of the stockholders. This is a typical binary classification problem. The response variable is the party affiliation of the stockholders (democrat or republican). We would use Accuracy as the metric for evaluating the model because we don't value FN or FP over each other, but simply want the model that produce as many accurate predictions as possible.

Baseline Model

The model we used is a Logistic Regression model with features of transaction_year and amount. We examined in the EDA that there is correlation between party affiliation and transaction_year and amount. We used accuracy as metric for evaluating the model. For the train set, the accuracy is around 0.71. For the test set, the accuracy is 0.72.

Final Model

Two extra features are if one trade has made over 200 usd gains and the number of times ticker has been traded. They are good for data and prediction because accuracy has improved after adding these features, especially the 4th one. The final model we used was a Decision Tree with max_depth=18, min_samples_split=15, and criterion='gini' as the best parameters. The model was compared to Logistic Regression and KNN, and it outperformed them without Grid Search. Then to find the best parameters, we used Grid Search over 140 combinations of parameters.

Fairness Analysis

We want to check whether the prediction model is fair based on number of tickers when making predictions. So we manually created two group given that the number of tickers of one group is higher than average and the other group is below average. The observed differnce in accuracy is 0.05678...which means that group that have higher number of tickers will also have higher accuracy when predicting. Then we did a permutaion test to see whether that phenomenon is just caused by chance. After 1000 simulations, the p-value of the permutation test is 0.035 which is lower then our significance level. So we proved that the model does not acheive accuray parity based on number of tickers.

Code

In [1]:
```
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
import seaborn as sns
%config InlineBackend.figure_format = 'retina'  # Higher resolution figures
```

Feature Engineering

In [2]:
```
merged_df = pd.read_csv('merged.csv')
merged_df = merged_df.dropna(subset=['Party'])
merged_df['Party'] = merged_df['Party'].map({'Democratic': 0, 'Republican': 1})
merged_df['transaction_year'] = merged_df['transaction_date'].apply(lambda x: x.split('-')[0])
merged_df['transaction_month'] = merged_df['transaction_date'].apply(lambda x: x.split('-')[1])
merged_df['transactions_representatives'] = merged_df.groupby('representative')['transaction_year'].transform('c
merged_df['transactions_tickers'] = merged_df.groupby('ticker')['transaction_year'].transform('count')

merged_df
```

Out[2]:

	disclosure_year	disclosure_date	transaction_date	owner	ticker	asset_description	type	amount	representative	district	
0	2021	2021-08-26	2012-06-19	NaN	BLFSD	BioLife Solutions Inc	purchase	1, 001 – 15,000	tom malinowski	NJ07	https clerk.house.g
1	2022	2022-03-03	2017-09-05	NaN	SUP	Superior Industries International Inc Common S...	purchase	1, 001 – 15,000	thomas suozzi	NY03	https clerk.house.g
2	2022	2022-03-03	2017-12-06	NaN	CAT	Caterpillar Inc	purchase	1, 001 – 15,000	thomas suozzi	NY03	https clerk.house.g
3	2022	2022-03-03	2018-04-17	NaN	BA	Boeing Company	purchase	15, 001– 50,000	thomas suozzi	NY03	https clerk.house.g
4	2022	2022-03-03	2018-04-30	NaN	CTRL	Control4 Corporation	purchase	1, 001 – 15,000	thomas suozzi	NY03	https clerk.house.g
...	...	...	...	...	...	...	...	...	...	...	
14268	2022	2022-05-04	2022-04-28	joint	MMP	Magellan Midstream Partners LP Limited Partner...	purchase	1, 001 – 15,000	virginia foxx	NC05	https clerk.house.g
14269	2022	2022-05-04	2022-04-29	joint	TTE	TotalEnergies Inc	purchase	1, 001 – 15,000	virginia foxx	NC05	https clerk.house.g
14270	2022	2022-05-04	2022-04-29	joint	ASML	ASML Holding NV - New York Registry Shares	purchase	1, 001 – 15,000	kathy manning	NC06	https clerk.house.g

	disclosure_year	disclosure_date	transaction_date	owner	ticker	asset_description	type	amount	representative	district	
14271	2022	2022-05-04	2022-04-29	joint	V	Visa Inc	sale_partial	1, 001 – 15,000	kathy manning	NC06	https clerk.house.g
14272	2022	2022-05-08	2022-05-04	joint	NaN	Sales Tax Securitization Corp 5% Due 1/1/2027	sale_full	250, 001– 500,000	suzan k. delbene	WA01	https clerk.house.g

14229 rows × 17 columns

Train Test Split

In [17]:
```python
x_train, x_test, y_train, y_test = train_test_split(merged_df.drop(['Party'], axis=1),\
                                                    merged_df['Party'],test_size=0.2, random_state=42)
```

Baseline Model:

Logistic Regression

Features:

- transaction_year
- amount

In [4]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline

x_train_base = x_train[['transaction_year', 'amount']]
x_test_base = x_test[['transaction_year', 'amount']]
y_train_base = y_train
y_test_base = y_test
pl = Pipeline([('ohe', OneHotEncoder(handle_unknown='ignore')),
                ('clf', LogisticRegression(max_iter=1000))])
pl.fit(x_train_base, y_train_base)
train_accuracy = pl.score(x_train_base, y_train_base)
y_pred_test = pl.predict(x_test_base)
test_accuracy = np.mean(y_pred_test == y_test)
print('Train accuracy:', train_accuracy)
print('Test accuracy:', test_accuracy)
```

```
Train accuracy: 0.7136958622507248
Test accuracy: 0.7210119465917076
```

Final Model

Features Tried:

- Type (no significant improvement)
- Representative, district and number of transactions by representatives cause data leakage because they expose reprsentatives information.

Features Added:

- cap_gains_over_200_usd
- number of transactions by tickers

3rd feature: cap_gains_over_200_usd (Average Accuracy 0.01 higher than baseline)

In [5]:
```
x_train_final = x_train[['transaction_year', 'amount', 'cap_gains_over_200_usd']]
x_test_final = x_test[['transaction_year', 'amount', 'cap_gains_over_200_usd']]
y_train_final = y_train
y_test_final = y_test
pl = Pipeline([('ohe', OneHotEncoder(handle_unknown='ignore')),
                ('clf', LogisticRegression(max_iter=1000))])
pl.fit(x_train_final, y_train_final)
train_accuracy = pl.score(x_train_final, y_train_final)
y_pred_final = pl.predict(x_test_final)
test_accuracy = np.mean(y_pred_final == y_test)
print('Train accuracy:', train_accuracy)
print('Test accuracy:', test_accuracy)
```

```
Train accuracy: 0.7245892998330844
Test accuracy: 0.7280393534785664
```

4th feature: transactions_tickers (Average Accuracy 0.03 higher than baseline)

In [6]:
```
x_train_final = x_train[['transaction_year', 'amount', 'cap_gains_over_200_usd', 'transactions_tickers']]
x_test_final = x_test[['transaction_year', 'amount', 'cap_gains_over_200_usd', 'transactions_tickers']]
pl = Pipeline([('ohe', OneHotEncoder(handle_unknown='ignore')),
                ('clf', LogisticRegression(max_iter=1000))])
pl.fit(x_train_final, y_train_final)
train_accuracy = pl.score(x_train_final, y_train_final)
y_pred_final = pl.predict(x_test_final)
test_accuracy = np.mean(y_pred_final == y_test)
print('Train accuracy:', train_accuracy)
print('Test accuracy:', test_accuracy)
```

```
Train accuracy: 0.7447070192392163
Test accuracy: 0.7512297962052003
```

Selected Model:

Decision Tree Classifier (Average Accuracy 0.03 higher than Logistic Regression and KNN)

In [7]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier

pl = Pipeline([('ohe', OneHotEncoder(handle_unknown='ignore')),
               ('tree', DecisionTreeClassifier())])
pl.fit(x_train_final, y_train_final)
train_accuracy = pl.score(x_train_final, y_train_final)
y_pred_final = pl.predict(x_test_final)
test_accuracy = np.mean(y_pred_final == y_test)
print('Decision Tree Classifier:')
print('Train accuracy:', train_accuracy)
print('Test accuracy:', test_accuracy)

pl = Pipeline([('ohe', OneHotEncoder(handle_unknown='ignore')),
               ('tree', (KNeighborsClassifier()))])
pl.fit(x_train_final, y_train_final)
train_accuracy = pl.score(x_train_final, y_train_final)
y_pred_final = pl.predict(x_test_final)
test_accuracy = np.mean(y_pred_final == y_test)
print('\n')
print('KNeighborsClassifier:')
print('Train accuracy:', train_accuracy)
print('Test accuracy:', test_accuracy)
```

```
Decision Tree Classifier:
Train accuracy: 0.7852938592638145
Test accuracy: 0.7716092761770906


KNeighborsClassifier:
Train accuracy: 0.7550733550030747
Test accuracy: 0.7519325368938862
```

Best Parameters (Grid Search):

In [8]:
```
from sklearn.model_selection import GridSearchCV

hyperparameters = {
    'tree__max_depth': [2, 3, 4, 5, 7, 10, 13, 15, 18, None],
    'tree__min_samples_split': [2, 3, 5, 7, 10, 15, 20],
    'tree__criterion': ['gini', 'entropy']
}

pl = Pipeline([('ohe', OneHotEncoder(handle_unknown='ignore')),
                ('tree', DecisionTreeClassifier())])
pl.fit(x_train_final, y_train_final)
train_accuracy = pl.score(x_train_final, y_train_final)
y_pred_final = pl.predict(x_test_final)
test_accuracy = np.mean(y_pred_final == y_test)

searcher = GridSearchCV(pl, hyperparameters, cv=5)
searcher.fit(x_train_final, y_train_final)
print('Best parameters:', searcher.best_params_)
```

```
Best parameters: {'tree__criterion': 'gini', 'tree__max_depth': 18, 'tree__min_samples_split': 7}
```

In [9]:
```
pl = Pipeline([('ohe', OneHotEncoder(handle_unknown='ignore')),
                ('tree', DecisionTreeClassifier(max_depth=18, min_samples_split=15, criterion='gini'))])
pl.fit(x_train_final, y_train_final)
train_accuracy_final = pl.score(x_train_final, y_train_final)
y_pred_final = pl.predict(x_test_final)
test_accuracy_final = np.mean(y_pred_final == y_test)
print('Train accuracy:', train_accuracy_final)
print('Test accuracy:', test_accuracy_final)
```

```
Train accuracy: 0.7737854695598699
Test accuracy: 0.7698524244553759
```

Fairness Analysis

In [11]:
```
from sklearn import metrics
mean_tickers = merged_df.groupby('representative').count()['transactions_tickers'].mean()

results = x_test_final.copy()
results['ticker_above_avg'] = results['transactions_tickers'] > mean_tickers
results['actual'] = y_test
results['pred'] = y_pred_final
```

In [12]:
```
acc_df = results.groupby('ticker_above_avg').apply(lambda x: metrics.accuracy_score(x['actual'], x['pred']))
acc_df
```

Out[12]:
```
ticker_above_avg
False    0.765512
True     0.821918
dtype: float64
```

In [13]:
```
obs = acc_df.diff().iloc[-1]
obs
```

Out[13]:
```
0.05640581735507455
```

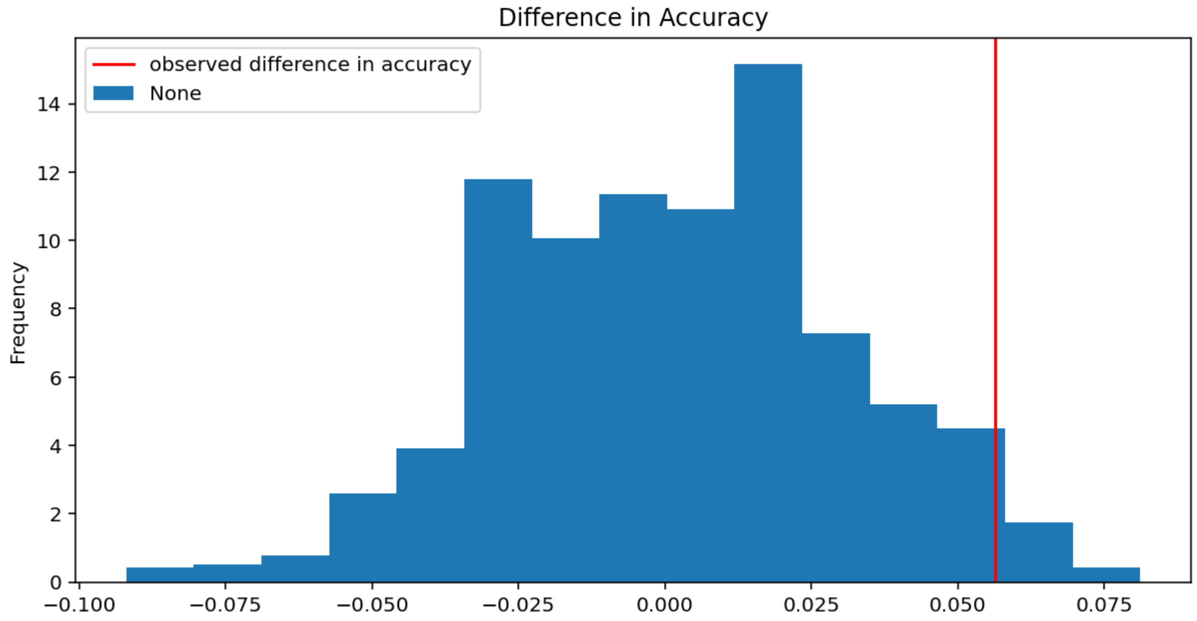
Permutation Test

- Null hypothesis: The classifier's accuracy is the same for both number of tickers that is above and number of tickers that is below average, and any differences are due to chance.
- Hypothesis: The accuracy is higher for data that number of tickers is higher than average.
- Test Statistic: The difference in accuracy
- Significance Level: 0.05

In [19]:

```
np.random.seed(123)
diff_in_acc = []

copy_for_shuffle = results.copy()
for _ in range(1000):

    shuffled_ticker_num = np.random.permutation(copy_for_shuffle['ticker_above_avg'].values)

    copy_for_shuffle['shuffled'] = shuffled_ticker_num

    acc = copy_for_shuffle.groupby('shuffled') \
                          .apply(lambda x: metrics.accuracy_score(x['actual'], x['pred'])) \
                          .diff() \
                          .iloc[-1]
    diff_in_acc.append(acc)
```

In [20]:
```
plt.figure(figsize=(10, 5))
pd.Series(diff_in_acc).plot(kind='hist', density=True, bins=15, title='Difference in Accuracy')
plt.axvline(x=obs, color='red', label='observed difference in accuracy')
plt.legend(loc='upper left');
```



In [21]:
```
p_value = np.mean(obs <= diff_in_acc)
p_value
```

Out[21]: 0.035

Result:

Under significance level of 0.05, the p-value of the permutation test is 0.035, which will reject the null hypothesis. The result shows that there is existing unfainess in the prediction model that it does not achieve accuracy parity.

In [ ]: