NYPD Allegations

- **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!
    - Predict the outcome of an allegation (might need to feature engineer your output column).
    - Predict the complainant or officer ethnicity.
    - Predict the amount of time between the month received vs month closed (difference of the two columns).
    - Predict the rank of the officer.

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

Summary of Findings

Introduction

In this project, we will use the data set which contains more than 12,000 civilian complaints filed against New York City. This data sets contains some basic informations for every cases it stored. That information is about age, gender, allegation type, id, ethinicity, fado types, the contact reasons and the outcome of this allegation. As given this data set, we will most focus on the prediction question:

Prediction the outcome of an allegation based on these basic information.

Using the accuracy as the measurement to evaluate the model. By the assistant of the classification report, we can also check if the model overfits the data. Due to we want to predict the outcome and which is a categorical variable, so this is an typical classifcation problem. So we cannot use the regression models. Instead this project will go as the following order:

1) Creating a naive model or a baseline model which uses RandomForestClassifier model. And set the target variabel as board_disposition. The features used here are:

month_received, year_received, month_closed, year_closed, rank_incident, complainant_ethnicity, complainant_gender, complainant_age_incident, fado_type, allegation, precinct

2) Analyzing the accuracy and classification report gained in the baseline model, set several directions of choosing the final model.

3) Improve the baseline model and decide the final model

4) Evaluating the fairness of the final model

Baseline Model

In this part, we used the RandomForestClassifier model as the predictio model. And setting board_disposition (the outcome of the allegation) as target variable. Since this is just a baseline model. So we choose:

month_received, year_received, month_closed, year_closed, rank_incident, complainant_ethnicity, complainant_gender, complainant_age_incident, fado_type, allegation, precinct

as features based on our understanding of the data set. And we seperated the feature to:

Quantitative Features: complainant_age_incident

Ordinal Features: rank_incident

Nominal Features: complainant_ethinicity, complainant_gender, fado_type, allegation, board_deposition, precinct

Using the 75% - 25% ratio as the training data - testing data, then we gained the accuracy 0.521, which means 52.1% predictions are correct. Comparing with the accuracies we learned from the lectures, this number is not good at all. In other words, this model is not a good predition model. And combining with the information of the complexity report, we found the consistency between the accuracy and precision, recall, and f-1 score. This means the model does not overfit the data, which is a good thing.

Standing at this point, we cannot say, this model is not a good model. Since it is just a naive model or baseline model. But we indeed can gain some hints or directions to improve for the following parts.

The potential reasons inducing this low accuracy are:

1) Choosing the wrong model

2) Features choice is not good

3) Various types of "Substantiated" outcomes reduce the overall accuracy of the model.

So following this two possible reasons. We can improve the baseline model by:

1) Trying several different models

2) Improving the features by adding or deleting some. Or creating some new and more representative features based on the data we have.

3) Combing all types of "Substantiated" outcomes into one.

Final Model

In this section, we followed the hints and directions gained from the baseline model. We started at improvement of the RandomForestClassifier model.

For features perspective, we added:

rank_abbrev_incident, mos_ethnicity, mos_gender, mos_age_incident, first_name, last_name

to the original feature list. Since the "mos" characteristics might also affect the prediction result due to sometimes the generality might dominant some minority, then affect the prediction. And there might be some certain pattern for the names affecting the prediction. Like some names have higher chance to some certain allegation outcome. Beyond that, we think the year and month that received and closed is not typical and strong. So we based on that created a brand-new feature -- duration which measures the time lasting from the case received to closed by month. Having this new feature, we deleted the year_received, month_recieved, month_closed, year_closed so as to avoiding overfitting. Also we added the PCA to the model and combining all "Susstantiated" outcomes into one. Making the total outcomes into three: Substantiated, Unsubstantiated, Exonerated.

By doing these improvements, we do get a better accuracy which is 0.5601. Even a slightly better accuracy, that means we derived the right direction from the baseline model.

Then we utilized GridSearchCV to find the best fit. And found the best fit occured at criterion= 'gini', max_depth= 6, min_samples_leaf= 5,min_samples_split=5. At this case the accuracy is 0.513. It is similiar with the accuracy of baseline model. By ploting the histogram of accuracies on Validation set for CV, we know that the region of accuracy is around (0.465, 0.514). And the STD is quite small which can even be ignored.

Finally we tried two other models: GradientBoostingClassifier model and SVC model. Impressively, GradientBoostingClassifier model produces the highest accuracy which is 0.57. And checking with the classification report, the precision, recall, and f1-score are all the highest among all models. Not to mention, this model does not have the problem of overfitting as well. For SVC model, it is not a approporiate model here, since the lowest accuracy generated and 0 precision, recall and f-1 score for Substantiated, Exonerated.

General and Comprehensive Thinking: After doing all above analysis and predictions, we can easily found that the accuracy of the prediction is all around 0.5 and less than 0.6. This means the model is not quite usable at all in the real life. So what might be the reason inducing this low accuracy no matter what model we use. Looking back to the data set, this data set contains more than 12,000 data points. So the size of data is enough. But as mentioned in the introduction part, this data points only contains some basic informations about the case, the people, and the type of the allegation then the result. Imagining this, if you tell you that there is a 30-aged Asian man who got the Action allegation and due to Moving Violation, and I want to you predict how the outcome will be of this allegation. These does not make sense at all, right! A huge data set does not mean a useful data set. Like this data set we used in this project. We lack more important and signifcant features to predict the outcome and improve the accuracy. The seriousness of same type of allegation is totally different. For example, I beat a guy by accident, and the guy is totaly good in all condition. I am possibly got the allegation. And the case that I beat a guy and almost kill him. There are tons of physical damage on him. They are all same kind of allegation, but totally different. Moreover, thinking about the attitudes and behavior of the guy got allegation during the cort trail. These are too many important features we do not have. And these important features are indeed determines how can the outcome might be. So based on the all information we have, the alomst 57% accuracy is quite good I think. Comparing with guess which has 1/3 possibility or accuracy to be right. Our model have 0.57 which is much larger than 1/3. And in real life more than 50% can significantly affect a person's decision making!

Fairness Evaluation

After find the final model. In this section we focused on the fairness evaluation. Still using accuracy as the key number, we compared the accuracy between elder age group and younger group within the final model. Setting the significance level as 5%, then we have:

Null Hyothesis:

The accuracies among these two group are same.

Alternative Hypothesis:

The accuracies among these two group are not same.

By permutation test, we finally gained the p-value which is 0.37. It is much larger than the default 5% significance level. So we fail to reject the null. That means the model has same accuracy to younger group and older group. The we conclude that the final model we made is not biased.

Code

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
import seaborn as sns
from sklearn.pipeline import Pipeline
from sklearn import preprocessing
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import FunctionTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
%matplotlib inline
%config InlineBackend.figure_format = 'retina'  # Higher resolution fi
```

Baseline Model

In [2]:
```
df = pd.read_csv("allegations_202007271729.csv")
```

In this part, we will use RandomForestClassifier as the naive(Baseline) model for the prediction. The prediction Question here is :

Predicting the outcome of an allegation.

Due to the target variable we are going to predict is a categorical variable. So we cannot use regression here. Then this question is an classification problem.

Target Variable:

board_disposition (the outcome of the allegation)

Features:

month_received, year_received, month_closed, year_closed, rank_incident, complainant_ethnicity, complainant_gender, complainant_age_incident, fado_type, allegation, precinct

Simply Data Processing and Information Gathering

creating a temporary data frame which contains all features that will use in the prediction model.

In [3]:
```
col_lst = ['month_received', 'year_received', 'month_closed', 'year_clo
```

In [4]:
```
newdf = df[col_lst]
newdf.head()
```

Out[4]:

year_received	month_closed	year_closed	rank_incident	complainant_ethnicity	complainant_gen
2019	5	2020	Police Officer	Black	Fen
2011	8	2012	Police Officer	Black	M
2011	8	2012	Police Officer	Black	M
2012	9	2013	Police Officer	Black	M
2018	2	2019	Police Officer	NaN	N

In [5]:
```
newdf.isnull().sum()
```

Out[5]:
```
month_received                0
year_received                 0
month_closed                  0
year_closed                   0
rank_incident                 0
complainant_ethnicity      4464
complainant_gender         4195
complainant_age_incident   4812
fado_type                     0
allegation                    1
precinct                     24
dtype: int64
```

In [6]:
```
newdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33358 entries, 0 to 33357
Data columns (total 11 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   month_received            33358 non-null  int64  
 1   year_received             33358 non-null  int64  
 2   month_closed              33358 non-null  int64  
 3   year_closed               33358 non-null  int64  
 4   rank_incident             33358 non-null  object 
 5   complainant_ethnicity     28894 non-null  object 
 6   complainant_gender        29163 non-null  object 
 7   complainant_age_incident  28546 non-null  float64
 8   fado_type                 33358 non-null  object 
 9   allegation                33357 non-null  object 
 10  precinct                  33334 non-null  float64
dtypes: float64(2), int64(4), object(5)
memory usage: 2.8+ MB
```

Data Transformation

Quantitative Features: complainant_age_incident

Ordinal Features: rank_incident

Nominal Features: complainant_ethinicity, complainant_gender, fado_type, allegation, board_deposition, precinct

In [7]:
```
# fill in null value and one hot encoding
category_ppl = Pipeline([
    ('impute', SimpleImputer(strategy='constant', fill_value='Missing'
    ('ohe', OneHotEncoder(handle_unknown='ignore')),
])

CT = ColumnTransformer([("category", category_ppl, ["rank_incident", "
    ('numeric',SimpleImputer(strategy='mean'), ["complainant_age_incic
])
```

In [8]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

# split the data into training set and test set
X = newdf
y = df.board_disposition
X_tr, X_ts, y_tr, y_ts = train_test_split(X,y, test_size=0.25)
```

Build the model

In [9]:
```
pl = Pipeline([('columntrans', CT), ('clf', RandomForestClassifier())])
pl.fit(X_tr, y_tr)
```

Out[9]:
```
Pipeline(steps=[('columntrans',
                 ColumnTransformer(transformers=[('category',
                                                  Pipeline(steps=[('i
mpute',
                                                                   Si
mpleImputer(fill_value='Missing',

strategy='constant')),
                                                                  ('o
he',
                                                                   On
eHotEncoder(handle_unknown='ignore'))]),
                                                  ['rank_incident',
                                                   'complainant_ethni
city',
                                                   'complainant_gende
r',
                                                   'fado_type', 'alle
gation']),
                                                 ('numeric', SimpleIm
puter(),
                                                  ['complainant_age_i
ncident',
                                                   'precinct'])])),
                ('clf', RandomForestClassifier())])
```

In [10]:
```
accuracy = pl.score(X_ts, y_ts)
accuracy
```

Out[10]:
```
0.5219424460431654
```

In [12]:
```
from sklearn.metrics import classification_report
preds = pl.predict(X_ts)
print(classification_report(y_ts, preds))
```

```
                                          precision    recall  f1-sco
re   support

                              Exonerated       0.53      0.54      0.
53      2451
             Substantiated (Charges)       0.31      0.22      0.
26       938
    Substantiated (Command Discipline A)       0.24      0.16      0.
19       239
    Substantiated (Command Discipline B)       0.31      0.17      0.
22       217
      Substantiated (Command Discipline)       0.21      0.13      0.
16       218
Substantiated (Command Lvl Instructions)       0.48      0.33      0.
39       126
     Substantiated (Formalized Training)       0.26      0.19      0.
22       234
           Substantiated (Instructions)       0.08      0.03      0.
04        64
      Substantiated (No Recommendations)       0.13      0.13      0.
13        38
                         Unsubstantiated       0.60      0.69      0.
64      3815

                                accuracy                           0.
52      8340
                               macro avg       0.31      0.26      0.
28      8340
                            weighted avg       0.50      0.52      0.
51      8340
```

From the above, we can find that the accuracy for the Baseline Model is 0.519 which is not good. And combining the informaton provided from the classification report, the precision, recall, f1-score are also not good. But there is an obvious consistency between the accuracy and the precision, recall and f1-score. That means the model does not overfit the data.

From the above report, we can notice that Unsubstantiated and Exonerated have much higher score compared with those "Substantiated" outcomes. The reason here might be "Substantiated" has to many types. So the performance has been affected.

Gerneally, the Baseline Model do provide us some meaningful informations and directions that can be helpful for following analysis.

1) the potential reason for the low accuracy is the features is not good. We can improve it by adding or deleting one of the feature. And also we can create some new features based on the info we have

2) Another reason is the model we chose here is not a propor one. We can test severl more models in the next section.

Final Model

So as to improve the general model proformance, we first need to combine the outcome of the allegation into three types: Substantiated, Unsubstantiated and Exonerated. The reason to do so is by the classification report from the baseline model, we found that the various types of "Substantiated" do have low precision, recall and f1-score. Then it makes the overall accuracy lower.

Then, in this part we will follow the directions and informationed we gained from the baseline model.

1) We will add more features to the original col_lst, which are:

rank_abbrev_incident, mos_ethnicity, mos_gender, mos_age_incident, first_name, last_name

2) Creating a brand new feature -- duration. This feature measures the time lasting from the case received to closed (measured in month). Same time, deleting the month_received, month_closed, year_received, year_closed

3) Adding the PCA to the model

4) Use GridSearchCV to find the best fit

5) Try GradientBoostingClassifier model

6) Try SVC model

Data Processing

Making the outcome result into Substantiated, Unsubstantiated and Exonerated

In [13]:
```
# the helper function that can combining all 'Substantiated' outcomes
helper = lambda x: 'Substantiated' if 'Substantiated' in x else x
```

In [14]:
```
df['board_disposition'] = df['board_disposition'].apply(helper)
```

In [15]:
```
new_col_lst = ['first_name','last_name','month_received', 'year_receiv
               'rank_incident', 'complainant_ethnicity', 'complainant_
               'fado_type', 'allegation', 'precinct','rank_abbrev_inci
               'mos_age_incident']
```

In [16]:
```
w_df = df[new_col_lst]
n_month = (new_df['year_closed'] * 12 + new_df['month_closed']) - (new_
w_df['duration'] = num_month
w_df = new_df.drop(['month_received', 'year_received', 'month_closed',
w_df.head()
```

```
<ipython-input-16-75f907753069>: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/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  new_df['duration'] = num_month
```

Out[16]:

	first_name	last_name	rank_incident	complainant_ethnicity	complainant_gender	complainar
0	Jonathan	Ruiz	Police Officer	Black	Female	
1	John	Sears	Police Officer	Black	Male	
2	John	Sears	Police Officer	Black	Male	
3	John	Sears	Police Officer	Black	Male	
4	Noemi	Sierra	Police Officer	NaN	NaN	

Adding PCA to the model

In [17]:
```
from sklearn.decomposition import PCA
new_ohe = Pipeline([
    ('impute', SimpleImputer(strategy='constant', fill_value='Missing'))
    ('ohe', OneHotEncoder(handle_unknown='ignore',sparse = False)),
    ('pca', PCA(svd_solver='full', n_components=0.99))
)

new_CT = ColumnTransformer([("new_ohe", new_ohe, ["first_name","last_na
        'mos_gender','allegation',"rank_abbrev_incident"]),
    ('numeric',SimpleImputer(strategy='mean'), ["complainant_age_incide
)
```

In [18]:
```
X = new_df
y = df.board_disposition
X_tr, X_ts, y_tr, y_ts = train_test_split(X,y, test_size=0.25)
```

In [19]:
```
pl = Pipeline([('new_CT', CT), ('clf', RandomForestClassifier())])
pl.fit(X_tr, y_tr)
```

Out[19]:
```
Pipeline(steps=[('new_CT',
                 ColumnTransformer(transformers=[('category',
                                                  Pipeline(steps=[('i
mpute',

                                                                   Si
mpleImputer(fill_value='Missing',

strategy='constant')),
                                                                  ('o
he',
                                                                   On
eHotEncoder(handle_unknown='ignore'))]),
                                                  ['rank_incident',
                                                   'complainant_ethni
city',
                                                   'complainant_gende
r',
                                                   'fado_type', 'alle
gation']),
                                                 ('numeric', SimpleIm
puter(),
                                                  ['complainant_age_i
ncident',
                                                   'precinct'])])),
                ('clf', RandomForestClassifier())])
```

In [20]:
```
accuracy = pl.score(X_ts, y_ts)
accuracy
```

Out[20]:
```
0.5601918465227818
```

In [21]:
```
preds = pl.predict(X_ts)
print(classification_report(y_ts, preds))
```

```
                precision    recall  f1-score   support

     Exonerated       0.53      0.52      0.53      2457
  Substantiated       0.47      0.42      0.44      2085
Unsubstantiated       0.62      0.66      0.64      3798

       accuracy                           0.56      8340
      macro avg       0.54      0.53      0.54      8340
   weighted avg       0.56      0.56      0.56      8340
```

Comparing the result we gained from the baseline model, we get a better accuracy after the improvement. Now the accuracy is 0.555, and the precision, recall, f1-score have a increasing for Exonerated. And these number does not change for Unsubstantiated. But the generally performance for Substantiated has big increasing compared with various type of Substantiated types. These make the overall accuracy increase

Use GridSearchCV to find the best fit

In [22]:
```
from sklearn.model_selection import GridSearchCV
parameters = {
    'clf__max_depth': [3,4,5,6],
    'clf__min_samples_split':[3,5,7,8],
    'clf__min_samples_leaf':[2,3,5,7],
    'clf__criterion': ['gini', 'entropy'],
}

cv = GridSearchCV(pl, parameters, cv=5)
cv.fit(X_tr, y_tr)
```

Out[22]:
```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('new_CT',
                                        ColumnTransformer(transformer
s=[('category',

Pipeline(steps=[('impute',

SimpleImputer(fill_value='Missing',

strategy='constant')),

('ohe',

OneHotEncoder(handle_unknown='ignore'))]),

['rank_incident',

'complainant_ethnicity',

'complainant_gender',

'fado_type',

'allegation']),

('numeric',

SimpleImputer(),

['complainant_age_incident',

'precinct'])])),
                                       ('clf', RandomForestClassifier
())]),
             param_grid={'clf__criterion': ['gini', 'entropy'],
                         'clf__max_depth': [3, 4, 5, 6],
                         'clf__min_samples_leaf': [2, 3, 5, 7],
                         'clf__min_samples_split': [3, 5, 7, 8]})
```

Find the best fit parameters

In [23]:
```
cv.best_params_
```

Out[23]:
```
{'clf__criterion': 'gini',
 'clf__max_depth': 6,
 'clf__min_samples_leaf': 5,
 'clf__min_samples_split': 5}
```

In [24]:
```
T', CT), ('clf', RandomForestClassifier(criterion= 'gini', max_depth=
```

Out[24]:
```
Pipeline(steps=[('new_CT',
                 ColumnTransformer(transformers=[('category',
                                                  Pipeline(steps=[('i
mpute',
                                                                   Si
mpleImputer(fill_value='Missing',

strategy='constant')),
                                                                  ('o
he',
                                                                   On
eHotEncoder(handle_unknown='ignore'))]),
                                                  ['rank_incident',
                                                   'complainant_ethni
city',
                                                   'complainant_gende
r',
                                                   'fado_type', 'alle
gation']),
                                                 ('numeric', SimpleIm
puter(),
                                                  ['complainant_age_i
ncident',
                                                   'precinct'])])),
                ('clf',
                 RandomForestClassifier(max_depth=6, min_samples_leaf
=5,
                                        min_samples_split=5))])
```

In [25]:
```
accuracy = pl.score(X_ts, y_ts)
accuracy
```

Out[25]:
```
0.5137889688249401
```

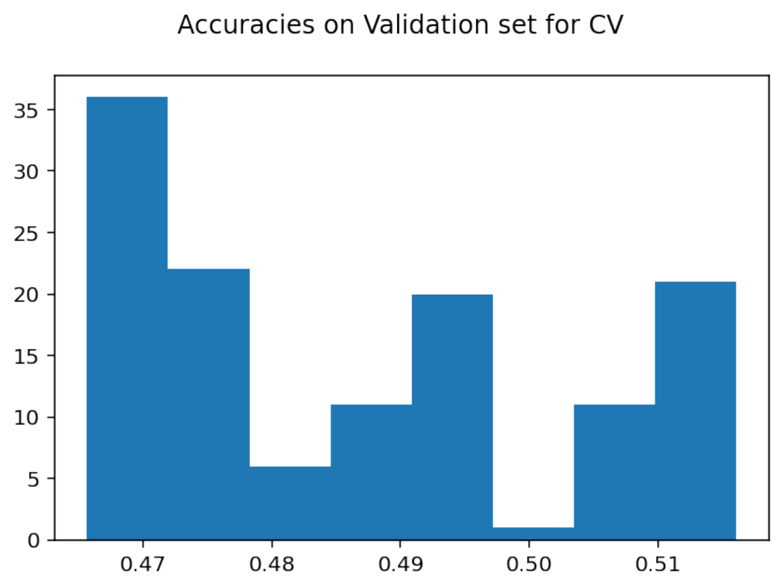
In [26]:
```
preds = pl.predict(X_ts)
print(classification_report(y_ts, preds))
```

```
                precision    recall  f1-score   support

     Exonerated       0.55      0.38      0.45      2457
   Substantiated      0.75      0.03      0.06      2085
Unsubstantiated       0.50      0.86      0.63      3798

       accuracy                           0.51      8340
      macro avg       0.60      0.43      0.38      8340
   weighted avg       0.58      0.51      0.44      8340
```

We can find that even the best fit, the overall accuracy is not higher than the baseline model. But noticablly, the precision of Substantiated outcome has a huge improvement, which is even higher than prvious.

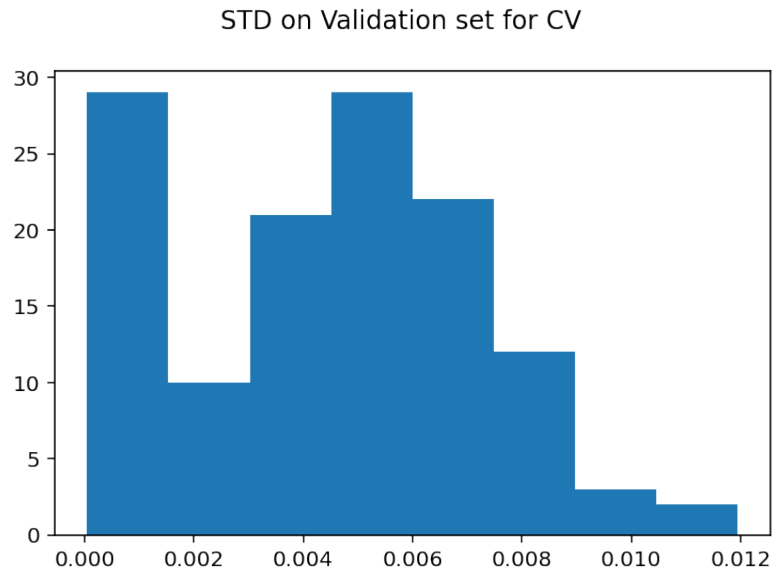
In [27]:
```
plt.hist(cv.cv_results_['mean_test_score'],bins = 8)
plt.suptitle('Accuracies on Validation set for CV')
```

Out[27]:
```
Text(0.5, 0.98, 'Accuracies on Validation set for CV')
```



```
In [28]: plt.hist(cv.cv_results_['std_test_score'],bins = 8)
         plt.suptitle('STD on Validation set for CV')
```

```
Out[28]: Text(0.5, 0.98, 'STD on Validation set for CV')
```



From the two histograms above, we can know that the region of the accuracies on Validation set for CV is around (0.465, 0.514). And the STD is quite small which can even be ignored.

Try the GradientBoostingClassfier model

In [29]:
```
from sklearn.ensemble import GradientBoostingClassifier
pl2 = Pipeline([('new_CT', CT), ('gbc', GradientBoostingClassifier())]
pl2.fit(X_tr, y_tr)
```

Out[29]:
```
Pipeline(steps=[('new_CT',
                 ColumnTransformer(transformers=[('category',
                                                  Pipeline(steps=[('i
mpute',
                                                                   Si
mpleImputer(fill_value='Missing',

strategy='constant')),
                                                                  ('o
he',
                                                                   On
eHotEncoder(handle_unknown='ignore'))]),
                                                  ['rank_incident',
                                                   'complainant_ethni
city',
                                                   'complainant_gende
r',
                                                   'fado_type', 'alle
gation']),
                                                 ('numeric', SimpleIm
puter(),
                                                  ['complainant_age_i
ncident',
                                                   'precinct'])])),
                ('gbc', GradientBoostingClassifier())])
```

Accuracy for GradientBoostingClassfier model

In [30]:
```
pl2.score(X_ts, y_ts)
```

Out[30]:
```
0.5694244604316546
```

In [31]:
```python
preds = pl2.predict(X_ts)
print(classification_report(y_ts, preds))
```

```
                 precision    recall  f1-score   support

     Exonerated       0.54      0.57      0.55      2457
  Substantiated       0.48      0.33      0.39      2085
Unsubstantiated       0.62      0.70      0.66      3798

       accuracy                           0.57      8340
      macro avg       0.55      0.53      0.53      8340
   weighted avg       0.56      0.57      0.56      8340
```

By using GradientBoostingClassfier model as the prediction model and keep all other thing the same, we can find a huge jump for the accuracy of the prediction. This is the best model we get. And the classification report is similar as RandomForestClassifier model

Try the SVC model

In [32]:
```
from sklearn.svm import SVC
pl3 = Pipeline([('new_CT', CT), ('gbc', SVC())])
pl3.fit(X_tr, y_tr)
```

Out[32]:
```
Pipeline(steps=[('new_CT',
                 ColumnTransformer(transformers=[('category',
                                                  Pipeline(steps=[('i
mpute',
                                                                   Si
mpleImputer(fill_value='Missing',

strategy='constant')),
                                                                  ('o
he',
                                                                   On
eHotEncoder(handle_unknown='ignore'))]),
                                                  ['rank_incident',
                                                   'complainant_ethni
city',
                                                   'complainant_gende
r',
                                                   'fado_type', 'alle
gation']),
                                                 ('numeric', SimpleIm
puter(),
                                                  ['complainant_age_i
ncident',
                                                   'precinct'])])),
                ('gbc', SVC())])
```

Accuracy for SVC model

In [33]:
```
pl3.score(X_ts, y_ts)
```

Out[33]:
```
0.4553956834532374
```

In [34]:
```
preds = pl3.predict(X_ts)
print(classification_report(y_ts, preds))
```

```
                 precision    recall  f1-score   support

     Exonerated       0.00      0.00      0.00      2457
  Substantiated       0.00      0.00      0.00      2085
Unsubstantiated       0.46      1.00      0.63      3798

       accuracy                           0.46      8340
      macro avg       0.15      0.33      0.21      8340
   weighted avg       0.21      0.46      0.28      8340


/opt/anaconda3/lib/python3.8/site-packages/sklearn/metrics/_classific
ation.py:1245: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/anaconda3/lib/python3.8/site-packages/sklearn/metrics/_classific
ation.py:1245: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/anaconda3/lib/python3.8/site-packages/sklearn/metrics/_classific
ation.py:1245: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

From the accuracy and the classification perspective, this model is apparently not propor here to be the prediction model. The precision, recall, f1-score are even 0 for Exonerated and Substantiated.

From all above work. We can find that the improvement we made for features are indeed useful. And this work helped the accuracy increase from 0.51 to 0.55. And we find parameters that best fit the model, but the accuracy is not good. By that, we know the region is (0.465, 0.518). Impresively, the another model we tried here, GradientBoostingClassifier model generated highest accuracy. While this situation changed to the other side, when it comes to SVC model.

Fairness Evaluation

Find the final model, it's time to evaluate the fairness of the model. In this section, we will still use accuracy as the key measurment. And we will focus on the model accuracy for the elder and younger age.

In this section

Null hypothesis:

The accuracies among these two group are same.

Alternative hypothesis:

The accuracies among these two group are not same.

In [35]:
```
from sklearn.preprocessing import binarize
```

In [36]:
```
prediction = pl2.predict(X_ts)
```

In [37]:
```
output = pd.DataFrame()
output['age'] = X_ts['mos_age_incident']
output['prediction'] = prediction
output['true_number'] = y_ts
output.head()
```

Out[37]:

	age	prediction	true_number
27882	26	Unsubstantiated	Substantiated
23550	28	Substantiated	Substantiated
30010	31	Unsubstantiated	Exonerated
31345	47	Substantiated	Unsubstantiated
18128	32	Unsubstantiated	Unsubstantiated

In [38]:
```
output['young_old'] = output['age'] <= 40
output.groupby("young_old").count()
```

Out[38]:

	age	prediction	true_number
young_old			
False	866	866	866
True	7474	7474	7474

In [39]:
```
from sklearn.metrics import accuracy_score
```

In [40]:
```
difference = (output.groupby('young_old').apply(lambda x: accuracy_scc
difference
```

Out[40]:
```
young_old
False    0.584296
True     0.567701
dtype: float64
```

Then we should use a permutation test to check if the difference is by chance or not.

In [41]:
```
stats = abs(difference[0] - difference[1])

lst = []
for _ in range(100):
    outputs = abs(output[['age', 'prediction', 'true_number']].assign(y
        .groupby('young_old').apply(lambda x: accuracy_score(x.true_nur
    lst.append(outputs)

pval = (np.array(lst) <= stats).mean()
pval
```

Out[41]:
```
0.37
```

Here our p-vlue is 0.37 which is much largr than the default 5% significance level. So we fail to reject the null. That means the model has same accuracy to younger group and older group. The we conclude that the final model we made is not biased.

In [ ]:

In [ ]: