NYPD Officer Ethnicity Prediction Task

Kethan Bethamcharla | Varun Dinesh

Summary of Findings

Introduction

Our Prediction Problem that we selected is 'Predicting officer ethnicity'. The type of Prediction Problem we are looking at is a classification problem because officer ethnicity is a categorical distribution. Since we have multiple ethnicities, binary classification can't be used to represent multiple officer ethinicities. It isn't a T/F problem. Therefore we are building a classifer that can result in multiclass classifcation. The response variable is officer ethnicity. We chose this response variable because it relates to our previous analysis in Project 3 of checking to see if board_disposition amongst officer ethincities differed. We thought it would be interesting to create a model that could predict officer ethnicities given case information to see if our predictions would match the results from our hypothesis test in Project 3. To evaluate our model we are using a accuracy over F1-Score because we have a multiclass classifcation here. Both are more apt for binary classification so we decided to go with accuracy as our metric because it is more straightforward in understanding our data overall.

Baseline Model

The two features we have selected for our base model are 'last_name' and 'allegation'. Names in general are related to ethinicity because each ethinicity has a certain group of names they choose from. Sometimes it relates to the ethinicites' culture or beliefs allowing us to make an educated guess on what their ethincity is. Therefore we believed for our base model we can start with 'last_name' as one of our features. Last names in general usually come from the fathers and they are the 'Home Name'. They usually relate to the families culture and religion which the rest of the family uses as their last name so we believed this column would lead us into insight on guessing Officer Ethnicites. Our second feature was 'allegation' because there is a stigma that certain ethinicity of officers dominate a certain allegation type than another. Therefore we believe that officer ethinicity can be decided from allegation type as well.

Both our columns are categorical columns so we one-hot encoded both these columns. We have no quantitative and ordinal columns in our features. Both our columns are nominal columns.

Our baseline model consists of using a pipeline of One-Hot Encoding followed by GridSearchCV(). We tested a small amount of hyperparameters as it was only the basic model to get a feel of how these columns helped with our prediction task. The hyperparameters we modified were max_depth, min_samples_split, and criterion. We used 5-fold cross validation for

our GridSeachCV() hyperparemeter. Along with that we used a train-test split to train and score our model. Our metric is accuracy and our testing accuracy for this model was around 59.1%.

We believe this is a good start to our model as a base implementation but we don't believe it is 'good' because we can definitely improve on our accuracy by adding other features that can potentially lead to insight on officer ethinicty. Therefore this is a good start but we can defenitely improve classification by incorporating other columns in our model. Our model generalizes for unseen data because we use a train-test split to train and score our model. If we completely trained our model on all the data we wouldn't really know how our model would perform. Plus our model could just overfit this sample. Therefore the train-test split accounts for generalizing unseen data.

Final Model

For our Final Model we decided to add the features 'fado_type', 'complainant_ethnicity', and 'mos_age_incident'. We added 'fado_type' because it would serve as an extra classification overlay for 'allegation'. Since 'allegation' stems from the 'fado_type' we wanted to add this feature as well to better organize and complete the data input that covers the complainant category information. We believe that 'complainant_ethnicity' is a valid feature to add because there is a stigma and relation seen between the complainant's ethnicity and the officer's ethnicity. This can be seen through recent news and police wrongdoings across the country. Because of this stigma we believed the complainant's ethnicity could be a valid insight into correctly predicting an officer's ethincity. Coming to 'mos_age_ethnicity' we added this feature to add a quantitative feature that could help us predict officer ethnicity. Younger people tend to make more mistakes then older people as they have less experience. So we believed this information could be valuable when comparing officers of different ethnicities as their age could play a key factor in separating cases of young and older officers.

Our model aka Pipeline we decided to go with has two steps to it. First we have a ColumnTransformer followed by a GridSearchCV that we perform on a DecisionTreeClassifier. We used a ColumnTransformer in our initial step of engineering our features because we had to account for numerical and categorical columns. For our categorical columns we used OneHotEncoder while for our numerical column we used StandardScaler. The method we used to select the model was based on our prediction task. Our prediction task was a multi-class classification task so we decided to go with GridSearchCV and DecisionTreeClassifier to come up with the best classifcation. We also passed in a list of hyperparemeters into GridSearchCV so it could come up with the best hyperparameters. The best hyperparameters that we ended up getting were {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 3}. We also used a train-test split so we can better generalize for unseen data because if we trained our model on the whole data we wouldn't really know how it would perform overall for any other given data.

Fairness Analysis

Null Hypothesis: Our model is fair. Its accuracy for white complainants and non-white complainants are roughly the same, and any differences are due to random chance.

Alternative Hypothesis: Our model is unfair. Its accuracy for white complainants is not the same as its accuracy for non-white complainants.

Test Statistic: Absolute Difference in Accuracy

Significance Value: 0.05

P-val: 0.57

Therefore our conclusion is that we don't reject the null hypothesis. This means that the accuracy for white and non-white complainants are roughly the same, and any differences are due to random chance.

The test statistic we chose was the absolute difference in accuracy because that was our metric and we wanted to see if there was any difference in accuracy at all. We wanted to see if there was a accuracy disparity on both the higher or lower end so we used absolute difference in accuracy. The significance level we chose is the default alpha = 0.05 because we believed we didn't need to change it from the widely-used significance value to compare with our observed values.

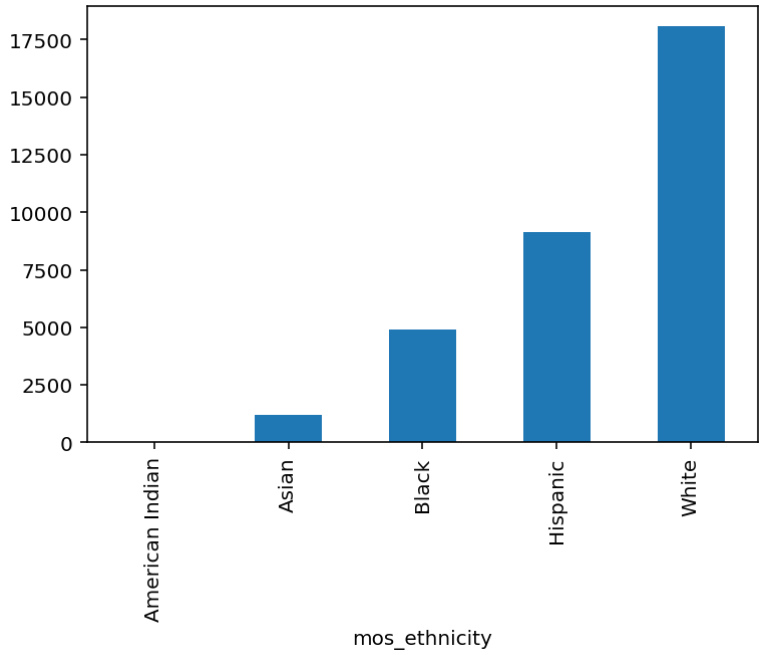
Code

In [1]:
```
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
```

In [2]:
```
filename = 'data/allegations_202007271729.csv'
df = pd.read_csv(filename)
```

In [3]:
```
df.groupby('mos_ethnicity').count()['unique_mos_id'].plot(kind='bar')
```

Out[3]:
```
<AxesSubplot:xlabel='mos_ethnicity'>
```



Baseline Model

In [4]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.tree import plot_tree
from sklearn.model_selection import GridSearchCV
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
```

In [5]:
```
dfBaseModel = df.dropna()[['last_name', 'mos_ethnicity', 'allegation']]

hyperparameters = {
    'max_depth': [3, 6, 15],
    'min_samples_split': [3, 7, 15],
    'criterion': ['gini', 'entropy']
}

plBase = Pipeline([
    ('one-hot', OneHotEncoder(handle_unknown='ignore')),
    ('gs', GridSearchCV(DecisionTreeClassifier(), hyperparameters, cv=5))
])

X_train, X_test, y_train, y_test = train_test_split(dfBaseModel[['last_name', 'a
                                                    dfBaseModel['mos_ethnicity']
                                                    random_state=1)
plBase.fit(X_train, y_train)
plBase.score(X_test, y_test)
```

Out[5]:
```
0.5908896187348279
```

Final Model

In [6]:
```
dfBaseModel = df.dropna()[['last_name', 'mos_ethnicity', 'allegation',
                           'complainant_ethnicity', 'fado_type', 'mos_age_incide

hyperparameters = {
    'max_depth': [2, 3, 4, 5, 7, 10, 13, 15, 18, None],
    'min_samples_split': [2, 3, 5, 7, 10, 15, 20],
    'criterion': ['gini', 'entropy']
}

preproc = ColumnTransformer(
    transformers = [
        ('quant', StandardScaler(), ['mos_age_incident']),
        ('cat', OneHotEncoder(handle_unknown='ignore'), ['last_name', 'allegatio
                                                         'complainant_ethnicity'
    ]
)
plFinal = Pipeline([
    ('preprocess', preproc),
    ('gs', GridSearchCV(DecisionTreeClassifier(), hyperparameters, cv=5))
])

X_train, X_test, y_train, y_test = train_test_split(dfBaseModel[['last_name', 'a
                                                                 'fado_type', 'm
                                                    dfBaseModel['mos_ethnicity']
                                                    random_state=1)
plFinal.fit(X_train, y_train)
plFinal.score(X_test, y_test)
```

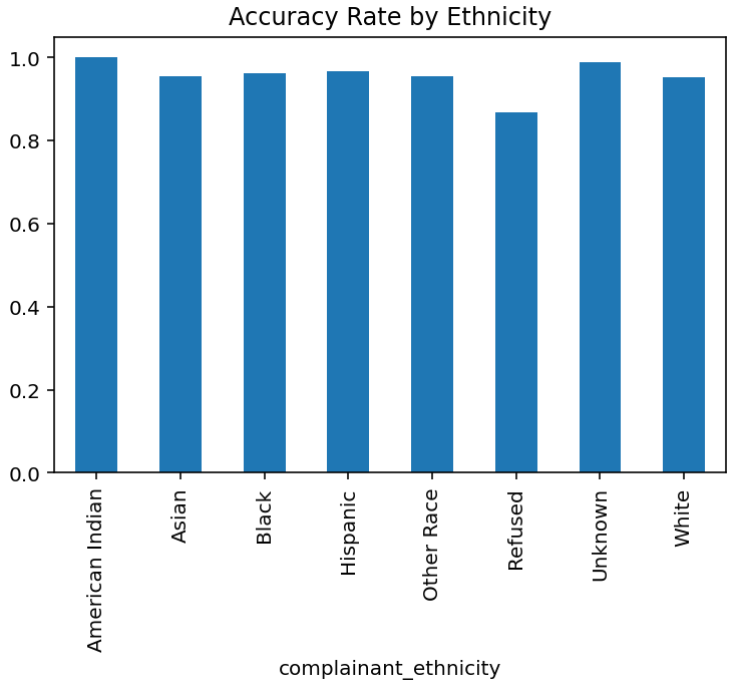
Out[6]:
```
0.9625874625160645
```

In [7]:
```
plFinal['gs'].best_params_
```

Out[7]:
```
{'criterion': 'gini', 'max_depth': None, 'min_samples_split': 3}
```

Fairness Analysis

In [8]:
```
y_pred = plFinal.predict(X_test)
results = X_test
results['complainant_isWhite'] = results['complainant_ethnicity']=='White'
results['prediction'] = y_pred
results['tag'] = y_test
(
    results
    .groupby('complainant_ethnicity')
    .apply(lambda x: metrics.accuracy_score(x['tag'], x['prediction']))
    .plot(kind='bar', title='Accuracy Rate by Ethnicity')
);
```



In [9]:

```
print(
    results
    .groupby('complainant_isWhite')
    .apply(lambda x: metrics.accuracy_score(x['tag'], x['prediction']))
    .rename('accuracy')
    .to_frame()
)
obs = results.groupby('complainant_isWhite').apply(lambda x: metrics.accuracy_sc
obs
```

```
                      accuracy
complainant_isWhite
False                 0.963542
True                  0.953523
```

Out[9]:

```
0.010018428285857084
```

In [10]:

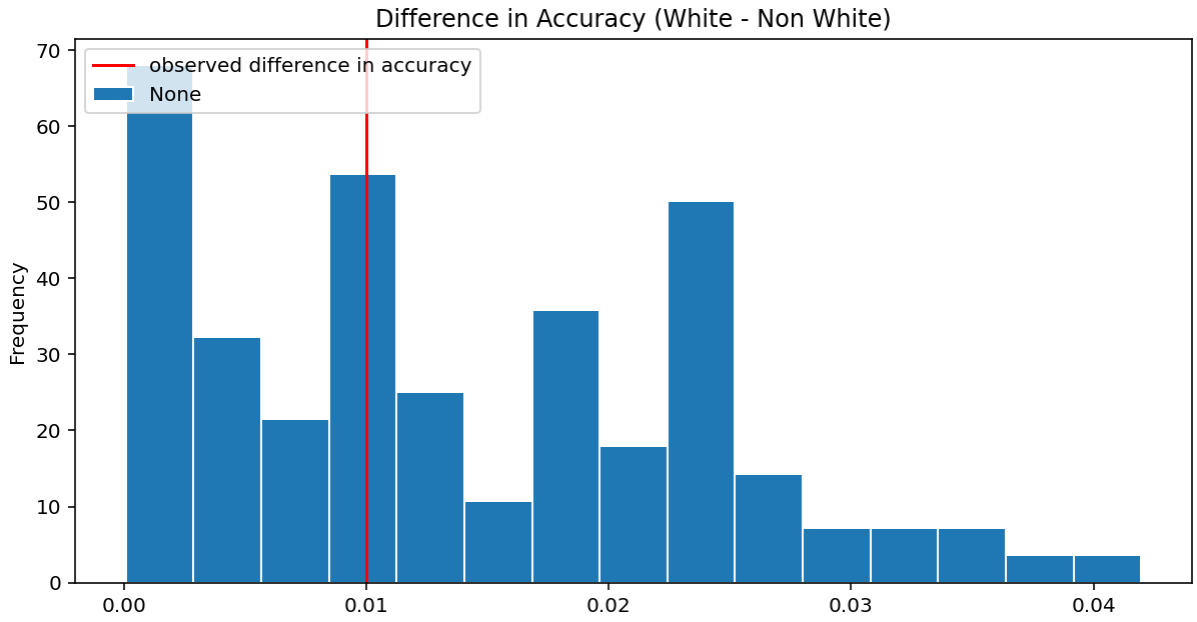
```
diff_in_acc = []
for _ in range(100):
    s = (
        results[['complainant_isWhite', 'prediction', 'tag']]
        .assign(complainant_isWhite=results.complainant_isWhite.sample(frac=1.0,
        .groupby('complainant_isWhite')
        .apply(lambda x: metrics.accuracy_score(x['tag'], x['prediction']))
        .diff()
        .abs()
        .iloc[-1]
    )

    diff_in_acc.append(s)

pval = (obs <= diff_in_acc).mean()
plt.figure(figsize=(10, 5))
pd.Series(diff_in_acc).plot(kind='hist', ec='w', density=True, bins=15,
                            title='Difference in Accuracy (White - Non White)')
plt.axvline(x=obs, color='red', label='observed difference in accuracy')
```

```
plt.legend(loc='upper left');
pval
```

Out[10]: 0.57



In [ ]: