Machine Learning Engineer Nanodegree

Capstone Project

Omar Ahmed Kassem
August 17th, 2018

San Francisco Crime Classification

Project Overview

As in all cities, crime is a reality San Francisco: Everyone who lives in San Francisco seems to know someone whose car window has been smashed in, or whose bicycle was stolen within the past year or two. Even Prius' car batteries are apparently considered fair game by the city's diligent thieves. The challenge we tackle today involves attempting to guess the class of a crime committed within the city, given the time and location it took place. Such studies are representative of efforts by many police forces today: Using machine learning approaches, one can get an improved understanding of which crimes occur where and when in a city — this then allows for better, dynamic allocation of police resources.

Domain Background

Supervised learning is one of the most promising field in machine learning and used to achieve many predictions used nowadays even in business goals. Using machine learning techniques to predict the crimes area and category shows the power of technology in the field of safety. Supervised learning is used in many feild similar to this problem like Predicting Crime Using Time and Location Data in this paper or Weather Forecasting using the weather data of the past two days, which include the maximum temperature, minimum temperature, mean humidity, mean atmospheric pressure, and weather classification for each day in this paper. Also This problem was a challenge on Kaggle on this link and many papers provide solutions for this kind of problem like this one.

Problem Statement

This is a Multi-Class Classification problem given the time and location of the crime, We need to do the following :

1. Reformat the features (e.g Data,Day) into a computer readable format.
2. Decrease the number of training points using K-Mean cluster Algorithm to speed up the training and testing process.
3. Splitting the training data into trainig and validation set.
4. Applying KNN and MLP Algorithms and evaluate the result.
5. Using the best Algorithm, predict the categories of the testing set and submit on Kaggle.

Metrics

Models will be evaluated using the multi-class logarithmic loss. Each incident has been labeled with one true class. For each incident,the model will provide a set of predicted probabilities (one for every class). The

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} log(P_{ij})$$

formula is then, where N is the number of cases in the test set, M is the number of class labels, log is the natural logarithm, y_{ij} is 1 if observation i is in class j and 0 otherwise, and p_{ij} is the predicted probability that observation i belongs to class j.

Log loss, also called logistic regression loss or cross-entropy loss, has very useful property in that it penalizes heavily when the model makes emphatically incorrect labels. In case the model predicts 1 for the label but the truth is 0, you will end up with natural log of (0), which reaches negative infinity. This will greatly increase the log loss term. It works well for multi-class classification, (multinomial) logistic regression and neural networks, as well as in some variants of expectation-maximization, and can be used to evaluate the probability outputs `predict_proba` of a classifier instead of its discrete predictions.

Log Loss vs Accuracy
Accuracy is the count of predictions where your predicted value equals the actual value. Accuracy is not always a good indicator because of its yes or no nature.
Log Loss takes into account the uncertainty of your prediction based on how much it varies from the actual label. This gives us a more nuanced view into the performance of our model.

II. Analysis

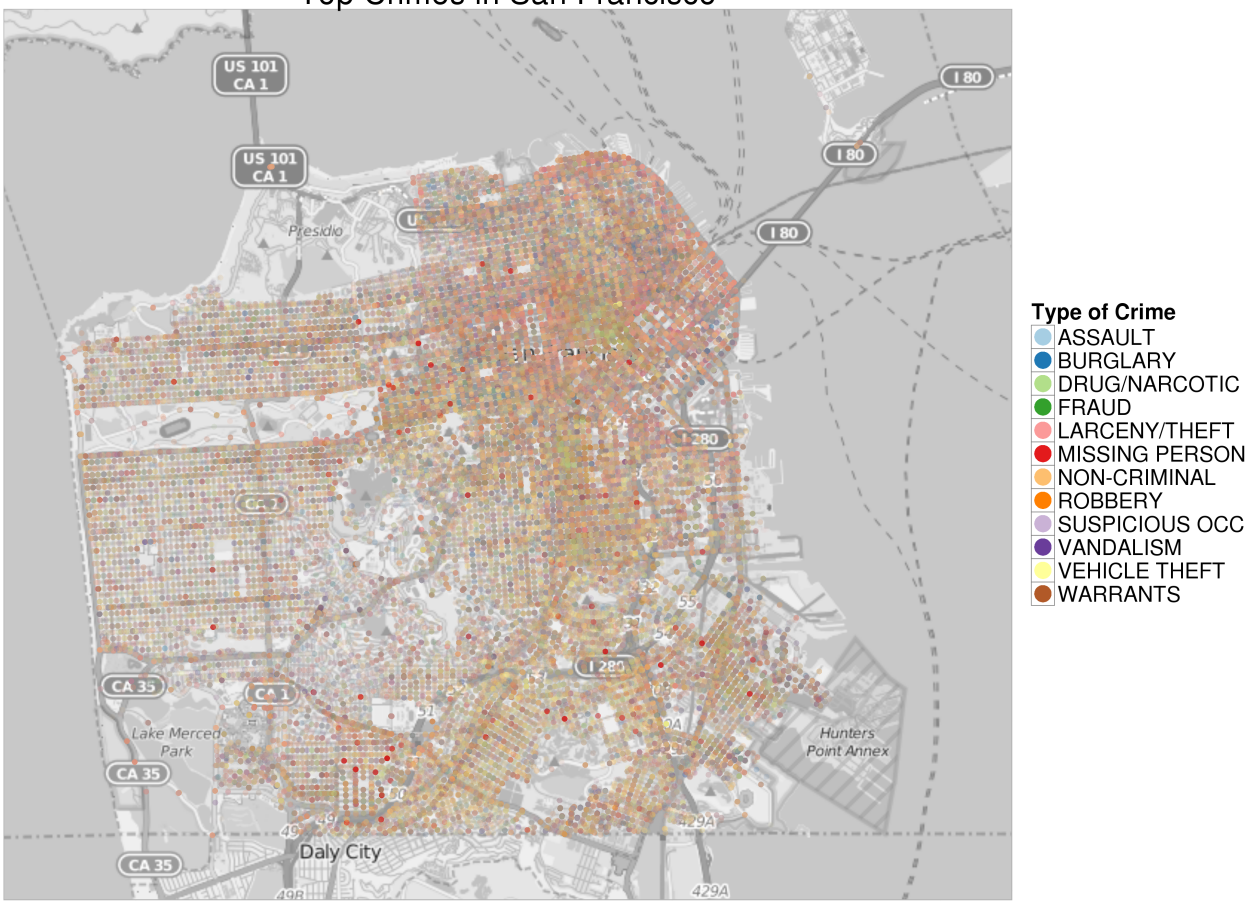
Data Exploration

To aid in the SF Crime Classification, Kaggle has provided about 12 years of crime reports from all over the city — a data set that is pretty interesting to comb through. We can find the dataset in this link. The dataset contains incidents derived from SFPD Crime Incident Reporting system. The data ranges from 1/1/2003 to 5/13/2015. The training set and test set rotate every week, meaning week 1,3,5,7... belong to test set, week 2,4,6,8 belong to training set. Each record contains the following information:

- Dates: a timestamp of the moment that the crime occurred. It is in the following format: Y-m-d H:i:s. E.g.: 2015-05-13 23:53:00
- Category: the category of the crime (the target, it has 39 unique class). E.g.: WARRANTS
- Descript: a short description of the crime. E.g.: WARRANT ARREST
- DayOfWeek: the day of the week in which the crime occurred. E.g.: Wednesday
- PdDistrict: the district of the city where the crime was committed. E.g.: NORTHERN
- Resolution: short description of the crime resolution. E.g.: “ARREST, BOOKED”
- Address: the address where the crime was located. E.g.: OAK ST / LAGUNA ST
- X: latitude of the crime position. E.g.: -122.425891675136
- Y: longitude of the crime position. E.g.: 37.7745985956747

If we draw different graphics we will see more clearly how the data is distributed.

Top Crimes in San Francisco



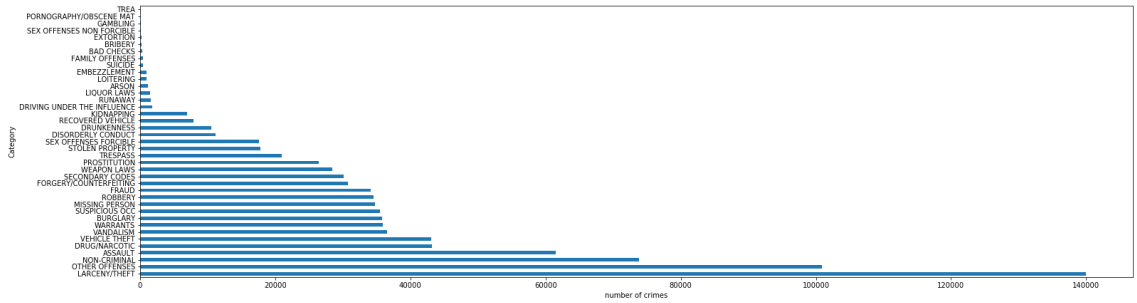
As it can be seen in the Figure, crimes are distributed all around the city, having a hot-spot in the upper right corner.

The dataset contains 39 unique category which are : WARRANTS, OTHER OFFENSES, LARCENY/THEFT, VEHICLE THEFT, VANDALISM, NON-CRIMINAL, ROBBERY, ASSAULT, WEAPON LAWS, BURGLARY, SUSPICIOUS OCC, DRUNKENNESS, FORGERY/COUNTERFEITING, DRUG/NARCOTIC, STOLEN PROPERTY, SECONDARY CODES, TRESPASS, MISSING PERSON, FRAUD, KIDNAPPING, RUNAWAY, DRIVING UNDER THE INFLUENCE, SEX OFFENSES FORCIBLE, PROSTITUTION, DISORDERLY CONDUCT, ARSON, FAMILY OFFENSES, LIQUOR LAWS, BRIBERY, EMBEZZLEMENT, SUICIDE, LOITERING, SEX OFFENSES NON FORCIBLE, EXTORTION, GAMBLING, BAD CHECKS, TREA, RECOVERED VEHICLE, PORNOGRAPHY/OBSCENE MAT.

Exploratory Visualization

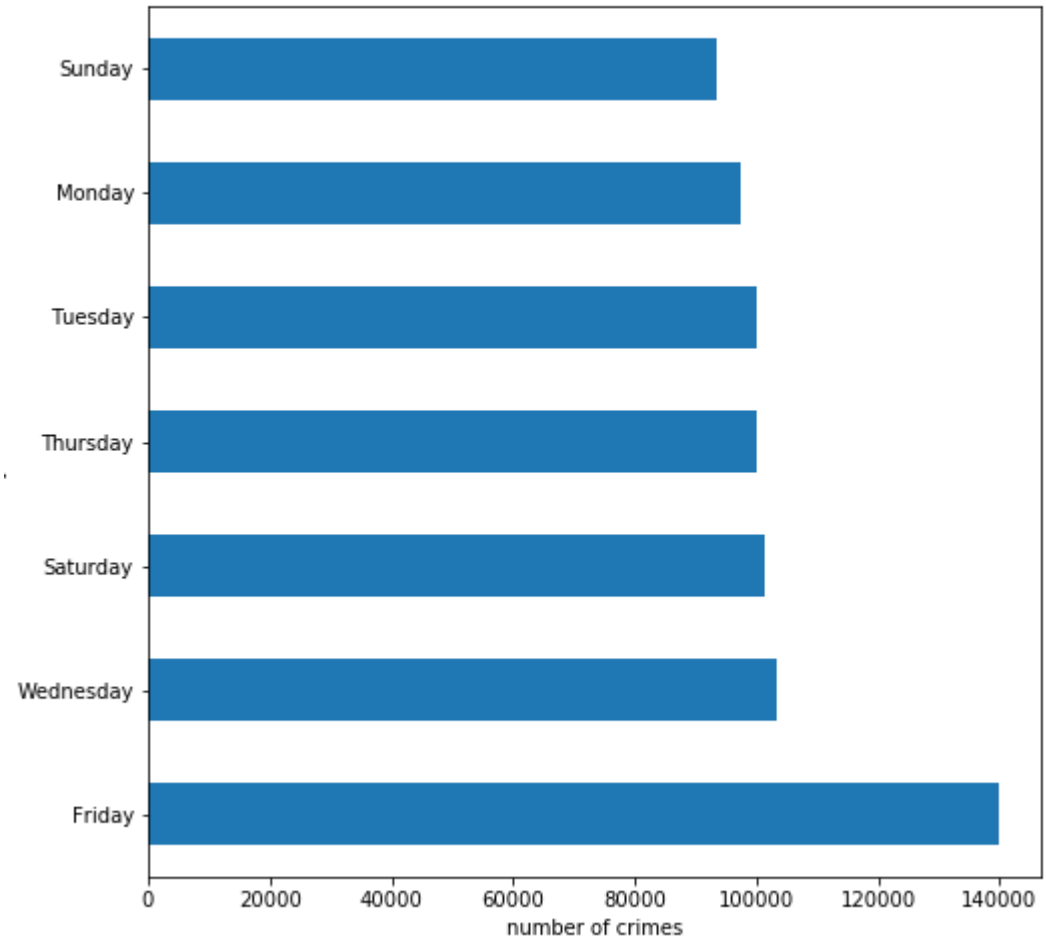
In this section we will explore the relation of some features and the target (category of crime):

- First we will show the number of occurrence of each Category :



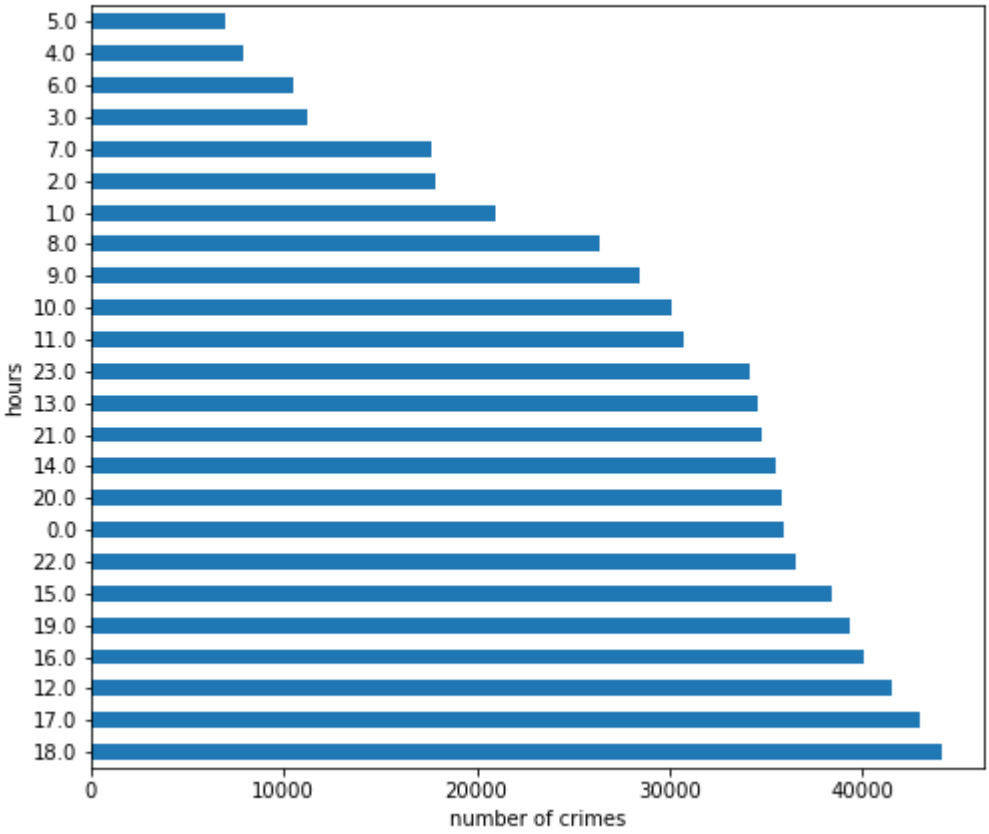
We could see that there are 39 different categories and the above plot shows that crimes are not equally distributed among categories but there is a huge difference between them.

- Second the relation between Categories and the day of week :



From the plot we could see that crime also distributed over days and Friday is the most selected day by criminals, while the rest of the week is pretty equally distributed.

- Third the relation between Categories and the hour of day:



From the plot we could see that the majority of crimes were committed between midday and midnight (including both).

In this section we will explore the abnormalities in the dataset which are the location information (logitude and latitude of each crime):

```
Longitudes summary:
count    878049.000000
mean       -122.422616
std           0.030354
min        -122.513642
25%        -122.432952
50%        -122.416420
75%        -122.406959
max        -120.500000
Name: X, dtype: float64

Latitudes summary:
count    878049.000000
mean         37.771020
std           0.456893
min          37.707879
25%          37.752427
50%          37.775421
75%          37.784369
max          90.000000
Name: Y, dtype: float64
```

We could see that longitudes values are between [-122.52, -120.5] and latitudes values are between [37.708, 90]. Here we can see bad values and thats because san fransisco latitudes are between [37.707, 37.83269]and longitudes are between [-122.517652, -122.3275] , so we can say that there are some abnormalities in the dataset. Now, to demonstrate the locations, let's plot them using scatter plot

Algorithms and Techniques

Techniques used at data preprocessing are `Label Encoding` and `K-Mean`:

- Label Encoding: is a utility class to help normalize Categories such that they contain only values between 0 and number 0f categories minus 1.
- K-Mean clustering: is a type of unsupervised learning, which is used when we have unlabeled data (i.e., data without defined categories or groups) or to reduce the size of data as in our case. The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity. The results of the K-means clustering algorithm are:

    - The centroids of the K clusters, which can be used to label new data, and these centroids (clusters' centres) act as the representatives of all the samples in their group.
    - Labels for the training data (each data point is assigned to a single cluster)

Cross Validation Algorithm used is `K-fold`:

- K-fold cross-validation: is a model validation technique where the training set is randomly partitioned into k equal sized subsamples. A single subsample is used as testing data for, and the remaining k − 1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the testing data. The k results from the folds are then averaged to produce a single estimation. We're ensured that we're using the entire dataset for training and validation, and that we're not overoptimizing to perform well towards the single validation set that was selected.

supervised learning Algorithms used are `KNN` and `MLP`:

- K-Nearest Neighbours (KNeighborsClassifier) : The k-nearest neighbour classification rule is the simplest and most intuitively appealing nonparametric classifier. It assigns a sample z to class X if the majority (i.e. more than 1⁄2) of the k values in the training dataset that are near z are from X, otherwise it assigns it to class Y. For this Model the parameter which is tunned is the number of neighbours, A very large number will guarantee more accuracy, however, computational efficiency would decay. A very small number will make the execution very fast, but won't provide accurate results. So after testing the model with many values the best choice is 600 neighbours.
- Multi-layer Perceptron (MLPClassifier) : it is a supervised learning algorithm that learns a function by training on a dataset. It trains iteratively since at each time step the partial derivatives of the loss function with respect to the model parameters are computed to update the parameters. It can also have a regularization term added to the loss function that shrinks model parameters to prevent overfitting. Given a set of features X = {x_1, x_2, ..., x_m} and a target y, it can learn a non-linear function approximator for either classification or regression. The number of hidden units can also be specified, large numbers produce long training times. In the output layer, the sigmoid function is used for classification. If the approximate sigmoid is specified for the hidden layers, it is also used for the output layer. The advantages of MLP are Capability to learn non-linear models and Capability to learn models in real-time (on-line learning).

Benchmark

There are 3 types of results that the model can be compared to:

- comparing with the best kernels solving this problem in Kaggle competition for example:
    - careyai (score = 2.42381)
    - EDA and classification (score = 2.56180)
- Test the model with it self using several algorithms and comparing them with each other.
- Kaggle scores: Comparing the results with Kaggle public scores which is a great indicator for how our model performe.

III. Methodology

Data Preprocessing

Relevant information : The following information about each record is provided: Dates, Category, Descript, DayOfWeek, PdDistrict, Resolution, Address, X and Y. In new data samples, of course, Category will not be there (as this is what it has to be inferred).

- Selection 1: X, Y (latitude and longitude)
  A priori, it might seem that the location of a crime might be enough to distinguish from different types of them, as, probably, same types of crime are committed in the same zone. However, as it has been seen when analysing the dataset, pretty much all categories have the same “hotspot” location. So a second selection including more information is necessary.
- Selection 2: X, Y, Dates
  Bearing in mind the analysis, it can be seen that another distinguishing property of the crimes is the time when they happened. So, including this Dates feature (which also includes the time) and formatting it correctly, could be enough to discern between crimes.
- Selection 3: X, Y, Dates, DayOfWeek
  The last feature combination would be also including the day of the week when the crime was committed. As it has been seen in the analysis, samples differ in this information, so it can be useful. It will also have to be converted to a different format.

This last feature extraction seems logic and, a priori, it seems that it would probably work.

Data Transformation : To use the dataSet in classification, we must reformate it to numbers instead of strings to be used in classification algorithms.

- Dates : the current format is : yyyy-mm-dd hh:mm:ss (not useful),date part is not useful and will be erased and the time will be converted to Cartesian coordinates with equation TimeX = cos(φ)time , TimeY = sin(φ)time where φ = (2*π) / 24 and time = (hours + (minutes / 60) + (seconds / 3600))
- DayOfWeek: it will be treated in the same way as Dates with equation dx = cos(φdayNumber), dy = sin(φdayNumber) where φ = (2*π) / 7 and dayNumber is 0 for Monday, 1 for Tuesday and so on.
- Category: it will be converted to a number in range (0,39) using sklearn.preprocessing
- PdDistrict and Address: will be removed from features because they are related to X,Y (they could be generated from these features).
- Resolution and Descript: will be removed because they are both a description for Category which is the target.
- X and Y (longitudes and latitudes): those features have outliers and Those outliers should be removed.

Data Clustering : The initial training database had nearly 900.000 records, which is pretty high. So, it is required to reduce the training dataset until it has a manageable size. Deleting all the “extra” samples that are not wanted would end up with a messed up database that wouldn’t have the same elements per category proportion than the original one. To avoid this happening, the clustering algorithm used to reduce the database size will be applied for each category. This way, it is ensured that the resulting set will maintain the same proportions from the initial one. A clustering algorithm groups elements in a specified number of clusters. This clusters’ centres are calculated and they act as the representatives of all the samples in their group. Because not all clusters will have the same number of elements, some will be more relevant than others. Their relevance is measured by their weight, which is the number of elements they contain. The algorithm used for this purpose has been K-means. K-means clustering proceeds by selecting k initial cluster centres and then, iteratively refining them as follows:

- Each instance d is assigned to its closest cluster centre.

- Each cluster centre C is updated to be the mean of its constituent instances.

The algorithm which has been applied for each category in the following way:

1. Getting all the elements of a category
2. Obtaining the number of centroids necessary to reduce the database to the number of elements that we want, bearing in mind that some classes might have very few members.
3. Applying K-means for this category.
4. Getting the number of samples per cluster.
5. Adding the resulting cluster centres with the number of elements they represent to the reduced database.
6. Following up with the next category.

Data Splitting : Now all categorical variables have been converted into numerical features, we will now split the data (both features and their labels) into training and test sets. 80% of the data will be used for training and 20% for testing. we have to use the reduced data set to decrease the number of training data and decrease the time of training and predicting.

Implementation

Creating a Training and Predicting Pipeline `train_predict(learner, X_train, y_train, X_test, y_test):` where the inputs are:

- learner: the learning algorithm to be trained and predicted on
- X_train: features training set
- y_train: crime category training set
- X_test: features testing set
- y_test: crime category testing set

It uses the leaner to fit the training points then predict the testing points and Because the results must be submitted (in the competition) giving a probability of each sample to belong to all categories, and not just the predicted category the function `pred_proba()` is used in each model along with `pred()`. Also this method will save all time taken for every step.

Evaluating all Algorithms and finding the best one `evaluate(learners, X_train, y_train, X_test, y_test)` where inputs are:

- learners: the learning algorithms to be evaluated
- X_train: features training set
- y_train: crime category training set
- X_test: features testing set
- y_test: crime category testing set

It uses the `train_predict(learner, X_train, y_train, X_test, y_test):` to evaluate all learners. The models used for evaulations are :

- `KNeighborsClassifier(n_neighbors=600,n_jobs=-1, weights='distance')` as we said before 600 neighbors was the best choice for Knn.
- `learner_MLP = MLPClassifier(learning_rate='invscaling', shuffle=True)` where learning_rate='invscaling' is used to make the learning rate speed up with time.

Results visualization `vs_evaluate(results)` where inputs are :

- results: those are results generate by `evaluate(learners, X_train, y_train, X_test, y_test)` method using the selected models.

It display results of various learners in plots to compare time and accuracy of different models.

Complications The most complicated parts were :

- Clustering the data took too much time to finish even on kaggle kernel and formating the data to fit in the algorithm took too much time.
- Choosing the best Algorithm was challenging especially choosing the best number of neighbors for KNN algorithm as the algorithm took alot of time and memory on predicting the testing set.

Refinement

MLPClassifier (neural net) model gives better results than KNN (k-nearest neighbors) in form of the F-score and Accuracy score. So we will tune its parameters using `GridSearchCV`. The parameter to be tunned are :

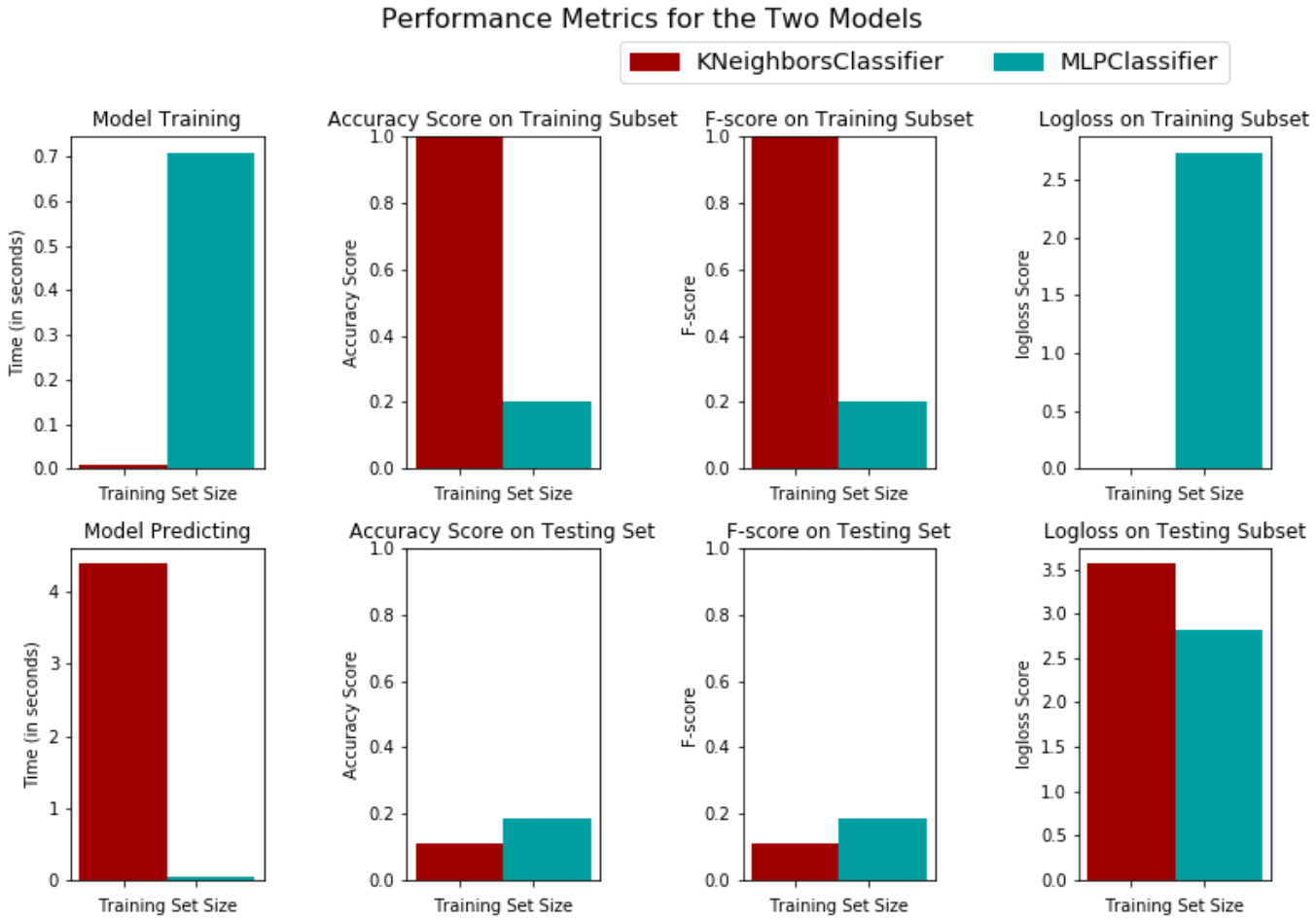
- activation : 'relu', 'logistic', 'tanh'
- solver : 'sgd',
- learning_rate : 'adaptive'
- alpha : from 0.1 to 0.00001
- learning_rate_init : 0.001
- max_iter : 1200, 1300, 1400, 1500

Tunnig those parameter make the logloss score (for the full testingset on Kaggle) improved from 2.71985 to 2.68353 and the best set of paremeters are :

- activation = 'relu'
- alpha = 0.0108
- learning_rate = 'adaptive'
- learning_rate_init = 0.001
- max_iter = 1200
- solver = 'sgd'

IV. Results

Model Evaluation and Validation

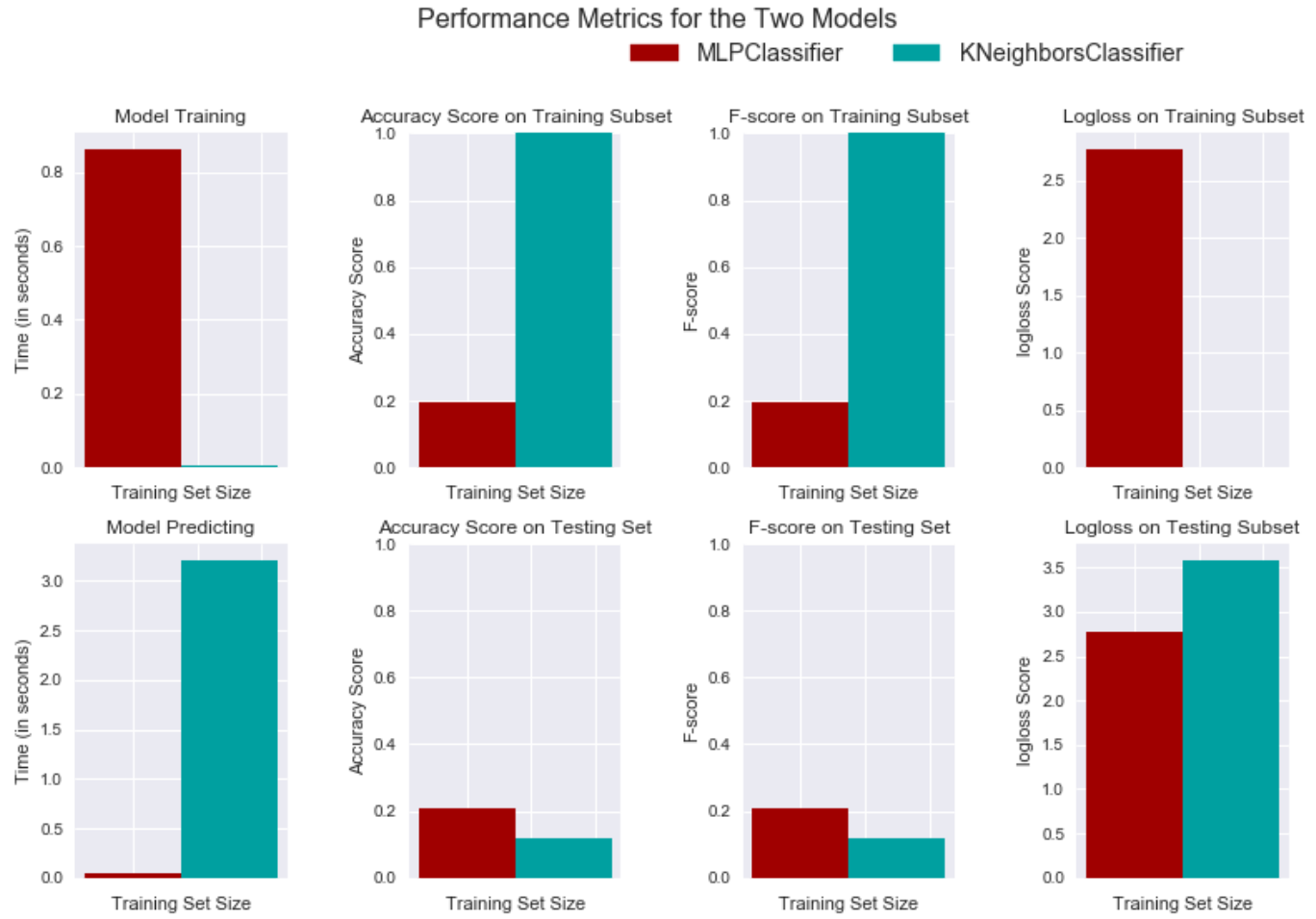


As we can see in the plot that each model is better than the other in some metrics for example

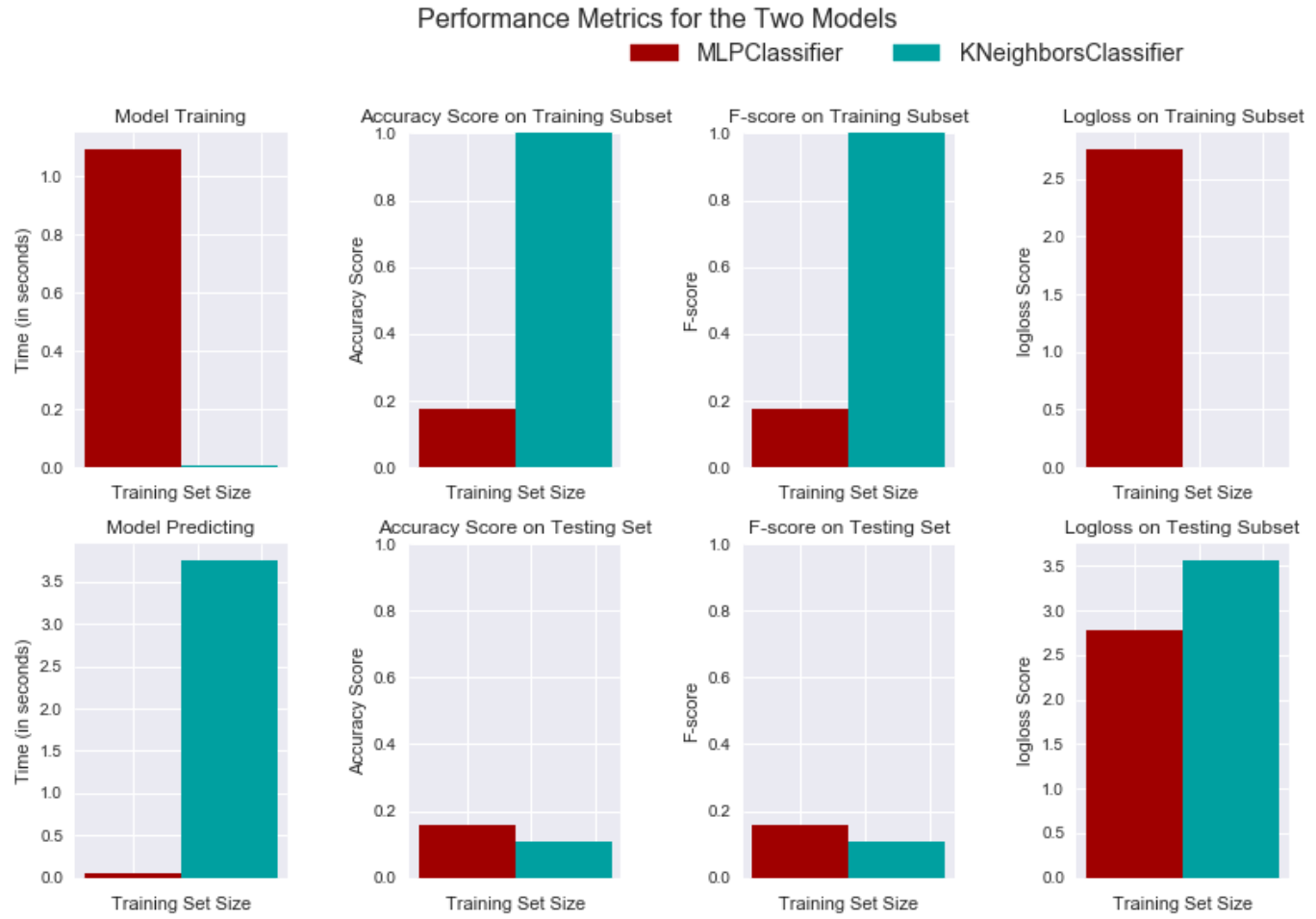
- K-Nearest Neighbours :
    - takes less time in training.
    - gives high score in prediction on training sets.
- MLPClassifier :
    - takes less time in testing.
    - gives higher score in Accuracy and F-beta score.
    - gives better (lower) logloss score

We also use K-Fold Cross Validation to validate or models and those are the resulted plots when K = 4 (spliting the data 4 time as explained above)

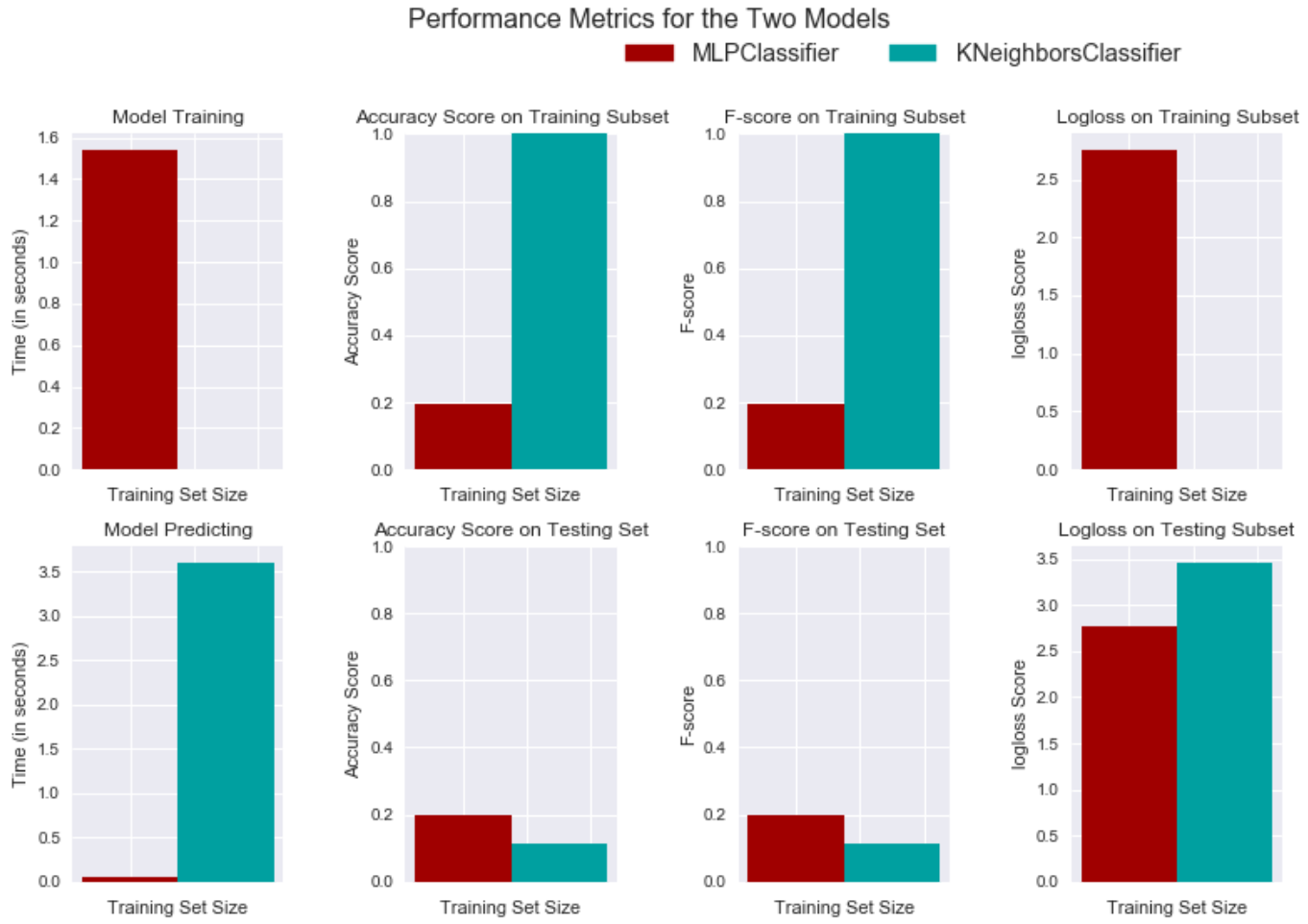
Split number 1



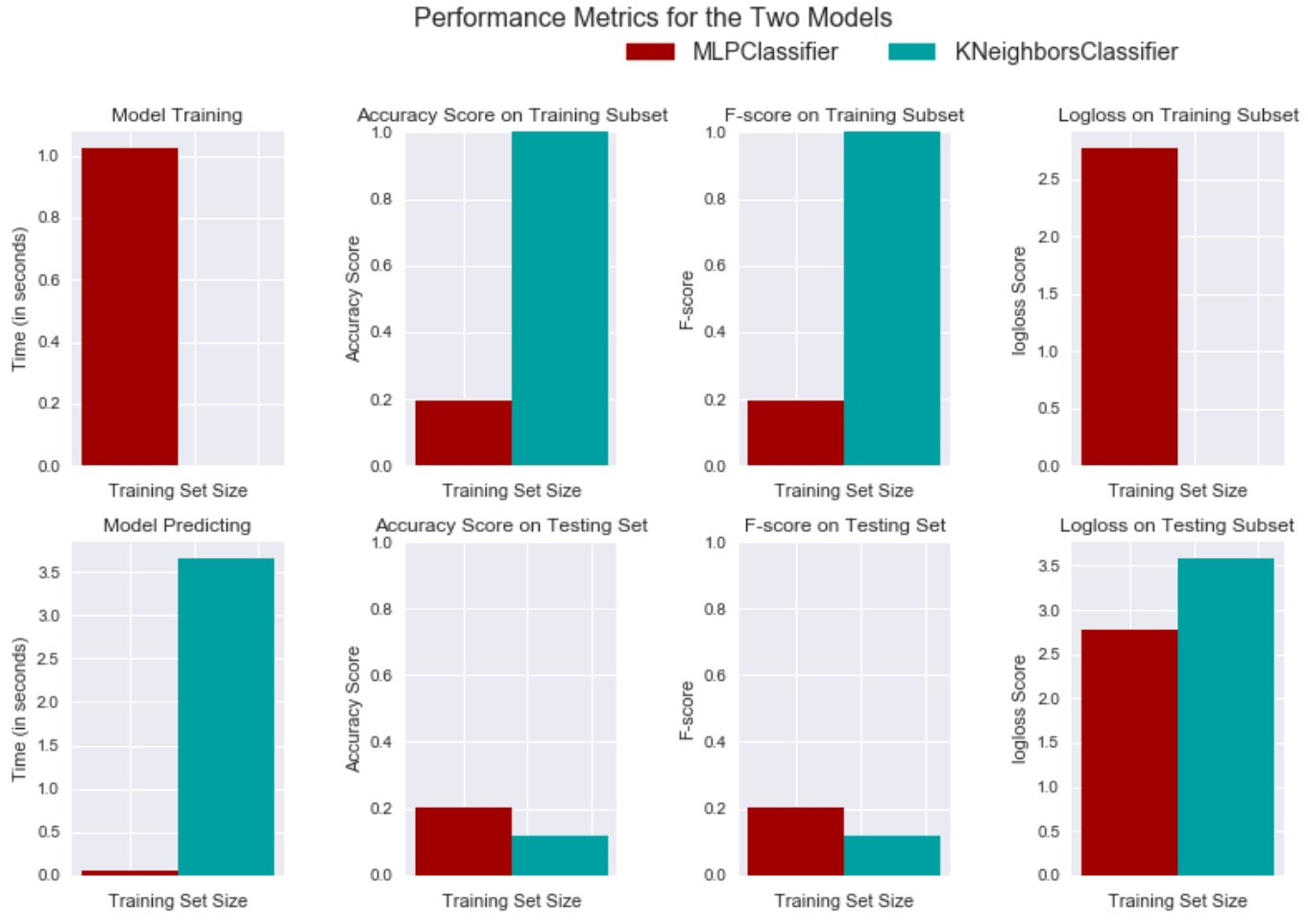
Split number 2



Split number 3



Split number 4



As we can see that the models keep their performance as KNN keeps giving high score in predicting the training sets and MLP keeps giving higher score in Accuracy and logloss scores.

Justification

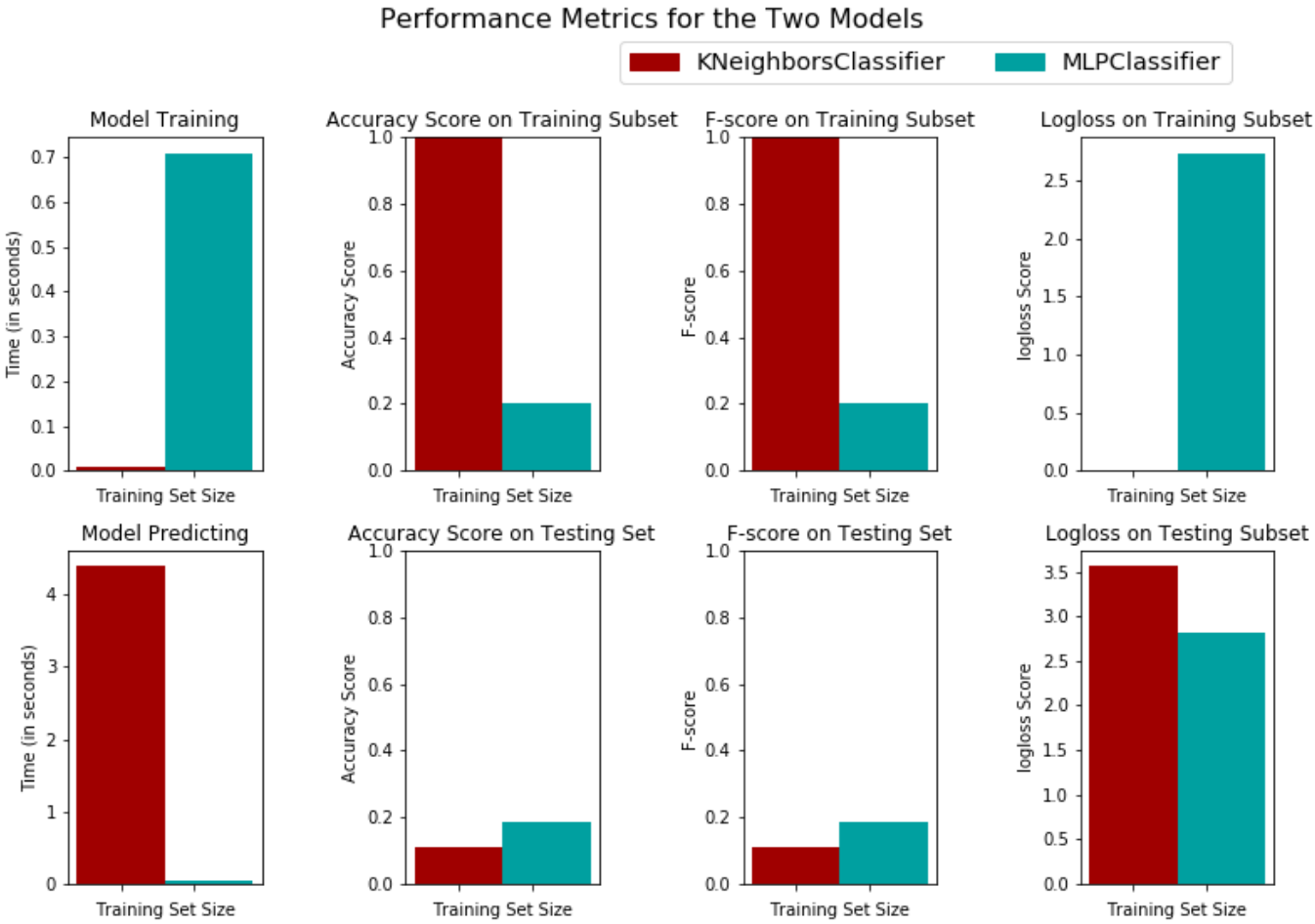
- K-Nearest Neighbours : As explained before, several tests using different number of nearst neighbours were performed with this algorithm, it was seen that the optimal k = 600 (number of neighbours to check), and the best results obtained by this model was a Logloss of 2.68045.
- MLPClassifier : Using the best model generated from the `GridSearchCV`, it gives a score = 2.68353

Those scores are very close to the Benchmark scores which are 2.42381, 2.56180.

V. Conclusion

Free-Form Visualization

The good quality of the project is represented in comparing various model with each other and observing the results. For example the following plot demonstrates the differences between MLPClassifier and KNneighbors. The plot shows that the best for predicting training data is KNN but they do very bad at testing data. On the other hand, MLP takes more time on training but gives good results on both training and testing data.



Reflection

Machine Learning is a really powerful field when it comes to AI and, if a model is done right, the level of accuracy that some algorithms can achieve can be astonishing. Certainly, the present and future of intelligent

systems goes through ML and big data analysis.

With the crime classification problem, it has been seen that the most accurate algorithm was the K-Nearest Neighbours. Although it is true that MLPClassifier was better than KNN in some metric but KNN gives me better score on Kaggle. This crime classification problem might have some utility in the academic field, however it might not very useful in real life (or at least, it is difficult to see a practical application of it).

A possible extension of this work would be to create a crime prediction system that would anticipate which zones of the city might be a crime hotspot on certain dates. This way, police could be warned and they could, for example, double the surveillance on this zone. Designing a Machine Learning algorithm (probably a Neural Network) to identify patterns among the data (unsupervised learning) would be the starting point of this future work.

There exist some difficulties which were represented in visualizing huge amount of data, Clustering and training this huge amount of data using heavy models such as K-Mean, KNN and MLP which lead to crash the device several times and also stop working on Kaggle kernel due to exceeding run time.

The interesting part of the project was inspite of the fact that KNN is simpler than MLPClassifier but it gives a better score in a very simple parameters configuration but I think that MLPClassifier could gives better result by increasing the number of iteration and number of hidden layers which require a high computation power.

Summarizing the entire problem solution

1. The dataset has been transformed to be used in training and testing
2. The dataset is clusterd unsing K-Means to speed up the training and testing process
3. Different Algorithms have been implemented and trained on the data.
4. Algorithms are evaluated and the best one is used to predict the given testingSet.
5. Results have been submitted on Kaggle Competition to check score.

Improvement

Of-course there exist better solutions for the problem (for example the benchmark models). The part that I think should be has more improvement is the implementation of Knn, predicting different setup of Knn models tooks alot of time and memory even on Kaggle kernel so it was hard to tune the model to get better results and I think that tunnnig this model parameters could give better results. Also I think I had to try more models as SVC,decision tree,XGB,... but the reason why I didn't use one of them that I found many people try to solve this problem with those models to I'm trying to make something new and challenging.

References

- https://www.kaggle.com/benhamner/san-francisco-top-crimes-map
- https://www.kaggle.com/c/sf-crime/
- https://www.kaggle.com/rudikruger/liblinear/code
- https://www.kaggle.com/vivekyadav/sfo-rmd-kaggle
- https://www.kaggle.com/hatone/mlpclassifier-with-gridsearchcv
- https://www.datascience.com/blog/k-means-clustering
- https://upcommons.upc.edu/handle/2117/96580
- visualization code from finding_donors project in this link https://github.com/omarahmed10/machine-learning/tree/master/projects/finding_donors

- https://datawookie.netlify.com/blog/2015/12/making-sense-of-logarithmic-loss/
- https://www.quora.com/What-is-log-loss-in-Kaggle-competitions

---