DATA 602

FINAL PROJECT PRESENTATION

MOVING VIOLATIONS ISSUED IN D.C

By
Jaswanth Sai Nathani

Introduction

- Data used in this project pertains to moving citations issued by law enforcement of various DC agencies and federal partners to violators.
- If a vehicle is in motion when the transgression occurs, it is deemed a moving violation. This includes speeding, running a stop sign or red light, reckless driving, drunk driving (DUI/DWI), racing, and eluding an officer.
- This data contains 70,458 records which includes 35 columns.
- This data is available for open usage in open DC data portal and is free to access for everyone.

OBJECTIVE

- Traffic violations are one of the major concerns in any part of the world. Understanding this data will help the government in taking necessary actions to prevent road accidents in the future. My objective is to understand and predict the reasons behind the most repeated violations which are causing road accidents and help reduce them

DATA CLEANING

Finding out the nulls in each column

```
In [10]: df.isnull().sum()
```

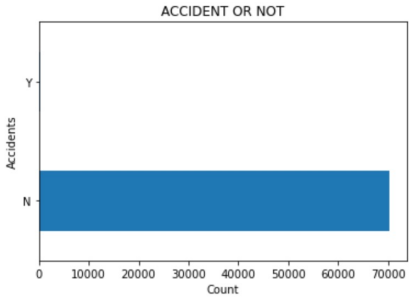
```
Out[10]: OBJECTID                    0
         TICKET_NUMBER               0
         VIOLATION_TYPE_DESC         0
         ISSUE_DATE                  0
         ISSUE_TIME                 70458
         ISSUING_AGENCY_CODE         0
         ISSUING_AGENCY_NAME         0
         ISSUING_AGENCY_SHORT        0
         VIOLATION_CODE              0
         VIOLATION_PROCESS_DESC      0
         LOCATION                    4
         PLATE_STATE                67989
         ACCIDENT_INDICATOR         67949
         DISPOSITION_CODE           70345
         DISPOSITION_TYPE            0
         DISPOSITION_DATE           70345
         FINE_AMOUNT                 0
         TOTAL_PAID                  0
         PENALTY_1                  70458
         PENALTY_2                  70458
         PENALTY_3                  70458
         PENALTY_4                  70458
         PENALTY_5                  70458
         RP_MULT_OWNER_NO           69111
         BODY_STYLE                 70458
         XCOORD                     15746
         YCOORD                     15746
         LATITUDE                   15746
         LONGITUDE                  15746
         MAR_ID                     15746
         GIS_LAST_MOD_DTTM           0
         DRV_LIC_STATE              59175
         DOB_YEAR                   59158
         VEH_YEAR                   31736
         VEH_MAKE                      34
```

DATA VISUALIZATION

In [26]:
```
rate_of_accidents = df['ACCIDENT_INDICATOR'].value_counts()
print(rate_of_accidents)
```

```
N    70202
Y      256
Name: ACCIDENT_INDICATOR, dtype: int64
```

In [27]:
```
rate_of_accidents.plot(kind='barh')
plt.title('ACCIDENT OR NOT')
plt.ylabel('Accidents')
plt.xlabel('Count')
plt.show()
```



In [28]:
```
rate_of_accidents = rate_of_accidents['Y']/rate_of_accidents.sum()
print(f'rate_of_accidents: {rate_of_accidents:.2%}')
```

```
rate_of_accidents: 0.36%
```

Visualization to understand which state registered vehicles committed more number of violations

Visualizing to find out which plate state cars have most number of violations

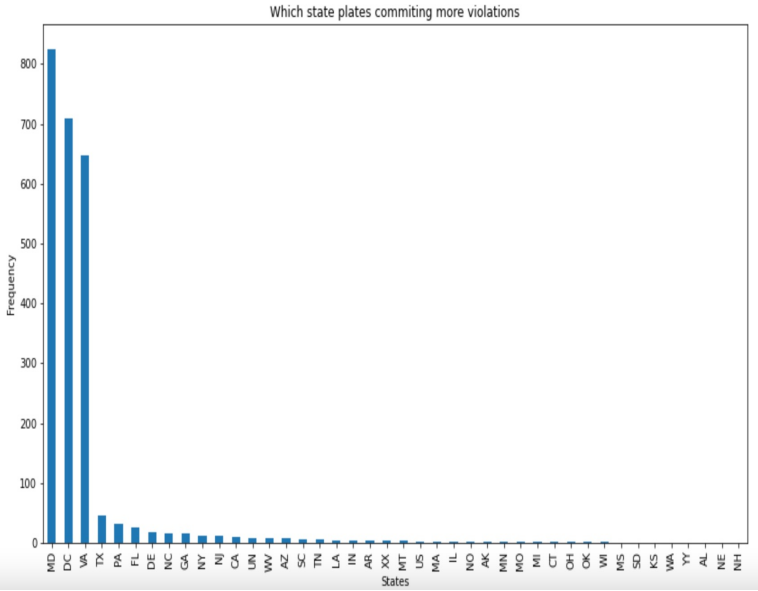
In [30]:
```
count_values_dl = df['PLATE_STATE'].value_counts()
count_values_dl
```

Out[30]:

MD	825
DC	710
VA	648
TX	46
PA	31
FL	26
DE	18
NC	17
GA	16
NY	13
NJ	12
CA	11
UN	9
WV	8
AZ	8
SC	7
TN	6
LA	5
IN	5
AR	4
XX	4
MT	4
US	3
MA	3
IL	3
NO	3
AK	2
MN	2
MO	2
MI	2
CT	2
OH	2
OK	2

In [31]:
```
ax = df['PLATE_STATE'].value_counts().plot(kind='bar',
                                           figsize=(14,8),
                                           title="Which state plates commiting more violations")
ax.set_xlabel("States")
ax.set_ylabel("Frequency")
```

Out[31]: Text(0, 0.5, 'Frequency')



TOTAL TRAFFIC VIOLATION DUES FOR THE MONTH OF SEPTEMBER IN D.C AREA

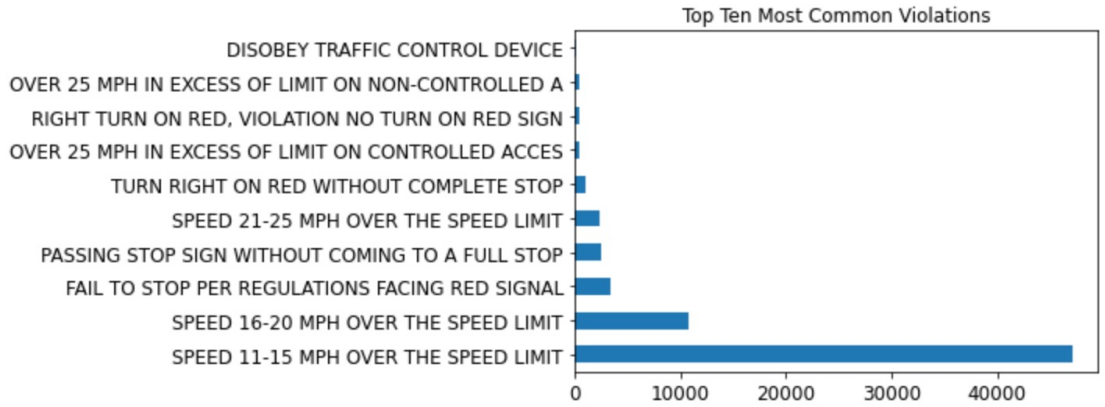
In [33]:
```
outstanding_dues = df['DUES'].sum()
print("The total outstanding dues to the D.C state:{}$".format(outstanding_dues))
```

```
The total outstanding dues to the D.C state:8289886$
```

TOP TEN MOST FREQUENT TYPES OF VIOLATION IN DC AREA

In [34]:
```
df['VIOLATION_PROCESS_DESC'].value_counts()[:10].plot(kind='barh', title='Top Ten Most Common Violations',fontsize=1
```

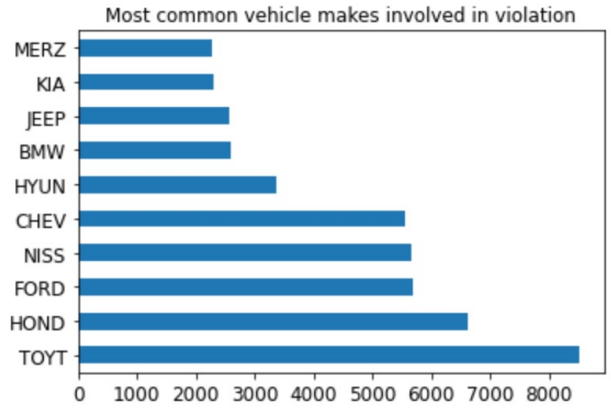
Out[34]:
```
<AxesSubplot:title={'center':'Top Ten Most Common Violations'}>
```



TOP TEN MOST FREQUENT VEHICLES INVOLVING IN VIOLATIONS IN D.C AREA

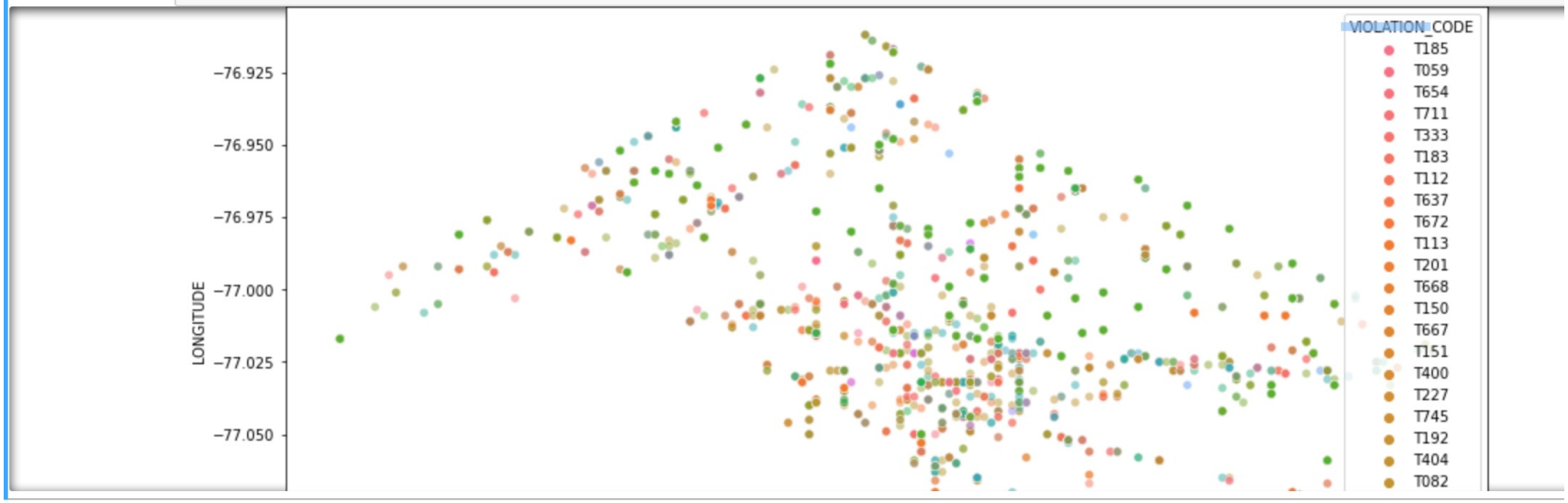
In [35]:
```
df['VEH_MAKE'].value_counts()[:10].plot(kind='barh',title='Most common vehicle makes involved in violation',fontsize
```

Out[35]:
```
<AxesSubplot:title={'center':'Most common vehicle makes involved in violation'}>
```



Latitude and Longitude Map for different type of violations in DC

In [36]:
```
fig = plt.figure(figsize=(15,8))
sns.scatterplot(data=df, x= 'LATITUDE', y = 'LONGITUDE', hue = 'VIOLATION_CODE' , alpha = 0.5)
plt.tight_layout()
```



One Hot Encoding of Columns which we will be using for classification

ONE HOT ENCODING OF CATEGORICAL COLUMNS

In [38]:
```
df.drop(['OBJECTID','TICKET_NUMBER','ISSUING_AGENCY_CODE','ISSUING_AGENCY_SHORT','VIOLATION_PROCESS_DESC','GIS_LAST_
```

In [39]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['ACCIDENT_INDICATOR'] = le.fit_transform(df['ACCIDENT_INDICATOR'])
df['VIOLATION_TYPE_DESC'] = le.fit_transform(df['VIOLATION_TYPE_DESC'])
```

In [40]:
```
for col in df.columns:
    print(col, ': ',len(df[col].unique()),'labels')
```

```
VIOLATION_TYPE_DESC :  2 labels
ISSUING_AGENCY_NAME :  20 labels
VIOLATION_CODE :  136 labels
PLATE_STATE :  43 labels
ACCIDENT_INDICATOR :  2 labels
FINE_AMOUNT :  26 labels
TOTAL_PAID :  16 labels
DRV_LIC_STATE :  47 labels
VEH_MAKE :  235 labels
DUES :  27 labels
```

In [42]:
```
def one_hot_top(df,variable,top_10_labels):
    for label in top_10_labels:
        df[variable+'_'+label]=np.where(df[variable]==label,1,0)

one_hot_top(df,'ISSUING_AGENCY_NAME',top_10_issuing_name)
df.head()
```

	ISSUING_AGENCY_NAME_SPECIAL OPERATION DIV & TRAFFIC DIV	ISSUING_AGENCY_NAME_METROPOLITAN POLICE DPT-DISTRICT 3	ISSUING_AGENCY_NAME_METROPOLITAN POLICE DPT-DISTRICT 2	ISSUING_AGENCY_NAME_METROPOLI POLICE DPT-DISTRIC
)	0	0	0	
)	0	0	0	
)	0	0	0	
)	0	0	0	
)	0	0	0	

Splitting the data into test and train sets

In [50]:
```
from sklearn.model_selection import train_test_split
def generate_splits():
    y = df['ACCIDENT_INDICATOR']
    X = df[[x for x in df.columns if x != 'ACCIDENT_INDICATOR']]

    return train_test_split(X,y,test_size=0.2)

X_train, X_test, y_train, y_test = generate_splits()

print(f'Training examples: {X_train.shape[0]:,}')
print(f'Test examples: {X_test.shape[0]:,}')
```

```
Training examples: 56,366
Test examples: 14,092
```

LOGISTIC REGRESSION MODEL

Logistic Regression

In [52]:
```
param_lr = {'lr_classifier__C':[0.001,0.1, 1, 10]}
gcv_results_lr = GridSearchCV(estimator=num_pipeline_lr, param_grid=param_lr, scoring='accuracy', cv=5)
gcv_results_lr = gcv_results_lr.fit(X_train, y_train)
y_predict=gcv_results_lr.predict(X_test)
print(f'The accuracy when we use logistic regression classifier is {gcv_results_lr.score(X_test,y_test)}')
```

```
The accuracy when we use logistic regression classifier is 0.9970195855804712
```

In [53]:
```
gcv_results_lr.best_params_
```

Out[53]:
```
{'lr_classifier__C': 1}
```

DECISION TREE CLASSIFIER

Decision Tree Classifier

In [56]:
```
param_dt = {'dt_classifier__max_depth':[1,2,3,4]}
gcv_results_dt = GridSearchCV(estimator=num_pipeline_dt, param_grid=param_dt, scoring='accuracy', re
gcv_results_dt = gcv_results_dt.fit(X_train, y_train)
y_predict = gcv_results_dt.predict(X_test)
print(f'The accuracy of decision tree classifier is {gcv_results_dt.score(X_test,y_test)}')
```

```
The accuracy of decision tree classifier is 0.9964518875957991
```

In [57]:
```
gcv_results_dt.best_params_
```

Out[57]:
```
{'dt_classifier__max_depth': 3}
```

KNN CLASSIFIER

KNN Classifier

In [60]:
```
param_knn = {'knn__n_neighbors':[1, 4, 6, 10]}
gcv_results_knn = GridSearchCV(estimator=num_pipeline_knn, param_grid=param_knn, scoring='accuracy',
gcv_results_knn = gcv_results_knn.fit(X_train, y_train)
y_predict = gcv_results_knn.predict(X_test)
gcv_results_knn.score(X_test,y_test)
print(f'The accuracy of KNN classifier is {gcv_results_knn.score(X_test,y_test)}')
```

```
The accuracy of KNN classifier is 0.9965938120919671
```

In [61]:
```
gcv_results_knn.best_params_
```

Out[61]:
```
{'knn__n_neighbors': 10}
```

CONCLUSION

This dataset we have choosen have lot of data that is nulls which impacted our predictions

Also, the uneven distribution of target variable did not help in our cause.

Better data would have helped us in creating more estimations like age group of people who are most frequently committing violations.

It would also help us in identifying repeated violators and would help control them.

Thank You!