Project 2 – Decision Trees, Linear Regression, Model Trees, Regression Trees
CS548 / BCB503 / CS583 Knowledge Discovery and Data Mining - Fall 2019
Prof. Carolina Ruiz

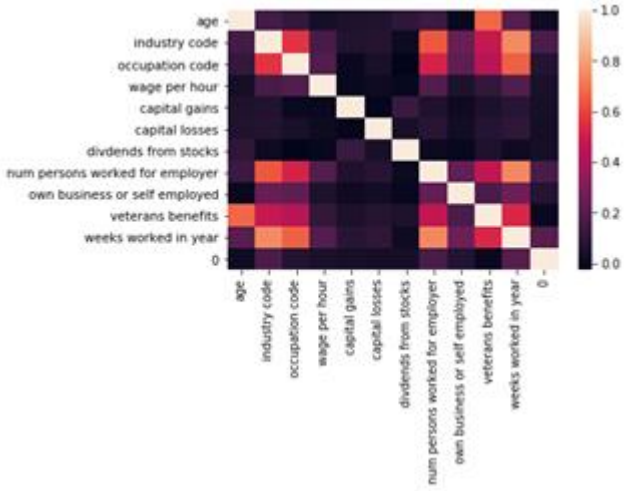
Students: Bhoomi Patel and Srinarayan Srikanthan

	Classification	Regression
Dataset : ● Dataset Description ● Data Exploration ● Initial Data Preprocessing (if any)	/05 /10 /05	
Code Description: Python libraries and functions, and/or your own code	/10	/20
Experiments: ● Guiding Questions	/10	/10
● Sufficient & coherent set of experiments	/10	/10
● Objectives, Parameters, Additional Pre/Post-processing	/10	/10
● Presentation of results	/10	/10
● Analysis of individual experiments' results	/10	/10
Summary of Results, Analysis, Discussion, and Visualizations	/20	/20
Advanced Topic	/30	
Total Written Report	/220 = /100	

Dataset Description, Exploration, and Initial Preprocessing:

[05 points] Dataset Description: (e.g., dataset domain, number of instances, number of attributes, distribution of target attribute, % missing values, …)
The dataset is from the census bureau database to determine the income levels of a person. The number of instances is 199523, with 40 attributes considered for modelling. The target has two classes each having a distribution of 93.80% and 6.20% respectively. The following fields have missing values: State of previous residence(709-0.35%),migration code-change in msa(99696-49.96%),migration code-change in reg(99696-49.96%),migration code-change within reg(99696-49.96%),migration prev res in sunbelt(99696-49.96%), country of birth father(6713-3.36%), country of birth mother(6119-3.06%), country of birth self(3393-1.70%)

[10 points] Data Exploration: (e.g., comments on interesting or salient aspects of the dataset, visualizations, correlation, issues with the data, …)
The dataset has 33 Nominal attributes and 7 Continuous attributes .The dataset is an extensive one, with nearly 2 lakh instance representing the actual population. From the dataset it is observed that among numeric attributes capital gains has the highest standard deviation of 4697.53 and year has the lowest standard deviation of 0.5. It is also observed that capital losses has maximum number of zeros with a percentage of 98.04. By plotting the correlation matrix, we interpreted that *Capital Gains* column is highly correlated to the target variable *income*. The major issue with the dataset is the % of missing values it contains. Of the total columns that contained the missing values, majority columns has missing data almost equal to 50%. Another issue was the distribution of the income attribute, that was used as a classification predictor. It was clearly dominated by one class of values, making it difficult to develop a model to learn patterns in the weaker class.



[05 points] Initial data preprocessing, if any, based on data exploration findings: (e.g., removing IDs, strings, …)
Removed the entire column *instance weight*. Further there were missing values represented as '?' and the column hispanic Origin has values NA. These values were converted to categorical values using the following: df = df.replace('?', 'Missing'); df = df.fillna(df['hispanic Origin'].mode()[0])

Code Description: Python Libraries and Functions you used and with what parameters. If you wrote your own Python code, describe it here. (at most 2 pages)

[10 points] Classification Techniques:

Preprocessing Techniques for Classification: For zeroR: we did not perform any preprocessing because we implemented it from scratch
For OneR: We just replaced NA values in *hispanic Origin* column using function *df.fillna('Not Available')* as we implemented it from scratch
For DecisionTrees & Random Forest: First, we used label encoding for column *education* using function *LabelEncoder()* and then, we *OneHotEncoded* all the nominal values using the function *pd.get_dummies(df) i.e.* OneHotEncoding with drop = 'first'
Libraries used for Accuracy, Precision, Recall, AUC : *sklearn.metrics.accuracy_score, sklearn.metrics.precision_score, sklearn.metrics.recall_score, sklearn.metrics.f1_score*
"Zero-R": Split the entire dataset into test & train using the library *sklearn.model_selection.KFold* and function *KFold* with the *parameters: n_splits=10*. Compute train_index and test_index for the entire DataFrame using *kf.split* with *parameter as dataframe.* Now, do the following for each split: On the train dataset, find the maximum occurring target value. Now compare the max target value obtained on the train dataset with test dataset, compute the correctly predicted targets and then calculate the accuracy. Add this accuracy to the list. Finally, after the entire loop, compute average accuracy using accuracyList.

"One-R": Library used: *pandas, sklearn.model_selection.KFold*. Initial set maxAccuracy=0. Apply preprocessing mentioned above for OneR. Using function *KFold* with the *parameters: n_splits=10*, compute train_index and test_index for the entire DataFrame using *kf.split* with *parameter as dataframe.* Now, do the following for each split: For each column in the dataset, use *pd.crosstab* with parameters: *DataFrame.column* and *target*. To this, compute the occurrence of maximum using function *idxmax* with *parameters: axis=1.* For each row compare with the maximum value obtained and assign it to a variable. Use this variable to compute accuracy. if (accuracy > maxAccuracy), accuracy = maxAccuracy. Add this to the list accuracyList. Also store the corresponding column name and print the column name with maximum accuracy. Finally, after the entire loop, compute average accuracy using accuracyList.

Decision Trees: Library used:pandas, *sklearn.tree.DecisionTreeClassifier, sklearn.model_selection.KFold*. Using function *DecisionTreeClassifier, parameters* used are *criterion*, *max_depth* and *min_samples*. Based on the criterion used the best attribute to split is determined. This process is repeated in each child with reduced data instances until the sample size is equal to or less than the min_samples or the max_depth reached. This process is repeated on each folds having different training and test samples. This is done using *KFold* with the *parameters: n_splits=10*. *classifier=tree.DecisionTreeClassifier(min_samples_split=100, criterion = "entropy", random_state=0) classifier = classifier.fit(x_train,y_train)*

Random Forests: Library used: *pandas*, *sklearn.ensemble*, *sklearn.model_selection.KFold*. Function *RandomForestClassifier* is invoked with the *parameters n_estimaters* which is the number of trees being generated in the forest. Each predict the class of the target, and the target with maximum votes from there trees is the final prediction, the *next parameter* is the *criterion* which is either gini or entropy(entropy used for the project but experimented with gini).The other parameters used are max_depth to limit the depth of each tree and min_samples which gives the minimum number of samples to split so that we don't overfit to the particular dataset on which it is being trained.This is repeated for 10 folds with varying data instances for modelling and prediction generated using *KFold* with the *parameters: n_splits=10*. *classifier= RandomForestClassifier(n_estimators=, criterion = "entropy", random_state=0) classifier=classifier.fit(x_train, y_train)*

[20 points] Regression Techniques:

Preprocessing Techniques for Regression: Used label encoding for column *education* using function *LabelEncoder()* and then, we *OneHotEncoded* all the nominal values using the function *pd.get_dummies(df) i.e.* OneHotEncoding with drop = 'first'. Used *sklearn.preprocessing.scale* to scale the attributes for Linear Regression and Regression Trees.

Linear Regression: Libraries used: *sklearn.linear_model.LinearRegression,sklearn.preprocessing.OneHotEncoder, sklearn.preprocessing.LabelEncoder, sklearn.model_selection.KFold, sklearn.preprocessing.scale, sklearn.metrics.mean_absolute_error, sklearn.metrics.mean_squared_error, sklearn.metrics.r2_score.* The object of linear regression model is created and is fit to the training data instance with the parameter normalize=False. This function finds a hyperplane between all the features with minimum difference between the hyperplane and the point in the test set. Later predictions on the test data are made using this hyperplane and the error values are calculated.

Regression Trees: Libraries used: *sklearn.preprocessing.OneHotEncoder, sklearn.preprocessing.LabelEncoder, sklearn.model_selection.KFold, sklearn.preprocessing.scale, sklearn.tree.DecisionTreeRegressor, sklearn.metrics.mean_absolute_error, sklearn.metrics.mean_squared_error, sklearn.metrics.r2_score*. Using *KFold=10,* split the dataset into training and testing. For each split, do the following: use *DecisionTreeRegressor* using the *parameters* such as *random_state, min_sample_leaf, etc.* Using *fit* function, fit the training dataset and then using *predict* function, compute y_pred for for the test dataset. Finally compute mean_absolute_error and mean_squared error using the libraries mentioned above. Here, the *DecisionTreeRegressor* function, obtains the decision tree for continuous variables and predicts the target based on the model constructed. It outputs the mean value of the samples in the leaf node.

Model Trees [10 points]: Libraries used: *sklearn.preprocessing.OneHotEncoder, sklearn.preprocessing.LabelEncoder, pandas, numpy, sklearn.metrics.mean_absolute_error, sklearn.tree.DecisionTreeRegressor, sklearn.linear_model.LinearRegression,* Apply preprocessing to the dataset by converting nominal attributes to continuous using *OneHotEncoder* and scaling the attributes using *preprocessing.scale*. Then, split the dataset using the library *sklearn.model_selection.train_test_split* and function *train_test_split* with the *parameters: Dataset, target, test size = 0.3 and random_state=0*. Compute regression tree using *DecisionTreeRegressor* with the *parameters random_state=0, min_sample_split=2, max_leaf_nodes=5* and then apply *fit* function to X_train, y_train. Now using the *tree_*, get the *node_count, children_right, children_left, feature and threshold using regressionTree.tree_.node_count*, *regressionTree.tree_.children_right, regressionTree.tree_.children_left, regressionTree.tree_.feature, regressionTree.tree_.threshold.* Maintain a stack for the intermediate and the leaf nodes. Traverse the tree structure to obtain depth, right and left child of each node and all the leaf nodes. Now for each node in node_count, split the data set to obtain get X_left, y_left, X_right, y_right using feature and threshold of each of the nodes. Fit the data on these nodes using *LinearRegression to obtain leftModel and rightModel.* Compute *y_predLeft* , *y_predRight* and then calculate the mean_absolute_error and root mean squared error for each left and right model. Calculate the combined loss of left and right child using the *formula (N_left*lossLeft + N_right*lossRight) / N; where N_left* is the length of the data in the left instances and *N_right* is the length of the data in the right instances and *N* is the *X.shape*. Obtain the mean_absolute error for the data.

Guiding Questions

[10 points] Three Guiding Questions for the Classification Experiments: (at most 1/3 page)

1. Predict a person's income based on capital gains, dividends from stocks, weeks per year and age.
2. Predict the income of a degree holder given age, sex and marital status.
3. Predict the income of a person based on his class of worker, occupation code and industry code.

[10 points] Three Guiding Questions for the Regression Experiments: (at most 1/3 page)

1. Predict the age of person based on capital gains and tax filer status.
2. Predict the age of person given income, education and marital status.
3. Predict the age of a person based on marital status, capital losses who earns less than 50k.

[40 points] Summary of Classification Experiments Use 10-fold cross-validation *At most 1 page.*									
Tech.	**Guiding questions**	**Pre-process**	**Parameters**	**Post-process & Pruning**	**Accuracy, Precision, Recall, ROC Area**	**Time to build model**	**Size of model**	**Analysis & observations about experiment, and interesting patterns in the trees**	**Comparison against GINI**
ZeroR	1	NA	Test Set	NA	Accuracy = 93.79, Recall = 1, Precision = 0.9379, ROC Area = 0.5	14 secs	depth = 0	ZeroR predicts the majority class always. Here since the data is dominated by one class the accuracy is over 93%	NA
OneR	1	NA	Test Set	NA	Accuracy = 94.57 Recall = 0.997 Precision = 0.948 ROC Area = 0.581	11 mins	depth = 2	In OneR, since each instance of the feature is compared with the target, the time taken to build the model is very large as compared to others	NA
DT	1	Label Encode the target	criterion='entropy', random_state=0	min_samples_split=60	Accuracy = 94.6 Recall = 0.24 Precision = 0.70 ROC Area = 0.62	4 sec	depth = 38	With min_sample_split = 60, we are able to obtain a model with a good precision and ROC Area. As we increase/ decrease the min_sample_split, these values keep on decreasing. Hence, 60 is the best possible value	Accuracy = 94.3 Depth = 20
DT	2	OneHotEncoding, LabelEncoder	criterion='entropy'	max_depth=20	Accuracy = 79.74 Recall = 0.43 Precision = 0.58 ROC Area = 0.67	2.5 sec	depth =20	With all the combinations of pre-pruning parameters, the value for recall and precision were low. Which implied that the attributes were not highly correlated to the target to predict the guiding question	Accuracy= 79.76 max_depth= 20
RF	2	OneHotEncoding, LabelEncoder	n_estimators=7, criterion = "entropy", random_state=0	min_sample_leaves=10	Accuracy = 79.25 Recall = 0.56 Precision = 0.45 ROC Area =0.67	8.2 sec	max depth= 28	In random forest, reducing the number of estimators from 10 to 7 reduces the time by nearly 3 seconds with very little change in accuracy and using min_sample_leaves prevents the model from overfitting to the data.	Accuracy= 79.25 max_depth= 27
RF	3	OneHotEncoding, LabelEncoder	n_estimators=10, criterion = "entropy", random_state=0	max_features=”auto”	Accuracy = 94.03 Recall = 0.16 Precision = 0.56 ROC Area = 0.57	22 sec	max depth = 31	By using max_features=auto, we limit the no.of features to be considered from n to sq-root(n). This makes the model run in the same time with 10 estimators which provides slightly better Recall and Precision values.	Accuracy = 94.03 max_depth = 30

[20 points] Summary of Python Classification Results, Analysis, Discussion, and Visualizations (at most 1 page) 1. Analyze the effect of varying parameters/experimental settings on the results. 2. Analyze the results from the point of view of the dataset domain, and discuss the answers that the experiments provided to your guiding questions. 3. Include (a part of) the best classification model obtained.

1. Based on the various parameters used, following was our observation:

a. Max depth: As we gradually increased the max depth, we reached a limit where the model gave the best results. Increasing this attribute after that led to overfitting of the data and hence, the model's precision, recall and AUC started falling.

b. Random State: Two possible values; 0, 1. With random state = 0, the results obtained after running the experiment 10 times were the same, whereas with random state = 1, the results varied by a marginal value of 0.1 - 0.2%

c. min samples leaf: With increasing the min sample leaf, it was observed that the model started predicting good results which further implied that there was a decrease in the variance.

d. n_estimators, max_features: As we increased the value of n_estimators, the model started performing better. With max_features = auto, each estimator randomly picks up the features and then returns the results. Both of the features were best suited to reduce overfitting of the model

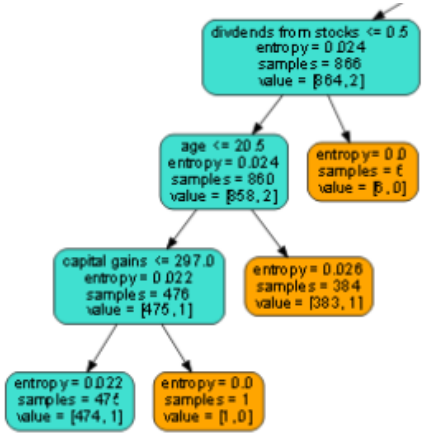
2. From the perspective of the domain, capital gains had a clear correlation with the target attribute ie., income.This is the reason for it being the attribute used by the oneR classification algorithm. In this dataset, the number of people with an income of more than 50000 was very sparse. Finding patterns among them was a challenging task in which capital gains helped in identifying the sparse entries. There were interesting instances that were encountered while analysing the relationship between education and income such as people with less than 1st grade in their education making more than 50000.

   Guiding Question 1: With the dataset, we decided to go with the attributes such as capital gains, dividends from stocks, weeks per year and age. The results obtained were indeed good, wherein not just the value of accuracy but the value of AUC was high too

   Guiding Question 2: On running experiments with attributes for this guiding question, we figured out that no matter what parameters we tried, the true negative and false positives values were high thus giving low recall and low precision values which implied they were not good predictors

   Guiding Question 3: Though for this the accuracy was high, the model could not predict the minority class and hence are not good predictors.

3.



[40 points] Summary of Regression Experiments. Use 10-fold cross-validation. *At most 1 page.*									
Tech.	**Guiding questions**	**Pre-process**	**Parameters**	**Post-process & Pruning**	**Correlation Coefficient and Error Metric(s)**	**Time to build model**	**Size of model**	**Salient observations about experiment; and Interesting patterns in the model**	**Root Mean Square Error**
DT	1	Scaling, OHE, Label Encoding	random_state = np.random	min_weight_fraction_leaf=0.05	Corr coef = 0.30 Mean Abs Error = 0.63	3 sec	3	With min_weight_fraction_leaf, the depth of the tree reduced drastically leading to having just a small model size	0.83
LR	1	Scaling the attributes, OneHotEncoding	normalize= False	NA	Correlation Coefficient = 0.59 Mean Absolute Error = 0.49	1 sec	-	Here, as we plot the equation of linear regression, we are able to find a correlation between the attributes and the target value i.e age	0.70
ZeroR	2	Scaling the attributes, OHE, Label Encoder	NA	NA	Mean Absolute Error = 0.83	94 sec	1	ZeroR always predicts the majority class,but for a regression problem it would be meaningless to output majority value and hence mean value.	0.99
DT	2	OHE, Label Encoder	random_state=0	max_leaf_nodes = 20	Correlation Coefficient = 0.70 Mean Absolute Error = 8.96	2 sec	7	As we increase the max_leaf_nodes, the depth of the tree increases and the absolute error improves. But the change in metrics is only until a specific point, after which, there are just marginal changes	12.12
MT	2	OneHotEncoder, LabelEncoder	random_state=0, min_sample_split=2	max_leaf_nodes =5	Mean Absolute Error = 21.04 (w/o scaling)	7 secs	5	Though model trees performs linear regression on leaf instead of mean, the error is high as we limit the no. of leaf nodes further down to 5	24.46
DT	3	Scaling the attributes, One Hot Encoding	random_state=1	min_samples_leaf=10	Correlation Coefficient = 0.60 Mean Absolute Error = 0.49	2 sec	14	As we change increase the min_sample_leaf, it is observed that the Correlation Coefficient, Mean absolute error and Mean Square error do not change. It is just the depth of the tree that decreases.	0.62

[20 points] Summary of Python Regression Results, Analysis, Discussion, and Visualizations (at most 1 page) 1. Analyze the effect of varying parameters/experimental settings on the results. 2. Analyze the results from the point of view of the dataset domain, and discuss the answers that the experiments provided to your guiding questions. 3. Include (a part of) the best regression model obtained.

1. Based on the various parameters used, following was our observation:

a. Random State: Unlike decision trees, if we apply a random int value to the Random State, the results do not change and always produce the same output

b. max_leaf_nodes: This parameter decides the depth of the tree. Suppose for two experiments, if we have values that are close like say for exp 1 it is 7 and for exp 2 it is 10, the depth of the tree will still remain the same. Hence, to change the depth, a sufficient number should be inputted to observe the changes in the results

c. min_sample_leaf: This is the minimum number of samples required at the leaf nodes. As we increase the min_sample_leaf value, the decision tree starts underfitting and hence the absolute error for the tree starts increasing.

d. min_weight_fraction_leaf: As we increased the value of min_weight_fraction_leaf, the depth of the tree started decreasing

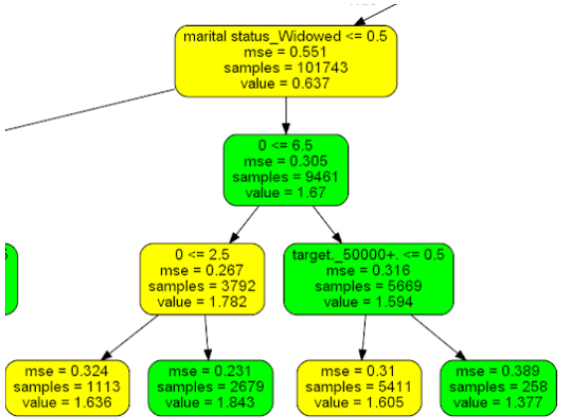
2. By analysing the data, by common sense we felt that education and marital status could be good predictors of an age of a person which we framed as one of the guiding questions along with sex. This turned out to make very good predictions on unseen data. From the domain point of view, we felt knowing the relation between tax filer status and age would be useful which led us to another guiding question involving capital gains and tax filer status. Though the relation between education,marital status and age seemed straight forward, there was an interesting relationship between age and capital loss.

Guiding Question 1: The results obtained for this experiment were good if not that great. The mean absolute error for this guiding question was high, thus implying that even though capital loss and tax filer status seemed correlated to age, they weren't the good predictors

Guiding Question 2: This experiment was performed by comparing the results of model trees and regression trees. Average prediction error was around 8. But as we limit the number of leaf nodes for the model trees the predictions started becoming less accurate.

Guiding Question 3: For this experiment, we did not scale the attributes and obtained good results. The mean absolute error wasn't bad implying that is closely predicted age of people whose income was less than 50k

3.



Advanced Topic : Instability of Decision Trees

[7 points] List of sources/books/papers used for this topic (include URLs if available). *Provide full references (authors, title, where published, year, …).*

- Ruey-Hsia Li and Geneva G.Belford,Instability of decision tree classification algorithms,ACM SIGKDD international conference on Knowledge discovery and data mining, July2002(https://dl-acm-org.ezproxy.wpi.edu/citation.cfm?id=775131)
- Leo Breiman,Bagging Predictors,Springer,August 1996, Volume 24, Issue 2, pp 123–140(https://link.springer.com/article/10.1007/BF00058655)
- Anuja Nagpal, Decision Tree Ensembles-Bagging and Boosting,TowardsDataScience blog Oct2017(https://towardsdatascience.com/decision-tree-ensembles-bagging-and-boosting-266a8ba60fd9)

[20 points] In your own words, provide an in-depth, yet concise, description of your chosen topic. Make sure to cover all relevant data mining aspects of your topic. *Your description here should be comprehensive and in-depth (it should reflect work at the graduate level).*

The decision tree classifiers are highly sensitive to the variation in input data, any modifications made on the input data have a great impact on the decision boundaries. This instability is mainly due to the instability in choosing the splitting criterion. The rules generated vary to small change in the data. We modify the data before building the model, the predictions made by this model are not perfectly in-relation with the original data. This instability in decision trees can be tackled by bagging and boosting which are based on the concept that several weak learners combine together to form a strong learner. Bagging a powerful method referred to as *bootstrap aggregation*, is based on the premise that, if the underlying predictor is unstable, then aggregating the predictor over multiple bootstrap samples will produce a more accurate, and more stable, procedure.This is used when the goal is to reduce the variance in the decision tree generated.Calculate fˆ1 (x), fˆ2 (x), . . . , fˆB (x) using B separate training sets and average them in order to obtain a single low-variance statistical learning model: fˆ ave(x) 1 B X B b=1 fˆb (x).Having different sets may not be possible so we bootstrap the existing dataset. Hence bagging achieves higher accuracy and less sensitivity to change in data at the expense of interpretability,it is only possible to get an overall summary of the importance of each predictor. The random forest algorithm is an extension of this concept where in addition to taking random subset of data,random subset of features are used to grow the tree rather than using all the features . Boosting provides better models through a combination of reweighting and aggregation. In this method the weights of the instances that were incorrectly predicted are increased and weights of the correct predictions decreased accordingly. This process is done on different samples with modified weights and finally an aggregated classifier is formed. By this assigning and reassigning weights we make the model train on the observations that are difficult to classify. But this comes at an expense too. If we do not carefully tune the hyper-parameter(the number of trees, the shrinking parameter and the number splits in each tree) there is a chance of over-fitting to the data on which it is being trained. The gradient boosting is an extension of this concept involving gradient descent and boosting.

[3 points] Describe how this topic relates to trees and the material covered in this course.

In this course we apply classification techniques such as Decision Trees and Random Forest. Since Bagging and Boosting is one of the concepts used by Random Forest to increase the stability of the trees, this topic is highly related to the trees and the materials covered.

Authorship: The initial Data Exploration was done by Sri whereas visualizing the dataset to obtain correlation was done by Bhoomi. Both of the members came up with different guiding questions and both built their code, ran experiments and then based on the results, merged the code to obtain the best output for the guiding questions. Mutually decided which would be the best part of the model to incorporate in the report and it was RandomForest concept that motivated Bhoomi to choose the advanced topic. Based on the topic, Sri came up with the concept of Bagging and Boosting.