University of Bucharest
Faculty of Mathematics and Computer Science
Department of Computer Science

Andrei Dumitriu
MSC in Computer Science
April 2020

Practical Machine Learning

Crime in Chicago dataset

Dataset: https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2/data
Purpose:

1. Compare 2 supervised algorithms on the dataset
2. Compare 2 unsupervised algorithms on the dataset

Understanding the dataset

Chicago's situation:

Chicago is the most populous city in the Illinois state and is the 3rd most populous city in the US. It is an international hub for finance, culture, commerce, industry, education, technology, telecommunication and transportation.

With 58 million domestic and international visitors in 2018, it is the second most visited city in the US. It ranked first in the 2018 Time Out magazine life index.

Chicago had a murder rate of 18.5 per 100.000 residents in 2012, ranking 16th among US cities with 100.000 people or more. Analysis in 2015 showed a 12.5% increase, with 468 murders, compared to 416 in 2014. It also noticed 2900 shootings, 13% increase from 2014 and 29% increase since 2013. In 2016, the Chicago Police Department reported a dramatic increase in gun violence, with 4331 victims. It also reported 7623 murders in 2016, an increase of 62.79% from 2015. To combat this, in June 2017, the Chicago Police Department and the Federal ATF announced a new task force.
In 2016 it accounted for 22% of the nationwide increase, making it an important study case.

Reports in 2013 state that most of Chicago's violent crime comes from gangs trying to maintain control of drug-selling territories. [1]

Exploratory Data Analysis and Feature Engineering

Dataset description:
It is provided by the Chicago Police Department and is updated daily, containing information from 2001 to present, excluding the last 7 days.
I chose to focus on 2015 - 2019, excluding 2020 due to possible different behavior due to Covid-19 and an increased chance of ongoing, subject to change, investigations. I chose

2015 since 2009 - 2013 contains the large portion of null values, exposing us to an increased bias based on the Year column.

Restrictions: Since the data visualisation maps should be considerate approximate, **attempts to derive specific addresses are strictly prohibited.** (source: Chicago Police Department) [2]

Limitations: Since the crime details change with new information, the dataset is not guaranteed for accuracy, completion, timeliness or correct sequencing.

For the purpose of this project, we will not address the limitations, taking the dataset “as-is”. For an in-depth conclusion-driven analysis, these should not be ignored.

At the time of download (**May** 2020), the dataset contains **22 features** and **4899980** observations. Out of these, selecting the 2015 - 2019 period yields **1282805** observations
Data columns (total 22 columns):

#	Column	Dtype
0	ID	object
1	Case Number	object
2	Date	object
3	Block	object
4	IUCR	object
5	Primary Type	object
6	Description	object
7	Location Description	object
8	Arrest	object
9	Domestic	object
10	Beat	object
11	District	object
12	Ward	object
13	Community Area	object
14	FBI Code	object
15	X Coordinate	object
16	Y Coordinate	object
17	Year	object
18	Updated On	object
19	Latitude	object
20	Longitude	object
21	Location	object

Due to inaccuracies in data types in the same column, and significantly decreased loading time, I have loaded the dataset with **dtype=object**, applying feature engineering later on.

Initial feature and data engineering:

1. Removing the obviously unused columns from the beginning:
    - ID
    - Updated On
    - Description
    - X Coordinate, Y Coordinate, Block, Location, and Location Description, due to the multitude of Location information.

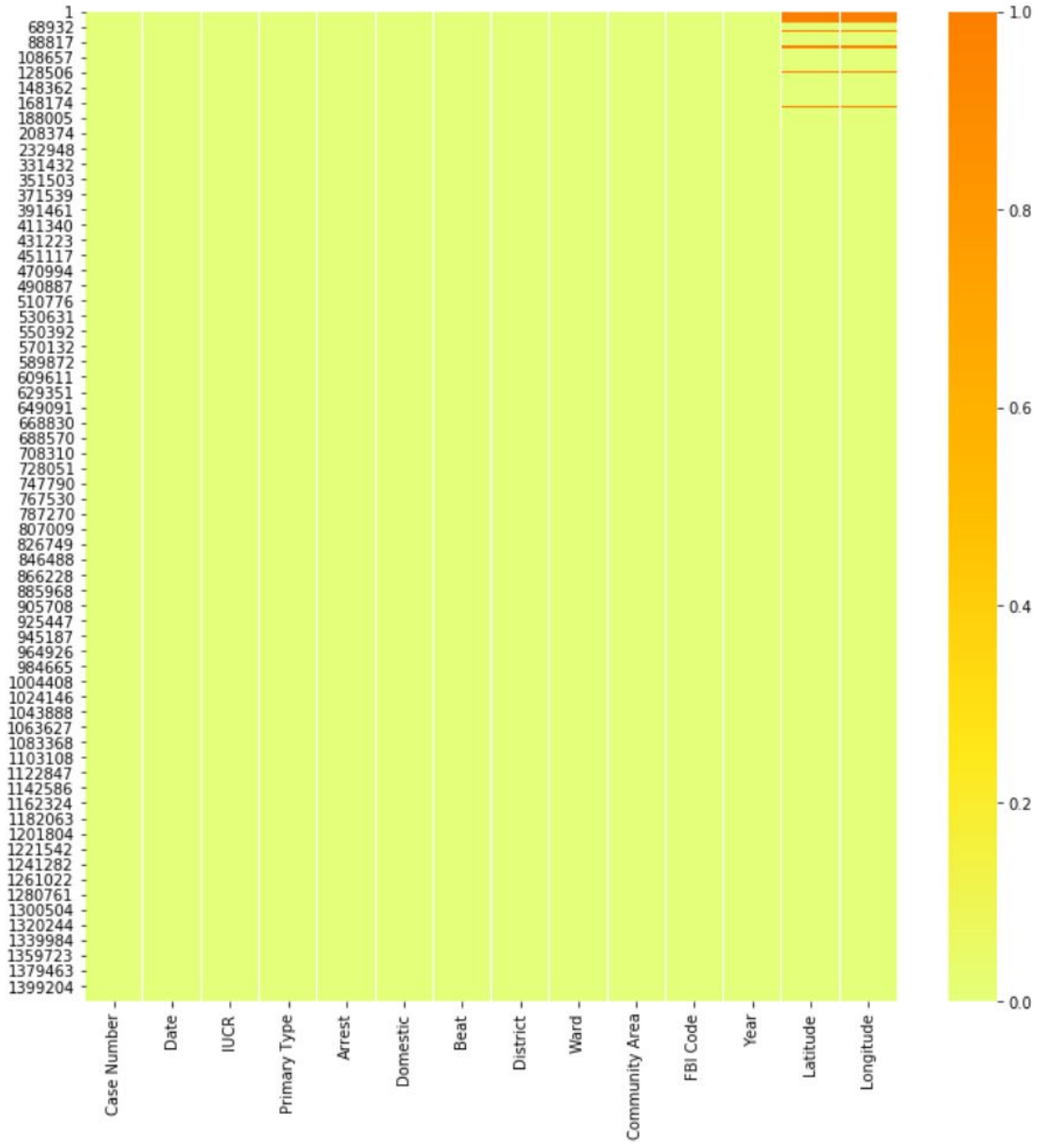
2. Splitting the Date column into Year, Month, Day and Hour
3. Taking only the years between 2001 and 2019, inclusive

Data visualisation:

Calculate the number of missing values for each column:

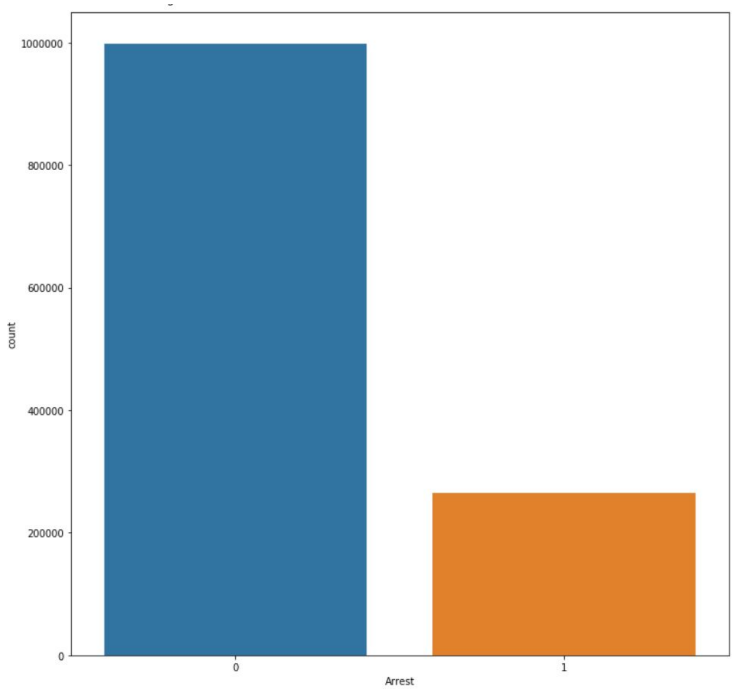
```
Case Number           0
Date                  0
IUCR                  0
Primary Type          0
Arrest                0
Domestic              0
Beat                  0
District              1
Ward                 22
Community Area        0
FBI Code              0
Year                  0
Latitude          18005
Longitude         18005
dtype: int64

Calculate the % of missing values for each column:

Case Number       0.0
Date              0.0
IUCR              0.0
Primary Type      0.0
Arrest            0.0
Domestic          0.0
Beat              0.0
District          0.0
Ward              0.0
Community Area    0.0
FBI Code          0.0
Year              0.0
Latitude          1.4
Longitude         1.4
```

Plotting the missing values are contained within the dataset

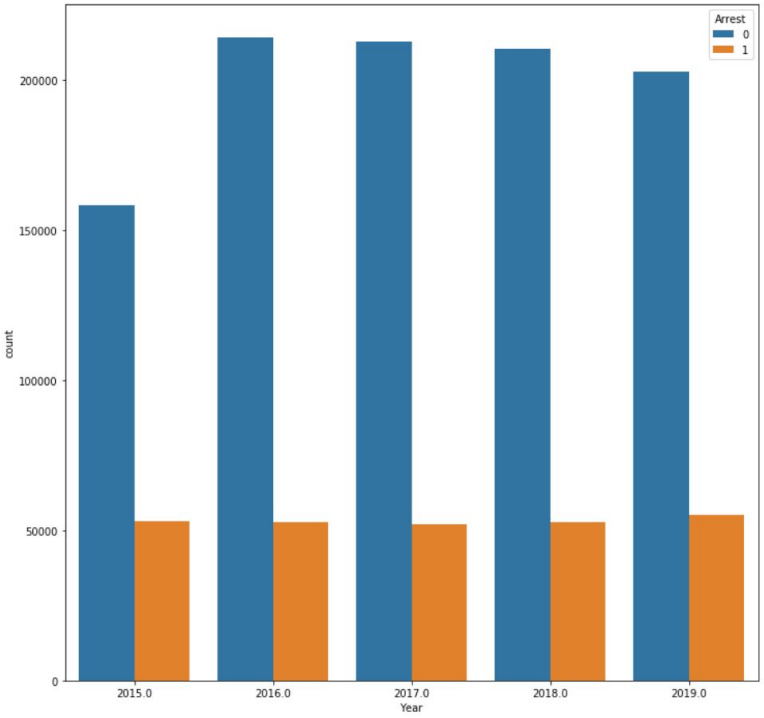


Since the dataset is large (and we will be using a subset later on), we can afford to drop all the rows that contain null values, instead of applying other methods, such as filling in with the mean value.

Total number of cases ending without arrests: 999008 (78.986%)
Total number of cases ending with arrests: 265770 (21.014%)

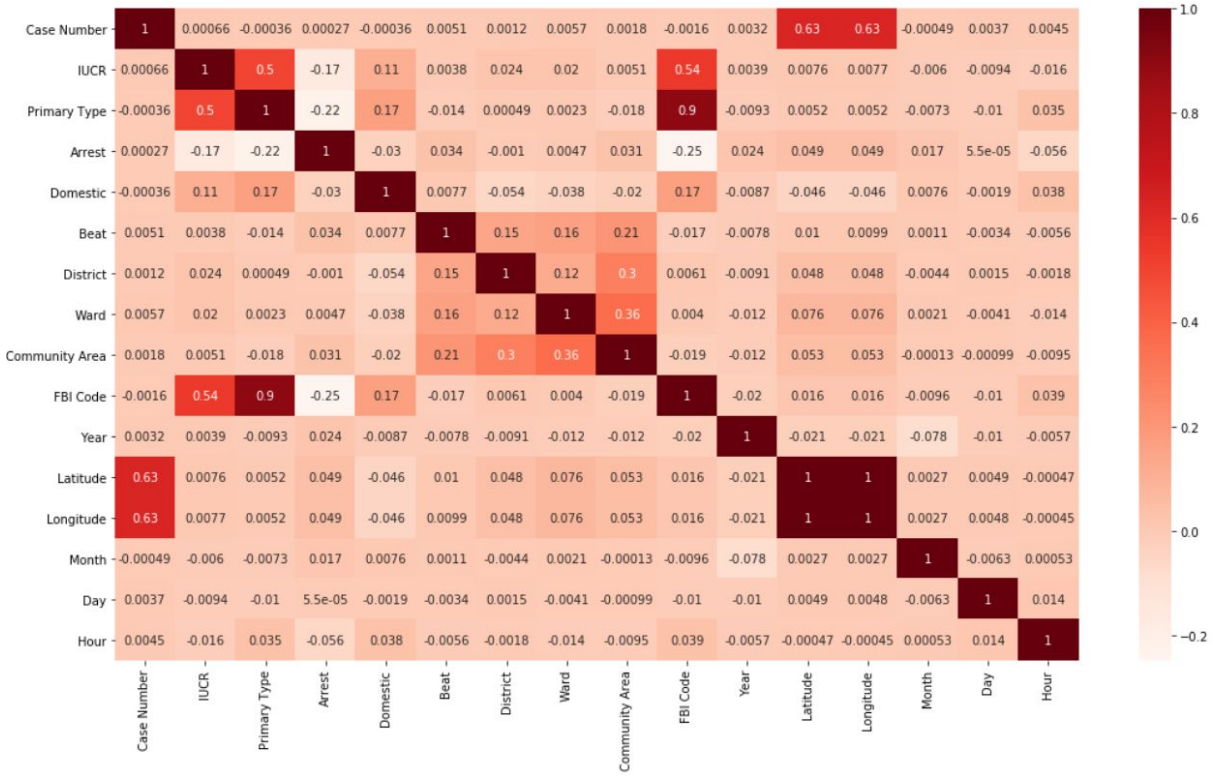


Number of cases ending in arrests per year



We've also randomly sampled **100.000** entries from the dataset, due to its large size. We're going to use the same sample both for supervised and unsupervised learning.

Heatmap for feature correlation



Given all the information, we're going to use the following features from the dataset:

1. **IUCR** - Illinois Uniform Crime Reporting code
2. **Primary Type** - The primary description of the IUCR code
3. **Domestic** - Indicates whether the incident was domestic-related as defined by the Illinois Domestic Violence Act
4. **Beat** - Indicates the beat where the incident occurred. A beat is the smallest police geographic area – each beat has a dedicated police beat car. Three to five beats make up a police sector, and three sectors make up a police district. The Chicago Police Department has 22 police districts. See the beats at
5. **District** - Indicates the police district where the incident occurred
6. **Ward** - The ward (City Council district) where the incident occurred
7. **Community Area** - Indicates the community area where the incident occurred
8. **FBI Code** - Indicates the crime classification as outlined in the FBI's National Incident-Based Reporting System (NIBRS)
9. **Year** - Year the incident occurred
10. **Month** - Month the incident occurred
11. **Day** - Day the incident occurred
12. **Hour** - Hour the incident occurred
13. **Latitude** - The latitude of the location where the incident occurred. This location is shifted from the actual location for partial redaction but falls on the same block
14. **Longitude** - The longitude of the location where the incident occurred. This location is shifted from the actual location for partial redaction but falls on the same block

Target: **Arrest** - Indicates whether an arrest was made

First of all, we scale all the features using StandardScaler

Target selection
We will try to define whether an incident resulted in an **arrest** or not.

Data split

1. Extract the X_Features and the y_target from the dataset
2. Split **twice** using train_test_split, ending up with **50% for train**, **25% for validation and 25% for test**

Baseline definition:

For this, we used the DummyClassifier from sklearn, with the “most_frequent” strategy. Of all the strategies (stratified, most_frequent, prior, uniform and constant) this predicted the best accuracy, along with constant and prior, which are basically the same thing in this case.

The baseline accuracy for the model is **79%**

While accuracy is important, in this report we will take into consideration the precision, recall, and f1 score as well.

Confusion matrix:

Actual \ Predicted	Predicted 0	Predicted 1
Actual 0	19871	0
Actual 1	5129	0

Classification report

	Precision	Recall	F1-score	support
0	0.79	1	0.89	19871
1	0	0	0.00	5129
accuracy			0.79	25000
Macro avg	0.40	0.50	0.44	25000
Weighted avg	0.63	0.79	0.70	25000

Supervised Learning

For this part, the data has been split into **train (50%), validation (25%) and test (25%)**

K-nearest neighbors algorithm

It is a method used for classification and regression. The input consists of the K closest training examples in the feature space. In KNN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. [4]

Hyperparameter tuning:
For hyperparameter tuning, we are going to use **GridSearchCV**, an exhaustive search over specified parameter values.

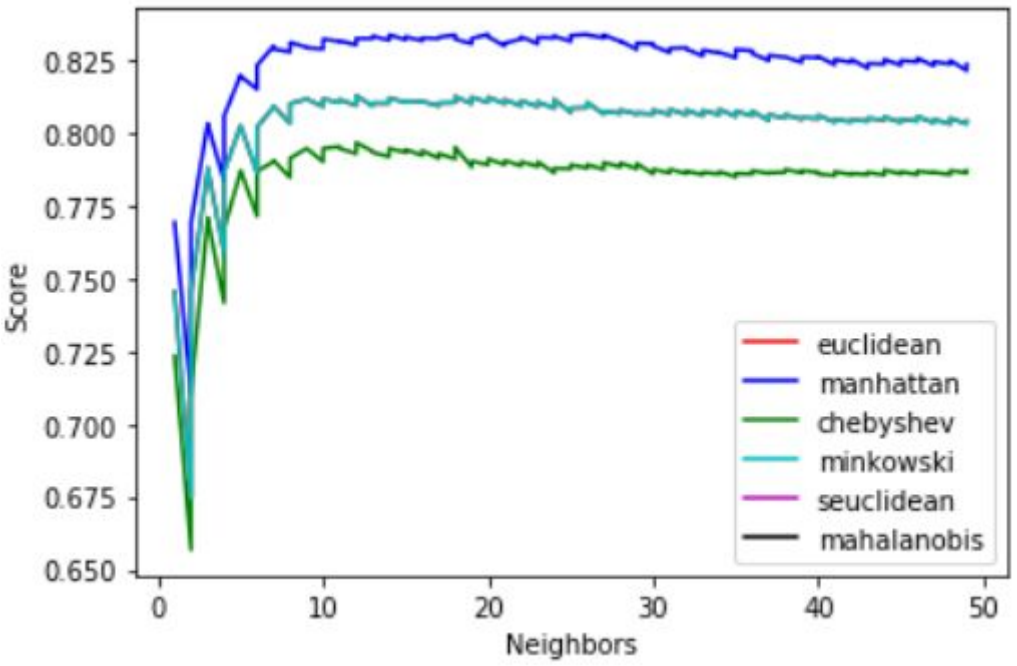
We are doing an exhaustive search over the following parameters for KNN:

1. 'weights': ['uniform', 'distance'],
2. 'n_neighbors': range(1, 50),
3. 'metric': ['euclidean', 'manhattan', 'chebyshev', 'minkowski', 'seuclidean', 'mahalanobis'],

We are leaving the algorithm to ‘auto’ so it can pick the best choice based on the provided parameters combination.

To obtain the best results, we ended up fitting 5 folds for each of 588 candidates, totaling 2940 fits

Metrics comparison plot. On The X axis we have K, the # of neighbors and on the Y axis we have the accuracy.



The main observable metrics in the plot are Manhattan, Minkowski and Chebyshev, with the rest being an almost identical, indistinguishable match to one of these 3. The rest result is with **K = 15 using the Manhattan metric, with an accuracy of 83.18% on validation data and 84.82% on test data**

Results for KNN, for K = 15:

Confusion matrix:

Actual \ Predicted	Predicted 0	Predicted 1
Actual 0	19460	411
Actual 1	3383	1746

Classification report

	Precision	Recall	F1-score	support
0	0.85	0.98	0.91	19871
1	0.81	0.34	0.48	5129
accuracy			0.85	25000
Macro avg	0.83	0.66	0.70	25000
Weighted avg	0.84	0.85	0.82	25000

Random Forest algorithm

It is an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees. It consists of a large number of individual decision trees that work together. Each tree produces a class prediction and the class with the most votes is chosen as the final prediction. [5]

Relationship with KNN:
A relationship between RF and KNN algorithm was pointed out by Lin and Jeon in 2020. Both can be viewed as so-called weighted neighborhoods schemes.

Hyperparameter tuning:
In order to implement RF, there are two parameters that are important. The number of trees (n_estimators) and the number of features in each split (max_features).
According to Liaw & Wiener, a large number of trees will provide a stable result. Breinman stated that while using a higher number of trees will not improve the result, it will not negatively impact it either. [6]
There are many studies on the max_features parameters, but a good guess is log(n_features).
For hyperparameter tuning, we are going to use GridSearchCV, an exhaustive search over specified parameter values.

We are doing an exhaustive search over the following parameters for RF:

1. 'n_estimators': [50, 100, 200, 300],
2. 'max_features': ['auto', 'sqrt', 'log2'],
3. 'max_depth' : [None, 4, 5, 6, 7, 8],
4. 'criterion' :['gini', 'entropy'],
5. 'bootstrap': ['False', 'True']

To obtain the best results, we ended up fitting 5 folds for each of 288 candidates, totalling 1440 fits.
We found the best results with the following parameters:

1. ‘n_estimators’: 100
2. ‘max_features’: ‘log2’
3. ‘max_depth’: None
4. ‘criterion’: ‘gini’
5. ‘bootstrap’: ‘False’

With an accuracy of **88% on validation data and 89% on test data**

Results for Random Forest:

Confusion matrix:

Actual \ Predicted	Predicted 0	Predicted 1
Actual 0	19524	347
Actual 1	2351	2778

Classification report

	Precision	Recall	F1-score	support
0	0.89	0.98	0.94	19871
1	0.89	0.54	0.67	5129
accuracy			0.89	25000
Macro avg	0.89	0.76	0.80	25000
Weighted avg	0.89	0.89	0.88	25000

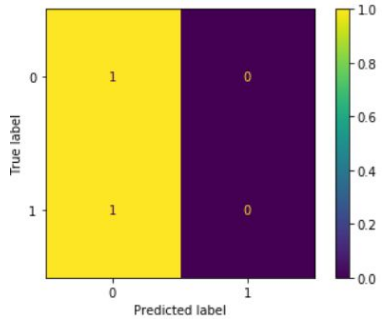
Comparison between KNN, RF and baseline model:
Both KNN (**85%)** and RF (**89%**) yield an increase of accuracy over the baseline model (**79%**). RF is slightly better, with 4% higher accuracy than KNN. This is due to the fact that in this example, K is slightly large and we suffer from the curse of dimensionality. Also, the data is imbalanced, as the label is split in 21% - 79%.

Random Forest yielded considerably better results with **89%** accuracy, compared with the baseline of **79%.**

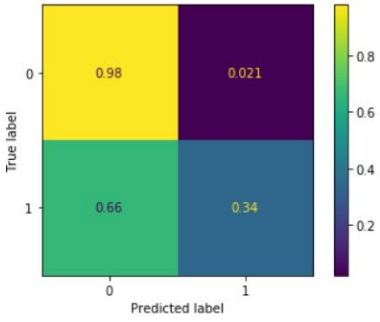
However, we can see that while KNN had lower overall accuracy, it had a slightly better result in predicting **True Negatives**. This is due to the imbalanced label value distribution.

Final results for supervised learning:

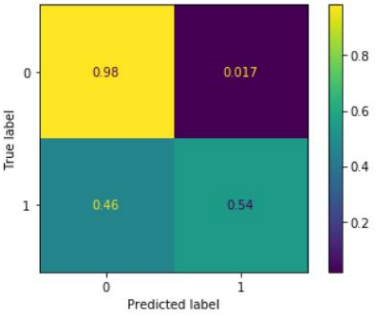
Base



KNN



RF

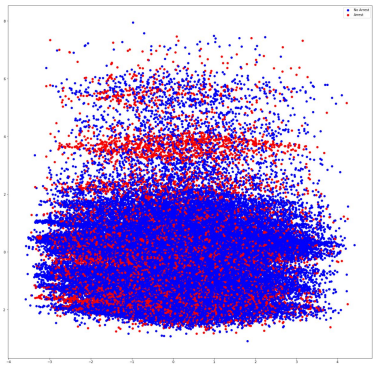
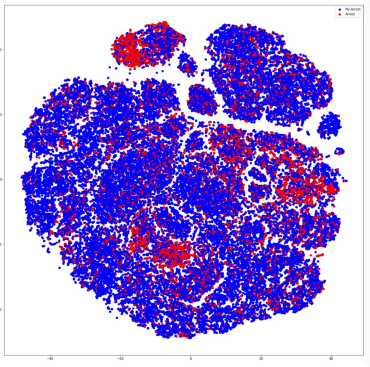
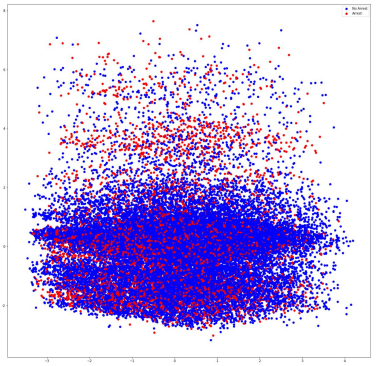
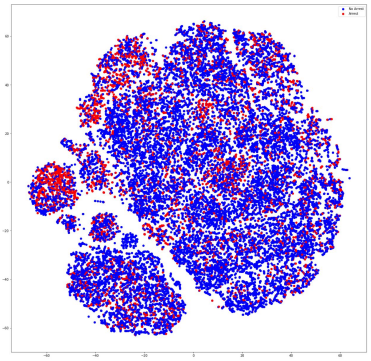


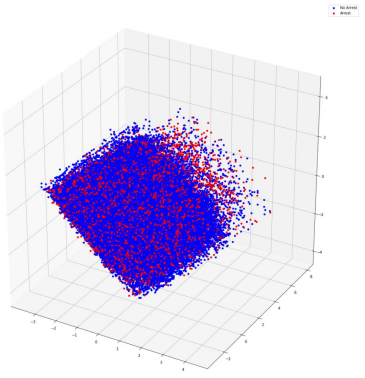
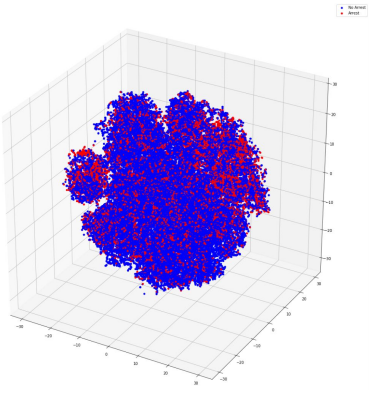
	Base	KNN	RF
Accuracy	79%	85%	**89%**
Precision (Macro avg)	40%	83%	**89%**
Precision (Weighted avg)	63%	84%	**89%**
Recall (Macro avg)	50%	66%	**76%**
Recall (Weighted avg)	79%	85%	**89%**
F1-score (Macro avg)	44%	70%	**80%**
F1-score (Weighted avg)	70%	82%	**88%**

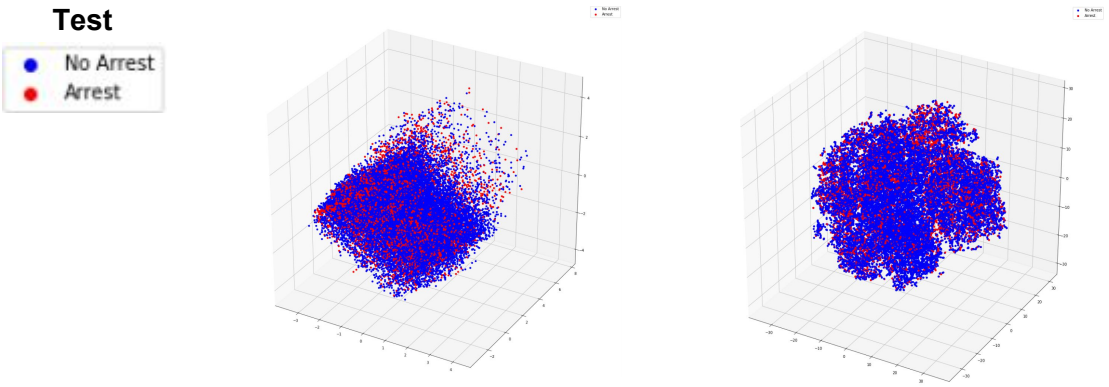
Unsupervised Learning

For this part, the data has been split into **train (70%) and test (30%),** using **K Means** and **Hierarchical clustering (agglomerative)**.

Visualizing the clusters

Data \ Method	PCA 2D	t-SNE 2D
Train		
Test		

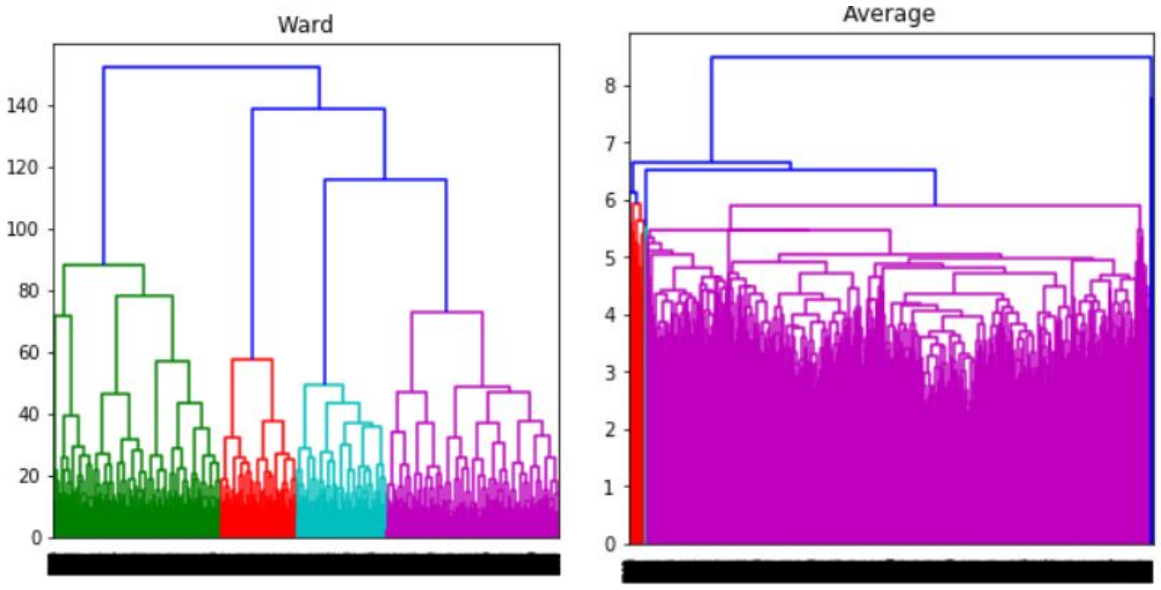
Data \ Method	PCA 3D	t-SNE 3D
Train		

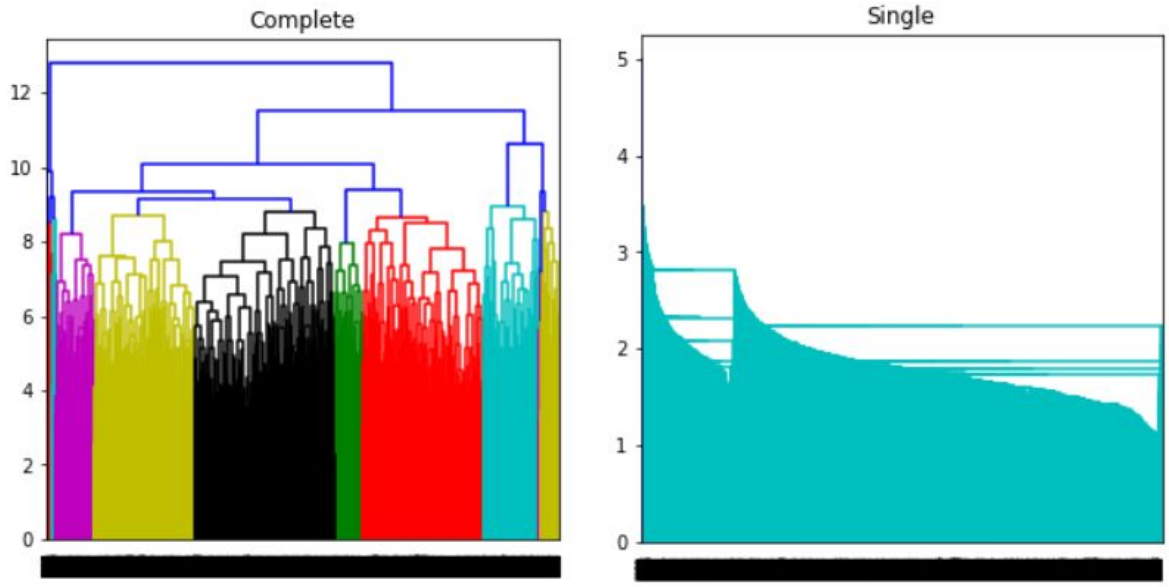


Hierarchical Clustering

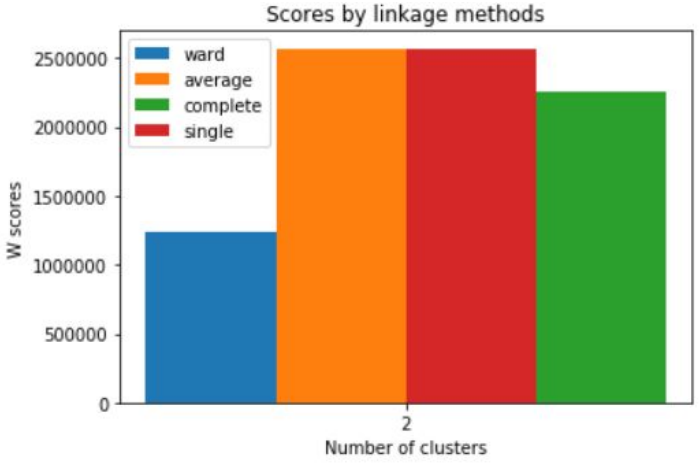
Is a method of cluster analysis which seeks to build a hierarchy of clusters, as we will see. In this example we will use **Agglomerative clustering.**

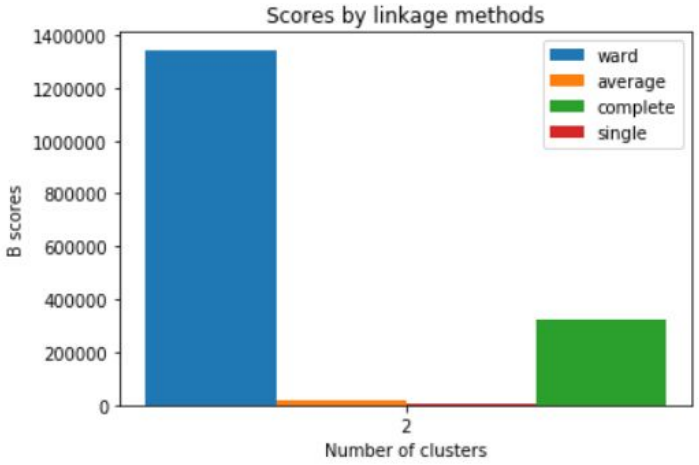
First of all, we build the dendrograms for all the linkages. (ward, average, complete and single)
Seeing that we know that we want to cluster the data into 2 clusters, based on the labels, we will be working on 2 examples for both HC and Kmeans. We will be splitting the data into 2 clusters, comparing the results with the baseline and supervised algorithms.





For **n = 2 clusters**,
The clustering solution consists in maximizing B(C) or minimizing W(C)
W(C) = The distance between points in the same cluster
B(C) = The distance between points from different clusters





HC Ward

We have an accuracy of **69%**

Confusion matrix:

Actual \ Predicted	Predicted 0	Predicted 1
Actual 0	19926	3744
Actual 1	5552	778

Classification report

	Precision	Recall	F1-score	support
0	0.78	0.84	0.81	23670
1	0.17	0.12	0.14	6330
accuracy			0.69	30000
Macro avg	0.48	0.48	0.48	30000
Weighted avg	0.65	0.69	0.67	30000

HC Complete

We have an accuracy of **21%**

Confusion matrix:

Actual \ Predicted	Predicted 0	Predicted 1
Actual 0	1166	1136
Actual 1	22504	5194

Classification report

	Precision	Recall	F1-score	support
0	0.51	0.05	0.09	23670
1	0.19	0.82	0.31	6330
accuracy			0.21	30000
Macro avg	0.35	0.43	0.20	30000
Weighted avg	0.44	0.21	0.14	30000

K-means

Is a method of vector quantization that aims to partition n observations into k clusters' each observation belongs to the cluster with the nearest mean.

Since we started this paper in mind with predicting whether it will end in an Arrest or not, we already know that we want to split the data into 2 clusters. Thus, we can do so directly and can compare the results with the baseline and supervised results.

Confusion matrix:

Actual \ Predicted	Predicted 0	Predicted 1
Actual 0	15154	8516
Actual 1	2587	3743

Classification report

	Precision	Recall	F1-score	support
0	0.85	0.64	0.73	23670
1	0.31	0.59	0.40	6330
accuracy			0.63	30000
Macro avg	0.58	0.62	0.57	30000
Weighted avg	0.74	0.63	0.66	30000

The resulting accuracy is **63%**. We can, however, see that the precision is quite unbalanced when it comes to 0 and 1. This is most likely due to the fact that the label values distribution is imbalanced.

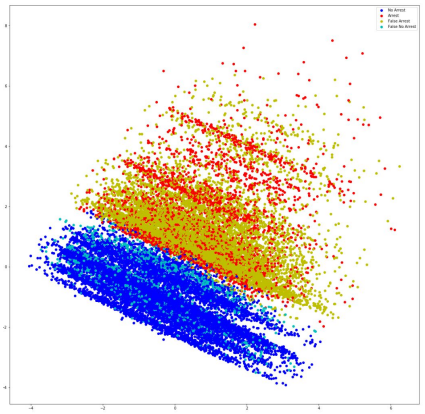
Final report (supervised and unsupervised)

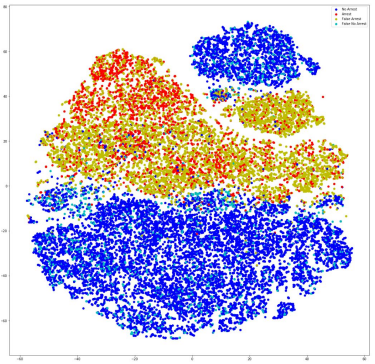
	Base	KNN	RF	HC ward	HC Complete	K Means
Accuracy	79%	85%	**89%**	69%	21%	63%
Precision (Macro avg)	40%	83%	**89%**	48%	35%	58%
Precision (Weighted avg)	63%	84%	**89%**	65%	44%	74%
Recall (Macro avg)	50%	66%	**76%**	48%	43%	62%
Recall (Weighted avg)	79%	85%	**89%**	69%	21%	63%
F1-score (Macro avg)	44%	70%	**80%**	48%	20%	57%
F1-score (Weighted avg)	70%	82%	**88%**	67%	41%	66%

PCA 2D

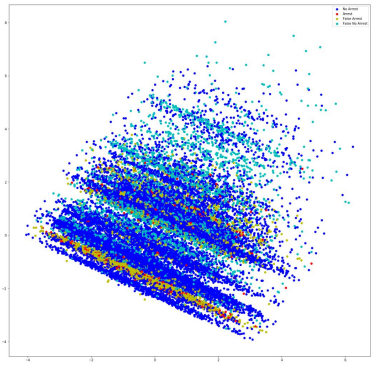
t-SNE 2D

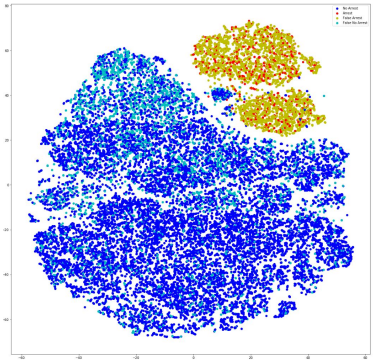
K Means



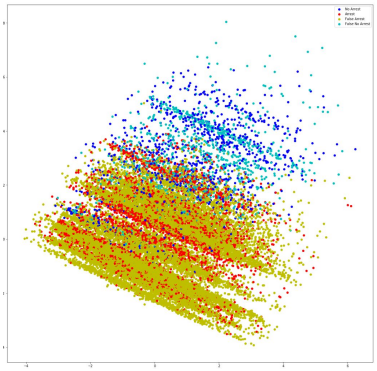


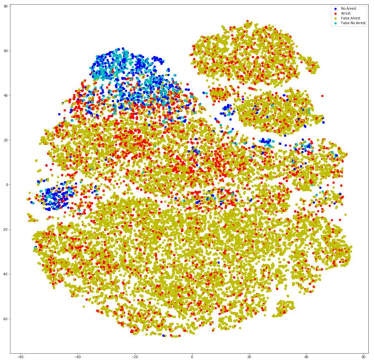
HC ward

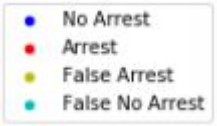




HC Complete



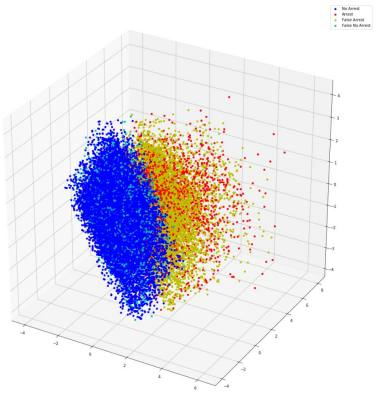


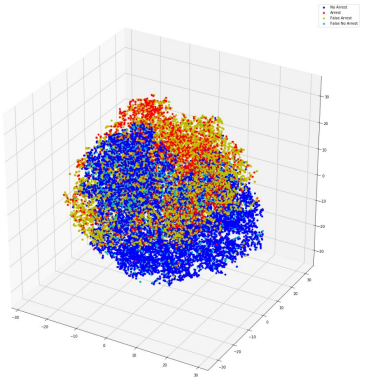


PCA 3D

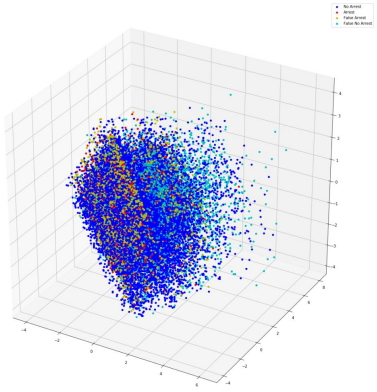
t-SNE 3D

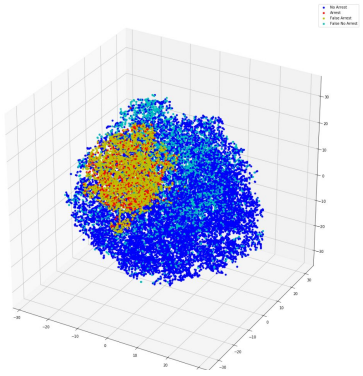
K Means



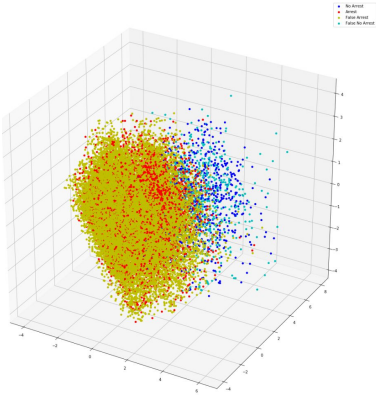


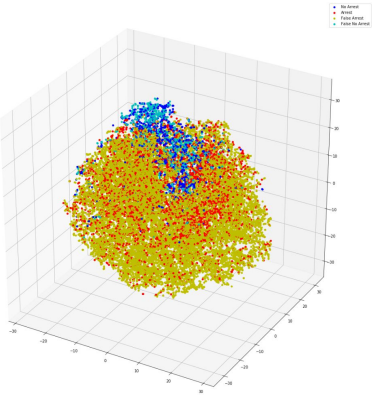
HC ward





HC Complete





Conclusion

In this report we've compared 4 algorithms, 2 supervised (KNN and RF) and 2 unsupervised (K Means and HC Agglomerative) between themselves and with a reference baseline model (DummyClassifier with most_frequent), trying to predict on the Chicago Police Crime data. whether an incident will occur in an arrest or not. For the supervised algorithms we used an exhaustive hyperparameter search using GridSearchCV. The exhaustive hyperparameter tuning yielded good results, considerably increasing our accuracy.
For unsupervised learning, we've used both PC and t-SNE to visualize the data in 2D and in 3D. In order to compare the unsupervised algorithms with the supervised algorithms as well, we went for 2 clusters and optimized the parameters as best as possible, comparing HC Ward, HC Complete and K Means, looking at W(C) and B(C) as we did. It is clear that our approach did not yield good unsupervised results, having less than the baseline model, but with decent results overall. Overall we've obtained a good understanding of the provided data and the used methods.

References:

1. https://en.wikipedia.org/wiki/Chicago
2. https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2/
3. https://towardsdatascience.com/understanding-random-forest-58381e0602d2
4. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
5. https://en.wikipedia.org/wiki/Random_forest#Preliminaries:_decision_tree_learning
6. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5796274/
7. https://www.researchgate.net/profile/Andy_Liaw/publication/228451484_Classification_and_Regression_by_RandomForest/links/53fb24cc0cf20a45497047ab/Classification-and-Regression-by-RandomForest.pdf
8. https://scikit-learn.org/stable/modules/grid_search.html
9. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV
10. https://en.wikipedia.org/wiki/K-means_clustering