Predicting and Classifying Crime Data

Ramya Kondapi[1] Yesaswini Valavala[2] Yaswanth Santhanam Raghavan[3]

rkondapi@knights.ucf.edu yasaswinivalavala@knight.ucf.edu yaswanthsr@knights.ucf.edu

1,2,3MS Computer Science, College of Engineering and Computer Science, University of Central Florida, 32816

***Abstract*: This work focuses on analysing the San Francisco Crime classification data available in Kaggle. Considering the number of increasing crimes in this city, this project focuses on analysing the insights into the crime location, and the day of the week when most of the crimes occur. Machine learning algorithms like Naïve Bayes, K-Nearest Neighbour, Linear Regression, Decision Tres, Gradient Boosting, logistic Regression and Random Forest are applied on this dataset which contains around 1.7 million samples. The results were obtained by applying the algorithms but varying the number of districts included in order to calculate the accuracy.**

Key words: Naïve Bayes, K-Nearest Neighbour, Linear Regression, Decision Tres, Gradient Boosting, logistic Regression, Random Forest, San Francisco, classification, samples, San Francisco Crime

I. INTRODUCTION

San Francisco, a city which is now currently famous for holding various booming Companies in this world. But this city also had a high record for housing some of most notorious criminals during the time span of 1934 to 1963. San Francisco Police Department does ensure the safety of the people in this city, but some crimes do tend to happen even after stringent safety measures. A dataset available from Kaggle contains the records of the crimes that happened in this City ranging from 2003 to 2015 and it contains around 1.7 million samples of data. Therefore, this project focuses on analysing the insights into the crime location, and the day of the week when most of the crimes occur. Machine learning algorithms like Naïve Bayes, K-Nearest Neighbour, Linear Regression, Decision Tres, Gradient Boosting, logistic Regression and Random Forest are applied on this dataset which contains around 1.7 million samples. Dimensionality reduction is performed in order to better understand this dataset and classify them in order to obtain a good classification accuracy. The results were obtained by applying the algorithms but varying the number of districts included in order to calculate the accuracy.

II. PROBLEM STATEMENT AND GOALS

Insight of Crime Locations: Designing a classification method in order to classify the various districts of San Francisco city based on the number of crimes occurring in each district.

Comparing the maximum crime rates in a year: Analysing the maximum crime rate pattern in San Francisco based on the obtained crime location results using ML algorithms.

Understand the trend of most frequently occurring crimes: Comparing the occurrences of most frequently occurred crimes from 2003 to 2015 across various districts in this city.

III. METHODOLOGY AND TECHNICAL DETAILS OF THE APPROACH

The first stage starts with preprocessing the data where all the irrelevant features are omitted, also new features are introduced to the dataset required for evaluation. In this stage, all the categorical data are converted to numerical data through a method called label encoding. In this type of encoding, all the unique values in a column are considered and assigned a unique integer value to each of the unique values. For instance, consider a column with unique values [‘A’, ‘B’, ‘C’], each of these categorical values are assigned an integer values: A→0, B→1, C→2. This can be done using a predefined method LabelEncoder() from the sklearn library. After preprocessing the, the data is split into train and test data required for evaluation and sent to each of the models below for predicting and classifying the crime data.

A. Naïve Bayes Classifier

In classification, for predicting the probability of every class we choose Naive Bayes classifier as it is a probabilistic machine learning. It mainly relies on Bayes theorem i.e.,

$$P\,(A/B) = \frac{P(B/A)\,P(A)}{P(B)}$$

From the above equation we predict that probability value of A given that B has already been occurred. We can conclude that A is hypothesis and B is the evidence. Assuming that predicting values are independent.

Naive Bayes algorithms are widely used in real world applications like sentiment analysis, spam filtering, recommendation systems etc. They are easy and simple to implement But, they don’t produce efficient results when the predicting values are dependent as always, the predicting values are independent in this scenario.

B. Linear regression

It is a supervised learning model. Which is used in predicting the value of dependent variable (b) from the independent variable(a). Therefore, it finds the linear relation among the input and output values i.e., between a and b respectively.

$$b = \theta_1 + \theta_2.a$$

Here θ_1 is the intercept, and θ_2 is the coefficient of x. While the model is being trained, it predicts the value of a from the value b till it best fits the curve. On finding the best values for θ_1 and θ_2 , we achieve in getting the best regression curve that is we get the best fit line. And which in turn results in finding the a and b values.

Statistics method is used to estimate the coefficients. When it is given only with a single input then we go for simple linear regression model else having more than a single input then to find the result we use the ordinary least square approach.

C.KNN

KNN is supervised machine learning algorithm. In KNN, the data we provide for learning will be labelled and classified in prior. KNN algorithm is used in classification of the input data. For example, we can use KNN to provide recommendation to customers based on their interests, identify the objects that are similar to each other.

Consider we have data that is already classified. The data present will be divided into different clusters which will be similar to output obtained from K-Means which is unsupervised machine learning algorithm. The data will be provided to KNN as supervised data set. Consider the data in the data set has three different clusters i.e. data is divided into three different classes, when a new data point is given as input, we should be able to find out, to which cluster will the data point belong to. Initially we need to identify the number of nearest neighbours to the given input point by calculating the distance to its surrounding points. We will consider the ‘K’ value while calculating the distance of the input point to its neighbours. If the ‘k’ is 1 we calculate the distance to the nearest of all points and assign it to that cluster. If ‘k’ is 6 we need to calculate distance from the six nearest data points and if four of them are from cluster 1 and 2 from cluster 2, we assign the input point of cluster 1 as it has more nearest neighbours from cluster 1. This works same for all cases. In short, ‘k’ in K- Nearest Neighbour refers to number of neighbours that are needed to be taken into account for classifying the input point.

Choosing K- value: The K- value should be large enough to make sure that we will have very less influence from the outliers i.e which are data points that are very far from everything else. Also make sure that K- value is small enough when the sample size is very small such that it will not lose the influence. There is also an alternate approach to choose k- value. We need to calculate F1 score or accuracy for different values of ‘k’ on test data set and we need to plot the graph. We will observe a sudden drop at some value to ‘k’(i.e when ‘k’ is increasing accuracy gets dropped) and reaches to minimum accuracy. When you observe the graph, there will be point where the graph will drop and the point at which the graph is dropped is called as ‘elbow’ point. You need to choose ‘k’ which is close to the elbow point. That will be the optimal value for ‘k’. You can use K-fold cross validation to evaluate the accuracy for KNN.

Calculating distance between points: In order to calculate the distance between the given point to its surrounding neighbours, you can use any of the metrics below to calculate the distance.

a) Euclidean distance: Square root of sum of squared distance between two points. Consider points (x_1, y_1) and (x_2, y_2). The Euclidean distance between them will be

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

b) Manhattan distance: It is sum of absolute value values for difference between the points. Consider points (x1, y1) and (x2, y2). The Manhattan distance between them will be

$$|x_1 - x_2| + |y_1 - y_2|$$

c) Hamming distance: It is used to determine whether the two given variables fall into same category or not. Mostly used for categorical variables. If it falls under same category, then it will assign ‘0’ else it will assign ‘1’.

d) Minkowski distance: When we wanted to find out the similarity of distance in between two given points then we use Minkowski distance. Consider points (x_1, y_1) and (x_2, y_2). The Minkowski distance between them will be

$$(|x_1 - x_2|^p + |y_1 - y_2|^p)^{\frac{1}{p}}$$

D. Decision tree

It is a supervised learning algorithm. Which stands as perfect example for classifying. It is a simple representation which represents data as a continuous split. The decision tree consists of nodes, edges/branch and leaf nodes (terminal nodes). The two types of decision tree are:

a) Regression tree: there are used for continuous data

b) Classification tree: they are used for categorical or discrete variable. And they tree is built by dividing the data into partitions repeatedly and this process is known as binary recursive partitioning.

Classification and regression models are represented in tree format which have been built from the decision tree algorithm. Here the data set is divided further into smaller and smaller sets. And we obtain the result that is tree containing decision nodes and leaf nodes implying it to be classification or decision. The deeper the tree is the fitter the curve is and more complex decisions. Though, there is slight change in training data the results of decision tree are drastic. And also, the trees size being large it is not easy to predict the values. It is widely used in fields of Biomedical Engineering, Financial analysis, Astronomy, System control, manufacturing and production.

E. Logistic Regression

Logistic Regression is supervised machine learning algorithm which can be used in binary classification where we get the output as '0' and '1', It is mainly used to identify relationships between the dependent variables with respect to one or more independent variables in the corpus. It uses logistic function and estimate the probabilities to do so. We get the probabilities and those probabilities are mapped to 0's and 1's using a function called 'Sigmoid' function. The main aim of logistic regression is to classify the given input to set of classes present.

For given input function $h_{w,b}(x) = f(w_T x + b)$, the activation function is expressed as:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Here 'f' is activation function, 'w' is weight of neuron, 'b' is bias, 'h' is hidden input, 'f(x)' is sigmoid function.

F. Random Forest

It is supervised machine learning algorithm where already labelled data is used to train the model, for a given input it should be able to decide which label does the input belongs to. The outputs for Random Forest algorithm can be binary, numeric or any type of classifications. Random Forest uses decision trees i.e it is collection of decision trees. It helps the decision trees to be more accurate.

Consider we have a data set, now it picks up random sample from the data set which has different features in it. It tries to construct a decision tree by taking the subset of the features in random. The root node is decided based on the feature having closest co-relation with the label. In order to decide the following branch, it randomly picks other features from other data set avoiding the one's already picked and pick subset of features in random and checks the co-relation with label resulting in decision tree that has variance in it. It repeats the process multiple times depending up on how longer you want to train the data. Once the training is done, if an input data given, it runs through all the decision trees and results will be generated at the end of each tree. It checks for maximum alike results and return that as output (i.e it counts all the occurrences for different possibilities and return the maximum occurrence as result).

G. Gradient boosting

Gradient boosting is a process for converting weak learners whose performance will be little better compared to the performance obtained for random choice, into strong learners. It trains the models in any of additive, gradual or sequential way. Here we will obtain a new tree every time we perform this which is made to fit on the original data which is modified. AdaBoost is one of the gradient boosting algorithms. In AdaBoost it starts with training the decision tree where all the observation present are given equal weight. We then evaluate that tree and categorize the observations into the ones that are easy to classify and the ones which are difficult to classify. We then decrease the weights for the ones which are easy to classify and increase the weights of those which are difficult to classify. Based on the previous weights the second tree is build which helps to better the predictions compared to first time. The summation of results from previous trees yields new model. Now using the previous two trees data, we need to calculate classification error and build a new tree to make predictions for the revised values. Repeat this process for some specific iterations. The results obtained from the final tree will be considered as it will be optimal (i.e it is nothing but sum of all the weights obtained from previous trees).

AdaBoost differs from gradient boosting in only how both algorithms work on identifying the drawbacks weak learners. In AdaBoost, it uses weights will be higher for the ones that are difficult to classify whereas in Gradient Boosting using the gradient which is obtained from the loss function

$$y = ax + b + e$$

Here, ‘e’ is the term used for error. Loss function helps us to calculate deviation of the result from the true outcome which decide how good the coefficients fit for inherited data. This helps us in optimising the model.

The main advantage of gradient boosting is that it helps us in optimising the cost function rather the value which is specified by the user because it is not good in corresponding with real world applications.

IV. DATA DESCRIPTION

‘San Francisco Crime Classification’ is used as the data set for our model. Kaggle was the source from where the data set has been taken. The data set contains 1.7 million samples of data in it. From which the available data used for training is 8.7 lakh samples (i.e 878049 to be precise) and 8.8 lakh samples for testing the model (i.e 884262 to be precise). There are totally 9 attributes in the training data and 7 attributes in the testing data. It comprises of 10 classes namely ‘CENTRAL’, ‘RICHMOND’, ‘INGLESIDE’, 'TENDERLOIN', 'BAYVIEW', 'TARAVAL', 'MISSION', 'SOUTHERN', 'NORTHERN', 'PARK’. The total number of attributes available in training and testing data are ‘Dates’, 'Category', 'Descript', 'DayOfWeek', 'PdDistrict’, 'Resolution', 'Address', 'X', 'Y’, 'Id', 'Dates', 'DayOfWeek', 'PdDistrict', 'Address', 'X', 'Y' in which we considered only ‘Coordinates’,’ Years’,’ DayOfWeek’,’PdDistrict’ and dropped "Resolution", "Descript", "Category", "Address" in training data and "Id", "Address" in testing data to obtain better accuracy.

V. CRITICAL EVALUATION AND COMPARISION

The evaluation metrics used for calculating the algorithms is accuracy. As it is best way to measure the quality of machine learning model. And it will be helpful in predicting how correct the label is in the given dataset.
On combing the classes in various combinations and performing the algorithms for each class combination we acquired the following results.

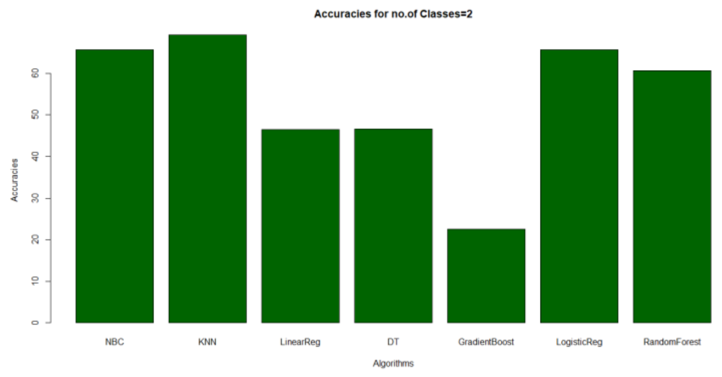


Fig.1: Accuracies of all algorithms considering 2 classes

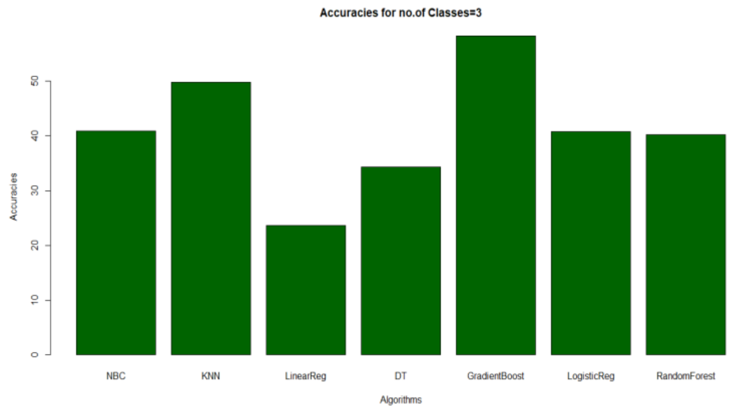


Fig.2: Accuracies of all algorithms considering 3 classes

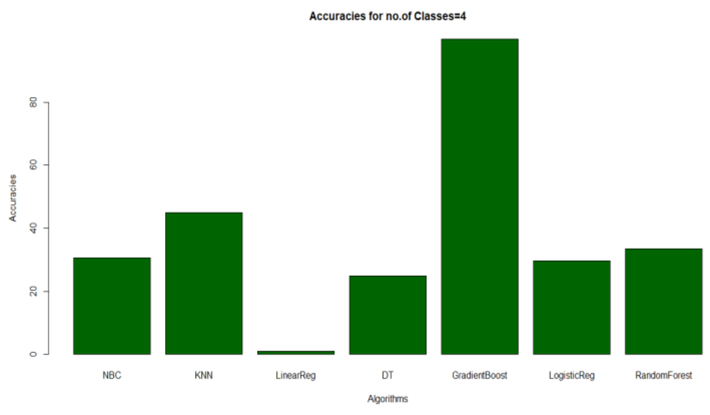


Fig.3: Accuracies of all algorithms considering 4 classes

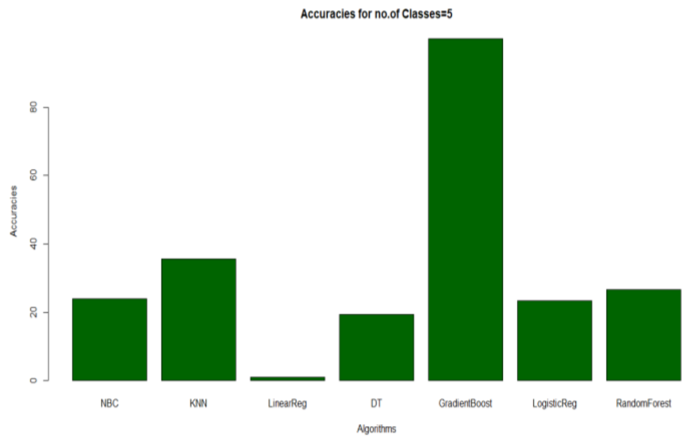


Fig.4: Accuracies of all algorithms considering 5 classes

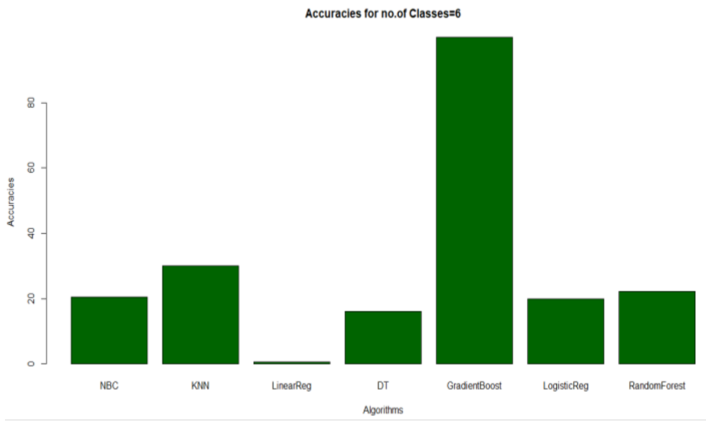


Fig.5: Accuracies of all algorithms considering 6 classes

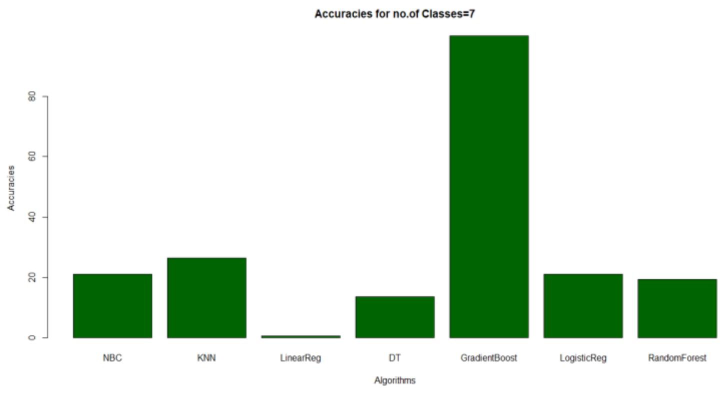


Fig.6: Accuracies of all algorithms considering 7 classes

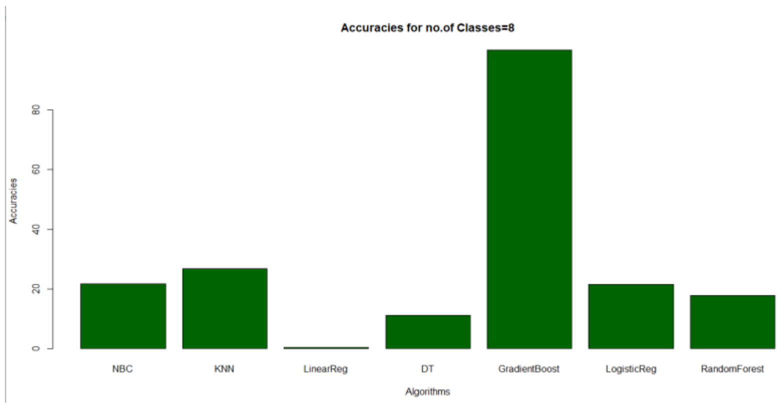


Fig.7: Accuracies of all algorithms considering 8 classes

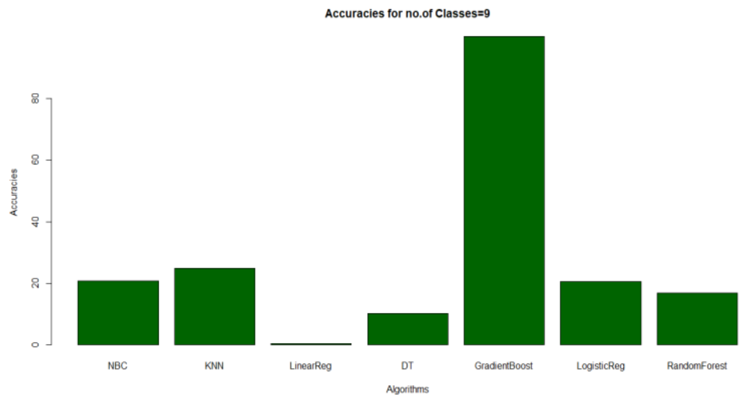


Fig.8: Accuracies of all algorithms considering 9 classes

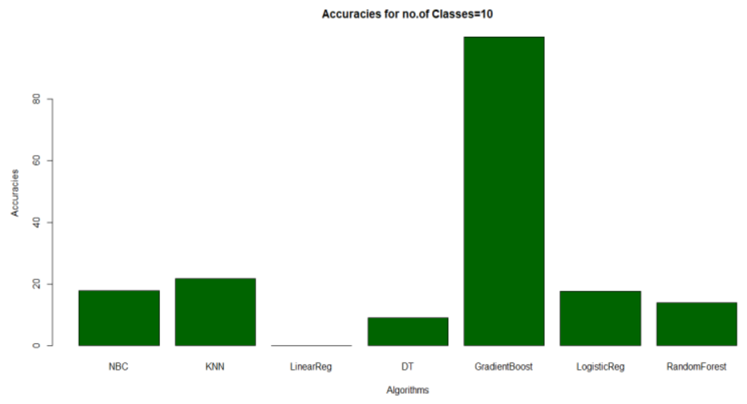
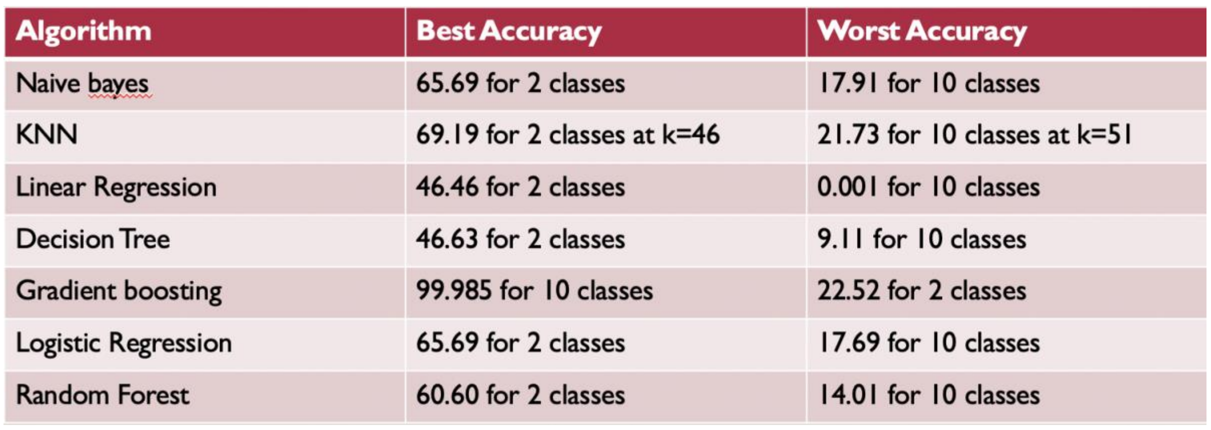


Fig.9: Accuracies of all algorithms considering all classes



Algorithm	Best Accuracy	Worst Accuracy
Naive bayes	65.69 for 2 classes	17.91 for 10 classes
KNN	69.19 for 2 classes at k=46	21.73 for 10 classes at k=51
Linear Regression	46.46 for 2 classes	0.001 for 10 classes
Decision Tree	46.63 for 2 classes	9.11 for 10 classes
Gradient boosting	99.985 for 10 classes	22.52 for 2 classes
Logistic Regression	65.69 for 2 classes	17.69 for 10 classes
Random Forest	60.60 for 2 classes	14.01 for 10 classes

Fig.10: Best and worst accuracies of all algorithms

From the above graphs and table we can conclude that the best accuracy we achieved is for Gradient boosting 99.985 for 10 classes and 69.19 when k=46 for 2 classes. Thus, we have observed that when a greater number of classes are considered for model evaluation, Gradient boosting provides the best results compared to other models shown in Fig.10. Whereas when a lesser number of classes are considered Gradient Boosting shows the worst accuracy of all, although KNN has close to 70% accuracy, it is considered as the best among all with fewer number of classes.

VI. FUTURE WORK AND CONCLUSION

In this project, San Francisco city's crime data is analyzed in order to obtain insights into the crime locations and frequency of occurrences**.** The dataset contains around 1.7 million samples and seven Machine Learning Algorithms are used for classification. Based on the obtained results, the gradient boosting algorithm gave the overall best results since its accuracy was almost consistent when the algorithm was tested with increasing the number of districts for evaluation. Furthermore, this work can be improved by adding and dropping certain attributes during feature selection and analyzing the accuracies for different types of class combinations.

Link to the data and code: https://github.com/sryaswanth/Crime-Classification-Analysis_ML

REFERENCES

[1] San Francisco Crime data from Kaggle - https://www.kaggle.com/c/sf-crime/data

[2] Sathyadevan S., Nair R.R. (2015) Comparative Analysis of Decision Tree Algorithms: ID3, C4.5 and Random Forest. In: Jain L., H., Mandal J., Mohapatra D. (eds) Computational Intelligence in Data Mining - Volume 1. Smart Innovation, Systems and Technologies, vol 31. Springer, New Delhi

[3] Will Koehrsen, Towards Data Science, An Implementation and Explanation of the Random Forest in Python, Aug 2018

[4] Pedregosa et al. Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830, 2011.

[5] Jason Brownlee, How to Fix FutureWarning Messages in scikit-learn, Feb 2019

[6] Gareth James et al. An Introduction to Statistical Learning, 2014

[7] Chalapathy, SurajSahoo, Global and Local Variables in Python, 2016

[8] Michael Galarnyk, Logistic Regression using Python (scikit-learn), Sep 2017

[9] Dannar Mawardi, Linear Regression in Python, Aug 2017

[10] Susan Li, Building A Logistic Regression in Python, Step by Step, Sep 2017

[11] Mirko Stojiljkovic, Logistic Regression in Python, Jan 2020

[12] Avinash Navlani, Understanding Random Forests Classifiers in Python, 2018

[13] Varun, Find the index of value in Numpy Array using numpy.where, Dec 2018

[14] Sanjay.M, MachineLearning — KNN using scikit-learn, Oct 2018

[15] Eijaz Allibhai, Building a k-Nearest-Neighbors Model with Scikit-learn, Sep 2018

[16] Avinash Navlani, Decision Tree Classification in Python, December 2018

[17] Adi Bronshtein, Simple and Multiple Linear Regression in Python, May 2017

[18] Scott Fortmann-Roe, Accurately Measuring Model Prediction Error, May 2012

[19] Chris Beaumont, Cross Validation: The Right and Wrong Way, Oct 2013

[20] Statistical Learning, Dec 2013

[21] Damjan Krstajic et al. Cross-validation pitfalls when selecting and assessing regression and classification models, 2014