Crime Predictor Tool

Problem Statement:-

The main problem is that day to day the population is going to be increasing and by that the crimes are also going to increase in different areas. The crime rate can‘t be accurately predicted by the officials. The officials as they focus on many issues may not predict the crimes to happen in the future. There has been countless work done related to crimes. Large datasets have been reviewed, and information such as location and the type of crimes have been extracted to help people follow law enforcements. Existing methods have used these databases to identify crime hotspots based on locations. There are several map applications that show the exact crime location along with the crime type for any given city. Even though crime locations have been identified, there is no information available that includes the crime occurrence date and time along with techniques that can accurately predict what crimes will occur in the future.

Objectives

The objective of our work is to:

· Predicting crime before it takes place.

· Understanding crime pattern.

· Classify crime based on location.

Implementation:-

The implementation of the project is done with the help of python language. To be particular, for the purpose of machine learning Anaconda is being used. The entries was done just to make the machine learn what all it has to do with the data and what actually the output is being demanded. As soon as the machine learnt the algorithms and the process, accuracy of different algorithms were measured & the algorithm with the most

accuracy is used for the prediction kernel i.e. Random forest. For the purpose of proper implementation and functioning several Algorithms and techniques were used. Following are the algorithms used:

- **KNN (K-Nearest neighbors)**

  A powerful classification algorithm used in pattern recognition K nearest neighbors stores all available cases and classifies new cases based on a similarity measure (e.g. distance function).One of the top data mining algorithms used today. A non-parametric lazy learning algorithm (An Instance based Learning method).

- **Decision Tree**

  As the name says all about it, it is a tree which helps us by assisting us in decision-making. Used for both classification and regression, it is a very basic and important predictive learning algorithm. It is different from others because it works intuitively i.e., taking decisions one-by-one. · Non-parametric: Fast and efficient.

- **Random forest**

  Random Forests is a very popular ensemble learning method which builds a number of classifiers on the training data and combines all their outputs to make the best predictions on the test data. Thus, the Random Forests algorithm is a variance minimizing algorithm that uses randomness when making split decision to help avoid overfitting on the training data.

Code:

Data Analysis

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
dataset=pd.read_csv('data.csv')
```

```
data=pd.read_csv('data.csv')

dataset.head()
for col in data:
   print (type(data[col][1]))
data['timestamp'] = pd.to_datetime(data['timestamp'], coerce=True)

data['timestamp'] = pd.to_datetime(data['timestamp'], format = '%d/%m/%Y %H:%M:%S')

data['timestamp']
# DATE TIME STAMP FUNCTION
column_1 = data.ix[:,0]

db=pd.DataFrame({"year": column_1.dt.year,
            "month": column_1.dt.month,
            "day": column_1.dt.day,
            "hour": column_1.dt.hour,
            "dayofyear": column_1.dt.dayofyear,
            "week": column_1.dt.week,
            "weekofyear": column_1.dt.weekofyear,
            "dayofweek": column_1.dt.dayofweek,
            "weekday": column_1.dt.weekday,
            "quarter": column_1.dt.quarter,
           })
dataset1=dataset.drop('timestamp',axis=1)
data1=pd.concat([db,dataset1],axis=1)
data1.info()
data1.dropna(inplace=True)
data1.head()
```

Data Visualization & Analysis

```
sns.pairplot(data1,hue='act363')
sns.boxplot(x='act379' ,y='hour' ,data=data1, palette='winter_r')
sns.boxplot(x='act13' ,y='hour' ,data=data1 , palette='winter_r')
sns.boxplot(x='act323' ,y='hour' ,data=data1, palette='winter_r')
sns.boxplot(x='act363' ,y='hour' ,data=data1, palette='winter_r')
df = pd.DataFrame(data=data1, columns=['act13', 'hour', 'day'])
df.plot.hexbin(x='act13',y='hour',gridsize=25)
df.plot(legend=False)
```

```
df1 = pd.DataFrame(data=data1, columns=['act13', 'act323', 'act379'])
df1.plot.kde()
```

X & Y array

```
X=data1.iloc[:,[1,2,3,4,6,16,17]].values
y=data1.iloc[:,[10,11,12,13,14,15]].values
y
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=50)
```

Creating & Training KNN Model

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(X_train,y_train)
knn.score(X_test,y_test)
knn.score(X_train,y_train)
```

Elbow Method For optimum value of K

```
error_rate = []
for i in range(1,140):

    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
plt.figure(figsize=(10,6))
plt.plot(range(1,140),error_rate,color='blue', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=5)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(max_depth=500, random_state=300)
dtree.fit(X_train,y_train)
y_pred=dtree.predict(X_test)
dtree.score(X_test,y_test)
dtree.score(X_train,y_train)
y_pred
treefeatures=dtree.feature_importances_
indices = np.argsort(treefeatures)
```

```
treefeatures
features = data1.iloc[:,[1,2,3,4,6,16,17]]
plt.figure(1)
plt.title('Feature Importances')
plt.barh(range(len(indices)), treefeatures[indices], color='b', align='center')
plt.yticks(range(len(indices)), features[indices])
plt.xlabel('Relative Importance')
```

Tree Visualization

```
feature_names=[ 'dayofweek', 'dayofyear', 'hour', 'month', 'week','latitude', 'longitude']
from IPython.display import Image
from sklearn.externals.six import StringIO
from sklearn.tree import export_graphviz
import pydot
import os
os.environ["PATH"] += os.pathsep + 'C:/Program Files (x86)/Graphviz2.38/bin/'
dot_data = StringIO()
export_graphviz(dtree,out_file=dot_data,feature_names=feature_names,filled=True)

graph = pydot.graph_from_dot_data(dot_data.getvalue())
Image(graph[0].create_png()
```

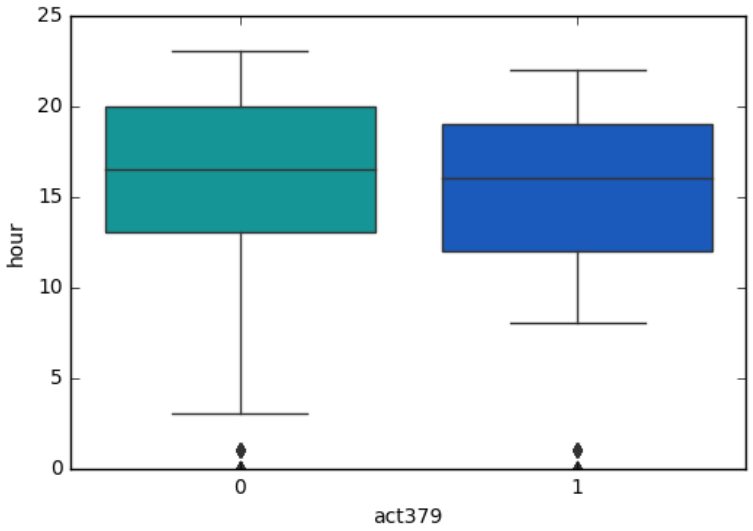
Creating & Training Random Tree Model

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=100)
rfc.fit(X_train, y_train)
y_pred=rfc.predict(X_test)
rfc.score(X_test,y_test)
rfc.score(X_train,y_train)
om=rfc.feature_importances_
indices = np.argsort(om)

om
features = data1.columns
plt.figure(1)
plt.title('Feature Importances')
plt.barh(range(len(indices)), om[indices], color='b', align='center')
plt.yticks(range(len(indices)), features[indices])
plt.xlabel('Relative Importance')
```

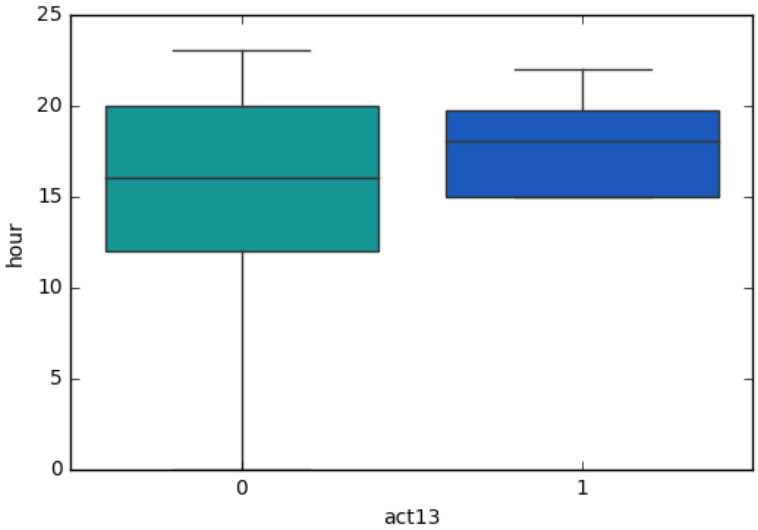
Visualization:-

The following graph represents the prediction of Robbery with respect to time.



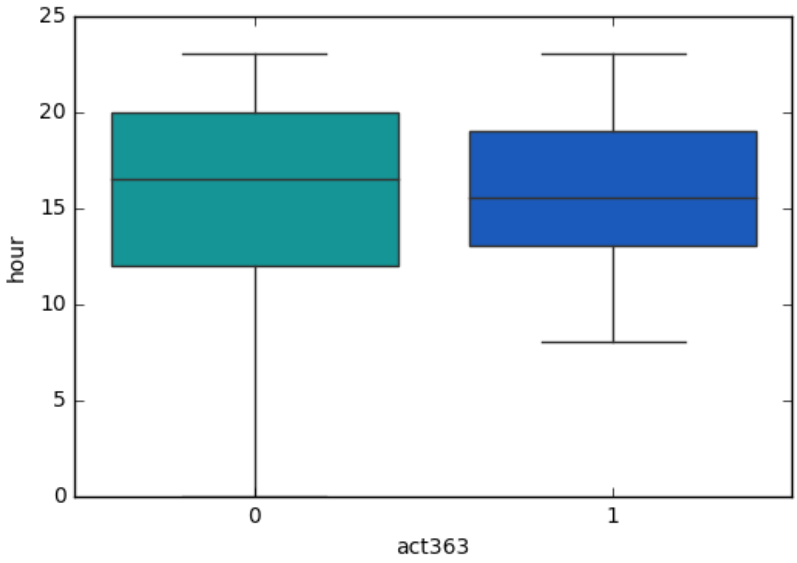
Act379(Robbery vs Hour)

The following graph represents the prediction of Gambling with respect to time.



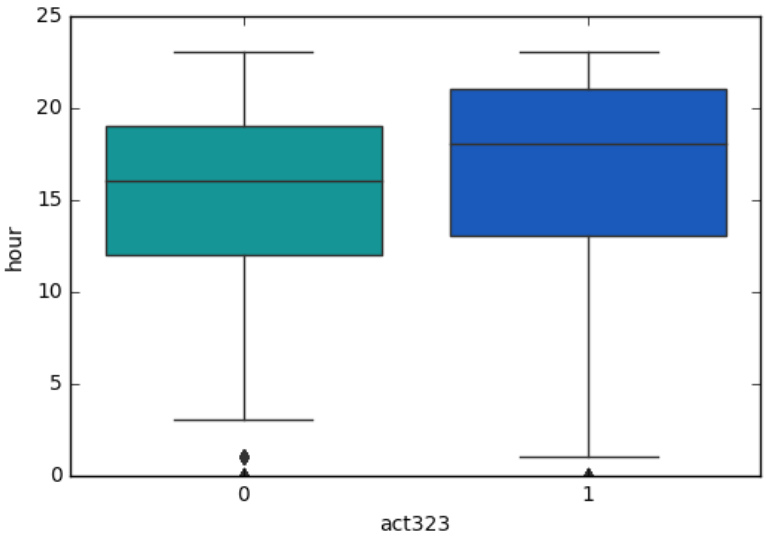
Act13(Gambling vs Hour)

The following graph represents the prediction of Kidnapping with respect to time.



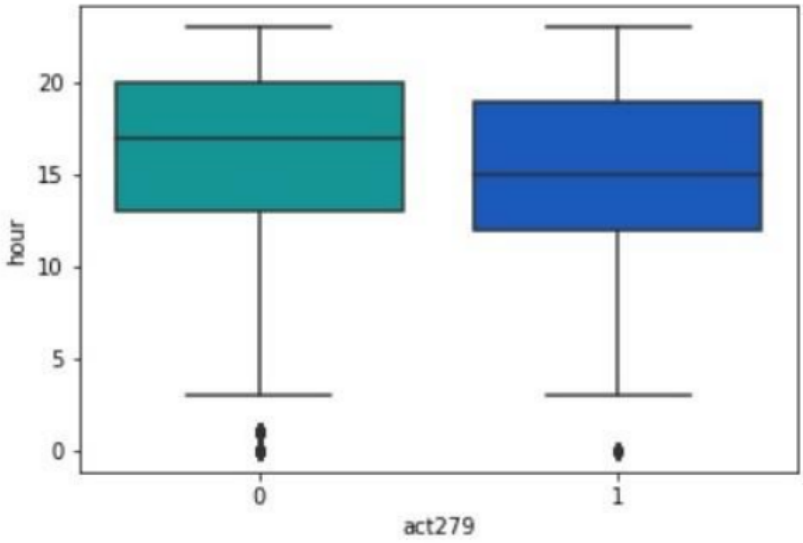
Act363(Kidnapping vs Hour)

The following graph represents the prediction of Violence with respect to time.



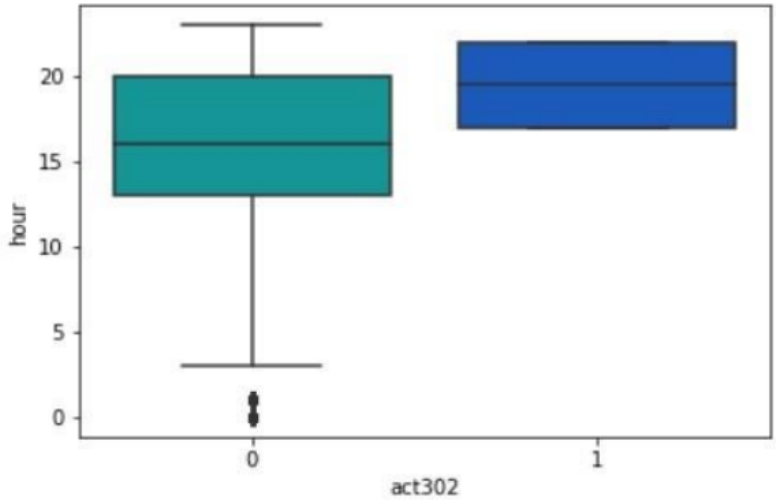
Act323(Violence vs Hour)

The following graph represents the prediction of Accident with respect to time.



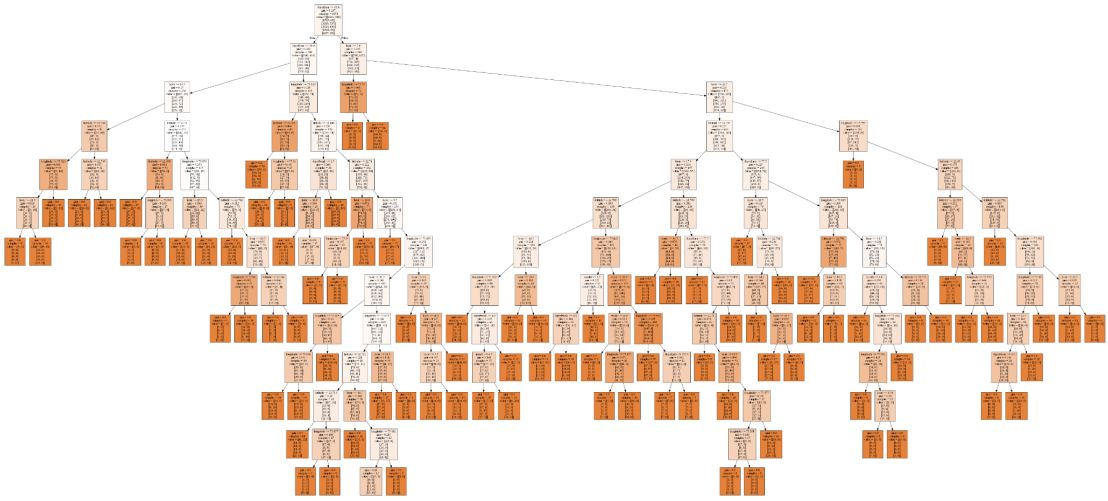
Act279(Accident vs Hour)

The following graph represents the prediction of Murder with respect to time.

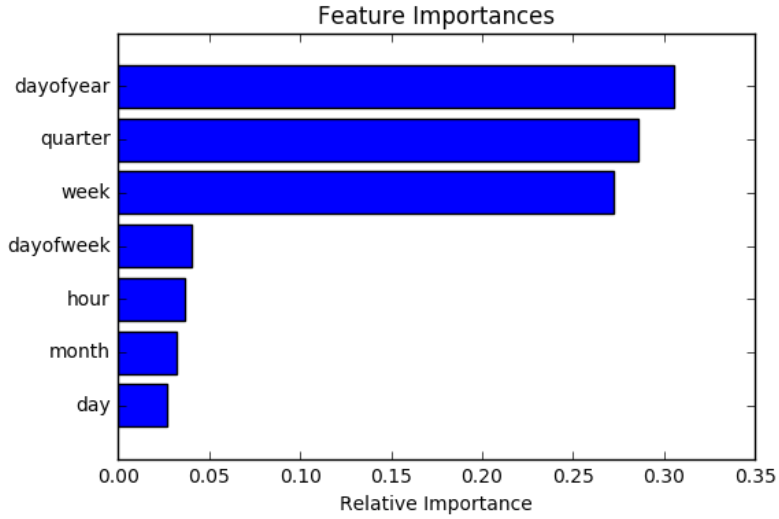


Act302(Murder vs Hour)

Decision Tree of Crime Rate Predictor.



The following bar graph shows Feature importances and Relative importance of crime rate prediction.



Results and Analysis:-

The inputs to our algorithms are time (hour, day, month, year), place (latitude and

longitude), class of crime

· Act 379-Robbery

· Act 13-Gambling

· Act 279-Accident

· Act 323-Violence

· Act 302-Murder

· Act 363-Kidnapping

The output is the class of crime that is likely to have occurred. We try out multiple classification algorithms, such as KNN (K-Nearest Neighbors), Decision Trees, and Random Forests.

The data was scraped from the publically available data from Indore police website which had been made by people in police station of different areas. Implementation of the idea started from the Indore city itself so as to limit an area for the prediction and making it less complex. The data was sorted and converted into a new format of timestamp, longitude, latitude, which was the input that machine would be taking so as to predict the crime rate in a particular location or city.

The initial problem of classifying 6 different crime categories was a challenging multi-class classification problem, and there was not enough predictability in our initial data-set to obtain very high accuracy on it. We found that a more meaningful approach was to collapse the crime categories into fewer, larger groups, in order to find structure in the data. We got high accuracy and precision on Prediction.

Accuracy of KNN: 0.97

Accuracy of Decision Tree: 0.98

Conclusion:-

Public safety and protection relate to crime, and a better understanding of crime is

beneficial in multiple ways: it can lead to targeted and sensitive practices by law enforcement authorities to mitigate crime, and more concerted efforts by citizens and authorities to create healthy neighborhood environments. With the advent of the Big Data era and the availability of fast, efficient algorithms for data analysis, understanding patterns in crime from data is an active and growing field of research.

References:-

[1] Bogomolov, Andrey and Lepri, Bruno and Staiano, Jacopo and Oliver, Nuria and Pianesi, Fabio and Pentland, Alex.2014. Once upon a crime: Towards crime prediction from demographics and mobile data, Proceedings of the 16th International Conference on Multimodal Interaction.

[2] Yu, Chung-Hsien and Ward, Max W and Morabito, Melissa and Ding, Wei.2011. Crime forecasting using data mining techniques, pages 779-786, IEEE 11th International Conference on Data Mining Workshops (ICDMW)

[3] Kianmehr, Keivan and Alhajj, Reda. 2008. Effectiveness of support vector machine for crime hot-spots prediction.