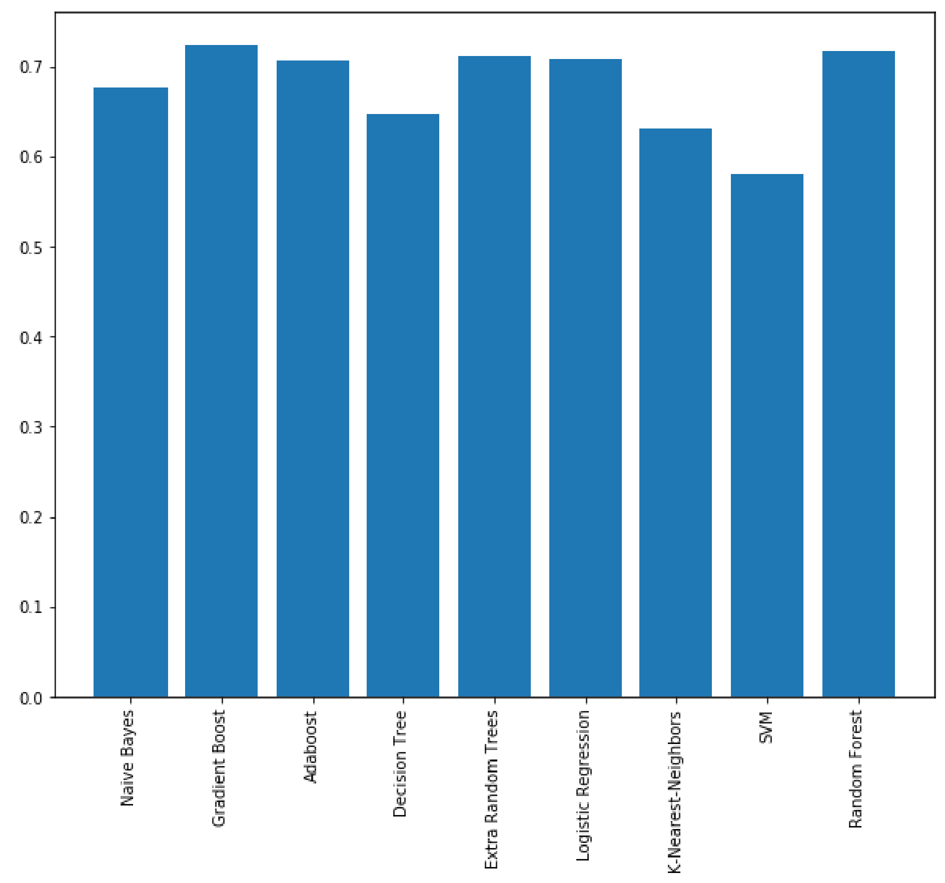
CS5011 – Assignment
June 2019

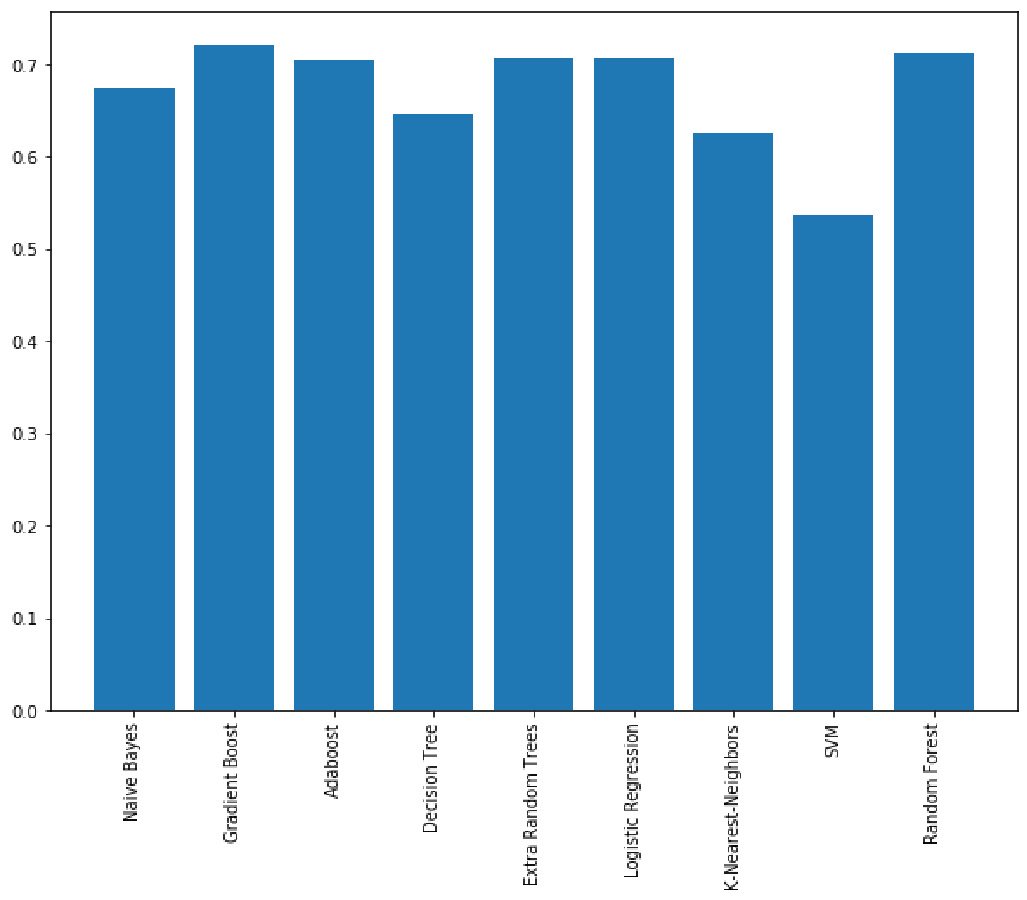
Introduction: The Knowledge Discovery and Data Mining (KDD) Cup in 2004 posed a binary classification problem for a particle generated inside a collider. There were a total of 78 attributes and training set of 50,000. The test data set had 100,000 samples. In this report we generated the best classifier in terms of the following four metrics (accuracy, area under curve, log loss, and SLQ). After doing the standard preprocessing to handle missing data points we also tuned the hyperparameters for each of the classifiers generated and submitted our final labels for the test data on the online platform.

Implementation:
Preprocessing: It was already mentioned in the data description that there were missing data in the dataset. This was a more or less **balanced classification task**. Missing values in columns 22,23,24 and 46,47,48 were indicated using a value of "999" to denote "not available", and columns 31 and 57 use "9999" to denote "not available". These are the column numbers in the data tables starting with 1 for the first column (the case ID numbers). The columns were not named in this competition so as not to give any undue advantage to people who had a working knowledge in particle physics. We used **mean** imputation technique (replacing NA values with the column means) to impute missing data in the dataset. That is the only preprocessing task we did.

Classifiers: After splitting the dataset into 66-33 train-test ratio, we started applying different classifiers. The cross-validated and test accuracy of each of the classifiers is listed below:
[Naive Bayes, Gradient Boost, Adaboost, Decision trees, Extra Randomtrees, Logistic regression, KNN, SVM, Random forest]



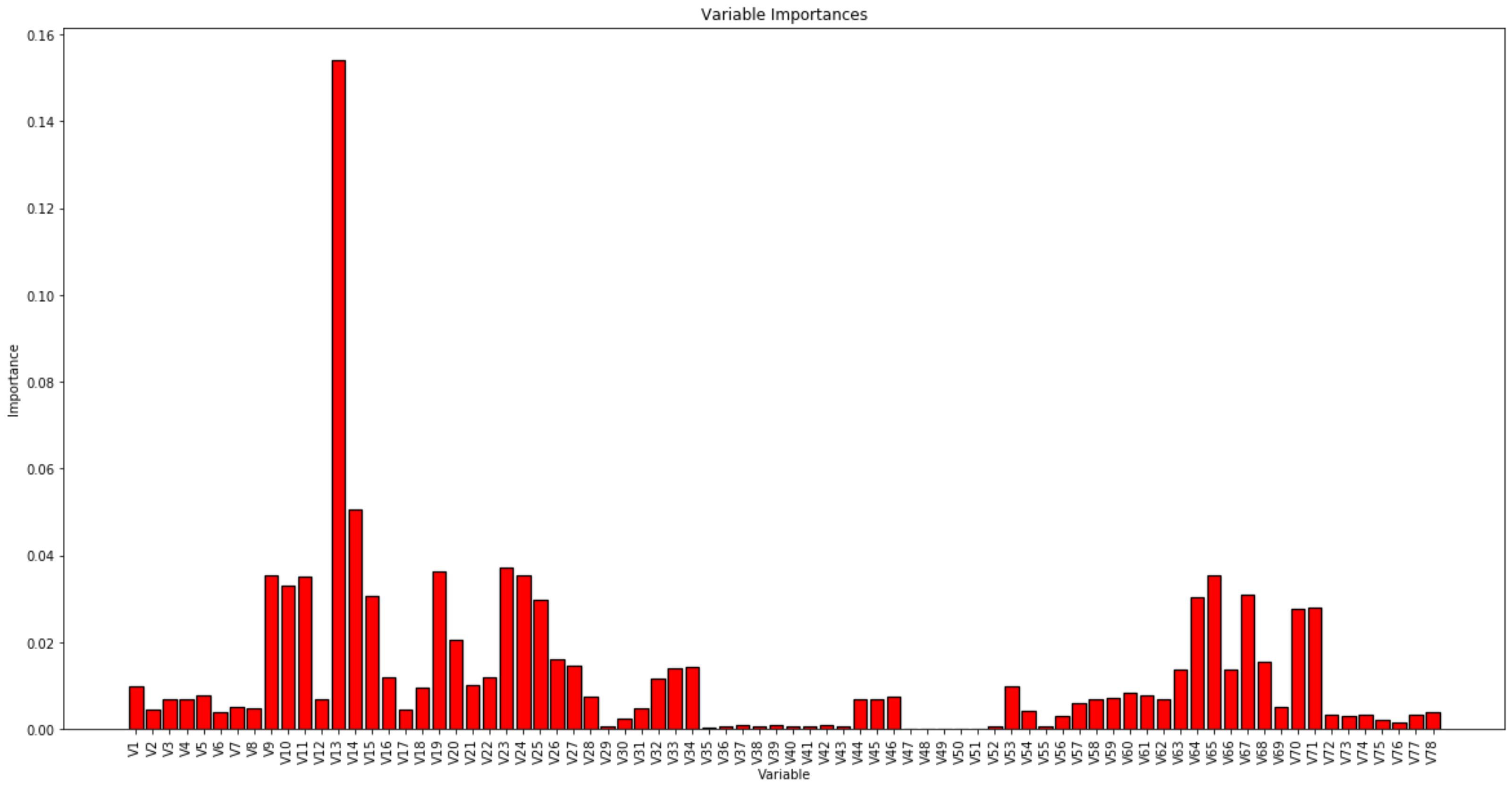
5 fold CV scores



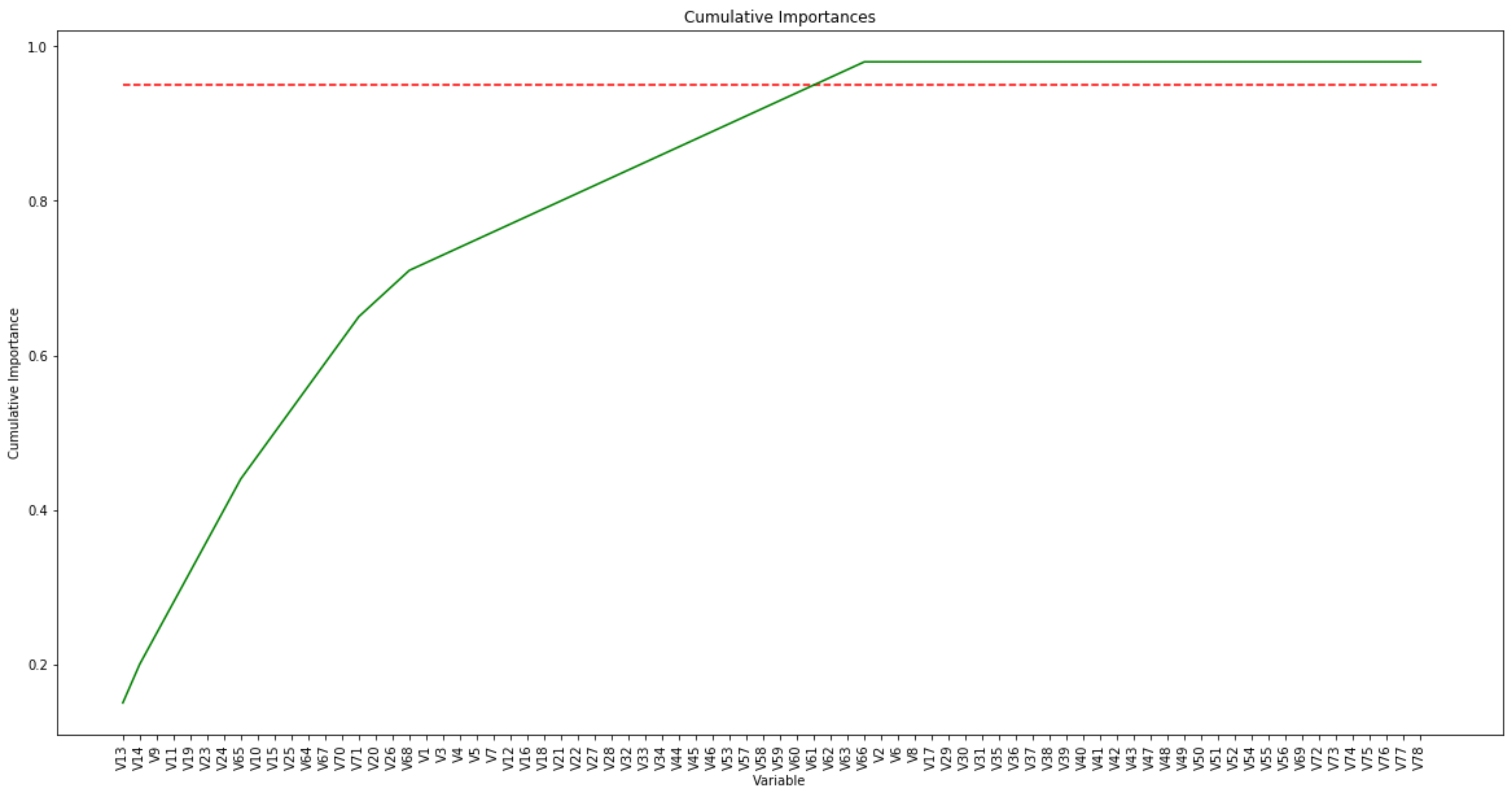
Test Accuracy

Feature selection:

Since there are 78 features and domain knowledge is not available, we have to do some sort of feature selection to reduce the complexity of our model. We use the feature selection tool built on the RandomForestclassifier in sklearn to implement this. Following is the result:



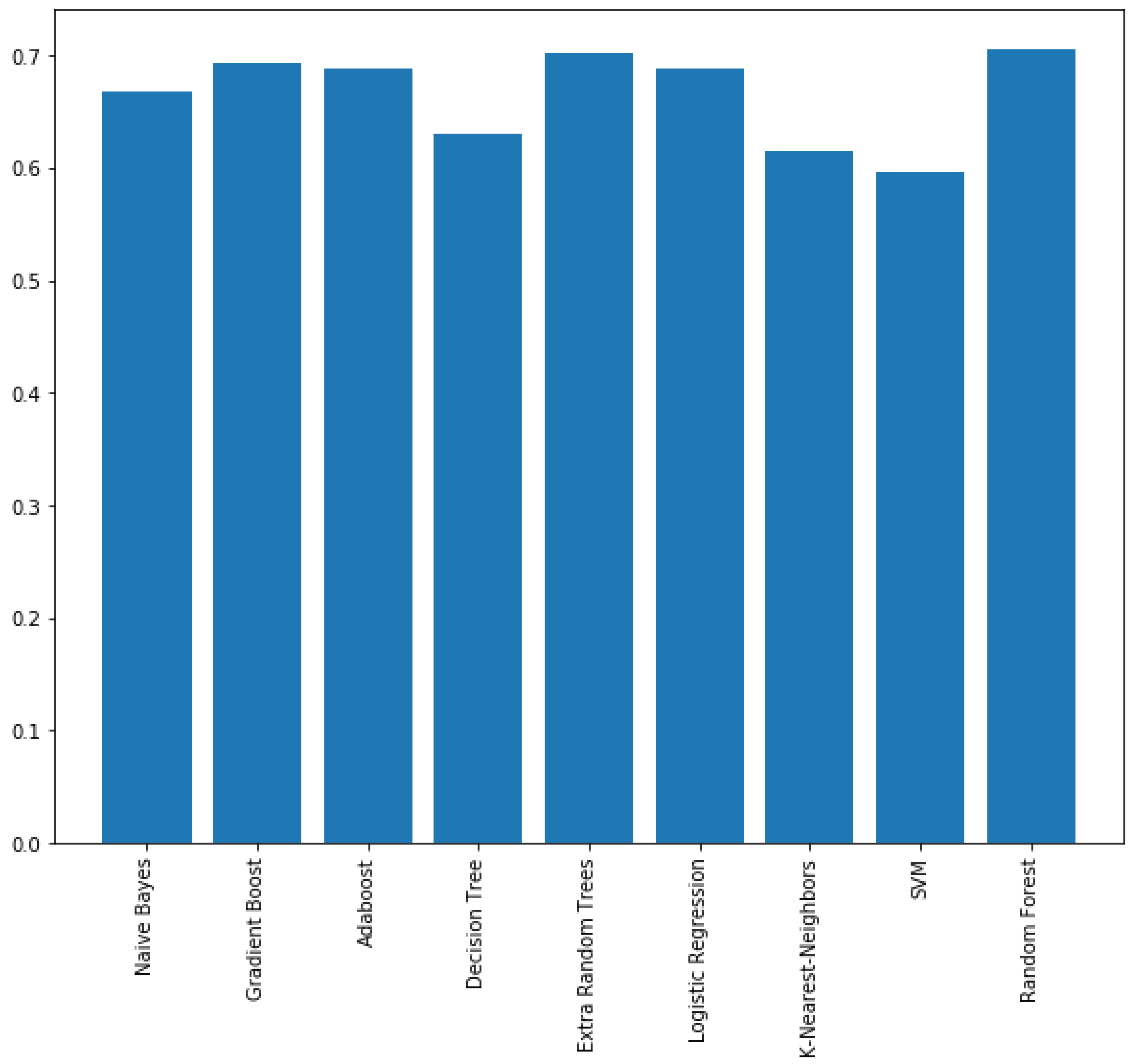
So we can easily see that there are some features with almost zero importance. We remove those and build the classifiers again. The first step in achieving that is to make a cumulative importance graph that shows the contribution to the overall importance of each additional variable. The dashed line is drawn at 95% of total importance accounted for.



Cumulative feature importance graph

‘

Now we use this to remove unnecessary features. 95% is the cutoff and we find the exact number of features above that threshold and consider them high importance features. There were around **42** such features. We effectively reduced the number of features from **78** to **42.** Now we build our models using these features and if there’s a noticeably poor performance, we can always adjust the value of the threshold.



Test Accuracy after feature selection

In the above plot the performance is more or less the same if we do feature selection and that goes on the show that there were indeed a lot of redundant features **(36 to be exact)**. From now on, we will build all our classifiers or ensembles of them using these most important features only.

Ensemble learning: Take the top performing individual classifiers and build an ensemble model. But before that we had to make sure that we have tuned the hyperparameters to our model and we are working with the best classifiers possible

1. **Hyperparameter tuning:** The next step was to take the top 5 classifiers and tune the parameters. We would use the GridsearchCV() of sklearn to do that job. Below are the test accuracies of our tuned classifiers.

Classifier	Best parameters (using sklearn GridsearchCV, cv=3)
Random Forest	'max_depth': 80, 'max_features': 3, 'min_samples_leaf': 3, 'min_samples_split': 8, 'n_estimators': 300
Gradient boosting	'max_depth': 8, 'n_estimators': 400
Adaboost	'base_estimator_criterion': 'entropy', 'base_estimator__splitter': 'best', 'n_estimators': 1
ExtraRandomTrees	'max_depth': 110, 'max_features': 3, 'min_samples_leaf': 3, 'min_samples_split': 12, 'n_estimators': 500
Logistic Regression	C=0.1

2. **Stacking tuned classifiers:** Next we tried combining the classification results for the top classifiers using majority voting rule:
   a. Adaboost + GradientBoost + RandomForests
   b. Random Forests+ Logistic Regression+ GradientBoost+ExtraTrees

We tested the ensemble learners using the unseen test data available and uploaded our predictions to the online platform. **The second ensemble made of Randomforests, logit, gradient boost and Extra trees gave the best results. It was even better than each of the individual classifiers.**
The performances of all the classifiers built up to now were compared by uploading their predictions on the unseen data and checking the leaderboard positions.

Discussion: We built our ensemble classifiers on the best feature subset and using the best parameters for each of the models**.** The fact that we achieved reasonable accuracy shows that feature selection step was extremely crucial as it led to a more generalizable model. One very important observation from this competition is the fact that the leaderboard standings are based on four different metrics: Accuracy, AUC, cross entropy and SLQ score. Optimising our models separately for each of the metrics would have definitely fetched better results. Methods for selecting and combining variations of ensemble models has helped us understand how to most effectively approach complex machine learning problems in our future research endeavours.

Final leaderboard rank for username **shayantan**: 87.750 (10 submissions in total) (Link)