Stock Trade Classifier

February 4, 2023

1 Introduction to this Report

This is a project assignment from DSC 80 at UCSD instructed by Professor Justin Eldridge during Fall 2022. In collaboration with Zhengyun Nie(znie@ucsd.edu), as a group, we analyzed the stock trades of the US House of Representatives. Using the clean data we obtained from the previous analysis, we built up a decision tree classifier model which predicts whether a transaction if purchare or sale given 12 features with ≈ 68 accurracy. Moreover, we analzed the fairness of our model with respect to the titles of different representatives.

2 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.

3 Summary of Findings

3.0.1 Introduction

The prediction problem we are attempting is to predict whether the type of a trade is a purchase or a sell. This is a classification questions. Our choice of target variable is the type columns in the dataframe as it directly tells whether it is a purchase or sell. The stocks has three main type: sale_full, sale_partial and purchase. We assign 1 to purchase and 0 to any kind of sales and disgard the rest minor noise. Finally, we use accuracy as our evaluation metric.

3.0.2 Baseline Model

We use 10 categorical features, where 6 of them are ordinal and the rest of them are nominal. For the nominal variables(district, name etc.), we apply one-hot encoder. Then, we directly pass in the cleaned ordinal features.

Using a decision tree classifier, we obtain a training score of 0.8828014184397163 and the test score is 0.6777452805105025.

Our model performs very well as the training score indicates that we are not overfitting or underfitting and the test score is also approriate as the ratio of purchase in the dataframe is around 0.5. Hence, our model should have an accuracy higher than 0.5 otherwise it is better to just guess all the type is a purchase.

3.0.3 Final Model

To improve the performance of our model, we add a feature called 'span' which is the number of days between the transaction_date and disclosure_date. We believe this is helpful since purchasing of stocks might have a tendency of not exposing so quickly.

Another feature,'lower bound', which represents the lower bound of the amount of a trade, is also introduced. Using decision tree classifier and with the help of Grid Search, we find the best parameter which in our case is: criterion='entropy' ,max_depth = None, min_samples_split = 2

3.0.4 Fairness Evaluation

We evaluate fairness on different titles(Hon., Mr., Mrs., None). Our parity measure is accuracy. The accuracy for people with different titles are exactly the same as 0.687317. Hence, we get a fair model with respect to titles.

4 Code

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

4.0.1 Import Data

[2]:
```
stocks = pd.read_csv('stocks_data.csv')
stocks.head()
```

[2]:

	disclosure_year	disclosure_date	transaction_date	owner	ticker	asset_description	type
0	2021	2021-10-04	2021-09-27	joint	BP	BP plc	purchase
1	2021	2021-10-04	2021-09-13	joint	XOM		
2	2021	2021-10-04	2021-09-10	joint	ILPT		
3	2021	2021-10-04	2021-09-28	joint	PM		
4	2021	2021-10-04	2021-09-17	self	BLK		

```
1                          Exxon Mobil Corporation      purchase
2  Industrial Logistics Properties Trust - Common…      purchase
3                  Phillip Morris International Inc      purchase
4                                       BlackRock Inc  sale_partial

              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  …

  transaction_weekday  transaction_year  transaction_month  transaction_day  \
0               Mondy              2021                  9               27
1               Mondy              2021                  9               13
2              Friday              2021                  9               10
3             Tuesday              2021                  9               28
4              Friday              2021                  9               17

  title        full_name  state  amount_lower_range amount_upper_range  \
0  Hon.          V. Foxx     NC              1001.0            15000.0
1  Hon.          V. Foxx     NC              1001.0            15000.0
2  Hon.          V. Foxx     NC             15001.0            50000.0
3  Hon.          V. Foxx     NC             15001.0            50000.0
4  Hon.  A. S. Lowenthal     CA              1001.0            15000.0

  day_of_week
0           0
1           0
2           4
3           1
4           4

[5 rows x 24 columns]
```

[3]:
```
# include data with type as purchase or any kind of sale
stocks = stocks[stocks['type'] != 'exchange']
stocks = stocks[stocks['type'] != 'sale']
# create a column records whether the trade is buy or not
stocks['is_buy'] = (stocks['type'] == 'purchase').astype(int)
# turn 'cap_gains_over_200_usd' to 0 and 1
stocks['cap_gains_over_200_usd'] = stocks['cap_gains_over_200_usd'].astype(int)
stocks.head()
```

[3]:
```
   disclosure_year disclosure_date transaction_date  owner ticker  \
0             2021      2021-10-04       2021-09-27  joint     BP
1             2021      2021-10-04       2021-09-13  joint    XOM
```

```
2             2021      2021-10-04       2021-09-10  joint   ILPT
3             2021      2021-10-04       2021-09-28  joint     PM
4             2021      2021-10-04       2021-09-17   self    BLK

                                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

              amount          representative district  … transaction_year  \
0   $1,001 - $15,000      Hon. Virginia Foxx     NC05  …              2021
1   $1,001 - $15,000      Hon. Virginia Foxx     NC05  …              2021
2  $15,001 - $50,000      Hon. Virginia Foxx     NC05  …              2021
3  $15,001 - $50,000      Hon. Virginia Foxx     NC05  …              2021
4   $1,001 - $15,000  Hon. Alan S. Lowenthal     CA47  …              2021

   transaction_month  transaction_day  title        full_name  state  \
0                  9               27   Hon.          V. Foxx     NC
1                  9               13   Hon.          V. Foxx     NC
2                  9               10   Hon.          V. Foxx     NC
3                  9               28   Hon.          V. Foxx     NC
4                  9               17   Hon.  A. S. Lowenthal     CA

   amount_lower_range  amount_upper_range day_of_week is_buy
0              1001.0             15000.0           0      1
1              1001.0             15000.0           0      1
2             15001.0             50000.0           4      1
3             15001.0             50000.0           1      1
4              1001.0             15000.0           4      0

[5 rows x 25 columns]
```

4.0.2 Baseline Model

[4]:
```
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import FunctionTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OrdinalEncoder
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
```

[5]:
```
# split train and test set
X = stocks[['owner', 'amount', 'district', 'cap_gains_over_200_usd',\
            'transaction_year', 'transaction_month', 'transaction_day',␣
 ↪'day_of_week', 'title', 'full_name']]
y = stocks['is_buy']
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

[6]:
```
# pre-processing
preproc = ColumnTransformer(
    transformers = [
        ('one_hot', OneHotEncoder(handle_unknown = 'ignore'),␣
 ↪['owner','amount', 'district', 'title', 'full_name'])
    ], remainder = 'passthrough'
)
```

[7]:
```
# define pipeline
pl = Pipeline([
    ('preproc', preproc),
    ('clf', DecisionTreeClassifier())
])
```

[8]:
```
pl.fit(X_train, y_train)
```

[8]:
```
Pipeline(steps=[('preproc',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('one_hot',
OneHotEncoder(handle_unknown='ignore'),
                                                  ['owner', 'amount',
                                                   'district', 'title',
                                                   'full_name'])])),
                ('clf', DecisionTreeClassifier())])
```

[9]:
```
pl.score(X_train, y_train)
```

[9]:
```
0.8820921985815603
```

[10]:
```
pl.score(X_test,y_test)
```

[10]:
```
0.6793406009040149
```

4.0.3 Final Model

[11]:
```
stocks['span'] = (pd.to_datetime(stocks['disclosure_date']) - pd.
 ↪to_datetime(stocks['transaction_date'])).apply(lambda x: x.days)
stocks['lower bound'] = (stocks['amount'].str.split(' - ')).apply(lambda x:␣
 ↪x[0].strip('$'))
```

[12]:
```
X_train, X_test, y_train, y_test = train_test_split(stocks.drop(columns =␣
↪['is_buy']), stocks['is_buy'])
```

[13]:
```
# split train and test set
x_col = ['amount_lower_range','transaction_year', 'transaction_month',␣
↪'ticker', 'district', 'lower bound', 'span']
X_train2 = X_train[x_col]
X_test2 = X_test[x_col]
# X_train, X_test, y_train, y_test = train_test_split(X, y)
```

[14]:
```
def ordinal(df):
    scale = {"1,001":1, '1,001 -':1, "15,001":2, "50,001":3, '100,001':4,␣
↪'250,001':5, '500,001':6, '1,000,001':7,
            '1,000,000 +':7, '5,000,001':8,'1,000':1, '15,000':3, '50,000,000␣
↪+':9, '1,000,000':7}
    return pd.DataFrame(df[df.columns[0]].replace(scale))
ordinal_hot = FunctionTransformer(ordinal)
```

[15]:
```
# pre-processing
preproc = ColumnTransformer(
    transformers = [
        ('ordinal_hot' , ordinal_hot, ['lower bound']),
        ('z-scale', StandardScaler(), ['span','amount_lower_range']),
        ('one_hot', OneHotEncoder(handle_unknown = 'ignore'),␣
↪['transaction_year', 'transaction_month','ticker','district'])
    ], remainder = 'passthrough'
)
```

[16]:
```
# define pipeline
p2 = Pipeline([
    ('preproc', preproc),
    ('clf', DecisionTreeClassifier())
])
```

[17]:
```
p2.fit(X_train2, y_train)
p2.score(X_train2, y_train), p2.score(X_test2,y_test)
```

[17]:
```
(0.9925531914893617, 0.693166710981122)
```

[18]:
```
import itertools

hyperparameters = {
    'clf__max_depth': [10,20,30,40,50,60,70,80,None],
    'clf__min_samples_split': [2, 3, 4, 6, 8, 10, 12, 14, 16, 18],
    'clf__criterion': ['gini', 'entropy']
}
```

[19]:
```
searcher = GridSearchCV(p2, hyperparameters, cv=5)
searcher.fit(X_train2, y_train)
searcher.best_params_
```

[19]:
```
{'clf__criterion': 'gini', 'clf__max_depth': None, 'clf__min_samples_split': 3}
```

[20]:
```
p3 = Pipeline([
    ('preproc', preproc),
    ('clf', DecisionTreeClassifier(criterion='gini',max_depth = None,␣
 ↪min_samples_split = 4))
])
```

[21]:
```
p3.fit(X_train2, y_train)
p3.score(X_train2, y_train), p3.score(X_test2,y_test)
```

[21]:
```
(0.9763297872340425, 0.6873172028715767)
```

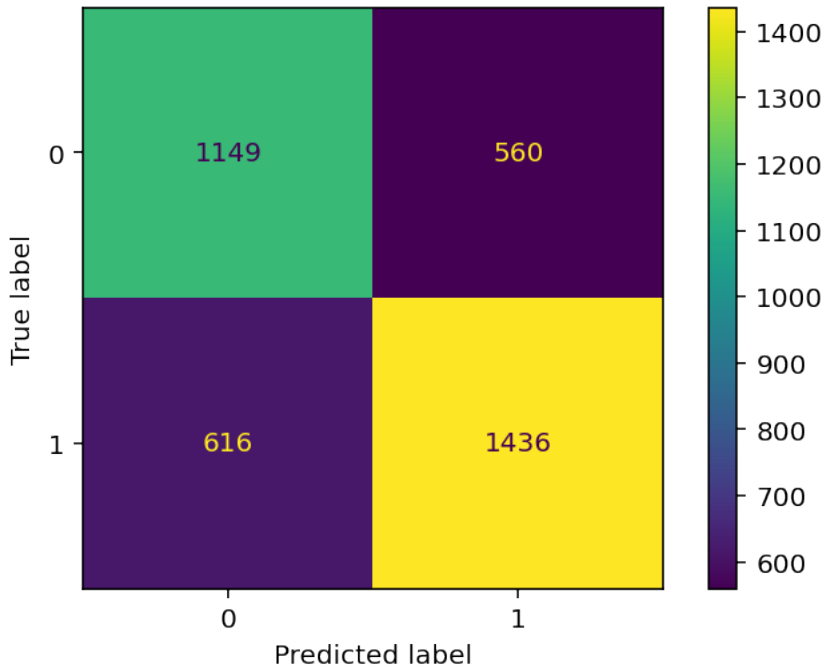
4.0.4 Fairness Evaluation

[22]:
```
from sklearn import metrics
```

[23]:
```
metrics.plot_confusion_matrix(p3, X_test2, y_test)
```

```
/Users/maowanting/opt/anaconda3/lib/python3.9/site-
packages/sklearn/utils/deprecation.py:87: FutureWarning: Function
plot_confusion_matrix is deprecated; Function `plot_confusion_matrix` is
deprecated in 1.0 and will be removed in 1.2. Use one of the class methods:
ConfusionMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

[23]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7fc1b8476cd0>
```



[24]:
```
results = X_test2
results['pred'] = p3.predict(X_test2)
results['title'] = X_test['title']

# accuracy for transactions of diff state
(
    results
    .groupby('title')
    .apply(lambda x: metrics.accuracy_score(y_test, results['pred']))
    .rename('accuracy')
    .to_frame()
)
```

```
/var/folders/62/t5nbbj213758m93dpkrfvlj40000gn/T/ipykernel_64050/648552205.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  results['pred'] = p3.predict(X_test2)
/var/folders/62/t5nbbj213758m93dpkrfvlj40000gn/T/ipykernel_64050/648552205.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  results['title'] = X_test['title']
```

[24]:

title	accuracy
Hon.	0.687317
Mr.	0.687317
Mrs.	0.687317
None	0.687317