stocks5

December 8, 2022

1 Stock Trades by Members of the US House of Representatives

- **See the main project notebook for instructions to be sure you satisfy the rubric!**
- See Project 03 for information on the dataset.
- A few example prediction questions to pursue are listed below. However, don't limit yourself to them!
  - Can you predict the party affiliation of a representative from their stock trades?
  - Can you predict the geographic region that the representative comes from using their stock trades? E.g., west coast, east coast, south, etc.
  - Can you predict whether a particular trade is a BUY or SELL?

Be careful to justify what information you would know at the "time of prediction" and train your model using only those features.

2 Summary of Findings

2.0.1 Introduction

Stocks Data

Dataset Name: Stock Trades by Members of the US House of Representatives

Link to Dataset: https://housestockwatcher.com/api

Number of Observations: 15699

Column Description: - disclosure_year: The year in which a company released important information about itself which may influence stock buyer's decisions. - disclosure_date: The date on which a company released important information about itself which may influence stock buyer's decisions. - transaction_date: The date that a stock was purchased/sold/exchanged. - owner: The type of ownership the US House of Representative had on a stock. Joint means that the stock is owned by multiple people, self means that the stock is only owned by one person - ticker: The symbol for the company - asset_description: The full name of the ticker (the company name) - type: The type of transaction the US House of Representative made (either a purchase, sale, or exchange) - amount: The amount of shares purchased/sold/exchanged - representative: The congress member who bought the stock - district: The district the congress member is representing - ptr_link: A link to where the data came from - cap_gains_over_200_usd: Whether or not someone made capital gains over 200 usd on their stock. (True indicates that they did make a gain over 200 usd, False indicates that they did not).

We plan on using the stocks dataset in order to solve the classification problem to predict the owner of a stock trade by an individual. The Response Variable for the above model is stocks_df.owner, i.e., the owner column in the stocks dataframe. The owner column has the unique values of joint, self, and dependent. In order to make the predictions for this columns values we will be using a RandomForestClassifier. Moreover, in the response variable the missing values will be removed altogether as their absence will not be of help for the model. The columns from the dataframe that will be used as features are as follows: - Nominal Categorial- One Hot Encoded using OneHotEncoder() - stocks_df.ticker - stocks_df.district - Nominal Categorical- Transformed into 0/1 columns using a FunctionTransformer() - stocks_df.cap_gains_over_200_usd - Numerical- There are no missing values present in the numerical column - stocks_df.disclosure_year - stocks_df.amount

- The success of the model devised will be based on the mean accuracy of the prediction result generated
- The fairness analysis will be conducted on the stocks_df.type column, because there are different types of transactions taking place which can affect the ownership status of the stocks

2.0.2 Baseline Model

Based on the features mentioned above that will be used in the model it is observed that: - There are three columns which are stocks_df.ticker, stocks_df.district, and stocks_df.cap_gains_over_200_usd which are categorical (nominal) columns such that stocks_df.cap_gains_over_200_usd is a binary column with 0/1 values, while stocks_df.ticker, and stocks_df.district are non-binary columns being used - There are two columns which are numerical columns such that stocks_df.amount is a continuous column, and stocks_df.disclosure_year is a discrete column

The result of the baseline model produced is 0.9721 proportion of accuracy indicating that the model produced has a high level of accuracy.

- In order to improve the accuracy of the model further there would be a need to ensure a greater consistency among the values in the owner column, because the number of values which are dependent is far fewer than other values of the owner column which could introduce bias in the model as the model may detect dependent fewer times owing to that.
- In order to improve the accuracy of the model further if there other features available in the dataset that would've been useful to train the model, because currently many of the columns had to be dropped since they were merely repetitive.

2.0.3 Final Model

Based on the features mentioned above that will be used in the model it is observed that: - There are three columns which are stocks_df.ticker, stocks_df.district, and stocks_df.cap_gains_over_200_usd which are categorical (nominal) columns such that stocks_df.cap_gains_over_200_usd is a binary column with 0/1 values, while stocks_df.ticker, and stocks_df.district are non-binary columns being used - There are two columns which are numerical columns such that stocks_df.amount is a continuous column, and stocks_df.disclosure_year is a discrete column

It is observed that the result of the final model is 0.9216 proportion of accuracy indicating that the model has produced a high level of accuracy.

- In order to improve the accuracy of the model further there would be a need to ensure more diversity in terms of the available features in the dataframe as the available data only consists of a few rows given that there were numerous missing values in the column values being predicted by the model, and the values have too high a discrepancy as observed with dependent value being less frequently observed in the dataframe than the other values available for owner due to which the model is biased.

2.0.4 Fairness Evaluation

The fairness evaluation would be conducted on the year columns in the stocks dataframe, because it is observed that there are years indicated for pandemic as well as post-pandemic years, and the pandemic may have affected the stock ownership status.

Hypothesis Test: A permutation test would be conducted for fairness evaluation. - Null Hypothesis: The accuracy, and recall scores are the same between the years of stock trades taking place. - Alternate Hypothesis: The accuracy, and recall scores are different between the years of stock trades taking place. - Significance level (alpha)=0.05

Conclusion: - From the below result it is observed that the p-value for accuracy scores is about 0.23 indicating that it is above the significance level, so null hypothesis can't be rejected for accuracy scores. - From the below result it is observed that the p-value for recall scores is about 0.24 indicating that it is above the significance level, so null hypothesis can't be rejected for recall scores.

Thus, both the accuracy, and recall scores are evenly spread out among the various years of stock trades observed in the stocks dataframe.

3 Code

[202]:

```
import matplotlib.pyplot as plt
import numpy as np
import os
import re
import pandas as pd
import seaborn as sns
%matplotlib inline
%config InlineBackend.figure_format = 'retina'  # Higher resolution figures
#read in the dataset
stocks_df=pd.read_csv('all_transactions.csv')
stocks_df
```

[202]:

```
       disclosure_year disclosure_date transaction_date  owner ticker  \
0                 2021      10/04/2021       2021-09-27  joint     BP
1                 2021      10/04/2021       2021-09-13  joint    XOM
2                 2021      10/04/2021       2021-09-10  joint   ILPT
3                 2021      10/04/2021       2021-09-28  joint     PM
4                 2021      10/04/2021       2021-09-17   self    BLK
…                  …             …                …     …     …
15694             2020      06/10/2020       2020-04-09     --    SWK
```

```
15695             2020      06/10/2020       2020-04-09     --    USB
15696             2020      06/10/2020       2020-03-13    NaN    BMY
15697             2020      06/10/2020       2020-03-13    NaN    LLY
15698             2020      06/10/2020       2020-03-13    NaN    DIS

                                       asset_description          type  \
0                                                 BP plc      purchase
1                                Exxon Mobil Corporation      purchase
2      Industrial Logistics Properties Trust - Common…      purchase
3                       Phillip Morris International Inc      purchase
4                                          BlackRock Inc  sale_partial
…                                                  …           …
15694                       Stanley Black & Decker, Inc.  sale_partial
15695                                       U.S. Bancorp  sale_partial
15696                       Bristol-Myers Squibb Company     sale_full
15697                              Eli Lilly and Company     sale_full
15698                                Walt Disney Company     sale_full

                      amount            representative district  \
0           $1,001 - $15,000        Hon. Virginia Foxx     NC05
1           $1,001 - $15,000        Hon. Virginia Foxx     NC05
2          $15,001 - $50,000        Hon. Virginia Foxx     NC05
3          $15,001 - $50,000        Hon. Virginia Foxx     NC05
4           $1,001 - $15,000    Hon. Alan S. Lowenthal     CA47
…                      …                       …    …
15694       $1,001 - $15,000        Hon. Ed Perlmutter     CO07
15695       $1,001 - $15,000        Hon. Ed Perlmutter     CO07
15696    $100,001 - $250,000  Hon. Nicholas Van Taylor     TX03
15697  $500,001 - $1,000,000  Hon. Nicholas Van Taylor     TX03
15698    $250,001 - $500,000  Hon. Nicholas Van Taylor     TX03

                                                ptr_link  \
0      https://disclosures-clerk.house.gov/public_dis…
1      https://disclosures-clerk.house.gov/public_dis…
2      https://disclosures-clerk.house.gov/public_dis…
3      https://disclosures-clerk.house.gov/public_dis…
4      https://disclosures-clerk.house.gov/public_dis…
…                                                  …
15694  https://disclosures-clerk.house.gov/public_dis…
15695  https://disclosures-clerk.house.gov/public_dis…
15696  https://disclosures-clerk.house.gov/public_dis…
15697  https://disclosures-clerk.house.gov/public_dis…
15698  https://disclosures-clerk.house.gov/public_dis…

       cap_gains_over_200_usd
0                       False
1                       False
```

```
2                        False
3                        False
4                        False
…                        …
15694                    False
15695                    False
15696                    False
15697                    False
15698                    False

[15699 rows x 12 columns]
```

[203]:

```
#Values in type column
vals=stocks_df['owner'].unique()
vals
```

[203]:

```
array(['joint', 'self', nan, 'dependent', '--'], dtype=object)
```

[204]:

```
#replacing characters in the string in amount column
stocks=stocks_df.copy()
stocks['amount']=stocks['amount'].str.replace('$','')
stocks['amount']=stocks['amount'].str.replace(',','')
stocks['amount']=stocks['amount'].str.replace('-','')
stocks['amount']=stocks['amount'].str.split()
stocks
```

[204]:

```
       disclosure_year disclosure_date transaction_date  owner ticker  \
0                 2021      10/04/2021       2021-09-27  joint     BP
1                 2021      10/04/2021       2021-09-13  joint    XOM
2                 2021      10/04/2021       2021-09-10  joint   ILPT
3                 2021      10/04/2021       2021-09-28  joint     PM
4                 2021      10/04/2021       2021-09-17   self    BLK
…                …             …              …    …    …
15694             2020      06/10/2020       2020-04-09     --    SWK
15695             2020      06/10/2020       2020-04-09     --    USB
15696             2020      06/10/2020       2020-03-13    NaN    BMY
15697             2020      06/10/2020       2020-03-13    NaN    LLY
15698             2020      06/10/2020       2020-03-13    NaN    DIS

                                       asset_description          type   \
0                                                 BP plc      purchase
1                                Exxon Mobil Corporation      purchase
2      Industrial Logistics Properties Trust - Common…      purchase
3                       Phillip Morris International Inc      purchase
4                                          BlackRock Inc  sale_partial
…                                                  …           …
15694                       Stanley Black & Decker, Inc.  sale_partial
```

```
15695                                       U.S. Bancorp  sale_partial
15696                       Bristol-Myers Squibb Company     sale_full
15697                              Eli Lilly and Company     sale_full
15698                                Walt Disney Company     sale_full

                  amount            representative district  \
0          [1001, 15000]        Hon. Virginia Foxx     NC05
1          [1001, 15000]        Hon. Virginia Foxx     NC05
2         [15001, 50000]        Hon. Virginia Foxx     NC05
3         [15001, 50000]        Hon. Virginia Foxx     NC05
4          [1001, 15000]    Hon. Alan S. Lowenthal     CA47
…                  …                       …      …
15694      [1001, 15000]        Hon. Ed Perlmutter     CO07
15695      [1001, 15000]        Hon. Ed Perlmutter     CO07
15696   [100001, 250000]  Hon. Nicholas Van Taylor     TX03
15697  [500001, 1000000]  Hon. Nicholas Van Taylor     TX03
15698   [250001, 500000]  Hon. Nicholas Van Taylor     TX03

                                                ptr_link  \
0      https://disclosures-clerk.house.gov/public_dis…
1      https://disclosures-clerk.house.gov/public_dis…
2      https://disclosures-clerk.house.gov/public_dis…
3      https://disclosures-clerk.house.gov/public_dis…
4      https://disclosures-clerk.house.gov/public_dis…
…                                                   …
15694  https://disclosures-clerk.house.gov/public_dis…
15695  https://disclosures-clerk.house.gov/public_dis…
15696  https://disclosures-clerk.house.gov/public_dis…
15697  https://disclosures-clerk.house.gov/public_dis…
15698  https://disclosures-clerk.house.gov/public_dis…

       cap_gains_over_200_usd
0                       False
1                       False
2                       False
3                       False
4                       False
…                       …
15694                   False
15695                   False
15696                   False
15697                   False
15698                   False

[15699 rows x 12 columns]
```

[205]:
```
#looping through the values in the amount column to find the median for each␣
 ↪value of the column
lst=[]
for i in stocks['amount']:
    if len(i)==2:
        if (i[0].isnumeric()) and (i[1].isnumeric()):
            summed=int(i[0])+int(i[1])
            lst.append(summed/2)
        else:
            lst.append(int(i[0]))
    else:
        lst.append(int(i[0]))
len(lst)
```

[205]:
```
15699
```

[206]:
```
#storing the median values in the column instead of the stringed interval values
stocks_df['amount']=lst
stocks_df
```

[206]:
```
       disclosure_year disclosure_date transaction_date  owner ticker  \
0                 2021      10/04/2021       2021-09-27  joint     BP
1                 2021      10/04/2021       2021-09-13  joint    XOM
2                 2021      10/04/2021       2021-09-10  joint   ILPT
3                 2021      10/04/2021       2021-09-28  joint     PM
4                 2021      10/04/2021       2021-09-17   self    BLK
…                    …               …                …     …     …
15694             2020      06/10/2020       2020-04-09     --    SWK
15695             2020      06/10/2020       2020-04-09     --    USB
15696             2020      06/10/2020       2020-03-13    NaN    BMY
15697             2020      06/10/2020       2020-03-13    NaN    LLY
15698             2020      06/10/2020       2020-03-13    NaN    DIS

                                   asset_description         type  \
0                                             BP plc     purchase
1                           Exxon Mobil Corporation     purchase
2      Industrial Logistics Properties Trust - Common…    purchase
3                       Phillip Morris International Inc     purchase
4                                       BlackRock Inc  sale_partial
…                                                   …            …
15694                        Stanley Black & Decker, Inc.  sale_partial
15695                                     U.S. Bancorp  sale_partial
15696                        Bristol-Myers Squibb Company     sale_full
15697                               Eli Lilly and Company     sale_full
15698                                 Walt Disney Company     sale_full

         amount            representative district  \
0        8000.5        Hon. Virginia Foxx     NC05
1        8000.5        Hon. Virginia Foxx     NC05
2       32500.5        Hon. Virginia Foxx     NC05
3       32500.5        Hon. Virginia Foxx     NC05
4        8000.5    Hon. Alan S. Lowenthal     CA47
…         …                       …      …
15694    8000.5        Hon. Ed Perlmutter     CO07
15695    8000.5        Hon. Ed Perlmutter     CO07
15696  175000.5  Hon. Nicholas Van Taylor     TX03
15697  750000.5  Hon. Nicholas Van Taylor     TX03
15698  375000.5  Hon. Nicholas Van Taylor     TX03

                                                ptr_link  \
0      https://disclosures-clerk.house.gov/public_dis…
1      https://disclosures-clerk.house.gov/public_dis…
2      https://disclosures-clerk.house.gov/public_dis…
3      https://disclosures-clerk.house.gov/public_dis…
4      https://disclosures-clerk.house.gov/public_dis…
…                                                  …
15694  https://disclosures-clerk.house.gov/public_dis…
15695  https://disclosures-clerk.house.gov/public_dis…
15696  https://disclosures-clerk.house.gov/public_dis…
15697  https://disclosures-clerk.house.gov/public_dis…
15698  https://disclosures-clerk.house.gov/public_dis…

       cap_gains_over_200_usd
0                       False
1                       False
2                       False
3                       False
4                       False
…                       …
15694                   False
15695                   False
15696                   False
15697                   False
15698                   False

[15699 rows x 12 columns]
```

[207]:

```
#removing the absurd values from transaction_date column
stocks_df=stocks_df[stocks_df['transaction_date']!='20222-08-09']
stocks_df
```

[207]:

```
       disclosure_year disclosure_date transaction_date  owner ticker  \
0                 2021      10/04/2021       2021-09-27  joint     BP
1                 2021      10/04/2021       2021-09-13  joint    XOM
```

```
2                 2021      10/04/2021       2021-09-10  joint   ILPT
3                 2021      10/04/2021       2021-09-28  joint     PM
4                 2021      10/04/2021       2021-09-17   self    BLK
…                …             …              …    …    …
15694             2020      06/10/2020       2020-04-09     --    SWK
15695             2020      06/10/2020       2020-04-09     --    USB
15696             2020      06/10/2020       2020-03-13    NaN    BMY
15697             2020      06/10/2020       2020-03-13    NaN    LLY
15698             2020      06/10/2020       2020-03-13    NaN    DIS

                                       asset_description          type  \
0                                                 BP plc      purchase
1                                Exxon Mobil Corporation      purchase
2      Industrial Logistics Properties Trust - Common…      purchase
3                       Phillip Morris International Inc      purchase
4                                          BlackRock Inc  sale_partial
…                                                  …           …
15694                       Stanley Black & Decker, Inc.  sale_partial
15695                                       U.S. Bancorp  sale_partial
15696                       Bristol-Myers Squibb Company     sale_full
15697                              Eli Lilly and Company     sale_full
15698                                Walt Disney Company     sale_full

         amount            representative district  \
0        8000.5        Hon. Virginia Foxx     NC05
1        8000.5        Hon. Virginia Foxx     NC05
2       32500.5        Hon. Virginia Foxx     NC05
3       32500.5        Hon. Virginia Foxx     NC05
4        8000.5    Hon. Alan S. Lowenthal     CA47
…         …                       …      …
15694    8000.5        Hon. Ed Perlmutter     CO07
15695    8000.5        Hon. Ed Perlmutter     CO07
15696  175000.5  Hon. Nicholas Van Taylor     TX03
15697  750000.5  Hon. Nicholas Van Taylor     TX03
15698  375000.5  Hon. Nicholas Van Taylor     TX03

                                                ptr_link  \
0      https://disclosures-clerk.house.gov/public_dis…
1      https://disclosures-clerk.house.gov/public_dis…
2      https://disclosures-clerk.house.gov/public_dis…
3      https://disclosures-clerk.house.gov/public_dis…
4      https://disclosures-clerk.house.gov/public_dis…
…                                                  …
15694  https://disclosures-clerk.house.gov/public_dis…
15695  https://disclosures-clerk.house.gov/public_dis…
15696  https://disclosures-clerk.house.gov/public_dis…
15697  https://disclosures-clerk.house.gov/public_dis…
```

```
15698  https://disclosures-clerk.house.gov/public_dis…

       cap_gains_over_200_usd
0                       False
1                       False
2                       False
3                       False
4                       False
…                         …
15694                   False
15695                   False
15696                   False
15697                   False
15698                   False

[15698 rows x 12 columns]
```

[208]:

```
#keeping only non-null owner values in the dataframe
stocks_df=stocks_df[(stocks_df['owner']=='joint')|(stocks_df['owner']=='self')|(stocks_df['own
stocks_df
```

[208]:

```
       disclosure_year disclosure_date transaction_date  owner ticker  \
0                 2021      10/04/2021       2021-09-27  joint     BP
1                 2021      10/04/2021       2021-09-13  joint    XOM
2                 2021      10/04/2021       2021-09-10  joint   ILPT
3                 2021      10/04/2021       2021-09-28  joint     PM
4                 2021      10/04/2021       2021-09-17   self    BLK
…                  …             …                …     …
15689             2020      06/10/2020       2020-04-22   self   AAPL
15690             2020      06/10/2020       2020-04-22   self   COST
15691             2020      06/10/2020       2020-03-18   self   COST
15692             2020      06/10/2020       2020-04-22   self     FB
15693             2020      06/10/2020       2020-04-22   self    KMI

                               asset_description          type  \
0                                         BP plc      purchase
1                         Exxon Mobil Corporation      purchase
2      Industrial Logistics Properties Trust - Common…      purchase
3                    Phillip Morris International Inc      purchase
4                                   BlackRock Inc  sale_partial
…                                             …            …
15689                                  Apple Inc.     sale_full
15690                   Costco Wholesale Corporation  sale_partial
15691                   Costco Wholesale Corporation      purchase
15692                      Facebook, Inc. - Class A     sale_full
15693                           Kinder Morgan, Inc.     sale_full
```

```
        amount          representative district  \
0       8000.5      Hon. Virginia Foxx     NC05
1       8000.5      Hon. Virginia Foxx     NC05
2      32500.5      Hon. Virginia Foxx     NC05
3      32500.5      Hon. Virginia Foxx     NC05
4       8000.5  Hon. Alan S. Lowenthal     CA47
…          …                     …      …
15689   8000.5      Hon. Ed Perlmutter     CO07
15690   8000.5      Hon. Ed Perlmutter     CO07
15691   8000.5      Hon. Ed Perlmutter     CO07
15692   8000.5      Hon. Ed Perlmutter     CO07
15693   8000.5      Hon. Ed Perlmutter     CO07

                                                ptr_link  \
0      https://disclosures-clerk.house.gov/public_dis…
1      https://disclosures-clerk.house.gov/public_dis…
2      https://disclosures-clerk.house.gov/public_dis…
3      https://disclosures-clerk.house.gov/public_dis…
4      https://disclosures-clerk.house.gov/public_dis…
…                                                    …
15689  https://disclosures-clerk.house.gov/public_dis…
15690  https://disclosures-clerk.house.gov/public_dis…
15691  https://disclosures-clerk.house.gov/public_dis…
15692  https://disclosures-clerk.house.gov/public_dis…
15693  https://disclosures-clerk.house.gov/public_dis…

       cap_gains_over_200_usd
0                       False
1                       False
2                       False
3                       False
4                       False
…                         …
15689                   False
15690                   False
15691                   False
15692                   False
15693                   False

[8351 rows x 12 columns]
```

3.0.1 Baseline Model

[209]:
```
#dividing the columns
response_col=stocks_df['owner']
Id_col=stocks_df[['representative']]
```

```
dropped_cols=stocks_df.
 ↪drop(['disclosure_date','asset_description','ptr_link','type'],axis=1)
numbers=dropped_cols[['disclosure_year','amount']]
binarys=dropped_cols[['cap_gains_over_200_usd']]
onehoten=dropped_cols[['ticker','district']]
dated=dropped_cols[['transaction_date']]
```

[210]:

```
#packages to import
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import FunctionTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.impute import SimpleImputer
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import binarize
```

[211]:

```
#transformer builder cell
binary_bool=lambda x: x.replace({True:1, False: 0})
numer=lambda x:x.fillna(0)
ohe_tf=Pipeline([('impute', SimpleImputer(strategy='constant',␣
 ↪fill_value='Other')), ('ohe', OneHotEncoder(handle_unknown='ignore')),])
tf=ColumnTransformer([('ohe_tf',ohe_tf, ['ticker','district']), ('binary',␣
 ↪FunctionTransformer(binary_bool),['cap_gains_over_200_usd']),('numeric',␣
 ↪FunctionTransformer(numer),['disclosure_year','amount'])])
```

[212]:

```
#data split into train, and test datasets
X, y=dropped_cols, response_col
X_train, X_test, y_train, y_test= train_test_split(X, y, random_state=10,␣
 ↪test_size=0.4)
```

[213]:

```
ohe_tf.fit(dropped_cols[['ticker','district']])
```

[213]:

```
Pipeline(memory=None,
         steps=[('impute',
                 SimpleImputer(add_indicator=False, copy=True,
                               fill_value='Other', missing_values=nan,
                               strategy='constant', verbose=0)),
                ('ohe',
                 OneHotEncoder(categories='auto', drop=None,
                               dtype=<class 'numpy.float64'>,
                               handle_unknown='ignore', sparse=True))],
         verbose=False)
```

[214]:

```
tf.fit(X_train, y_train)
```

[214]:
```
ColumnTransformer(n_jobs=None, remainder='drop', sparse_threshold=0.3,
                  transformer_weights=None,
                  transformers=[('ohe_tf',
                                 Pipeline(memory=None,
                                          steps=[('impute',
SimpleImputer(add_indicator=False,
                                                                copy=True,
fill_value='Other',
missing_values=nan,
strategy='constant',
                                                                verbose=0)),
                                                 ('ohe',
OneHotEncoder(categories='auto',
                                                                drop=None,
                                                                dtype=<class
'numpy.float64'>,
                                                                han…
                                                     func=<function <lambda> at
0x7f95aedf0ca0>,
                                                     inv_kw_args=None,
                                                     inverse_func=None,
                                                     kw_args=None,
                                                     validate=False),
                                 ['cap_gains_over_200_usd']),
                                ('numeric',
                                 FunctionTransformer(accept_sparse=False,
                                                     check_inverse=True,
                                                     func=<function <lambda> at
0x7f95aedf0b80>,
                                                     inv_kw_args=None,
                                                     inverse_func=None,
                                                     kw_args=None,
                                                     validate=False),
                                 ['disclosure_year', 'amount'])],
                  verbose=False)
```

[215]:
```
#building the pipeline
p=Pipeline([('tf', tf), ('clf',␣
 ↪RandomForestClassifier(n_estimators=100,criterion='gini',max_depth=None,min_samples_split=2
p.fit(X_train, y_train)
```

[215]:
```
Pipeline(memory=None,
         steps=[('tf',
                 ColumnTransformer(n_jobs=None, remainder='drop',
                                   sparse_threshold=0.3,
                                   transformer_weights=None,
                                   transformers=[('ohe_tf',
                                                  Pipeline(memory=None,
                                                           steps=[('impute',
SimpleImputer(add_indicator=False,
 copy=True,
 fill_value='Other',
 missing_values=nan,
 strategy='constant',
 verbose=0)),
                                                                  ('ohe',
OneHotEncoder(categories='auto',
 drop=Non…
                 RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                        class_weight=None, criterion='gini',
                                        max_depth=None, max_features='auto',
                                        max_leaf_nodes=None, max_samples=None,
                                        min_impurity_decrease=0.0,
                                        min_impurity_split=None,
                                        min_samples_leaf=1, min_samples_split=2,
                                        min_weight_fraction_leaf=0.0,
                                        n_estimators=100, n_jobs=-1,
                                        oob_score=False, random_state=None,
                                        verbose=0, warm_start=False))],
         verbose=False)
```

[216]:

```
#accuracy of the model
score=p.score(X_train, y_train)
score
```

[216]:

```
0.9720558882235529
```

[217]:

```
#actual counts for ownership of stocks
predicted=p.predict(X)
y.value_counts()
```

[217]:

```
joint        4938
self         3013
dependent     400
Name: owner, dtype: int64
```

[218]:

```
#prediction counts for ownership of stocks
ser=pd.Series(predicted)
ser.value_counts()
```

[218]:

```
joint        4900
self         3115
dependent     336
dtype: int64
```

3.0.2 Final Model

[223]:
```
#data split into train, and test datasets
X1, y1=dropped_cols, response_col
X1_train, X1_test, y1_train, y1_test=train_test_split(X1, y1, random_state=10,␣
 ↪test_size=0.4)
```

[224]:
```
#forming the model with the new features devised not previously present
binary_bool=lambda x: x.replace({True:1, False: 0})
numer=lambda x:x.fillna(0)
ohe_tf=Pipeline([('impute', SimpleImputer(strategy='constant',␣
 ↪fill_value='Other')), ('ohe', OneHotEncoder(handle_unknown='ignore')),])
tf2=ColumnTransformer([('ohe_tf',ohe_tf, ['ticker','district']), ('binary',␣
 ↪FunctionTransformer(binary_bool),['cap_gains_over_200_usd']),('numeric',FunctionTransformer
#building the pipeline
p1=Pipeline([('tf',␣
 ↪tf2),('clf',RandomForestClassifier(n_estimators=100,criterion='gini',max_depth=None,min_sam
p1.fit(X1_train, y1_train)
```

[224]:
```
Pipeline(memory=None,
         steps=[('tf',
                 ColumnTransformer(n_jobs=None, remainder='drop',
                                   sparse_threshold=0.3,
                                   transformer_weights=None,
                                   transformers=[('ohe_tf',
                                                  Pipeline(memory=None,
                                                           steps=[('impute',
SimpleImputer(add_indicator=False,
 copy=True,
 fill_value='Other',
 missing_values=nan,
 strategy='constant',
 verbose=0)),
                                                                  ('ohe',
OneHotEncoder(categories='auto',
 drop=Non…
                 RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                        class_weight=None, criterion='gini',
                                        max_depth=None, max_features='auto',
                                        max_leaf_nodes=None, max_samples=None,
                                        min_impurity_decrease=0.0,
                                        min_impurity_split=None,
                                        min_samples_leaf=1, min_samples_split=2,
                                        min_weight_fraction_leaf=0.0,
                                        n_estimators=100, n_jobs=-1,
                                        oob_score=False, random_state=None,
                                        verbose=0, warm_start=False))],
         verbose=False)
```

[225]:

```
#accuracy of the model with the previous pipeline
p1.score(X1_test, y1_test)
```

[225]:

```
0.9209817419934151
```

[227]:

```
#building a new pipeline for the model
p2=Pipeline([('tf',tf2),('clf',␣
 ↪RandomForestClassifier(n_estimators=900,criterion='entropy',max_depth=None,min_samples_spli
p2.fit(X1_train, y1_train)
```

[227]:

```
Pipeline(memory=None,
         steps=[('tf',
                 ColumnTransformer(n_jobs=None, remainder='drop',
                                   sparse_threshold=0.3,
                                   transformer_weights=None,
                                   transformers=[('ohe_tf',
                                                  Pipeline(memory=None,
                                                           steps=[('impute',
SimpleImputer(add_indicator=False,
 copy=True,
 fill_value='Other',
 missing_values=nan,
 strategy='constant',
 verbose=0)),
                                                                  ('ohe',
OneHotEncoder(categories='auto',
 drop=Non…
                 RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                        class_weight=None, criterion='entropy',
                                        max_depth=None, max_features='auto',
                                        max_leaf_nodes=None, max_samples=None,
                                        min_impurity_decrease=0.0,
                                        min_impurity_split=None,
                                        min_samples_leaf=1, min_samples_split=2,
                                        min_weight_fraction_leaf=0.0,
                                        n_estimators=900, n_jobs=-1,
                                        oob_score=False, random_state=None,
                                        verbose=0, warm_start=False))],
         verbose=False)
```

[302]:

```
#prediction counts for ownership of stocks
predicted1=p2.predict(X1_test)
predicted1
```

```
[302]: array(['self', 'self', 'joint', …, 'self', 'joint', 'dependent'],
             dtype=object)
```

```
[229]: #accuracy of the model with the new pipeline
       p2.score(X1_test, y1_test)
```

```
[229]: 0.9215803651601316
```

3.0.3 Fairness Evaluation

```
[230]: #importing the packages for fairness evaluations
       from sklearn import metrics
       from sklearn.preprocessing import KBinsDiscretizer
```

```
[231]: #accuracy score observed
       acc=metrics.accuracy_score(predicted1, y1_test)
       acc
```

```
[231]: 0.9215803651601316
```

```
[256]: #recall score observed
       recall_val=metrics.recall_score(predicted1,y1_test, average='weighted')
       recall_val
```

```
[256]: 0.9215803651601316
```

```
[261]: #precision score observed
       prec_val=metrics.precision_score(predicted1,y1_test,average='weighted')
       prec_val
```

```
[261]: 0.9236822245427132
```

```
[294]: df=pd.DataFrame()
       df['prediction']=predicted1
       df['observation']=y1_test
       df['year']=X1_test['disclosure_year']
       df=df.dropna()
```

```
[295]: kb=KBinsDiscretizer(n_bins=8,encode='ordinal',strategy='quantile')
       df['year_bins']=kb.fit_transform(df[['year']])
```

```
/opt/conda/lib/python3.8/site-
packages/sklearn/preprocessing/_discretization.py:195: UserWarning: Bins whose
width are too small (i.e., <= 1e-8) in feature 0 are removed. Consider
decreasing the number of bins.
  warnings.warn('Bins whose width are too small (i.e., <= '
```

[297]:

```
df['vals']=(df['year_bins']<=5).replace({True:'small',False:'large'})
accurate=df.groupby('vals').apply(lambda x: metrics.accuracy_score(x.
 ↪observation, x.prediction)).rename('accuracy').to_frame()
recalled=df.groupby('vals').apply(lambda x: metrics.recall_score(x.observation,␣
 ↪x.prediction,average='weighted')).rename('recall').to_frame()
display(accurate)
display(recalled)
```

```
       accuracy
vals
small  0.454416
```

```
         recall
vals
small  0.454416
```

[298]:

```
observed_val=accurate.iloc[-1,0]
observed_val
```

[298]: 0.4544159544159544

[299]:

```
#permutation test for accuracy scores hypothesis test
lst=[]
for i in range(300):
    l=df[['vals','prediction','observation']].assign(vals=df['vals'].
 ↪sample(frac=1,replace=False).reset_index()).groupby('vals').apply(lambda x:
 ↪metrics.accuracy_score(x.observation,x.prediction)).diff().iloc[-1]
    lst.append(l)
#p-value observed from the calculation
p_val=(np.array(lst)>=observed_val).mean()
p_val
```

[299]: 0.22666666666666666

[300]:

```
observed_val1=recalled.iloc[-1,0]
#permutation test for recall scores hypothesis test
lst1=[]
for i in range(300):
    l1=df[['vals','prediction','observation']].assign(vals=df['vals'].
 ↪sample(frac=1,replace=False).reset_index()).groupby('vals').apply(lambda x:
 ↪metrics.recall_score(x.observation,x.prediction, average='weighted')).diff().
 ↪iloc[-1]
    lst1.append(l1)
#p-value observed from the calculation
p_val1=(np.array(lst1)>=observed_val1).mean()
p_val1
```

```
/opt/conda/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1272:
UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in labels
with no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
[300]: 0.23666666666666666
```