COMP5318 - Machine Learning and Data Mining

Assignment 2

June 2018

Comparison of Machine Learning Algorithm Performance on UCI Forest Cover Dataset

Contributed by:

SRIRAM SWAMINATHAN (470495968)

SONALI BHATNAGAR (470531956)

BHARANI RAM BALASUNDARAM (470369588)

Table of Contents:

1.Abstract

In this report we are predicting the forest cover type to be assigned in the correct forest category by using different machine learning models namely, Logistic Regression, Random Forest, Neural Network, Support vector machines, Extreme Gradient Boosting (XGBoost) for accurate decision. This problem is worth solving for the officials who needs to take informative decisions for creating ecosystem management strategies. We received the best prediction result from Random forest followed by Logistic Regression. Due to lack of data, Neural network wasn't able to perform the best.

2.Introduction

2.1 What is the problem you intend to solve?

We are provided with dataset of Cover type which has non-linear distribution. This data is obtained from the US Geological Survey (USGS) and the US Forest Service(USFS) and includes four wilderness areas located in Roosevelt National Forest of northern Colorado. Dataset contains 581k entries with 54 attributes each. However, there are only 12 real features because two of them are represented as a vector opposed to number notation. Each entry is observation of 30 * 30 m cell of forest land and goal of the assignment is to predict cover type of the path based on the observation. We will start our model from initial simple logistic regression, and then will apply more advanced techniques like Random Forest, Neural Networks.

2.2 Why is this problem important?

The features provided represent the forests with minimal human-caused disturbances, which indicates that the resulting forest distribution is entirely due to the ecological processes rather than the practices deployed by forest management. Basic descriptive information is required to support decision making processes for forested lands by natural resource managers who develop ecosystem management strategies. However, Officials who need such kind of information for the neighbouring lands which doesn't fall under their managing area can utilise these strategies to predict and make decisions.

3.Previous work

Few research is done previously & published with size of the dataset with first paper related to the task was published by Blackard and Dean[1]. Feed forward ANN model was used & achieved accuracy of 70.58% based on the below sample size. XB Li[2] applied scalable decision tree system due to the size of the data set and achieved accuracy of 91.86%. Bo L Wang [3] achieved the accuracy of 97.4% with their GS(greedy stagewise) SVM in which they took class 2 as one class and all the other as another class, because the data set consists of 7 classes, and SVM can treat 2 class problem only. But, the results of SVMs cannot be used directly for classification task. Because each SVM could classify differently for a new instance, we need some ensemble method to classify the new instance.

Table 1 : Comparison of previous work done with accuracy on forest cover data

Model	Authors	Accuracy	Training	Test
GS SVM	Bo , L wang, L Jiao	97.4%	400,000 Instances	50,000 Instances
Feed-forward artificial neural network	Denis Blackard	70.58%	11340 instances	565892 Instances
Scalable decision tree	XB Li	91.86%	60%	40%

4.Methods

4.1 Pre Processing

4.1.1 Scaling

Scikit-Learn’s Standard Scalar module is used to implement scaling of our variables.

4.1.2 Data Exploration

The plots below show that data is normally distributed index values of hillshade index at various times. It is observed at 3pm, the index value is more normally distributed while the hillshade index at 9am and noon is more skewed towards the right.

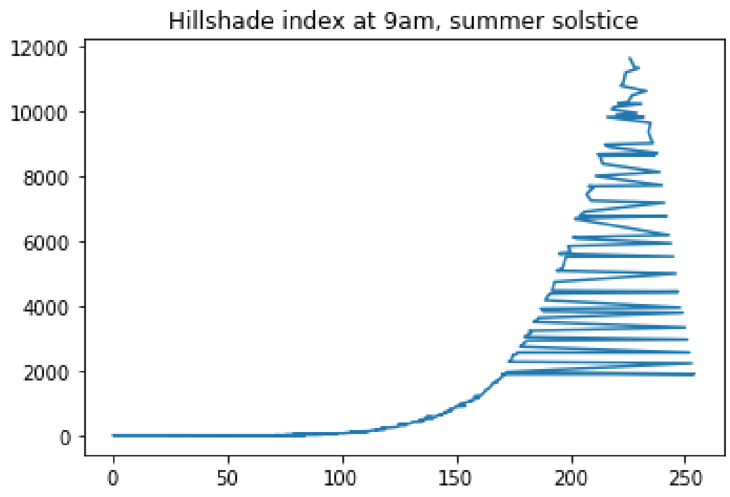

Fig 1: Hillshade Index at 9am

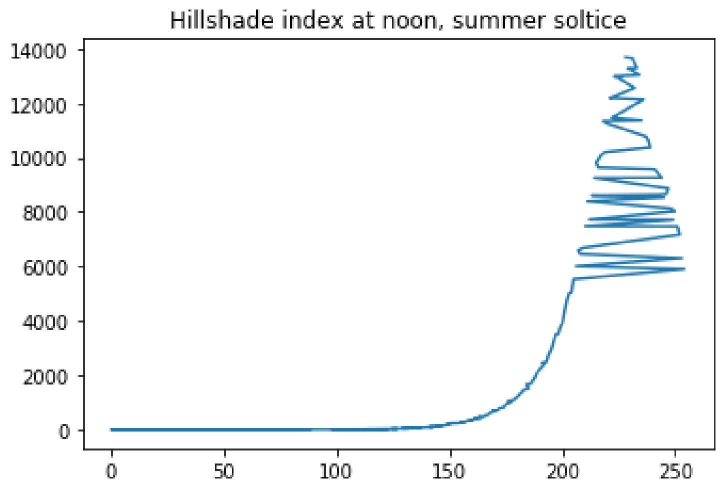

Fig 2: Hillshade Index at noon

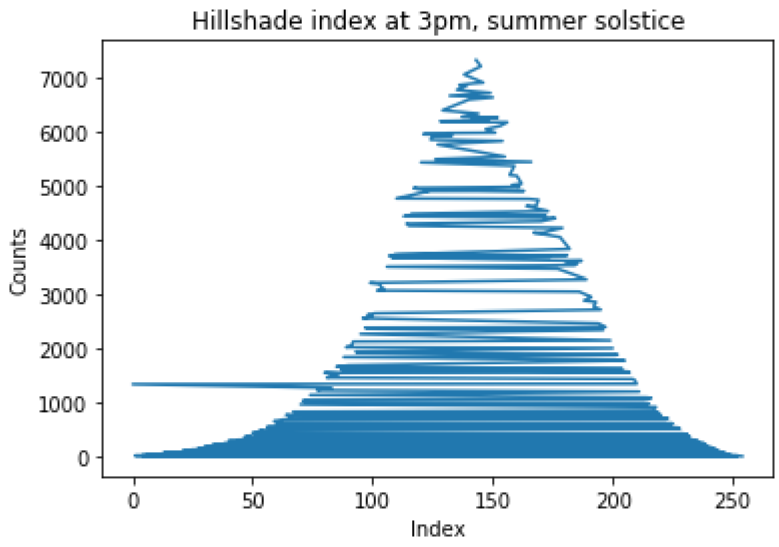

Fig 3: Hillshade Index at 3pm

The below histogram shows that the forest cover types are more for the cover type 1 and 2. The distribution of the forest cover types is more skewed towards forest cover type 1 and 2.

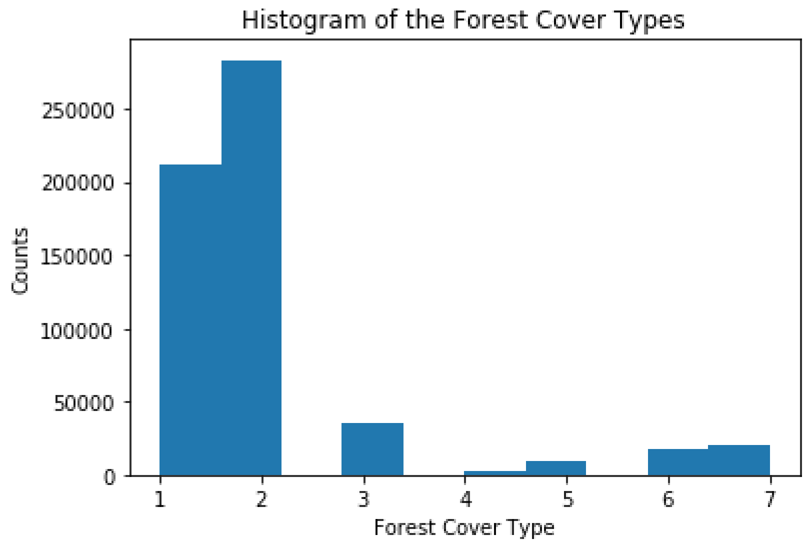

Fig 4: Histogram of forest cover types

The below correlation chart shows that the features are not highly correlated and hence there are no duplicate or redundant features in this dataset.

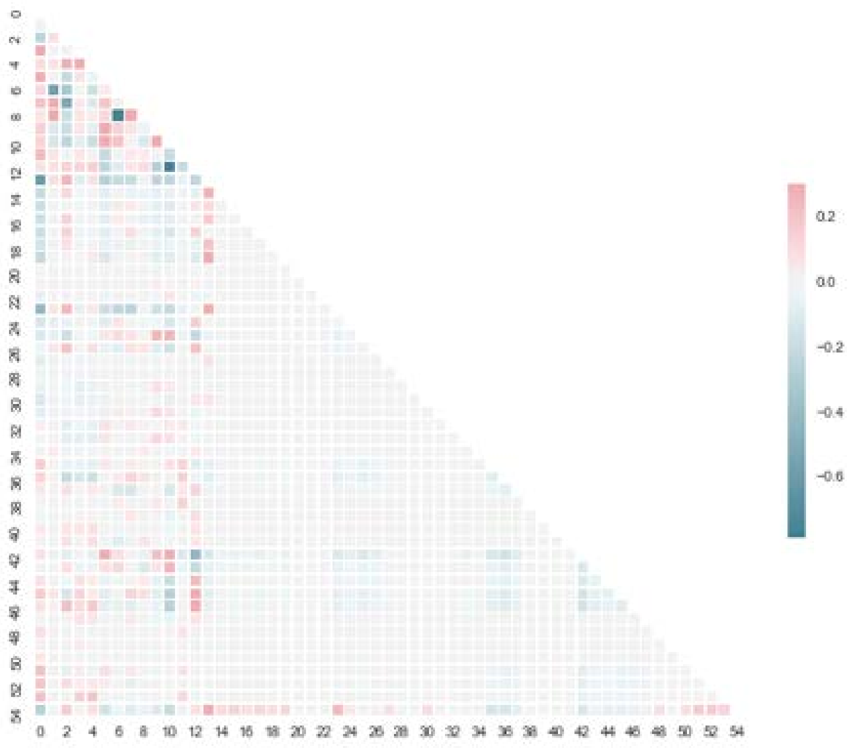

Fig 5:Correlation chart

Table 2 : Comparison of results achieved with the current methods

Algorithm	Average predictive accuracy	Computation Speed	Prone to over fitting	Performance w/ small no. of data?
Logistic regression	70%	1076s	No	Very Fast
Naive Bayes	38%	48s	No	Very Fast
Random Forest	71%	25.8s	No	Fast
Neural Network	91%	480s	Yes	Slow
SVM	61.2%	939s	Yes	Medium
XG Boost With Neural Network	91%	220s	No	Medium

4.2 Models

4.2.1 Logistic Regression

Our first Model is logistic regression. For each class y(k) we train regression on items that belong to class vs all other items. For Logistic regression, We assumes that probability of observation x belonging to class y is given by sigmoid function

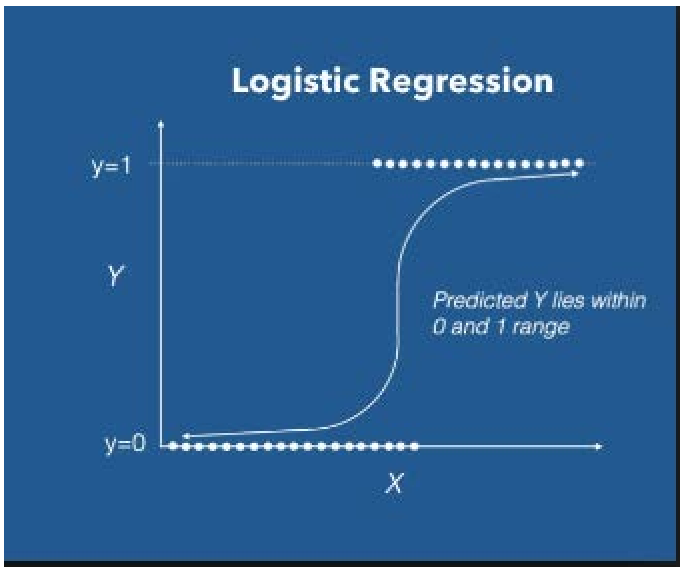

Fig 6: Logistic Regression

$$P(x|y) = \sigma(x) = \frac{1}{1+e^{-\theta \cdot x}}$$

Our goal is to maximize log likelihood L of train data set=

$$\begin{aligned}\mathcal{L}(data) &= \log P(data|\theta) \\ &= \sum_{i=1}^{N} (y^{(i)} \log \sigma(x^{(i)}) + (1-y^{(i)}) \log(1-\sigma(x^{(i)}))\end{aligned}$$

Maximizing it with respect to θ will provide us classifier for class yk Then, bayes optimal classifier h(x) for observation x is:

$$h(x) = \arg\max_k P(x|y^{(k)})$$

Unfortunately, performance of Logistic regression on this train set is not very satisfactorily. We were able to achieve 71 % of accuracy. However, one of the advantages of this models is it works very fast and actually for simple models it classifies with good accuracy.

4.2.2 Neural Network

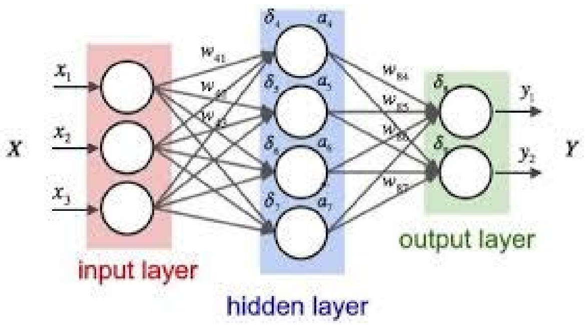

Figure 7 : Neural Network

Each circle on this figure called a neuron, and represents a simple function that gets some input and returns output. This network can be described by following function:

$$f(x) = G(b^{(2)} + W^{(2)} \sigma(b^{(1)} + W^{(1)}))$$

Where σ is a well known sigmoid function, b $^{(i)}$ is a bias vector for each layer *i*, and W$^{(i)}$ is weight matrix of layer ***i.*** This function maps from high dimensional feature space into 7 dimensional labels space, and for each x yields a vector $o \in R^7$. Then optimal classifier is h(x):

$$o = f(x)$$
$$h(x) = \arg\max_i o_i$$

Objective of this model is following loss function:

$$J(w,b) = \sum_{n=1}^{N} ||f(x^{(n)}) - y^{(n)}||^2$$

4.2.3 Support Vector Machines

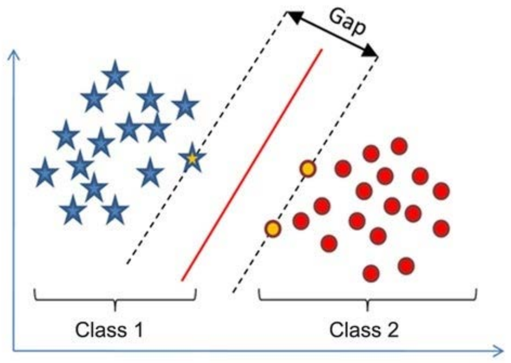

Figure 8: Support Vector Machines

A support vector machine (SVM) constructs a hyper-plane or set of hyper-planes in a high or infinite-dimensional space, which can be used for classification, regression, or other tasks. It selects a maximum margin classifier as in general the larger the margin the lower the generalization error of the classifier. C is called the regular factor. If C is set to be very large, the regularization would be very low, lower C would result in a higher regularization penalty. Thus tweaking the C parameter could give us a better performance in the new data

4,2.4 Naïve Bayes

Naïve Bayes is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. Naïve Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:

$$p(\mathbf{x}, \mathcal{C}_k) = p(\mathcal{C}_k)p(\mathbf{x}_n|\mathcal{C}_k) = \pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k)$$

class conditional density   class prior

class posterior

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_{\mathcal{C}_j} p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)}$$

normalising constant

P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).

P(c) is the prior probability of class.

P(x|c) is the likelihood which is the probability of predictor given class.

P(x) is the prior probability of predictor.

4.2.5 Random Forest:

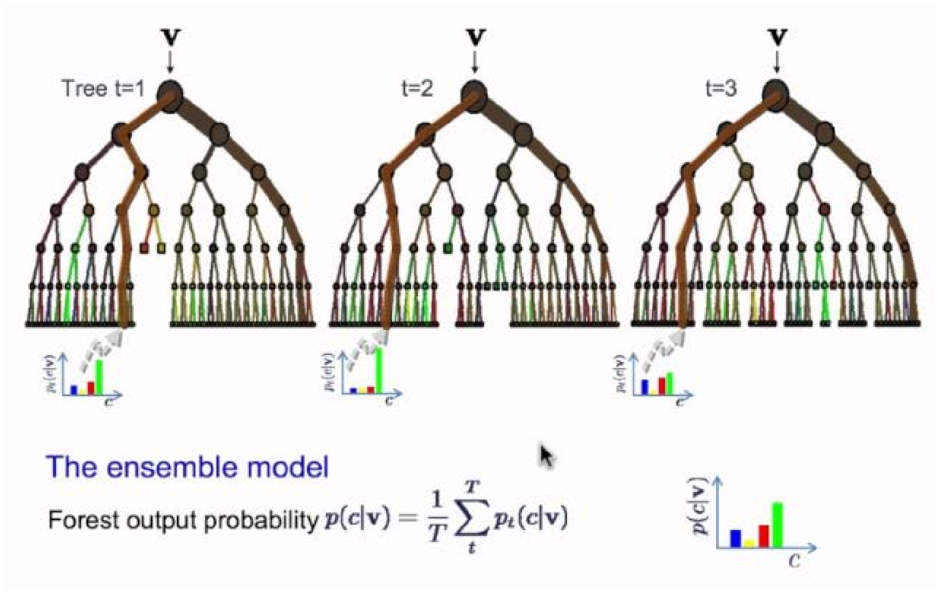
v
Tree t=1
t=2
t=3
$p_t(c|\mathbf{v})$
c
The ensemble model
Forest output probability $p(c|\mathbf{v}) = \frac{1}{T}\sum_t^T p_t(c|\mathbf{v})$
$p(c|\mathbf{v})$

Fig 9: Random Forest

Random forest uses an ensemble method by combining a multitude of decision trees. Random forest uses a bagging method, which averages the predictions of multiple models trained on different samples to reduce the variance and achieve higher accuracy. For a classification task, an instance from a dataset is classified according to each single tree, the trees will vote for their classification and the class having the most votes will be chosen as the final result.

4.2.6 XGBoost(Extreme Gradient Boosting)

XGBoost is an enhanced version of Gradient Boosting Machine which automatically approximates the non-linear transformation as well as deep interaction and treating missing values gracefully while offering powerful performance and fast execution. Gradient Boosting Machine is basically a type of ensemble network where you have multiple regression or classification tree that re-weight train sample based on the error of the previous tree

The highlight is its ability to perform automatic parallel computation. XGBoost builds model more stable by reducing chances of overfitting by leveraging mechanism to penalise error at every iteration, hence reducing the bias. The Core of algorithm is computing gradient function which measures the model performance by optimizing Parameter with unmatched accuracy including fast feature engineering, creating ensemble model or stacking. It can be used for both regression as well as classification problem and is 10 times faster than gradient boosting machine.

5.Experiments and Discussion

The original data set is split into 80% Training and 20% Testing data set

Comparison of results achieved for Training Data Set

	Logistic Regression	Naive Bayes	Random Forest	Neural Network	SVM	XG Boost
F1 score	0.70	0.42	0.72	ND	0.56	ND
Accuracy score	0.715	0.38	0.71	0.91	0.612	0.91
Precision score	0.69	0.57	0.73	ND	0.58	ND
Recall	0.71	0.39	0.72	ND	0.61	ND

Comparison of results achieved for Testing Data Set

	Logistic Regression	Naive Bayes	Random Forest	Neural Network (5 hidden layers)	SVM	XG Boost
F1 score	0.69	0.42	0.72	ND	0.55	ND
Accuracy score	0.70	0.38	0.71	0.70	0.612	0.70
Precision score	0.69	0.57	0.73	ND	0.58	ND
Recall	0.71	0.39	0.72	ND	0.61	ND

With the skewed data set with more number of data for forest cover types 1 and 2, there is a highly non-linear distribution of classes. Hence, a randomised dataset with the capability to develop a non-linear model for a multi-class dataset is what was required.

The first 3% of the data set when passed to these models gives an accuracy of 70% for Logistic Regression. When the entire data set is passed to the Logistic Regression, 71% accuracy is reached. This tells us that the distribution of first 3% data is synonymous with the rest of the dataset. However, entire data set is used for the various models that are studied and compared.

Gaussian Naïve Bayes was tried, although it gives a lower accuracy, to make use of the normal or skewed distribution of selective features. SVM model is used along with the OVR (One-versus-The Rest) concept to perform the multi-class classification and the non-linear dataset. However, the computation time taken is higher and the accuracy is less using the regularisation factor ‘1’. Though the regularisation factor is changed, there is a minimal change in the model accuracy for SVM due to the minimum trade-off in the variance-bias.

Logistic Regression was able to give an average accuracy of 71% mainly due to the linear dataset.

Neural network with 5 hidden layers was used to continuously learn the data set that passes through ELU (Exponential Linear Unit) and Softmax activation functions. The neural network could achieve an accuracy of up to 91% with a slow learning from 87% to 91% accuracy. The accuracy for the neural network could be increased with more number of data set or with addition of features with feature engineering.

Random Forest achieves up to 95% of the test accuracy due to the randomisation of the data and its tree structure. Due to the tree structure, the classification for the multi-class data set is computed much faster than the other algorithms and gives the best result compared to all algorithms.

In the aspiration for faster computation time, XGBoost (Extreme Gradient Boosting Trees) was used to identify top 10 features and the top 10 features was passed to neural network to achieve 89% accuracy with a computation time of 220 seconds.

6.Conclusions and future work

Due to the requirement for the randomisation and non-linear data set with multiple classes, Random Forest gives a higher accuracy with 99% Training accuracy and 95% Test accuracy. If the data set is not randomised, the random forest model causes overfitting and reduces the testing accuracy to 75%.

As a future work, it is decided to include feature engineering to add more features such that the learning gets more optimised for a neural network model. Addition of features will be created based on the given 54 features by feature multiplication or division.

7.References

1. J.A. Blackard, D.J. Dean: Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables
2. Bo,L.,Wang, L.,Jiao, L: Training Hard-margin Support Vector Machines Using Greedy Stagewise Algorithm. IEEE Transactions on neural networks 19(8), 1446-1455(2008)
3. Liefeng Bo, Ling Wang and Licheng Jiao :Training Hard Margin Support vector machines Using Greedy Stagewise Algorithm
4. https://www.kaggle.com/nitin007/forest-cover-type-prediction-complete-part-i
5. http://scikit-learn.org/stable/

Appendix:

Instructions to run the program:

1) Download the Forest Cover Dataset from the below link: https://archive.ics.uci.edu/ml/datasets/Covertype
2) Open Jupyter Notebook inside Tensorflow environment
3) Upload the sswa2435_bbal5837_sbha0604.ipynb and the forest cover dataset into the same folder in the Jupyter notebook
4) The first cell has the libraries and must be executed to import all the libraries to run the machine learning algorithms
5) The second cell must be executed to import the forest cover dataset and be converted to dataframe.
6) The next four cells can be executed to understand the data exploration that was done to understand the dataset
7) Every cell has a heading for the respective models. Each cell must be executed to understand the accuracy for the training and testing data set.

Contributions:

All members equally contributed to the discussions, brainstorming of ideas, exploration of the data and executing the machine algorithms.