Assignment 2 - Group: 13

A Comparison of Machine Learning Classifiers on Forest cover type

Tutors: Kelvin Hsu, Anthony Tompkins, Philippe Morere , Nicholas James, Rafael de Oliveira

Group members: Ruchita Manuja (rman5184), Faraz Ullah Mohammad (fmoh4434), Shakti Malhotra (smal0758)

Abstract

This study talks about classifying the Forest Cover Type dataset available on the UCI Machine Learning Repository. The Forest Cover Type dataset consists of cartographic features of four wilderness areas located in the Roosevelt National Forest of northern Colorado. The objective of this classification problem is to predict the forest cover type. The dataset is pretty large and also requires accurate prediction results so that Natural resource managers can develop ecosystem management strategies with the help of the predicted forest cover type descriptive information . This can be achieved by using Machine learning algorithms. In this study we have used three machine learning algorithms (Gradient Boosting Classifier, Logistic Regression, Support Vector Machine) to assist in solving the classification problem. Various preprocessing activities like Feature Engineering, Principal Component Analysis were used. Several parameters tuning techniques were also used in order to increase the performance of the classifiers. The dataset was split into 5 folds and each model was trained on the training set. To determine the best classifier we have compared accuracy, recall and F-measure scores of each classifier and determined that Gradient Boosting Classifier is the best classifier having obtained the highest accuracy compared to the other classifiers.

Introduction

Machine Learning techniques are becoming popular due to its ability of dealing with big data and the ability in solving various data related issues. The data can be categorized into several categories such as classification, regression, clustering and anomaly detection. Our study focuses on the classification problem for the Forest Cover type dataset. Forest Cover Type is one of the very basic characteristics that is recorded in these inventories. Cover type can be recorded by field personnel or estimated from remotely sensed data, however both these techniques can be time consuming and expensive in certain situations.[1] Accurate natural resource inventory information is important for any federal land management agency in order to classify areas of the forest based on the tree types.This is most important for older forests in the remote area where it maybe difficult to manually measure all types of trees. High transportation costs, difficult terrain are a few obstacles for obtaining accurate and up to date data from these areas .[2] A few more challenges have arisen in past few years such as conserving forest genetic resources where a shrinking land base due to the increasing demand of wood and other tree based products has affected the health and genetic diversity of the remaining forests. Another challenge for the land management agencies is determining appropriate forest practices. Lastly one of the major challenges is managing forests near human development. Certain species are more susceptible than others to human impact and therefore they need to be monitored carefully.[2] All the above challenges whether they arise from how remote the forest is or lack of time, manpower , funds or human impact lead to the need of easier and cost-effective prediction of forest cover types. Predictive models using machine learning algorithms can provide a useful alternative approach for obtaining such data. With the help of machine learning classification algorithms (Logistic Regression, Support Vector Machine, Random Forest) we aim to obtain inventory information which is crucial for the Forest Management agency for adjoining lands that are not directly under its control and are also often impossible to collect. Our main objective is to determine the most predominant tree species (Forest Cover type) in a 30m x 30m land using the cartographic variables provided in our data set.

Dataset Description

Forest Cover type dataset has been retrieved from the UCI Repository (https://archive.ics.uci.edu/ml/datasets/Covertype).
The Forest Cover Type dataset consists of only cartographic variables and has no remote sensed data. The

observations were taken from 30m x 30m patches of forest and were classified as one of the seven forest cover types classes as below.[3]

- C1: Spruce/fir (Picea engelmannii and Abies laciocarpa),
- C2: Lodgepole pine (Pinus contorta),
- C3: Ponderosa pine (Pinus ponderosa),
- C4: Cottonwood/willow (Populus angustifolia, Populus deltoides, Salix bebbiana, Salix amygdaloides),
- C5: Aspen (Populus tremuloides),
- C6: Douglas-fir (Pseudotsuga menziesii)
- C7: Krummholz (Engelmann spruce (Picea engelmannii), subalpine fir (Abies lasiocarpa) and Rocky Mountain bristlecone pine (Pinus aristata).

These classes were determined from US Forest Service (USFS) region 2 resource information system (RIS) data. Independent variables were also derived from data originally obtained from US Geological Survey (USGS) and USFS data. The data is mainly in raw form (not scaled) and contains binary information (0 or 1) for the qualitative independent variables. It includes four wilderness areas in Roosevelt National Forest in northern Colorado. i.e Neota, Rawah, Comache Peak and Cache la Poudre and 40 soil types based on the USFS Ecological Land Type Units. There are about 581012 observations and 54 attributes (comprising of 10 quantitative variables, 4 binary wilderness areas and 40 binary soil type variables) in the forest cover dataset in total. All four wilderness areas are also located far away from human land management activities and disturbance, thus making them good targets for predicting cover types based on natural conditions. [2]

A summary of the dataset and its associated task is provided below:

Title	Forest Covertype data
Dataset Associated Task	Classification
Number of observations	581,012
Number of features	12
Missing Values	Null
Target	Cover Type

Table 1: Class distributions

The Features information is displayed in Table 2

Name	**Data Type**	**Measurement**	**Description**
Elevation	quantitative	meters	Elevation in meters
Aspect	quantitative	azimuth	Aspect in degrees azimuth
Slope	quantitative	degrees	Slope in degrees
Horizontal_Distance_To_Hydrology	quantitative	meters	Horizontal Distance to nearest surface water features
Vertical_Distance_To_Hydrology	quantitative	meters	Vertical Distance to nearest surface water features
Horizontal_Distance_ToRoadways	quantitative	meters	Horizontal Distance to nearest roadway
Hillshade_9am	quantitative	0 to 255 index	Hillshade index at 9am, summer solstice
HillshadeNoon	quantitative	0 to 255 index	Hillshade index at noon, summer solstice
Hillshade_3pm	quantitative	0 to 255 index	Hillshade index at 3pm, summer solstice
Horizontal_Distance_To_Fire_Points	quantitative	meters	Horizontal Distance to nearest wildfire ignition points
Wilderness Area (4 binary columns)	quantitative	0 (absence) or 1 (presence)	Wilderness area designation
Soil_Type (40 binary columns)	quantitative	0 (absence) or 1 (presence)	Soil Type designation
Cover_type (7 types)	integer	1 to 7	Forest Cover Type designation

Table 2: Attribute information

Previous Work

Blackard and Dean have used artificial neural network (ANN) and traditional statistical model based on Gaussian discriminant analysis for performing the classification. They created three mutually exclusive and distinct data sets to train, validate and test the predicted models. They used the training data set for developing both the classifiers, the validation data set (only for artificial neural network) for developing the feedforward artificial neural networks to identify the point in the training process where network will start to overfit the training data set. Finally for both the classifiers the test data set was used which was used in determining how well each classifier performed against a data set that was not used in the creation of the model. They also had to perform a few pre-processing activities such as the variables in the data set were linearly scaled to lie in the range between zero and one, the dependent variables for all the three datasets for artificial neural network were coded as a series of binary variables similar to the soil type and wilderness areas. They used one hidden layer for their ANN implementation and systematically changed the number of nodes in this one layer while keeping the learning rate and momentum rate constant.. They reported achieving an accuracy of 70.58% using artificial neural network and 58.38% accuracy for discriminant analysis. [1] In another study, Mira and Nigel have used distributed Support Vector Machine(SVM). They defined a set of seven new data sets that observations falling into one of the two binary classes where the observations from a positive class were separated from the other six negative classes. Additional data sets with a smaller number of observations were also generated by randomly sampling the data observations. Similar to Blackard and Duncans approach, they also had to perform a few pre-processing activities. The variables in the data set were linearly scaled to lie in the range between zero and one. The training of the distributed SVM architecture was performed layer by layer from the first layer with four independent SVMs to the output layer with only one SVM. To show the performance of the classifier they seperated Class 1 and Class 2 from the other classes and reported the accuracy on each split data set. Their overall accuracy averaged out to be close to 95In another study conducted by Kelvin and Graham from Stanford University they also used Multi Class SVM. Similar to the previous approached they preprocessed their data by splitting it into 10 evenly-sized groups. For each group i they trained their models by combining the other 9 groups before testing on ki. They then average the percentage error on each testing set ki and determined the percentage error for the model. The also removed 44 binary features for testing to reduce overfitting and increase testing performance. They also used Principal Component Analysis (PCA) for reducing the dimensionality of data. While performing Multi-Class SVM they noticed as the number of training samples increases a convergence occurs between the training error and generalization error and concluded that not using booleans lowers the variance and reduces overfitting. They performed a 10-fold cross validation on the entire data and achieved an accuracy of 78.64%.When training the model on a random 70/30 split of the entire data set they obtained 81.35% training accuracy and 78.24% testing accuracy and by removing the boolean feature the accuracies dropped to 75.21% for training data and 72.75% for testing data. [5] Another study conducted by Yerlan from University of California four different classifiers were used. i.e Logistic Regression, Support Vector Machine, Neural Networks and Extremely randomized trees . They used feature engineering and combined Vertical distance to Hydrology and Horizontal distance to hydrology and applied the Pythagorean theorem to yield the Distance to hydrology. They also introduced two features for each pair of features that represent distance i.e their sum and difference. While using Logistic Regression they noticed its performance on the training set was not satisfactory. The second model they applied was SVM and ran this with linear kernel on normalized features. By running this on normalized features they their accuracy came out to be 20% better than Logistic Regression. The third model they used was Neural Network and used Stochastic gradient descent for optimization with back propagation of error in minibatches. They used the same set of normalized features from the SVM model. NN was able to outperform the reported scores by 15%. The last model they used was Extremely randomized trees. Comparing to SVM and NN this model performed the fastest and had the best cross validation score of 88%. [6]

Methods and Approach

4.1 Hardware and Software Specifications

- Hardware: Dell Laptop Windows 8 6GB RAM
- CPU: Core i5
- RAM: 6.0 GB
- Operating System: 64-bit Operating System, x64-based processor
- Software: Jupyter Notebook (Python v3)

4.2 Dataframe Creation

The dataset provided is in a .data format and does not contain a header. For this reason we had to manually specify the columns. The column names were explicitly provided when reading in the .data file using pandas read_csv function. [1]

4.3 Data Analysis

4.3.1 Class Imbalance

The tree cover type dataset is mostly clean i.e contains no missing or noisy data. Each data point is associated with one of the seven specific forest cover types. The distribution of these seven classes, however, is uneven. From Figure 1 we can see that out of the 7 classes , class 2 (Lodgepole pine class) and class 1 (Spruce/fir) form 85% of the data . On the other side, there are two classes i.e class 5 (Aspen) and class 4 (Cottonwood/willow) with less than 2% of records available.

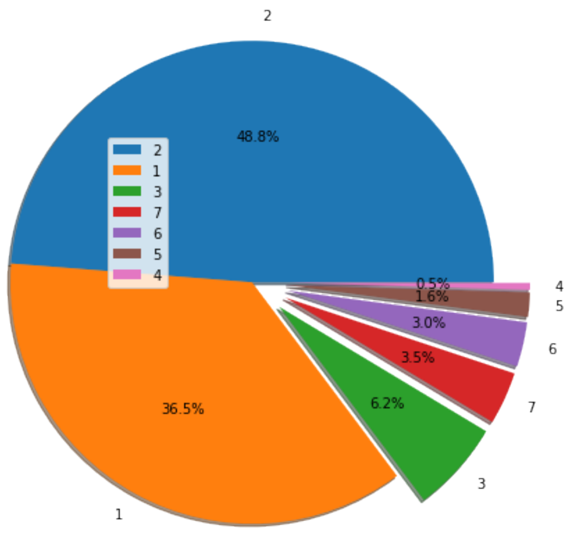


Figure 1: Class Distribution

Figure 1 and Table 3 clearly show that the classes are heavily skewed and thus our data analysis has to cater for this skewness in the distribution of the classes. We can also view the imbalance in the classes with respect to each Wilderness Area from Figure 2. From Figure 2 we can observe that there is a discrepancy in the presence of cover types in different wilderness areas with wilderness area 2 having the most diversity. For this reason one of our approaches would be to split the dataset with respect to wilderness areas and fitting classifiers on each of them.

4.3.2 Distribution of non-binary columns

From the distribution charts in table 4 we can see that most of the features are normally distributed across some mean e.g Slope or Hillshade. Hillshade_3pm is quite evenly distributed and closely resembles a normal distribution. However, in the case of Aspect this is slightly different. The reason for such a strange distribution may lie around the physical meaning of Aspect. Aspect is measured in azimuth *angles* , either positive or negative angle from 0 yields almost the same label for this path. Another point to notice here is that Vertical distance to hydrology has negative values indicating that some observation regions could be below the sea level or below the level of the water source and thus this will need to be catered for during feature engineering.

Forest Cover Type	**Number of Records**	**Percentage of Records**
C1: Spruce/fir	211840	36.5%
C2: Lodgepole pine	283301	48.8%
C3: Ponderosa pine	35754	6.2%
C4: Cottonwood/willow	2,747	0.5%
C5: Aspen	9,493	1.6%
C6: Douglas-fir	17367	3.0%
C7: Krummholz	20510	3.5%

Table 3: Class distributions

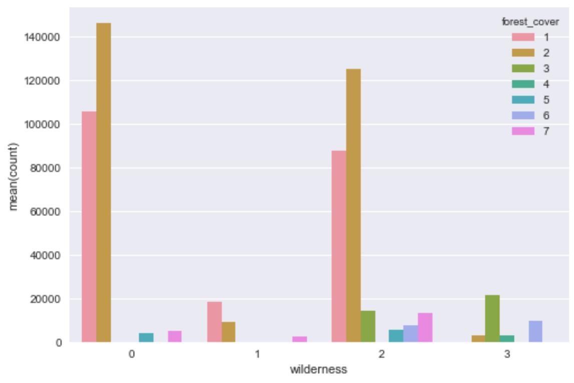


Figure 2: Class Distribution based on Wilderness Areas

4.3.3 Relationship between features and forest cover types

The charts in Table 5 show the comparison between the distributions of various features with respect to cover types. These could be useful features in predicting the cover types as the distributions are quite distinct for each type. As they are also present in different wilderness areas, we can check if these features are quite distinct with respect to wilderness areas as well. If it turns out that these are distinct then our approach of splitting the dataset could prove to be helpful for prediction

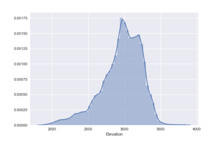
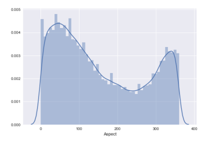
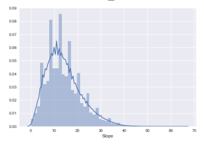
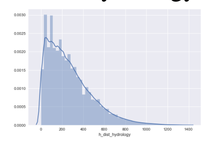
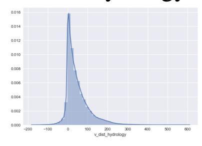
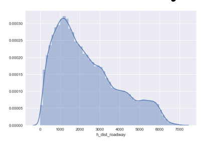
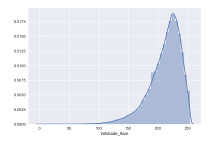
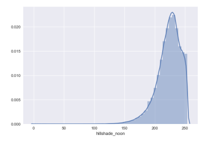
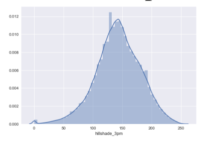
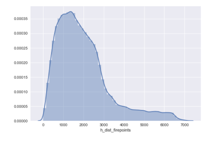
Elevation 	Aspect 	Slope 
h_dist_hydrology 	v_dist_hyrology 	h_dist_roadway 
hillshade_9am 	hillshade_noon 	hillshade_3pm 
h_dist_firepoints 		

Table 4: Distribution of non-binary columns

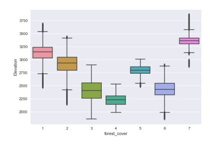
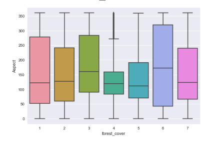
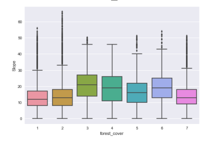
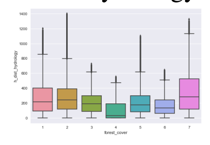
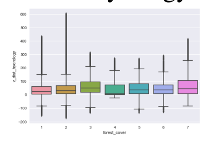
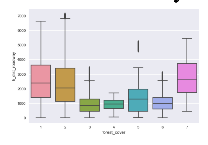
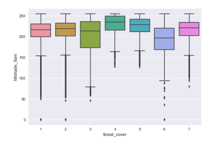
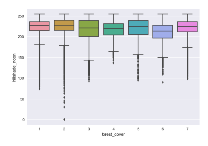
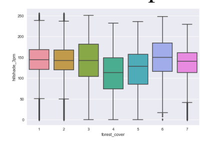
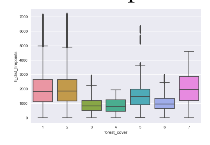
Elevation 	Aspect 	Slope 
h_dist_hydrology 	v_dist_hyrology 	h_dist_roadway 
hillshade_9am 	hillshade_noon 	hillshade_3pm 
h_dist_firepoints 		

Table 5: Distribution of features with respect to cover types

.

4.3.4 Relationship between features and wilderness areas

The charts in Table 6 show the distributions of features for each wilderness area. We have observed here that the distributions are different for each wilderness area , except for vertical distance to hydrology which looks quite similar across the wilderness areas.

4.3.5 Relationship between features , forest cover type and wilderness areas

The charts in Table 7 show the distribution of features for each forest type and wilderness area. By observing the distributions of the features for each cover type in different wilderness area , we have observed differences between forest cover types again. Therefore we can be sure that our approach of splitting the data set would work in this scenario. Another observation that we can infer is that the distribution of features of a cover type is mostly similar across wilderness areas if at all it is present in the area.

4.3.6 Records Selection

The histogram in Figure 3 was created for 11340 records only and shows that classes are completely balanced with respect to cover types at this point. Therefore our second approach would be to consider only the first 11340 rows of the dataset as the training set.

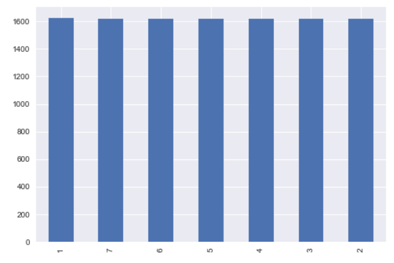


Figure 3: Histogram showing class balance for 11340 records

4.3.7 Correlation

Figure 4 shows us the correlation relationship between Elevation, Hydrology Roadway distance and the seven forest cover types.

Figure 5 shows us the correlation relationship between hill shades at various times and the seven forest cover types.

Figure 6 shows the correlation matrix of the 11340

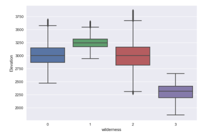
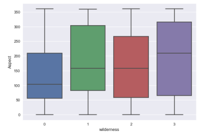
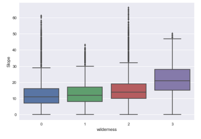
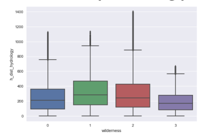
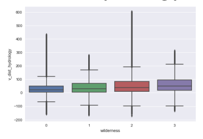
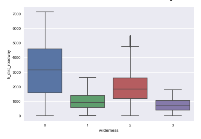
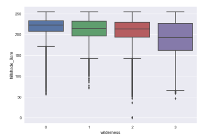
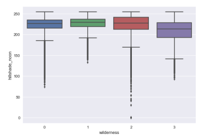
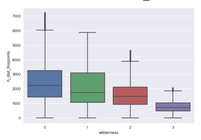
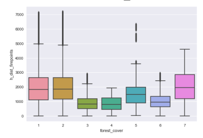
Elevation 	Aspect 	Slope 
h_dist_hydrology 	v_dist_hyrology 	h_dist_roadway 
hillshade_9am 	hillshade_noon 	hillshade_3pm 
h_dist_firepoints 		

Table 6: Distribution of features for each wilderness area

.

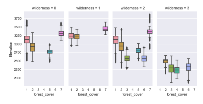
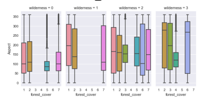
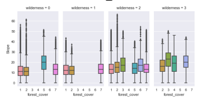
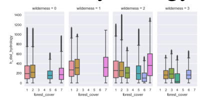
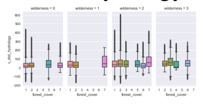
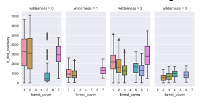
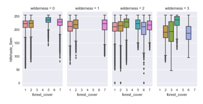
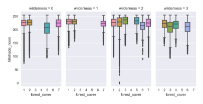
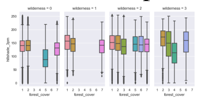
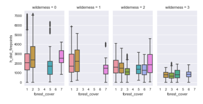
Elevation 	Aspect 	Slope 
h_dist_hydrology 	v_dist_hyrology 	h_dist_roadway 
hillshade_9am 	hillshade_noon 	hillshade_3pm 
h_dist_firepoints 		

Table 7: Distribution of features for each forest type wilderness area

.

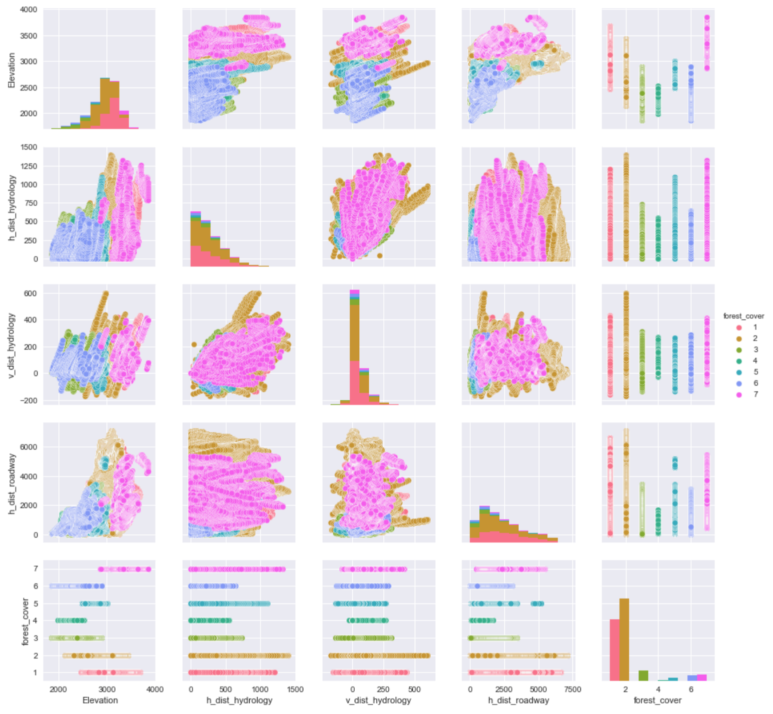


Figure 4: Pairplot for Correlation between features - 1

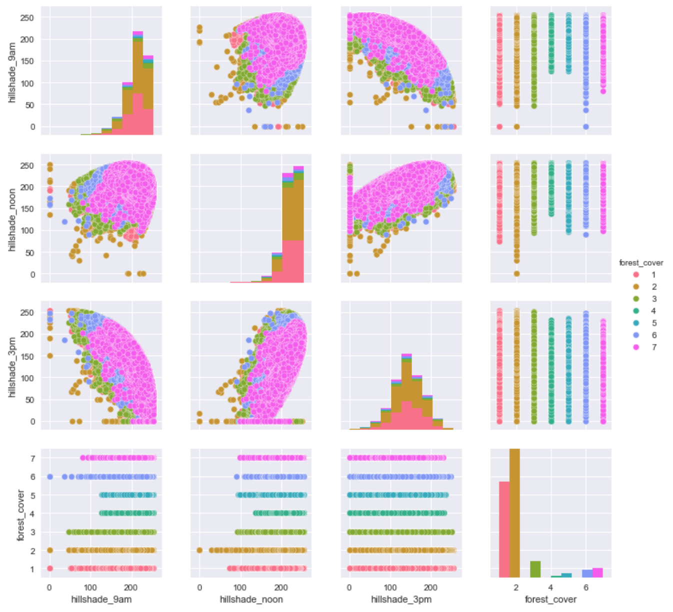


Figure 5: Pairplot for Correlation between features - 2.

records only. From the above we have observed the following:

- Elevation and horizontal distance are correlated. We can say this because the horizontal distance increases as we keep going higher up the region with a slope.

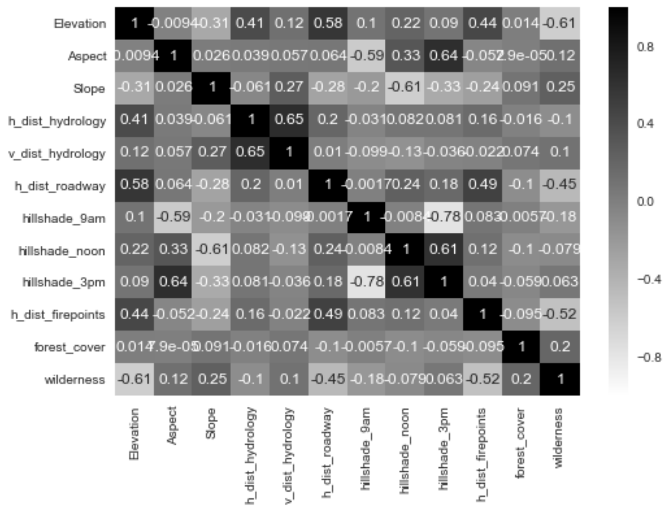


Figure 6: Correlation matrix for 11340 records.

- Aspect is correlated with all the hillshade features as it is the direction the steepest downslope direction is facing which could be any of the possible 8 directions with respect to the sun ranging from 0 to 360 degrees and hillshade values are calculated using aspect, therefore the correlation.[8]
- Horizontal distance and vertical distance to hydrology are correlated
- Horizontal distance to roadway and elevation are correlated
- Horizontal distance and horizontal distance to firepoints is correlated

All the above observations suggest that there is an underlying interaction between these features which could be used to engineer new features.

Based on the above analysis we have decided to work with the following two approaches on our dataset for our predictions:

- Split the dataset with respect to wilderness areas and consider the complete dataset for prediction and mitigate the class imbalance to some extent.
- Consider only the first 11340 rows of the dataset as the training set ,since it is completely balanced with respect to cover types for these number of rows.

4.4 Preprocessing

4.4.1 Feature Engineering

We had to engineer more features out of the existing ones to assist our models in differentiating different data points with respect to the cover types inferred from our data analysis above. dataset. We have used the feature engineering approach as described by Yerlan[6] and performed the following:

- Firstly, we combined Vertical distance to Hydrology and Horizontal distance to Hydrology by applying Pythagorean theorem and converted them to a single distance measure i.e Distance to Hydrology. This combined feature is more useful rather than using independent horizontal and vertical distances, because cover type of land should depend on one direct distance to water source. [6]
- Secondly, we converted the aspect measures from degrees to a sin value.
- Thirdly, we introduced a new binary feature to indicate whether the vertical distance is below the water source level due to its negative values as mentioned in section 4.3.2 above . If the vertical distance is below the water source level it got a value of 0 and if above it got a value of 1.
- Lastly, we have implemented new features by combining the horizontal distance to roadways and firepoints.

4.4.2 Data Cleaning

As per our observations of the counts of soil types we found that soil_type 7 8 and 15 had very low values,hence we dropped these columns since they would be least informative and won't help the models in prediction as well.

4.4.3 Data Normalization

We have only considered the numerical features and omitted out the wilderness and soil_type categorical features to observe the performance of our models. ScikitLearn external library sklearn.processing using the StandardScalar module was used to for data normalization. StandardScalar standardizes the features by removing the mean and then scaling it to unit variance. Scaling and Centring occur independently for each feature by calculating the relevant statistics in the training set.

4.4.4 Principal Component Analysis

Principal Component Analysis (PCA) is a method for reducing the dimensionality of the data maintaining most of the datas variance. In this method the important variables are extracted in the form of components from the large set of variables available in a data set. Low dimensional set of features are extracted from a high dimensional data set with a motive to capture as much information is possible. [4] The main goal of PCA is to find new set of dimensions (attributes) that can better capture the variability of the data and satisfy the following properties:

- Each pair of new attributes has zero covariance (for distinct attributes)
- The attributes are ordered with respect to how much of the variance of the data each attribute captures
- The first attribute captures as much of the variance of the data as possible
- Each successive attribute captures as much of the remaining variance as possible

We performed PCA on the normalized data using 10 components since the total total variance ratio was 97%.

4.4.5 Dataset Creation after Preprocessing

After all the preprocessing activities were completed the next step was to create our data set to be used by classifiers. For this the first 11340 rows were kept for training set and the rest for validation and testing

4.5 Classifiers

4.5.1 Dataset Preparation for Classifiers We will first train our model using the training data and the corresponding training labels to figure out the optimal W and b to fit the training data. We will use the sklearn Python library and train the multinomial logistic regression model by splitting the dataset into four datasets as below:

- train_x
- test_x
- train_y
- test_y

train_x and train_y will be used for modelling the model and test_x and test_y for calculating the accuracy of our trained regression model.

4.5.2 Multinomial Logistic Regression (MLR)

The very first model that was applied is Multinomial Logistic Regression which is a type of Logistic Regression with multi class problems where the dependent variable has more than two categories. This classifier is used when the dependent variable has multiple categories and those categories are not in any ordered form. Logistic regression is used for learning the relationship between the dependent variables i.e our label that we want to predict and the one or more independent variables i.e our features by estimating probabilities using the logistic/sigmoid function as below:

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

Our goal is to ensure that the log likelihood of the training data set is maximised. This can be done as follows:

$$L(data) = logP(data|\theta) = \sum_{i=1}^{\infty}(y^{(i)}log(x^{(i)}) + (1-y^{(i)})log(1-\sigma(x^{(i)}))$$

Multinomial Logistic Regression involves the following stages:

- **Inputs**: The inputs to the Multinomial Logistic Regression are the features of our dataset. For MLR the feature values should always be numerical.
- **Linear Model**: Multinomial Logistic Regression is a multi equation model that is similar to multiple linear regression.
- **Logits**: Logits are the outputs of the linear model. The logits will change with the changes in the calculated weights. They are also known as scores.
- **Softmax Function**: The softmax function is a probabilistic function that calculates probabilities for a given score. The softmax function will return the high probability value for high scores and fewer probabilities for the remaining scores. The calculated probabilities will always be in the range of 0 to 1 and the sum of all probabilities will be 1.
- **Cross Entropy**: Cross Entropy is the last stage of multinomial logistic regression. It uses the cross entropy function to find the similarity distance between the probabilities that is calculated from the softmax function and the target one-hot-encoding matrix.
- **One-Hot Encoding**: One-Hot Encoding is a method for representing the target values or categorical attributes into a binary representation. For each input feature (x1,x2,x3) the one-hot encoding matrix is with the value of 0 and 1 for the target class.

4.5.3 Support Vector Machine

Support Vector Machine is a supervised machine learning algorithm used in classification problems. The objective of SVM is to classify data by finding the linear decision boundary (hyperplane) that separates all the data points of one class from those of the other class. The best hyperplane for an SVM is considered as the one with the largest margin between the two classes, when the data is linearly separable and is known as the Optimal Hyperplane. If the data is not linearly separable, a loss function is used for penalizing points on the wrong side of the hyperplane. The points that are closest i.e have the minimum distance from the optimal hyperplane have the most influence and are known as support vectors.[7]

SVM has the following two properties:

- By maximising the margin , the bound can be minimized between the hyperplane which separates the two classes and the data points which are closest to the hyperplane. [7]
- The bound in the above property does not depend on the dimensionality of the space. [7]

Our assumption is that there is a hyperplane $\psi(x) = \psi^T x + b$ that separates the classes as below:
$\psi(x^{(i))} > 0, if y^{(i)} = 1$
$\psi(x^{(i)}) < 0, if y^{(i)} = -1$

We would want the margin $\rho(x^{(i)})$ between point and hyperplane to be as much as possible using the formula below:

$(x^{(i)}) = y^{(i)}(x^{(i))}||\psi||$

Thus our objective J is to maximise minimal distance:

$$J = \arg\max_{w,b}\{\frac{1}{||w||}\min(y_i\psi(x_i))\}$$

The kernel function can be any of the following:

- linear: . $(x, x^{'})$
- polynomial: $(\gamma(x, x^{'}) + r)^d$ is specified by keyword degree, by coef().
- rbf: $\exp(-\gamma||x, x^{'}||^2).\gamma$ is specified by keyword gamma, must be greater than 0.
- sigmoid $((tanh(\gamma(x, x^{'}) + r))$, where r is specified by coef().

4.5.4 Gradient Boosting Classifier

Gradient Boosting Classifier is a machine learning algorithm for regression and classification problems. It produces a prediction model in the form of an ensemble of weak prediction models. The idea behind gradient boosting classifier is to repetitively leverage the patterns in residuals and strengthen any model with weak predictions. Once a stage is reached where residuals do not meet any pattern that could be modelled , modelling residuals can be stopped.[9]

In summary Gradient Boosting Classifier involves the following: [9]

- Data is modelled with simple models and data is analyzed for errors. The errors signify data points that are difficult to fit by a simple model.
- Then for later models, particular focus is kept on those hard to fit data to get them right.
- In the end, all the predictors are combined by giving some weights to each predictor.

4.6 Programming Language and Libraries

Python is one of the most popular and commonly used programming language by aspiring data scientists for machine learning. It also offers a huge range of open source libraries that can be used easily. For this reason Python 3 was used making use of of the available Python libraries. Pandas library was used for reading the dataset and creating the dataframe. All the classifiers were implemented using sklearn machine learning library.

5 Experiments and Results

The following results are only for the approach of taking 11340 records as our training set and the rest as validation and test set.

5.1 Multinomial Logistic Regression Classifier

We prepared the train and test dataset without categorical attributes. ScikitLearn external library class sklearn.linear_model.LogisticRegression was used to implement this algorithm. We ran the GridSearchSV to find the suitable c values to choose. Though this command took a long time to compute it assisted us with choosing the c values that we should use for our classifier. On the training set we got the best accuracy score for c = 0.01 and thus we concluded that best C value on training data for 5 folds is 0.01 for both L1 L2. We then checked the accuracy on the validation set for c values in the range of [0.01,100,500,1000] to find the most suitable c-value for our classifier by performing hyperparameter tuning on our validation set and decided to go with c-value = 100 for L2 without PCA and c-value = 1 for L1 without PCA. On the other hand, with PCA the c-values were chosen as 10 for both L1 and L2. We have used 3 main parameters for tuning our model i.e

C-value, Penalty (L1 L2) and Class_weight = balanced.

The Performance Metrics for the 4 variations of MLR is as below:

	Accur-acy	**Preci-sion**	**Recall**	**F1-Score**	**Runtime**
L1 Penalty with-out PCA	56.08%	69%	56%	60%	3.08 sec
L2 Penalty with-out PCA	56.38%	69%	56%	60%	33.46 sec
L1 Penalty with PCA	56.16%	69%	56%	60%	3.56 sec
L2 Penalty with PCA	56.28%	69%	56%	60%	11.5 sec

Table 8: Performance Metrics - MLR

We can see that the parameter L2 Penalty gave us a slightly better Accuracy score of 56.38Therefore we can summarize the best parameters for our Multinomial Logistic Regression classifier are as below: C-value = 100 , Penalty = L2 and Class_weight = balanced. The classification report for our best tuned classifier i.e L2 Penalty without PCA is as in Table 9:

Forest Cover Type	**Precision**	**Recall**	**F1-Score**	**Support**
1	0.67	0.63	0.65	189208
2	0.79	0.48	0.59	253561
3	0.61	0.56	0.59	30661
4	0.16	0.89	0.27	1007
5	0.08	0.76	0.15	7121
6	0.29	0.65	0.40	14137
7	0.32	0.90	0.48	17009
avg / total	0.69	0.56	0.60	512704

Table 9: Classification Report - MLR with L2 Penalty without PCA

5.1.1 Confusion Matrix

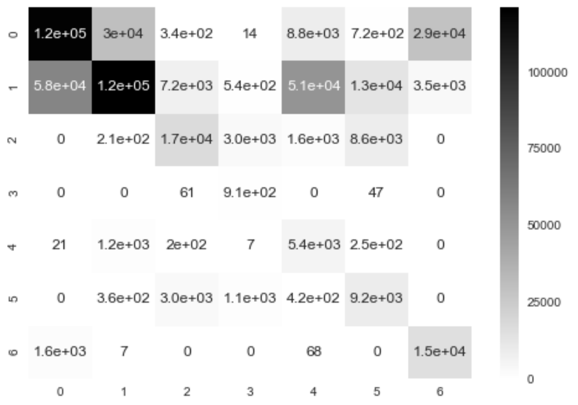


Figure 7: Feature Importance

5.2 Support Vector Machine Classifier

ScikitLearn external library was used to implement this algorithm with the Radial Basis Function (RBF) kernel for training our classifier .

ScikitLearn external library class sklearn.sklearn.svm.SVC was used to implement this algorithm. We ran the GridSearchSV to find the suitable c and gamma values to use. Though this command took a long time to compute it assisted us with choosing the c and gamma values that we should use for our classifier. We got the best results on c = 1 and gamma = 0.01 and thus concluded that we would get the best results with c value = 1 and gamma value = 0.01. We have used 4 main parameters for tuning our model i.e C Value, Gamma , Kernel = rbf and Class_weight = balanced. To find the most suitable c-value for our classifier we performed hyperparameter tuning on our validation set and decided to go with c-value = 10 and gamma value = 0.01. SVM was performed on reduced dataset (with PCA) and full data set (without PCA) and the following results were achieved:

	Accur-acy	**Preci-sion**	**Recall**	**F1-Score**	**Runtime**
SVM with PCA	58.25%	0.71	0.58	0.61	341.42 sec
SVM with-out PCA	67.03%	0.73	0.67	0.68	334.55 sec

Table 10: Performance Metrics - SVM

We can see that the parameter SVM without PCA gave us a better Accuracy score of 56.38Therefore we can summarize the best parameters for our SVM Classifier as below: C-value = 10 , Gamma = 0.01 , Kernel = rbf and Class_weight = balanced.

The classification report for our best tuned classifier i.e SVM without PCA is as below:

Forest Cover Type	Precision	Recall	F1-Score	Support
1	0.69	0.71	0.70	189208
2	0.81	0.60	0.69	253561
3	0.68	0.74	0.71	30661
4	0.34	0.96	0.51	1007
5	0.18	0.88	0.30	7121
6	0.45	0.80	0.57	14137
7	0.48	0.94	0.64	17009
avg / total	0.73	0.67	0.68	512704

Table 11: Classification Report - SVM without PCA

5.2.1 Confusion Matrix

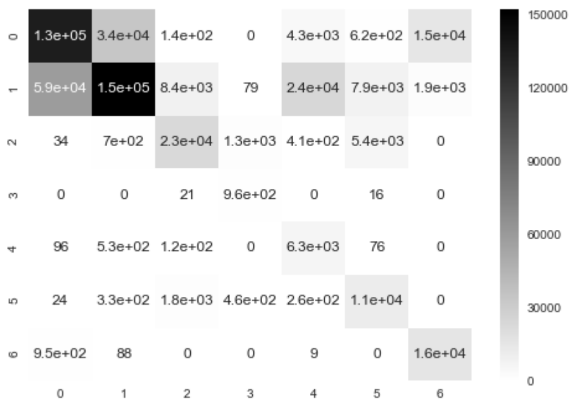


Figure 8: Feature Importance

5.3 Gradient Boosting Classifier

First, we prepared the train and test dataset by keeping all the features except wilderness and soil_type attributes, Secondly we used the full dataset and lastly we used random forest to select the subset of features from all the features and fit the new set with Gradient Boosting classifier.

ScikitLearn external library class sklearn.ensemble.GradientBoostingClassifier was used to implement this algorithm. We first tuned our learning rate and n_estimators. The n_estimators were set as $[50,80,100,150,200]$.We then ran the GridSearchSV command to find the suitable param_n_estimator value. Though this command took a long time to compute it assisted us with choosing the param_n_estimator that we should use for our classifier. We got rank_test_score = 1 for para_n_estimators = 50 and thus we decided to go with 50 and tune the rest of the parameters.

We have used various parameters for tuning our model as below:

- (learning_rate=0.1,n_estimators=50,min_samples_leaf =50, max_features='sqrt',subsample=0.8, random_state=10)

- The best parameters for all **our features except wilderness and soil_type** are as below: (learning_rate=0.01,n_estimators=600,max_depth=8, min_samples_split=800,subsample=0.8, min_samples_leaf=40,max_features=13, random_state=10)

- The best parameters for **full dataset** are as below: (learning_rate=0.01,n_estimators=600,max_depth=8, min_samples_split=200,subsample=0.8, min_samples_leaf=70,max_features=25, random_state=10)

- The best parameters after **feature selection** are as below: (learning_rate=0.01,n_estimators=600,max_depth=8, min_samples_split=600,subsample=0.8, _samples_leaf=70,max_features=15, random_state=10)

These best parameters were then used on our validation set to further tune our model and validated with the test data. The performance metrics for the three variations is as below:

The classification report for our best tuned classifier i.e Gradient Boosting on Full dataset is as below:

	Accuracy	Precision	Recall	F1-Score	Runtime
All the features except wilderness and soil_type attributes.	63.77%	73%	64%	66%	290.77 sec
Full dataset	70.74%	77%	71%	72%	393.83 sec
Feature Selection	66.99%	74%	67%	69%	309.79 sec

Table 12: Performance Metrics - Gradient Boosting Classifiers

Forest Cover Type	Precision	Recall	F1-Score	Support
1	0.73	0.73	0.73	189208
2	0.84	0.64	0.73	253561
3	0.79	0.84	0.81	30661
4	0.39	0.97	0.56	1007
5	0.18	0.94	0.30	7121
6	0.50	0.87	0.64	14137
7	0.50	0.96	0.66	17009
avg / total	0.77	0.71	0.72	512704

Table 13: Classification Report - Gradient Boosting on Full dataset

5.3.1 Confusion Matrix

Displayed below in figure 9

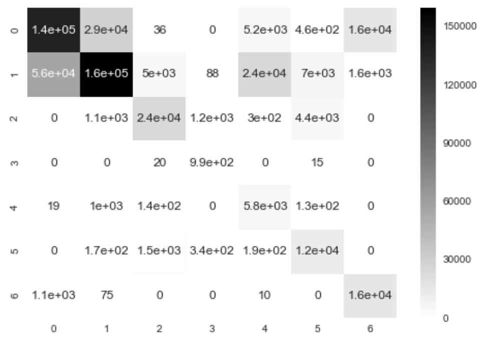


Figure 9: Feature Importance

5.4 Performance Comparison (10 -fold cross validation)

10-fold cross validation technique was used to compare the performances of our best tuned classifier using the cross_val function available within the sklearn library.

The performance results of each classifier for the best parameters are used for this comparison analysis only.

	Accuracy	Precision	Recall	F Measure
MLR - L2 Penalty without PCA	63.77%	73%	64%	66%
Gradient Boosting Classifier - Full Dataset	70.74%	77%	71%	72%
SVM without PCA	67.04%	73%	67%	68%

Table 14: 10 fold Performance Comparison

Based on the experiments and results obtained in this study **Gradient Boosting Classifier** obtained the highest accuracy compared to the other classifiers i.e Multinomial Logistic Regression Support Vector Machine.

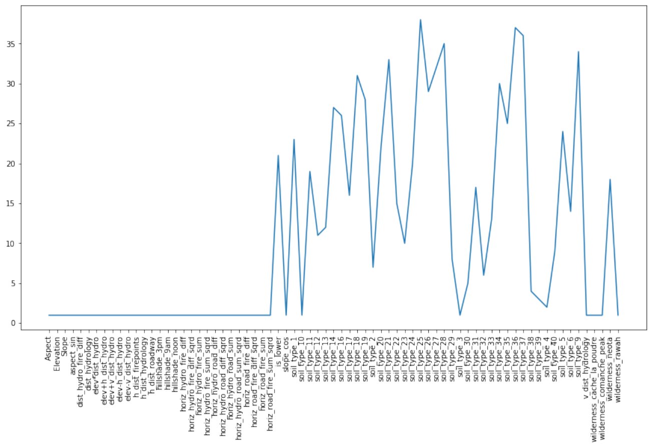


Figure 10: Feature Importance

6 Conclusion and Future Work

6.1 Conclusion

As a conclusion, parameter tuning has a large impact on the performances of each classifier. The performance results of different classifiers were compared in this study based on the accuracy, recall, precision, as well as F-measure scores. With accuracy of 70.74.%, **Gradient Boosting Classifier** stood out against other classifiers in term of performance ,accuracy and is more consistent than other classifiers which have been trained and tested. SVM classifier also gave similar accuracy to gradient boosting classifier.One notable observation is that we achieved the best results when we considered all the features in our dataset,given and engineered along with the soil type and wilderness features,which suggests that these features are important for classification and useful for differentiating between the cover types.

If we observe the feature importance after doing feature selection using Random Forest Classifier,we see that the original and the engineered features are given the highest rank,followed by soil type features.

Therefore,feature engineering proved beneficial in classifying the cover types and improving the accuracy of our classifier.

6.2 Suggested Future work

For future work we would like to implement the same classifiers using our second approach of splitting the dataset based on the four wilderness areas which would allow us to use more data for training,therefore capturing more variations and provding more examples for the classifiers to learn to classify the cover types,also helping us in mitigating the class imabalance that we have in our dataset as we have seen in our analysis that each wilderness area has atleast one or two dominant cover types.Therefore,one classifier's failure to classify a cover type would be compensated by the success of other classifier trained on a different wilderness area dataset.In addition to these we would like to use other tree classification classifiers and then apply the Random Forest classifier for feature selection and try more complex machine learning algorithms such as Neural Networks or Deep Learning techniques.

References

1. Blackard J, Dean D. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. Computers and Electronics in Agriculture. 1999;24(3):131-151.

2. MacMichael D, Si D. Addressing Forest Management Challenges by Refining Tree Cover Type Classification with Machine Learning Models. IEEE International Conference on Information Reuse and Integration (IRI. 2017;:177-183.

3. Trebar M, Steele N. Application of distributed SVM architectures in classifying forest data cover types. Computers and Electronics in Agriculture. 2008;63(2):119-130.

4. Practical Guide to Principal Component Analysis (PCA) in R Python [Internet]. Analytics Vidhya. 2016 [cited 26 May 2018]. Available from: https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/

5. Crain K, Davis G. Classifying Forest Cover Type using Cartographic Features [Internet]. Stanford University; 2014. Available from: http://cs229.stanford.edu/proj2014/Kevin

6. Idelbayev Y. Assignment 1. Predicting cover of forest [Internet]. San Diego; 2018. Available from: http://www.diva-portal.org/smash/get/diva2:1117814/FULLTEXT01.pdf

7. Sjqvist H. Classifying Forest Cover type with cartographic variables via the Support Vector Machine, Naive Bayes and Random Forest classifiers.

8. How Hillshade worksHelp — ArcGIS Desktop [Internet]. Pro.arcgis.com. 2018 [cited 1 June 2018]. Available from: http://pro.arcgis.com/en/pro-app/tool-reference/3d-analyst/how-hillshade-works.htm

9. Grover P. Gradient Boosting from scratch ML Review Medium [Internet]. Medium. 2017 [cited 3 June 2018]. Available from: https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d

8 Appendix

8.1 Files Submitted

The zip folder submitted consists of the following files:

- PDF Report (current document)
- 2 Python files as below:
  - forest_cover_dataset_eda.ipnyb - This file consists of Exploratory Data Analysis conducted
  - forest_cover_type_prediction.ipnyb -This file consists of Data Preprocessing, Feature Engineering the Machine Learning Algorithms implemented

8.2 Instructions on how to run code

- Use any Jupyter notebook
- Go to File $\rightarrow$ $Open.Draganddrop$

  $"forest\_cover\_dataset\_eda.ipnyb"$ & $"forest\_cover\_type\_prediction.ipnyb"$

  $filetothehomeinterface$ $andclickupload.$

  $Torunthecell, youcanpress$ $Ctrl - EnterorhitthePlaybuttonatthetop.$

8.3 Group Members Contribution

Group Member	**Tasks**
Ruchita Manuja	Abstract, Introduction, Dataset Description , Dataframe Creation, Data Analysis & Visualizations, Report Writing , Classifier Theory, Python Coding, Experiments & Results
Faraz Ullah Mohammad	Data Analysis & Visualizations , Data Preprocessing, Principal Component Analysis, Python Coding, Experiments & Results , Discussion
Shakti Malhotra	Previous Work, Report Writing & LaTex Report Writing and coding ,Python Coding, Data Analysis & Visualizations ,Conclusion and Future Work, Referencing

Table 15: Group Member Contribution