SUP'COM
Higher School of Communication of Tunis

Analysis and prediction of crimes in New York City using a machine learning approach and spatiotemporal data

By *AMINI Mohamed Omar , **FAKHFAKH Malek , ***GHRAB Rihab

*Prof. TEBOURBI Riadh

This paper reviews a web solution to fight and prevent crime. This solution, first, allows predicting a crime with 0,55 as acuuracy, based on provided user data. Second, it displays the spatial distribution and crime risk areas. This data originates from the collection of positions and information relating to crimes by citizens and authorities.

Abstract |

In today's world, security is the most prominent aspect which has been given higher priority. Recognizing the patterns of criminal activity in a location is essential to help authorities prevent it or to alert and advise citizens during their travels.

The objective of this project is to set up a web solution to fight and prevent crime. This solution should allow, first the collection of positions and information relating to crimes by citizens and authorities. Second, display of spatial distribution and crime risk areas. And third, predicting a crime based on user data.

In this work, unstructured crime data (over than 6 million records) from the crimes reported to the New York City Police Department (NYPD) through the previous ten years from 2006 to 2015, were extracted to predict the behavior of criminals' networks and transform it into useful information using machine learning approach and spatiotemporal data.

Keywords | Crime prediction; spatial analysis; Feature engineering; Prediction; ML model;

I. Introduction :

Crime is one of the dominant and alarming aspects of society. Everyday a huge number of crimes are committed. These frequent crimes have made the lives of common citizens restless. So, it is obvious to comprehend the patterns of criminal activity to prevent them.

Furthermore, the capability to predict any crime on the basis of time, location.. can help in providing useful information to law enforcement from a strategical perspective. And they can work effectively and respond faster if they have early information and pre-knowledge about criminal activities of the different points of a city.

However, predicting the crime accurately is a challenging task because crimes are increasing at an alarming rate.

Thus, the crime prediction and analysis methods are very important to detect future crimes and reduce them.

In recent times, artificial intelligence has shown its importance in almost all fields and crime prediction is one of them.

In this paper, a supervised learning technique is used to predict criminal activity.

The proposed web solution predicts crimes by analyzing New York city criminal activity data set for 10 years from 2006 to 2015. This data contains records of previously committed crimes and their patterns. The system stands on two main algorithms: Decision tree and k-nearest neighbor (KNN) algorithms are applied to predict crime. But these two algorithms provide low accuracy in prediction.

Then, random forest is applied as an ensemble method, and AdaBoost is used as a boosting method to increase the accuracy of prediction.

However, log-loss is used to measure the performance of classifiers by penalizing false classifications. As the dataset contains highly class imbalance problems, a random undersampling method for the random forest algorithm gives the best accuracy. The final accuracy is 99.16% with a 0.17% log loss.

The rest of the paper is organized as follows :

- In Section II, we describe the processing we made on our database to finally keep just relevant data.
- In section III, we introduce the followed steps to to choose the most efficient model
- Concluding remarks are offered in Section IV.

II. Data Preprocessing:

With the advancement in technology, criminal behavior is becoming more and more complex.

For crime control, the nature of crime must be understood. Also, the spatial analysis helps to decode the spatial behavior of criminal activities and assist law enforcement in making predictions about future crimes that may occur.

To develop our prediction model :

Data exploration is the initial step in data analysis, where we explored the NYPD Complaint Map. It is a large dataset of a valid felony, misdemeanor, and violation crimes reported to the New York City Police Department (NYPD) in an unstructured way to uncover initial patterns, characteristics, and points of interest.

The process of data exploration isn't meant to reveal every bit of information our dataset holds, but rather to assign meaning to thousands of rows and columns of data points and communicate that meaning using a combination of manual methods and automated tools such as data visualization, charts and statistical techniques to describe dataset characterizations, in order to better understand the nature of the data.

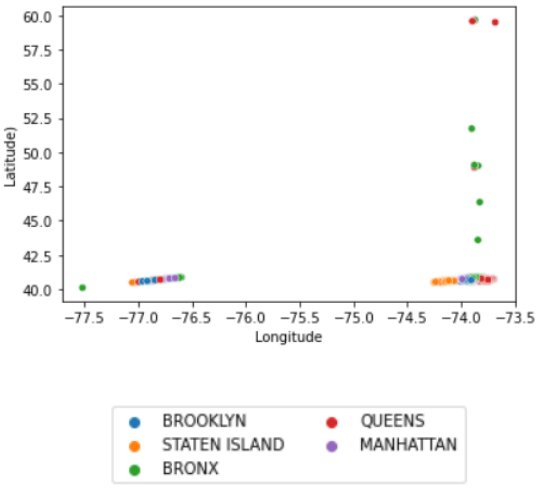


figure 1: Crimes distribution according to the state

In order to understand more the distribution of the data according to our target feature[OFNS_DESC]. We defined the heatmap related to two different states of the data which are BROOKLYN and MANHATTAN.

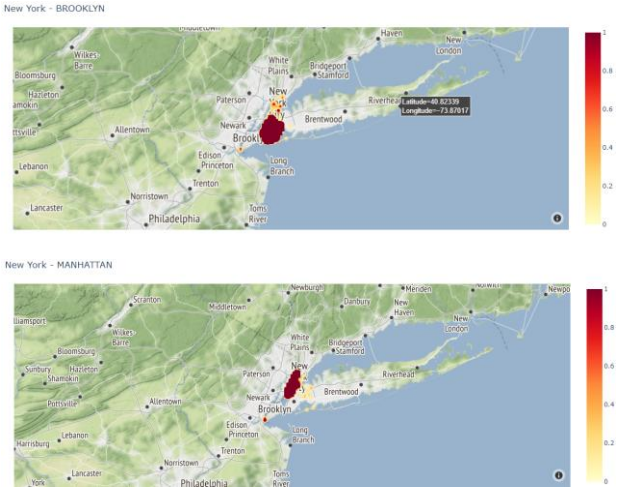


figure 2: BROOKLYN and MANHATTAN heatmaps

The dataset contains the different positions of the crimes in terms of **Latitude** and **Longitude.** Using these positions we can obviously delimit the zones that have high rates of crimes same as less and normal ones. The figure below shows the different zones defined by our dataset.

figure 3: Different zones of crimes

According to different records, we witnessed that crimes are committed during the weekdays with different rates and severity so we wanted to check the number of crimes per day. The bars in the following representation explain more the idea.

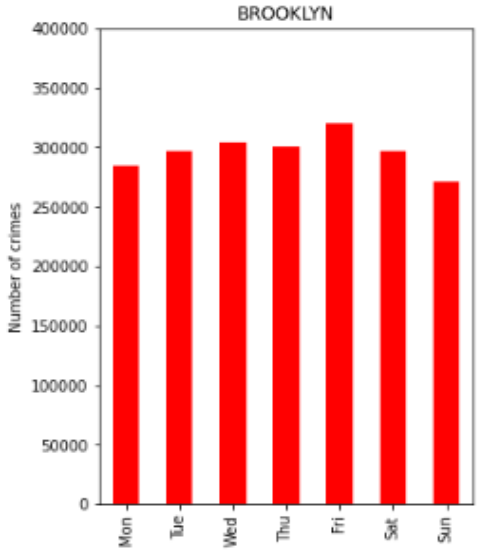


figure 4: Crimes distribution per day

It's clear that Friday is in the lead in the number of crimes per day. In advance, we wanted to illustrate this observation by plotting the number of crimes per hour for each day as shown in the figure below.

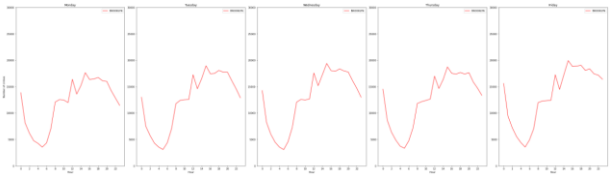


figure 5: Crimes distribution for each day per hour

Data cleaning: one of the most important steps for data decision-making is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within our large dataset.

So the first of all, by computing missing values in our features we noticed that the features [PARKS_NM, HADEVELOPT, HOUSING_PSA, TRANSIT_DISTRICT, STATION_NAME] are almost null values so it's better to delete them to keep just relevant features.

Then we also dropped data with negative time duration

Since the data set contain text and there are only a few algorithms such as CATBOAST, decision trees can handle categorical values very well but most of the algorithms expect numerical values to achieve state-of-the-art results.

There are many ways to convert categorical values into numerical values. Each approach has its own trade-offs and impact on the feature set. So we used **Label Encoding** to convert the labels into a numeric form so as to convert them into the machine-readable form.

```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()
# Encode labels in column 'species'.
data_final['CRM_ATPT_CPTD_CD']= label_encoder.fit_transform(

data_final['CRM_ATPT_CPTD_CD'].unique()
```

figure 6:Label Encoding of CRM_ATPT_CPTD_CD

Removing outliers is one of the most important tools to avoid overfitting. Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the model's ability to generalize. Overfitting is more likely with nonparametric and nonlinear models that have more flexibility when learning a target function. As such, many nonparametric machine learning algorithms also include parameters or techniques to limit and constrain how much detail the model learns.

In the figure below we eliminated outliers in the time duration feature.

```
# Drop the rows with td<0

neg_outliers=data_final[td]<=0
data_final.info()
data_final.head()

# Set outliers to NAN
data_final[neg_outliers] = np.nan
data_final.info()
data_final.head()

# Drop rows with negative td
data_final.dropna(subset=[td],axis=0,inplace=True)
data_final.info()
data_final.head()
```

figure 7:Removing outliers

Finally, we filled in nan values with the most occurrent value for each feature.

```
# filling with most common class
data_final = data_final.apply(lambda x: x.fillna(x.value_counts().index[0]))
data_final
```

figure 8:Filling in missing values

One of the crucial steps in feature engineering that gives a clearer overview of the relation between features and figure out the necessity of each one is feature correlation. Correlation is a statistical term which in common usage refers to how close two variables are to having a linear relationship with each other.
Features with high correlation are more linearly dependent and hence have almost the same effect on the dependent variable. So, when two features have a high correlation, we can drop one of the two features.

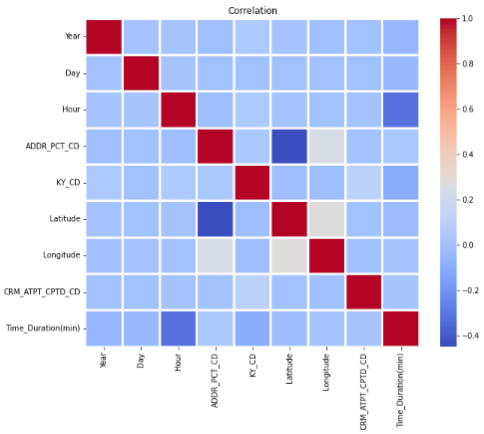


figure 9: Features correlation

III. Building the model

Model Selection: Model selection is the process of selecting one final machine learning model from among a collection of candidate machine learning models for a training dataset.

Model selection is a process that can be applied both across different types of models (e.g. logistic regression, SVM, KNN, etc.) and across models of the same type configured with different model hyperparameters.

All models have some predictive error, given the statistical noise in the data, the incompleteness of the data sample, and the limitations of each different model type.
Therefore, the notion of a perfect or best model is not useful. Instead, we must seek a model that is "good enough."
Therefore, a "good enough" model may refer to many things and is specific to your project, such as:

- A model that meets the requirements and constraints of project stakeholders.
- A model that is sufficiently skillful given the time and resources available.
- A model that is skillful as compared to naive models.
- A model that is skillful relative to other tested models.

- A model that is skillful relative to the state-of-the-art.

In our project since we are manipulating a multiclass classification problem with mainly a set of independent features we selected the following models to be trained and compared: KNN, RandomForest, Xgboost.

KNN:

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.

The KNN Algorithm :

1. Load the data
2. Initialize K to your chosen number of neighbors
3. For each example in the data

3.1. Calculate the distance between the query example and the

a current example from the data

3.2 Add the distance and the index of the example to an ordered collection

4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
5. Pick the first K entries from the sorted collection
6. Get the labels of the selected K entries
7. If regression, return the mean of the K labels
8. If classification, return the mode of the K labels

RandomForest :

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results for classification problems.

The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Put simply: random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

The RandomForest Algorithm :

1. Select random K data points from the training set.
2. Build the decision trees associated with the selected data points (Subsets).
3. Choose the number N for decision trees that you want to build.
4. Repeat Step 1 & 2.
5. For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

XGboost :

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks.

Gradient Boosting is a special case of boosting where errors are minimized by gradient descent algorithm e.g. the strategy consulting firms leverage by using case interviews to weed out less qualified candidates.

Think of XGBoost as gradient boosting on ‘steroids’ (‘Extreme Gradient Boosting’ for a reason)It is a perfect combination of software and hardware optimization techniques to yield superior results using less computing resources in the shortest amount of time.

The XGboost Algorithm :

The implementation of the algorithm was engineered for the efficiency of computing time and memory resources. A design goal was to make the best use of available resources to train the model. Some key algorithm implementation features include:

1. Sparse Aware implementation with automatic handling of missing data values.
2. Block Structure to support the parallelization of tree construction.
3. Continued Training so that you can further boost an already fitted model on new data.

XGBoost is free open source software available for use under the permissive Apache-2 license.

Model Training :

1. KNN: For the knn model we approached by setting the parameters to mainly :
   - n_neighbors=5

   Next, we fitted the model to train the dataset (X_train, y_train). We got an accuracy of exactly after predicting the test dataset and comparing it to real values.

```
# Import KNeighborsClassifier from sklearn.neighbors
from sklearn.neighbors import KNeighborsClassifier
# Create a k-NN classifier with 3 neighbors
knn = KNeighborsClassifier(n_neighbors=3)

# Fit the classifier to the data
knn.fit(X_train,y_train)

# Predict the labels for the training data X
y_pred = knn.predict(X_test)

# Get the accuracy score
acc_knn=accuracy_score(y_test, y_pred)

# Append to the accuracy list
accuracy_lst.append(acc_knn)

print('[K-Nearest Neighbors (KNN)] knn.score: {:.3f}.'.format(knn.score(X_test, y_test)))
print('[K-Nearest Neighbors (KNN)] accuracy_score: {:.3f}.'.format(acc_knn))
```

figure 10: KNN Implementation

2. RandomForest: We used the RandomForestClassifer as an ensemble learning tool in order to obtain a better accuracy value. The only needed parameter is :
   - n_estimators=10

   At the end of the process, we obtained 0.552 as an accuracy.

figure 11: RandomForets Implementation

We proceeded in advance to compute the feature importance value for each column of our dataset. This approach will help enhance the accuracy of the model since we will be able to train the model only on the most important features.

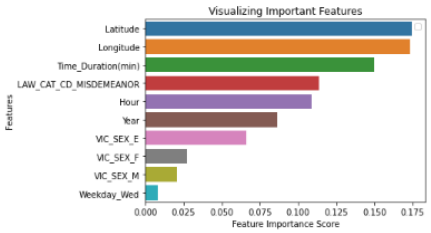
figure 12: Feature Importance

After the feature importance process, we found out that the most relevant features that are affecting the

```
feature_imp = pd.Series(clf.feature_importances_,index=X.columns).sort_values(ascending=False)

# Creating a bar plot, displaying only the top k features
k=10
sns.barplot(x=feature_imp[:10], y=feature_imp.index[:k])
# Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```



prediction result in the most dominant way are [Year, Hour, Latitude, Longitude, Time_Duration(min), LAW_CAT_CD_MISDEMEANOR, VIC_SEX_E]

```
# Create a selector object that will use the random forest classifier to identify
# features that have an importance of more than 0.03
from sklearn.feature_selection import SelectFromModel

sfm = SelectFromModel(clf, threshold=0.03)

# Train the selector
sfm.fit(X_train, y_train)

feat_labels=X.columns

# Print the names of the most important features
for feature_list_index in sfm.get_support(indices=True):
    print(feat_labels[feature_list_index])
```

```
Year
Hour
Latitude
Longitude
Time_Duration(min)
LAW_CAT_CD_MISDEMEANOR
VIC_SEX_E
```

figure 13: Most important features

Below is the training result of the modified RandomForest :

```
# Transform the data to create a new dataset containing only the most important features
# Note: We have to apply the transform to both the training X and test X data.
X_important_train = sfm.transform(X_train)
X_important_test = sfm.transform(X_test)

# Create a new random forest classifier for the most important features
clf_important = RandomForestClassifier(n_estimators=5, random_state=0, n_jobs=-1)

# Train the new classifier on the new dataset containing the most important features
clf_important.fit(X_important_train, y_train)
```

```
RandomForestClassifier(n_estimators=5, n_jobs=-1, random_state=0)
```

figure 14: RandomForetsmodified implementation

3. XGBoost: For the XGboost model and according to our dataset requirement we have defined the following values for our model's parameters :
   - n_estimatores = 3
   - max_depth =5

   We have trained the model on the

split dataset (X_train, y_train). After predicting test values and in order to check[1] the model's performance, we've calculated the value of the accuracy using **accuracy_score**.

We got **0.55** as final accuracy, here the value of the accuracy is affected by the dataset quality.

```
# fit model no training data
model = XGBClassifier(n_estimators=3, max_depth=5)
model.fit(X_train, y_train)
```

```
XGBClassifier(max_depth=5, n_estimators=3, objective='multi:softprob')
```

```
# make predictions for test data
y_pred = model.predict(X_test)
```

```
# evaluate predictions
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: %.2f%%" % (accuracy))
```

figure 15: XGBoost Implementation

After training all models we were able to compare all of them and get a clear idea about the model to choose. Here, since XGBoost marked the best accuracy among all other models we will be deploying it as a final model for this dataset.

In the below figure, we illustrate the performance of each model.

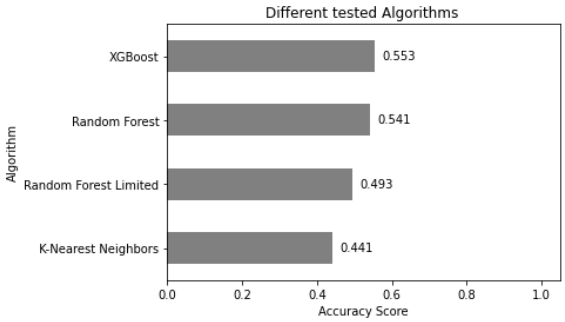


figure 16: Tested Algorithms

IV. Conclusion

The usage of digital information archives is a cost-effective way of predicting crime events occurring in a country.
Data related to crimes can be converted into useful information for the prediction of fruitful results.
The location-based geo-coded data, processed through GIS-based software, i.e., ArcMap provided locations based statistical information, which helped in the identification of the patterns, trends, and relationships between crime features.

In this paper, owing to using unstructured crime data (over than 6 million records) from the crimes reported to the New York City Police Department (NYPD) through the previous ten years from 2006 to 2015, we developed a useful prediction model.
The solution we have developed requires spatiotemporal data as input to to give a probability that a crime accurs with 0,55 in terms of accuracy.

[1] INDP3 AIM 2021-2022