Project

On

Crime In Chicago

By

Birwa Galia

Milony Mehta

Shantanu Deosthale

<u>INFO7390 Advances in Data Science and Architecture</u>

Spring 2018

Instructor: Prof. Sreekanth Krishnamurthy

Table of Contents

1. Abstract

Problem:

- Chicago is ranked one in homicide rate as compared to other metropolitan cities such as Los Angeles and New York. Homicide remains more than 50% of the crime in Chicago.

Dataset:

- The Dataset showcases reported incidents of crime that have been occurred in the city of Chicago from 2001 to 2016.
- For this project, we are using data provided by the open source Chicago Police Department's system.
- The entire dataset has 6.49M rows. Due to the size of this dataset, we decided to focus on data from 2012 to 2017 for prediction purposes and we used the data from 2008-2017 for analysis purposes.

Result:

- With the help of historical data, patterns, and fluctuations, a clear picture can be framed, about the reasons leading to increasing crime in Chicago.
- During the Presidential campaign, Chicago crime number and its analysis are widely used.

2. Introduction

2.1 Objective

- The inspiration behind this project is to help Chicago Police Department to improvise and derive suitable measures to reduce the crime.
- The following measures are projected:
  1. Is arrest rate equivalent to the crime rate?
  2. Analyze Number of Crime by Year, Month, Day and Week_days
  3. Deriving the most common locations of the crime will be beneficial to the city.
  4. Analyzing different types of crimes in Chicago
  5. Predicting the outcome if an arrest would occur or not.
- This dataset is used to correlate the types of the crimes occurred and number of criminals arrested

2.2 Background

- Chicago saw a major rise in crime in the late 1990's.
- From then, Chicago Crime Numbers have been in the media for all the wrong reasons.

Dataset is extracted from the Chicago Police Department's CLEAR (Citizen Law Enforcement Analysis and Reporting) system. To protect the privacy of crime victims, addresses are shown at the block level only and specific locations are not identified.

3. Methodology

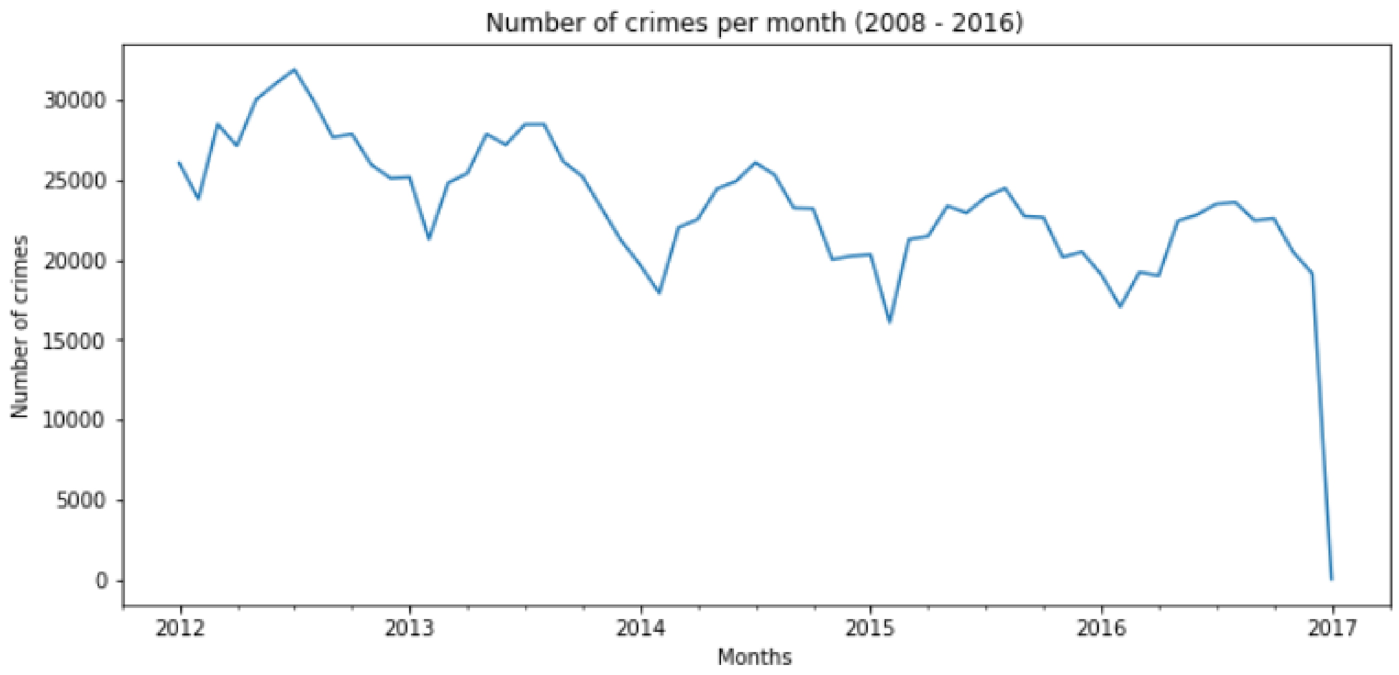
Performed analysis using matplotlib to determine:

1. Whether the city is safe or not?
2. Number of Crime by Year, Month, Day and Weekday
3. The most common locations of the crime

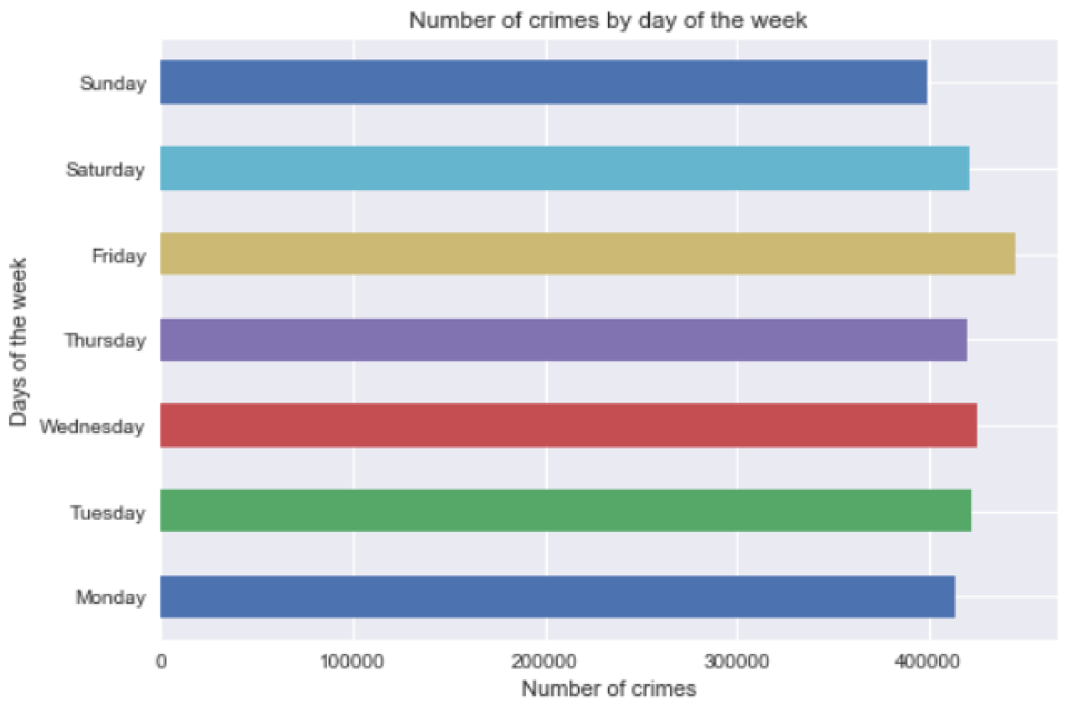
The following Classification algorithms were used to predict arrest:

1. Extra Tree Classification
2. Naïve Bayes Classification
3. Logistic Regression
4. Random Forest Classification
5. K-Nearest-Neighbours Classification

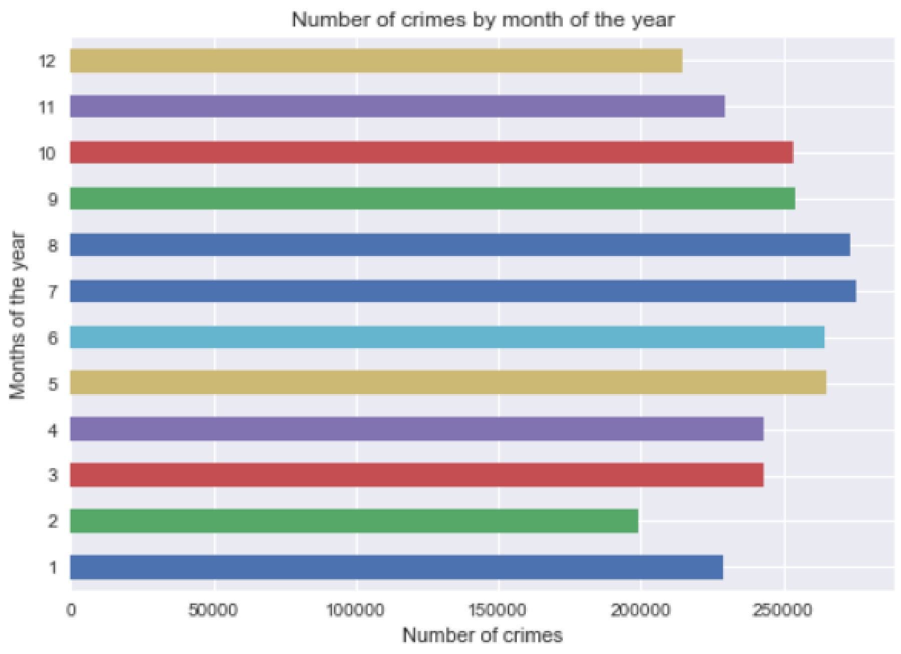
4. Analysis



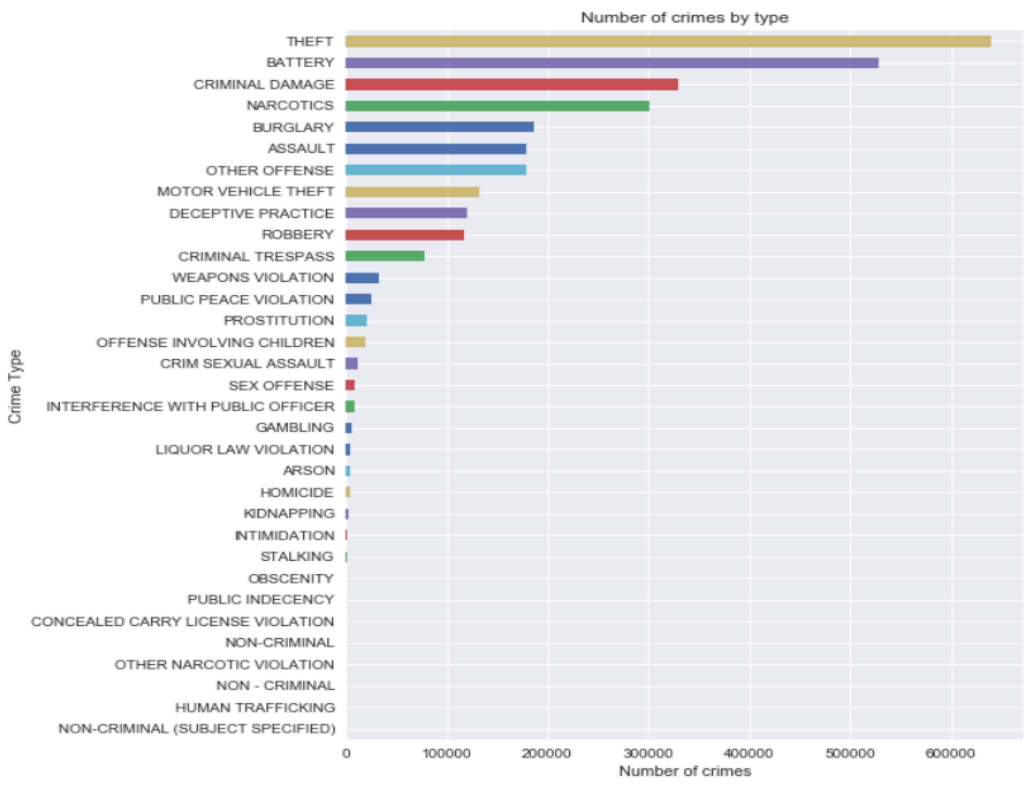
The above plot shows the total number of crimes committed per month for the years 2012-2017. We can see that the number of crimes committed has been decreasing over the years in turn making Chicago a safer city to live in.



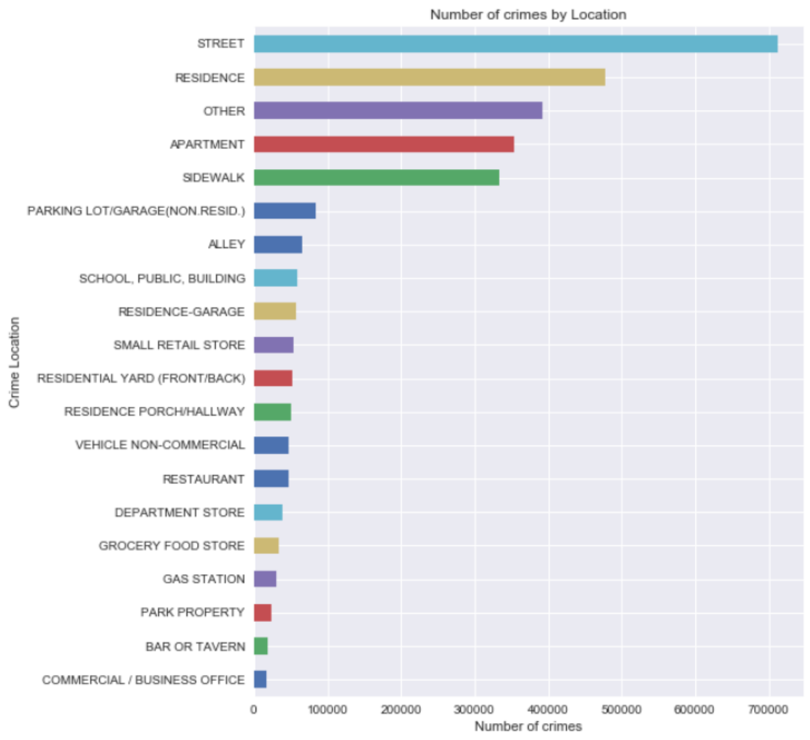
The above graph shows the total number of crimes for each day and it can be concluded that maximum amount of crimes are committed during the weekend



The above graph shows the total number of crimes for each month and it can be concluded that maximum amount of crimes are committed during the months July and August.

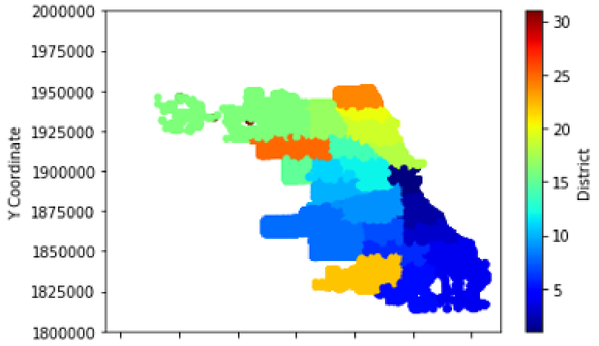


The above graph shows the count of all the crime types and it can be seen that the most frequently committed crime is Theft

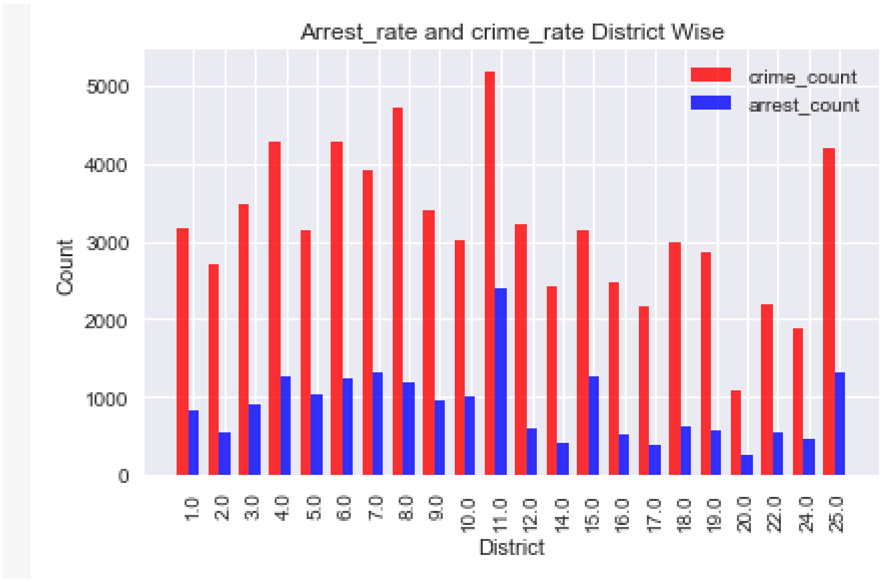


The above graph shows the count of crime committed in a location and it can be seen that the majority crimes are committed on the Streets or Residence

```
In [29]: data.plot(kind='scatter',x='X Coordinate', y='Y Coordinate', c='District', cmap=plt.get_cmap('jet'))
         plt.xlim(1075000,1210000)
         plt.ylim(1800000, 2000000)
         plt.figure(figsize=(30,30))

Out[29]: <matplotlib.figure.Figure at 0x269925d7748>
```



The above graph plots the Districts of Chicago



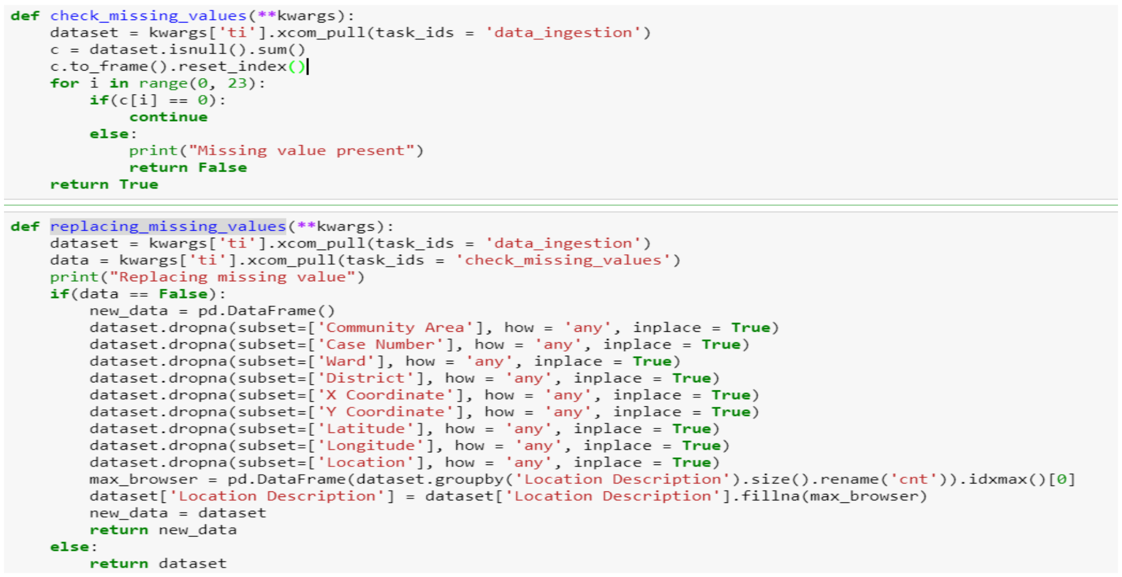
The above graph shows the count of the crimes and the count of the arrests that were actually made. As we can see that the ratio between the crimes committed and the arrests made is not desirable

5. Code

a. The data was ingested from S3 bucket using the library urllib

```
url = "https://s3.amazonaws.com/adsfinalproject/Chicago_Crimes_2012_to_2017.csv"
AWS_ACCESS_KEY_ID = str(sys.argv[1])
AWS_SECRET_ACCESS_KEY = str(sys.argv[2])
def data_ingestion(url):
    urllib.request.urlretrieve(url,'project.csv')
    data = pd.read_csv('project.csv')
    return data
```

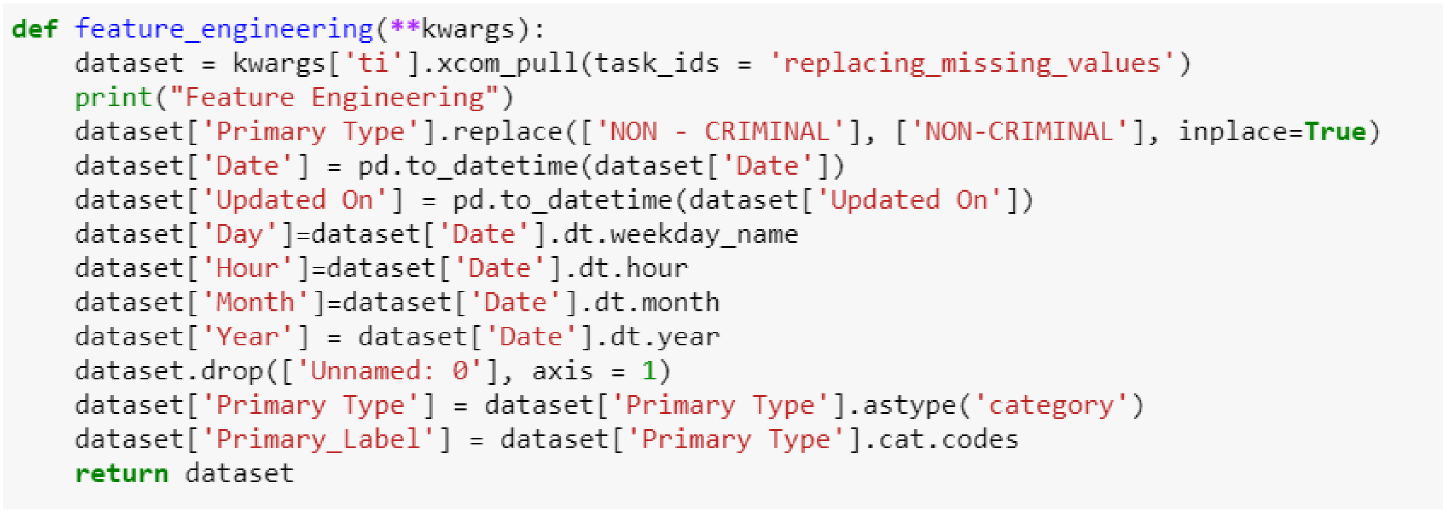
b. The code snippet below shows the functions that checks and replaces the missing values



```python
def check_missing_values(**kwargs):
    dataset = kwargs['ti'].xcom_pull(task_ids = 'data_ingestion')
    c = dataset.isnull().sum()
    c.to_frame().reset_index()
    for i in range(0, 23):
        if(c[i] == 0):
            continue
        else:
            print("Missing value present")
            return False
    return True
```

```python
def replacing_missing_values(**kwargs):
    dataset = kwargs['ti'].xcom_pull(task_ids = 'data_ingestion')
    data = kwargs['ti'].xcom_pull(task_ids = 'check_missing_values')
    print("Replacing missing value")
    if(data == False):
        new_data = pd.DataFrame()
        dataset.dropna(subset=['Community Area'], how = 'any', inplace = True)
        dataset.dropna(subset=['Case Number'], how = 'any', inplace = True)
        dataset.dropna(subset=['Ward'], how = 'any', inplace = True)
        dataset.dropna(subset=['District'], how = 'any', inplace = True)
        dataset.dropna(subset=['X Coordinate'], how = 'any', inplace = True)
        dataset.dropna(subset=['Y Coordinate'], how = 'any', inplace = True)
        dataset.dropna(subset=['Latitude'], how = 'any', inplace = True)
        dataset.dropna(subset=['Longitude'], how = 'any', inplace = True)
        dataset.dropna(subset=['Location'], how = 'any', inplace = True)
        max_browser = pd.DataFrame(dataset.groupby('Location Description').size().rename('cnt')).idxmax()[0]
        dataset['Location Description'] = dataset['Location Description'].fillna(max_browser)
        new_data = dataset
        return new_data
    else:
        return dataset
```

c. Next a function for feature engineering was created



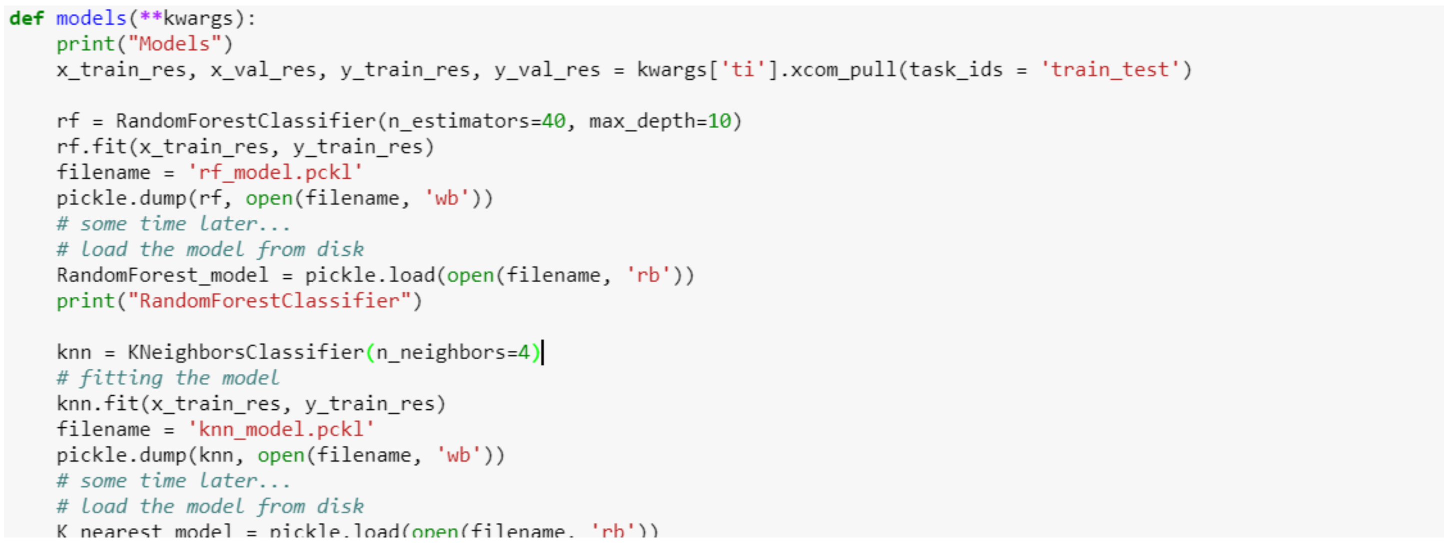
```python
def feature_engineering(**kwargs):
    dataset = kwargs['ti'].xcom_pull(task_ids = 'replacing_missing_values')
    print("Feature Engineering")
    dataset['Primary Type'].replace(['NON - CRIMINAL'], ['NON-CRIMINAL'], inplace=True)
    dataset['Date'] = pd.to_datetime(dataset['Date'])
    dataset['Updated On'] = pd.to_datetime(dataset['Updated On'])
    dataset['Day']=dataset['Date'].dt.weekday_name
    dataset['Hour']=dataset['Date'].dt.hour
    dataset['Month']=dataset['Date'].dt.month
    dataset['Year'] = dataset['Date'].dt.year
    dataset.drop(['Unnamed: 0'], axis = 1)
    dataset['Primary Type'] = dataset['Primary Type'].astype('category')
    dataset['Primary_Label'] = dataset['Primary Type'].cat.codes
    return dataset
```

- The datatype of 'Date' and 'Updated On' was changed from object to datetime
- New columns of Year, Month, Day and Hour were created using the column Date
- As the datatype for the column Primary Type was categorical Label Encoding was performed to convert it into an integer for prediction purposes

d. As the data was biased Under Sampling was performed

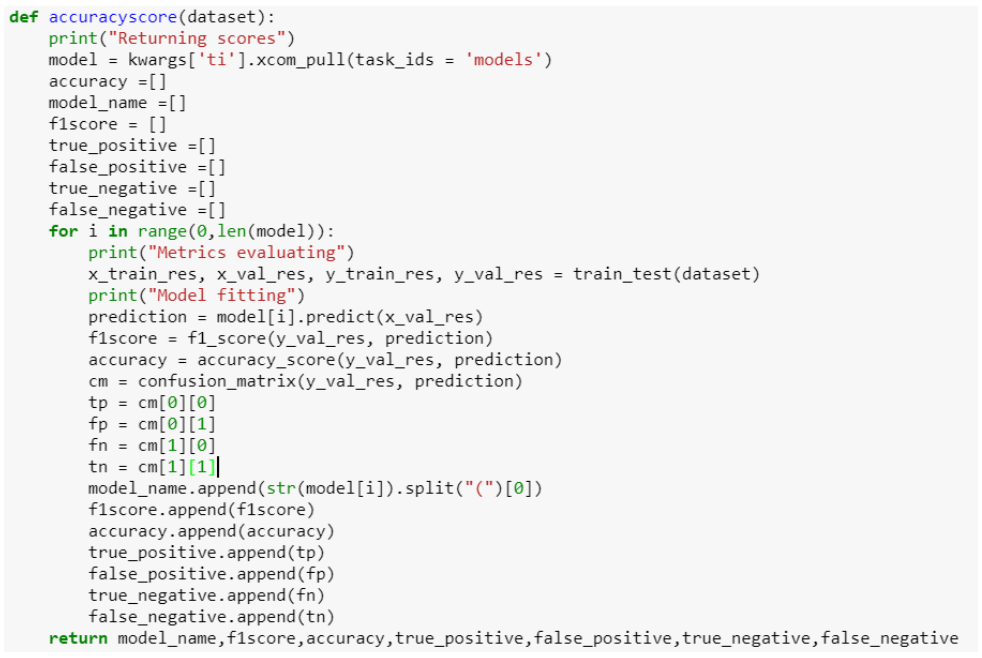
e. The data was split into train-test with 75% of the data being the training set and the remaining data being the test set

f. Next all the Classification model were pickled to persist the model for future use without having to retrain them



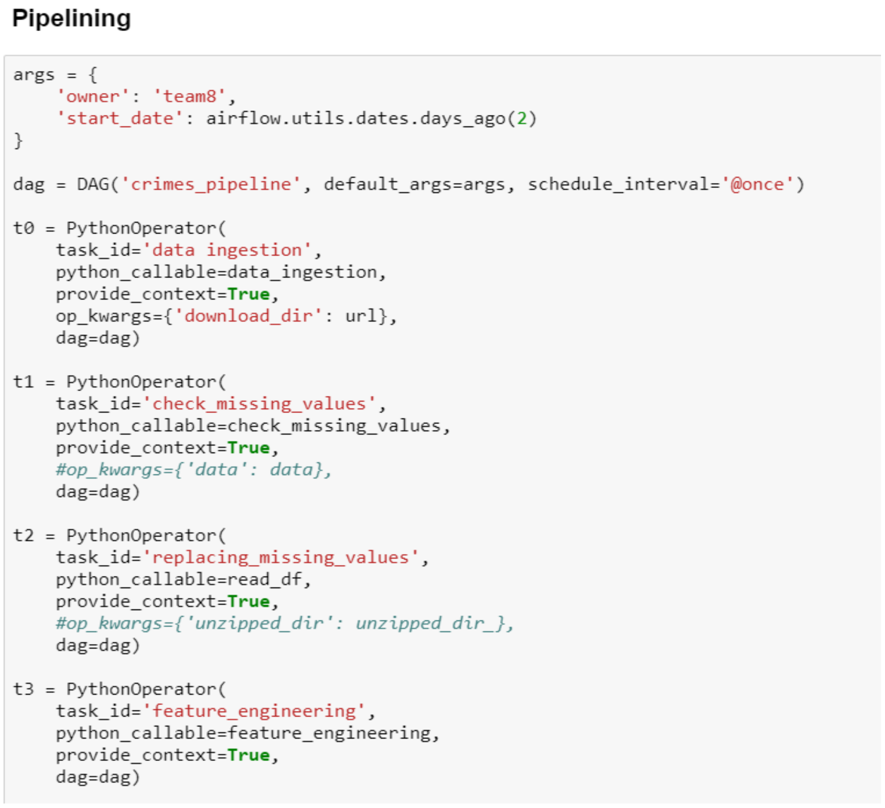
```
def models(**kwargs):
    print("Models")
    x_train_res, x_val_res, y_train_res, y_val_res = kwargs['ti'].xcom_pull(task_ids = 'train_test')

    rf = RandomForestClassifier(n_estimators=40, max_depth=10)
    rf.fit(x_train_res, y_train_res)
    filename = 'rf_model.pckl'
    pickle.dump(rf, open(filename, 'wb'))
    # some time later...
    # load the model from disk
    RandomForest_model = pickle.load(open(filename, 'rb'))
    print("RandomForestClassifier")

    knn = KNeighborsClassifier(n_neighbors=4)
    # fitting the model
    knn.fit(x_train_res, y_train_res)
    filename = 'knn_model.pckl'
    pickle.dump(knn, open(filename, 'wb'))
    # some time later...
    # load the model from disk
    K_nearest_model = pickle.load(open(filename, 'rb'))
```

g. The F1 score and Accuracy for each model was calculated in the following function. It also calculates the confusion matrix for each model



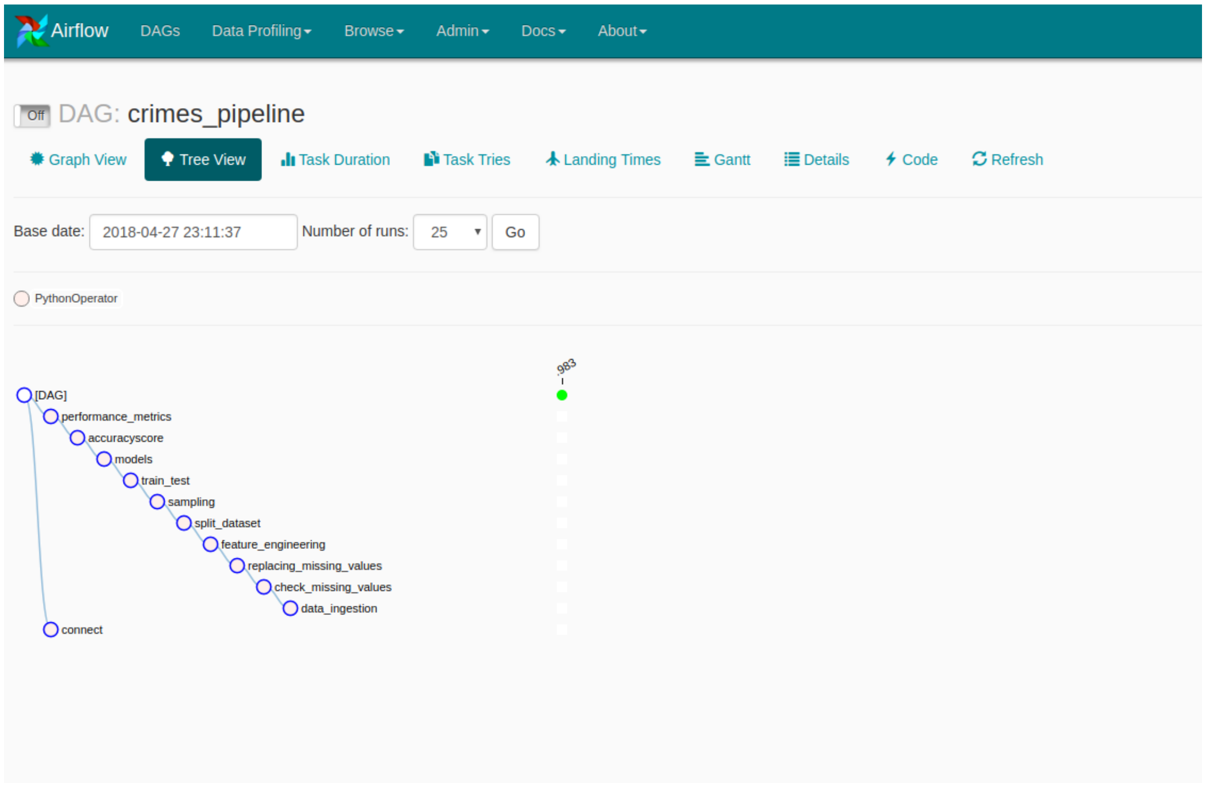
```
def accuracyscore(dataset):
    print("Returning scores")
    model = kwargs['ti'].xcom_pull(task_ids = 'models')
    accuracy =[]
    model_name =[]
    f1score = []
    true_positive =[]
    false_positive =[]
    true_negative =[]
    false_negative =[]
    for i in range(0,len(model)):
        print("Metrics evaluating")
        x_train_res, x_val_res, y_train_res, y_val_res = train_test(dataset)
        print("Model fitting")
        prediction = model[i].predict(x_val_res)
        f1score = f1_score(y_val_res, prediction)
        accuracy = accuracy_score(y_val_res, prediction)
        cm = confusion_matrix(y_val_res, prediction)
        tp = cm[0][0]
        fp = cm[0][1]
        fn = cm[1][0]
        tn = cm[1][1]
        model_name.append(str(model[i]).split("(")[0])
        f1score.append(f1score)
        accuracy.append(accuracy)
        true_positive.append(tp)
        false_positive.append(fp)
        true_negative.append(fn)
        false_negative.append(tn)
    return model_name,f1score,accuracy,true_positive,false_positive,true_negative,false_negative
```

h. Finally, the entire project was pipelined using AIRFLOW where individual tasks were created for each function



Pipelining

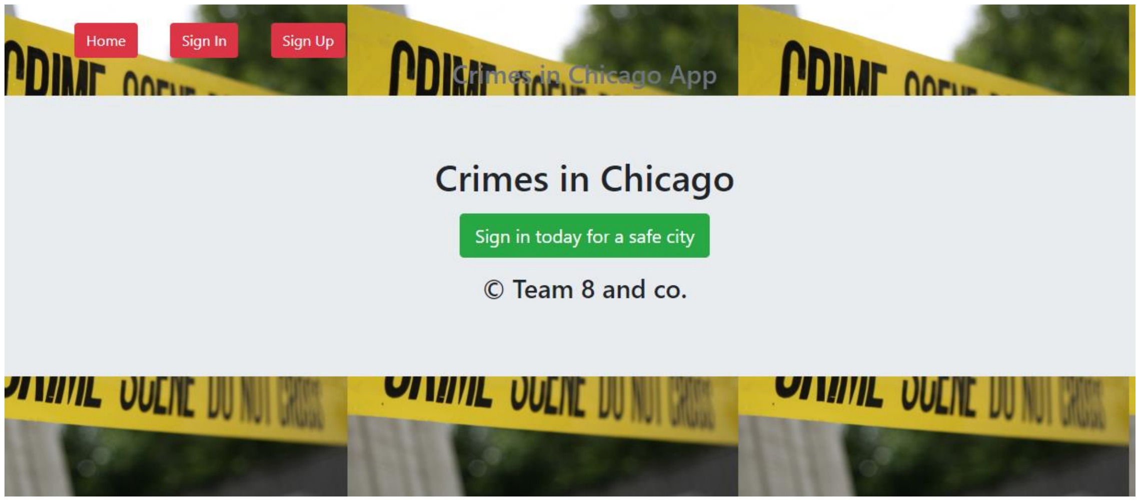
```
args = {
    'owner': 'team8',
    'start_date': airflow.utils.dates.days_ago(2)
}

dag = DAG('crimes_pipeline', default_args=args, schedule_interval='@once')

t0 = PythonOperator(
    task_id='data ingestion',
    python_callable=data_ingestion,
    provide_context=True,
    op_kwargs={'download_dir': url},
    dag=dag)

t1 = PythonOperator(
    task_id='check_missing_values',
    python_callable=check_missing_values,
    provide_context=True,
    #op_kwargs={'data': data},
    dag=dag)

t2 = PythonOperator(
    task_id='replacing_missing_values',
    python_callable=read_df,
    provide_context=True,
    #op_kwargs={'unzipped_dir': unzipped_dir_},
    dag=dag)

t3 = PythonOperator(
    task_id='feature_engineering',
    python_callable=feature_engineering,
    provide_context=True,
    dag=dag)
```



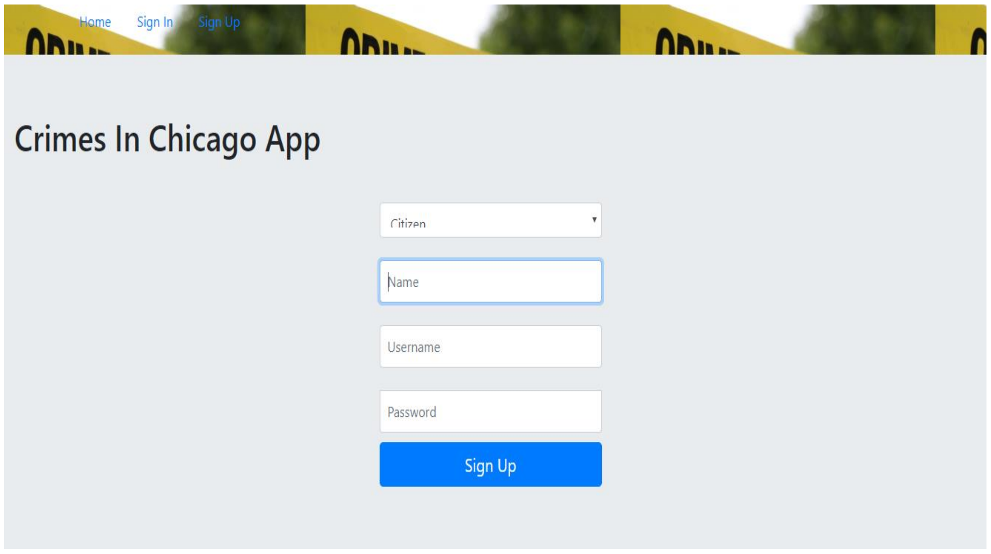
DAG: crimes_pipeline

Base date: 2018-04-27 23:11:37 Number of runs: 25

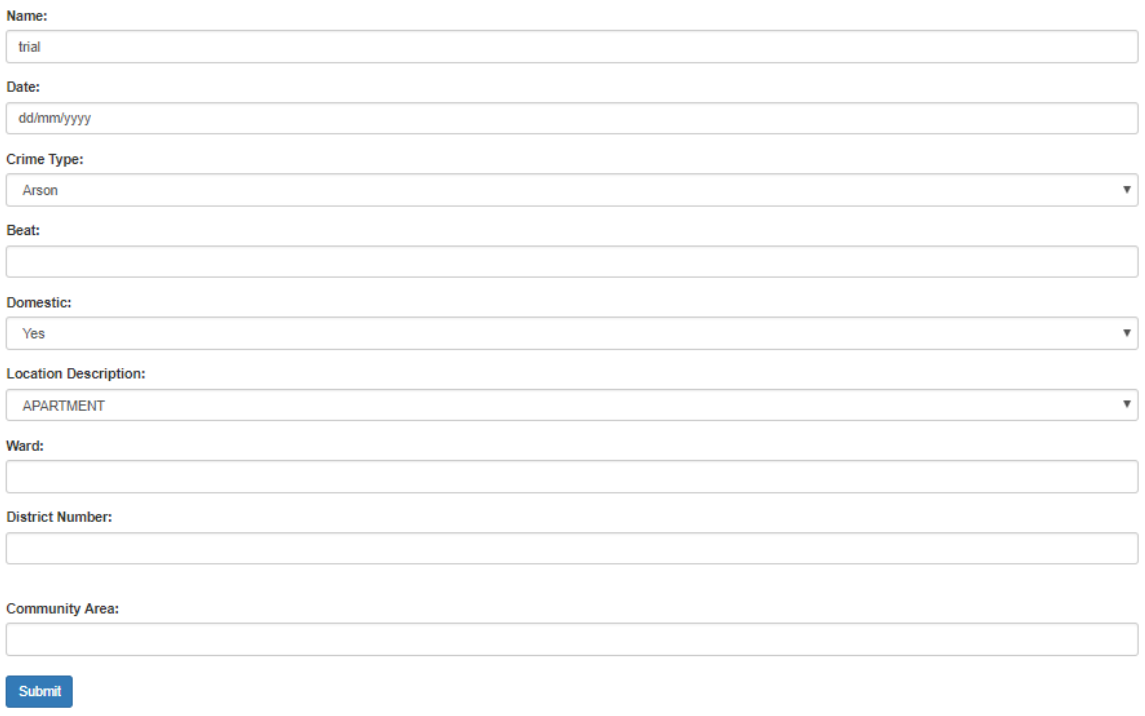
6. User Interface



This is the Home Page.



This page registers a new user. The user has two roles Police and Citizen.


Name: trial

Date: dd/mm/yyyy

Crime Type: Arson

Beat:

Domestic: Yes

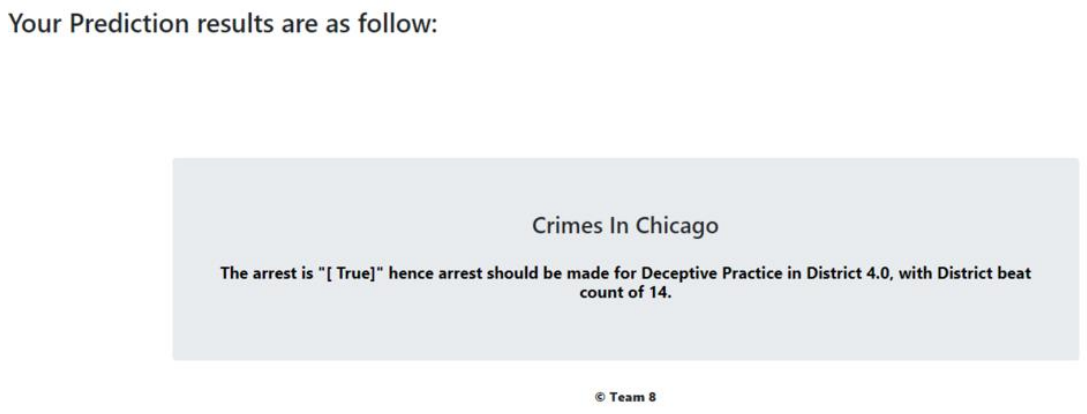
Location Description: APARTMENT

Ward:

District Number:

Community Area:

This takes the user input in order to predict whether the arrest will be made or not.

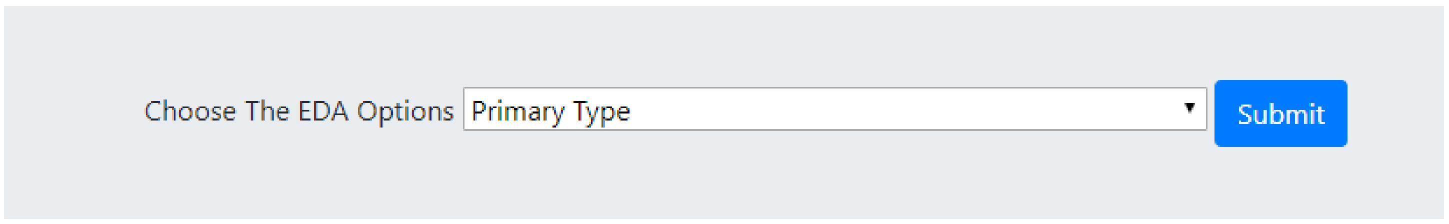

Your Prediction results are as follow:

Crimes In Chicago

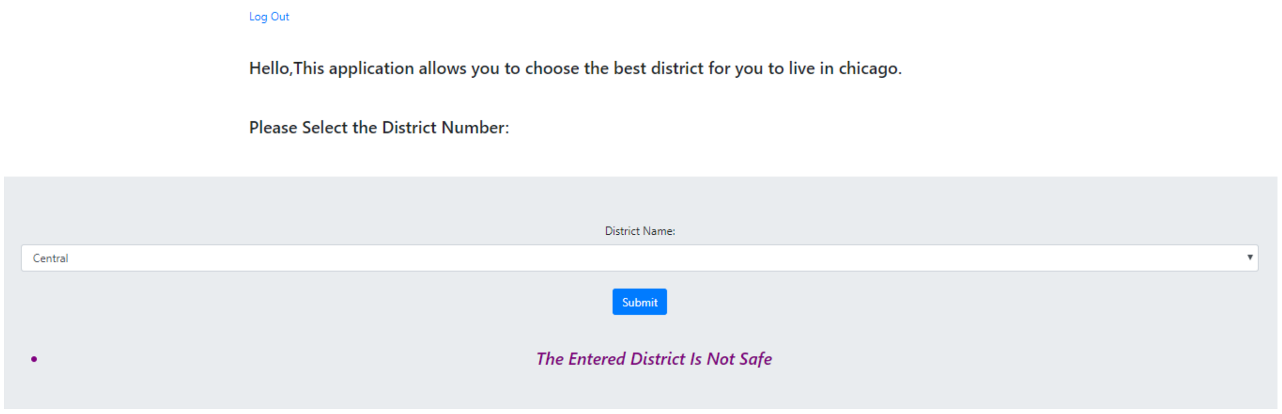
The arrest is "[True]" hence arrest should be made for Deceptive Practice in District 4.0, with District beat count of 14.

© Team 8

The above result was predicted with values that were given by the user. The prediction was performed using Extra Tree Classifier as it gave us the best accuracy


Choose The EDA Options: Primary Type

This page lets the user see the analysis made on the dataset that were mentioned above.


Log Out

Hello,This application allows you to choose the best district for you to live in chicago.

Please Select the District Number:

District Name:

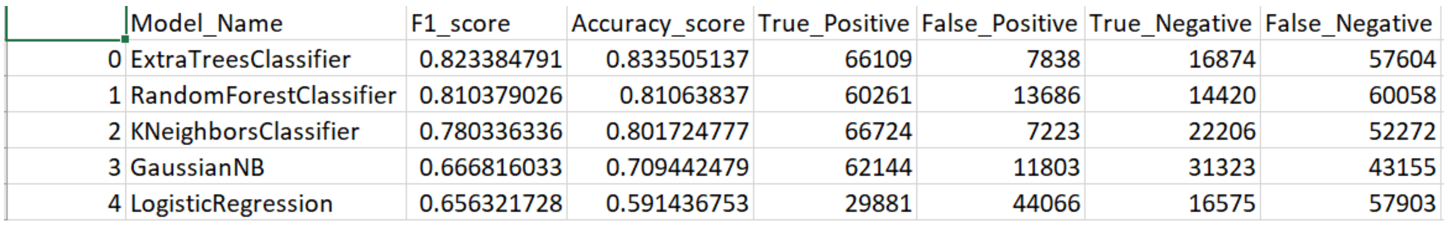
Central

Submit

- *The Entered District Is Not Safe*

This page takes district as an input and helps the user know if the District is safe or not.

7. Results



	Model_Name	F1_score	Accuracy_score	True_Positive	False_Positive	True_Negative	False_Negative
0	ExtraTreesClassifier	0.823384791	0.833505137	66109	7838	16874	57604
1	RandomForestClassifier	0.810379026	0.81063837	60261	13686	14420	60058
2	KNeighborsClassifier	0.780336336	0.801724777	66724	7223	22206	52272
3	GaussianNB	0.666816033	0.709442479	62144	11803	31323	43155
4	LogisticRegression	0.656321728	0.591436753	29881	44066	16575	57903

The above error_metrics was generated after training all the models. Extra Tree Classifier gave us the best accuracy hence we have used it to build our flask application.