CS4011 Contest Report

S Aakash CS15B060-S Ram Ananth ME15B153

November 12, 2017

0.1 Data Imputation

The first step was to fill in the missing data.We created 4 datasets: (i)overall mean imputation(in both training and test set):take the mean of each known values of features in dataset and use it to fill the missing values.
(ii)Per class mean imputation(in training).We felt it makes more sense to take the mean of those data points that belong to a particular class that taking overall mean.
(iii)Overall median imputation:Same as (i),but instead of mean,take median.
(iv)Per class median imputation:Similar to (ii)
(v)Per class mode imputation:(similar to ii)
So depending upon the method,we shifted between datasets.Eg.XGBoost,infact doesn't require any imputation.If any feature value is missing,it creates a new feature-'missing' to handle,and the manual says it performs better than any other imputation.Because the values were between 0 and 1,the mean and median didn't provide much difference in terms of performance.

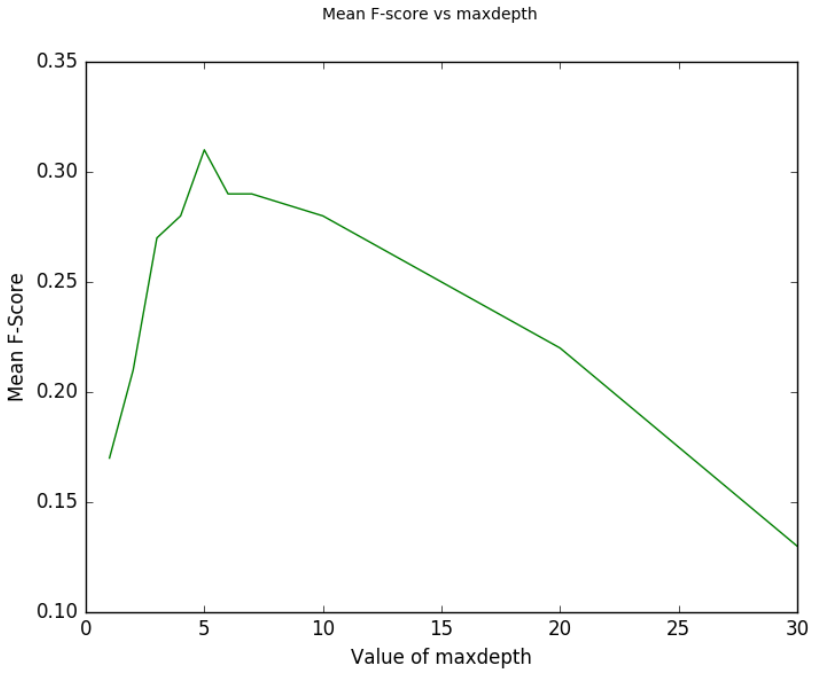
0.2 Preprocessing and visualization

We observed that all features were lying between 0 and 1,so we felt no need to perform any feature scaling(Infact we did first,but performance was slightly low).
Next,we performed Principal Component analysis(PCA).We observed the variances of each feature and found out that out of 2600 features,only about 400 seemed to be really important.
We found out the feature with maximum variance(say λ) and chose the ones whose variance were $\geq 0.005\times \lambda$(ie the ones whose variance is atleast 0.005 of the one with maximum variance).We obtained approximately 460 features and created a new dataset.(Note that PCA was done using overall median imputed data and we transformed test set also using PCA.)
But unfortunately,in some cases like Random-Forests,SVM PCA didn't help much.Maybe PCA eliminated very fine details that these classifiers depend on.So we were forced to use the original median imputed dataset.But PCA resulted a very good performance with XGBoost and was used here.
We didn't randomly come up with the threshold 0.005 ,we ran a grid search,with our XGBoost ie. the threshold was considered as a hyperparameter with XGBoost inherently.

0.3 Classifiers

0.3.1 XGBoost Classifier

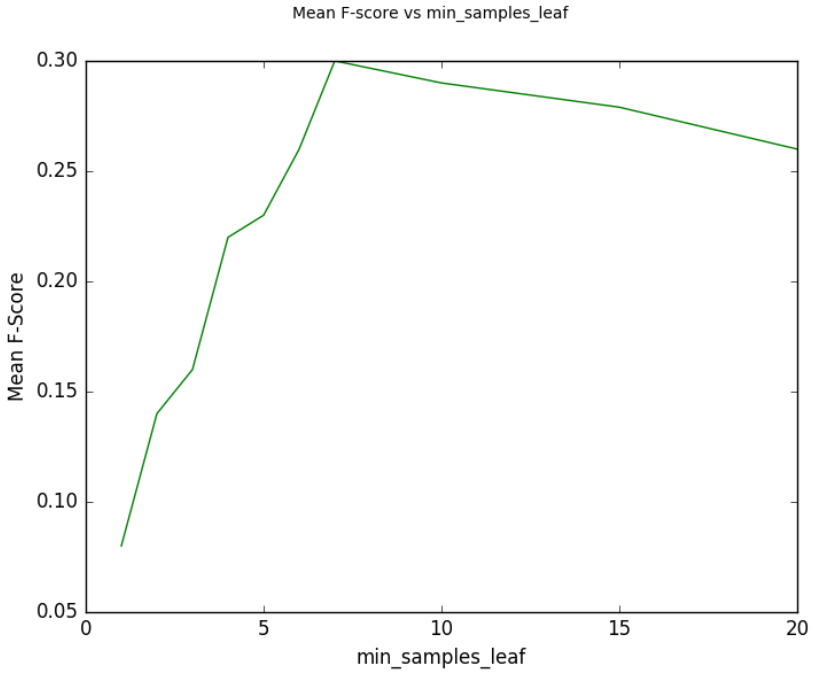
We started off with XGBoost Classifier,from XGBoost package(we didn't start with sklearn's XGBoost first).We used grid search to turn the parameters.
One of the most important parameters in XGBoost is the maxdepth.A larger value of maxdepth means too much overfitting,whereas a smaller one will result in high bias.So tuning this was the most important important thing and we obtained the following:

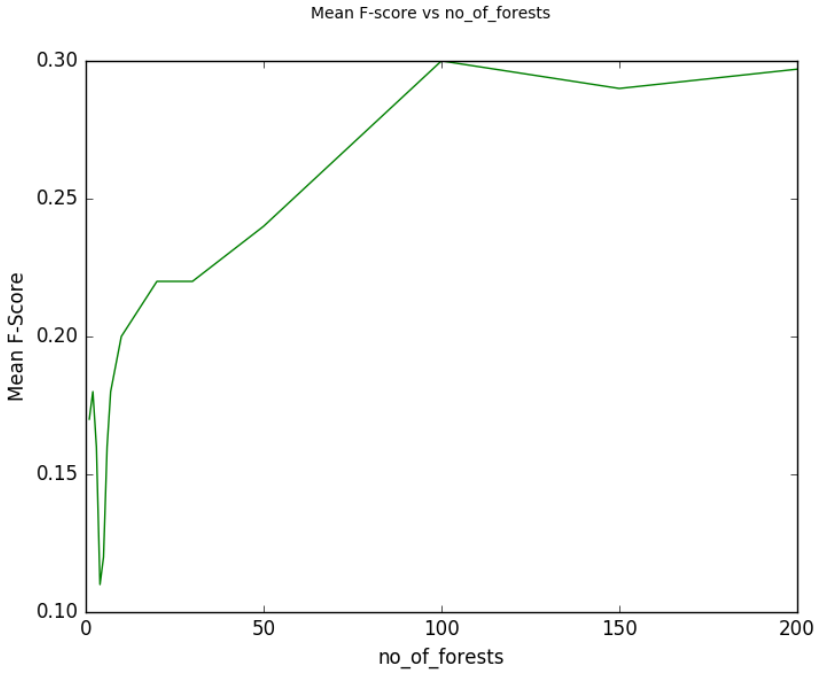


The fscore was an average of 5-fold Cross validation on training set.
We submitted the best one(on kaggle) which resulted in 0.30 as the mean F-Score.
Then we used Sklearn's XGBoost classifier,which was quite similar.But however it gave slightly lower performance than XGBoost's classifier.XGBoost's XGB-Classifier also had other hyperparameters like L1-reg. alpha etc. that helped us control overfitting.But the main thing we observed is that tuning the max depth matters the most.
But it was felt that XGBoost has more potential than 0.30,so we performed Bayesian Optimisation,understanding what it is and implementing it. We obtained F-Score:0.325

0.3.2 Random Forest

After performing boosting,next obvious thing was trying out random forests.Unfortunately,it didn't give as a better performance than XGBoost classifier.We varied a lot of parameters .N-estimator,minimum-no-of-samples-in-leaf and maxdepth were found to be most important parameters to tune.here were our results:

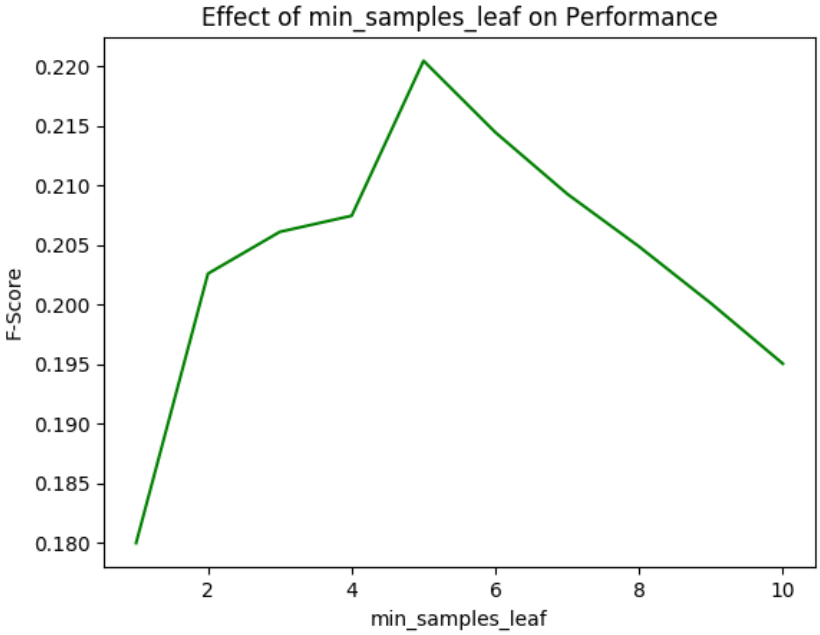


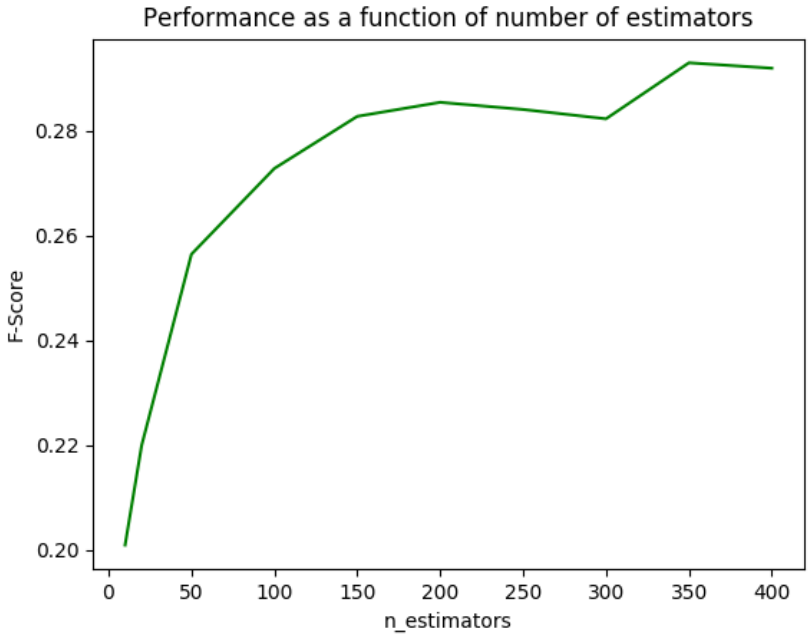


Note that it is not that we tuned the parameters seperately.We actually performed grid search and for the sake of illustration using graph,we fixed all parameters and varied only the parameter that was plotted in graph.
Best F-score submitted using random forests:0.30

0.3.3 Extra Trees Classifier

After performing random forests we came upon a classifier called ExtraTreesClassifier in ensemble methods of sklearn that is a meta estimator that fits a number of randomized decision trees (called extra-trees) on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control overfitting. The hyperparameters such as number of estimators and min samples in a leaf were varied using grid search and performance noted.


Effect of min_samples_leaf on Performance
F-Score
0.220
0.215
0.210
0.205
0.200
0.195
0.190
0.185
0.180
2
4
6
8
10
min_samples_leaf

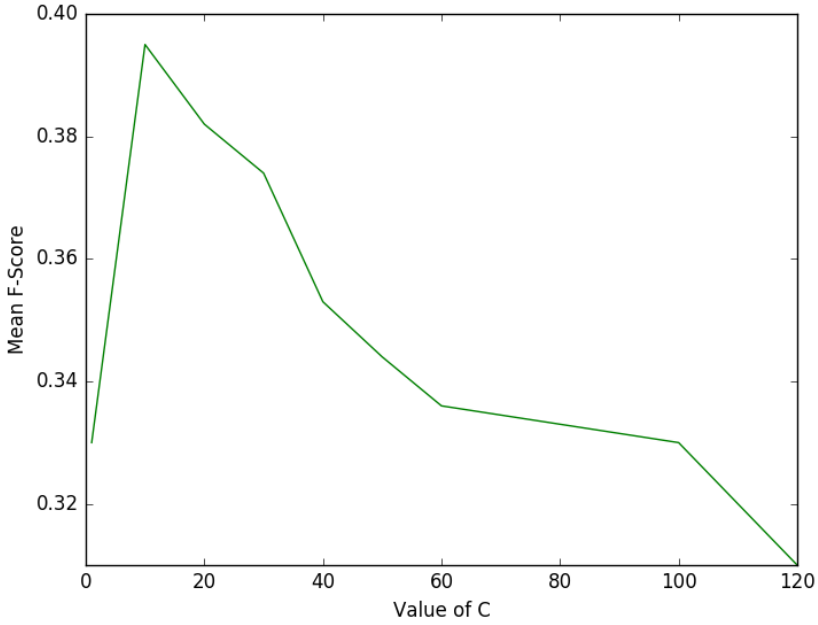

Performance as a function of number of estimators
F-Score
0.28
0.26
0.24
0.22
0.20
0
50
100
150
200
250
300
350
400
n_estimators

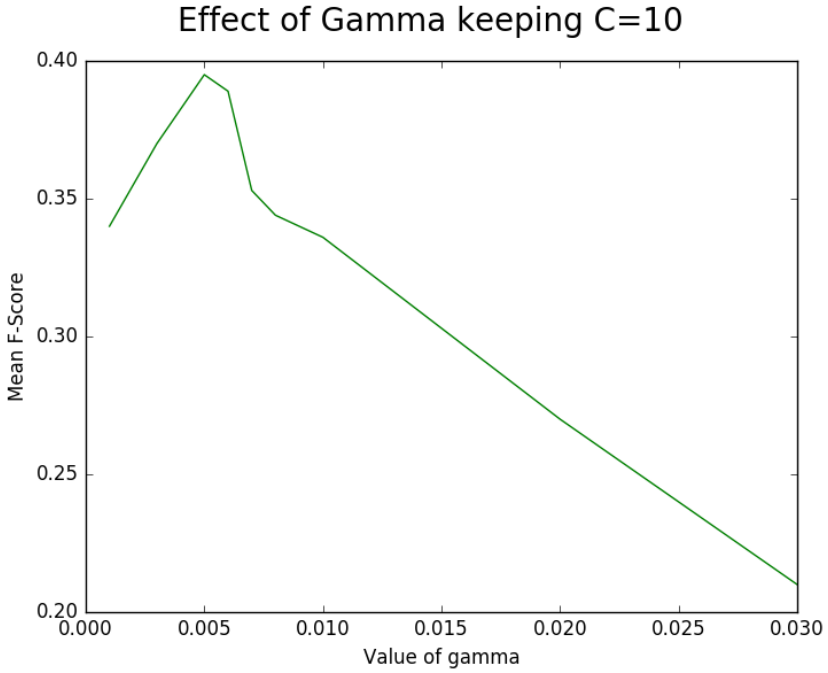
0.3.4 Support Vector Machines-Linear Kernel

We then explored with SVM,using linear kernel.We used several variants like SVC,LinearSVM,libSVM etc,performed with PCA,without PCA, but the best F-Score we could obtain was 0.31.This made as feel we need Basis expansions to increase F-Score and it opend us to SVM using RBF kernel.

0.3.5 Support Vector Machines-Guassian Kernel

We know that C,gamma were the determining hyperparameters for using SVM with RBF kernel.We observed that the dataset was very sensitive to the value of gamma!.A decrease in gamma by 5% caused quite an observable in F-score.Here also we ran a grid search with C,gamma





Another interesting thing we observed was the effect of One-vs-One and One-vs-Rest.One-vs-Rest performed significantly performed better than One-Vs-One in this case.
Since we found out that gamma was very sensitive,we performed 2-steps of grid search for gamma. (i)Region search: Identify which region(range) of gamma is likely to provide a good performance.Gamma was increased logarithmically.
(ii)Perform region search again on this smaller region,with smaller factors of range and continue this to narrow down the range of gamma,till we reach the optimal gamma.In this way,optimal gamma was found to be 0.0058.
For estimating C,a normal grid search was done.We used median imputed dataset for this case
The performance with C=10,gamma=0.058 was 0.395(mean F-Score).

0.3.6 Majority vote with top three

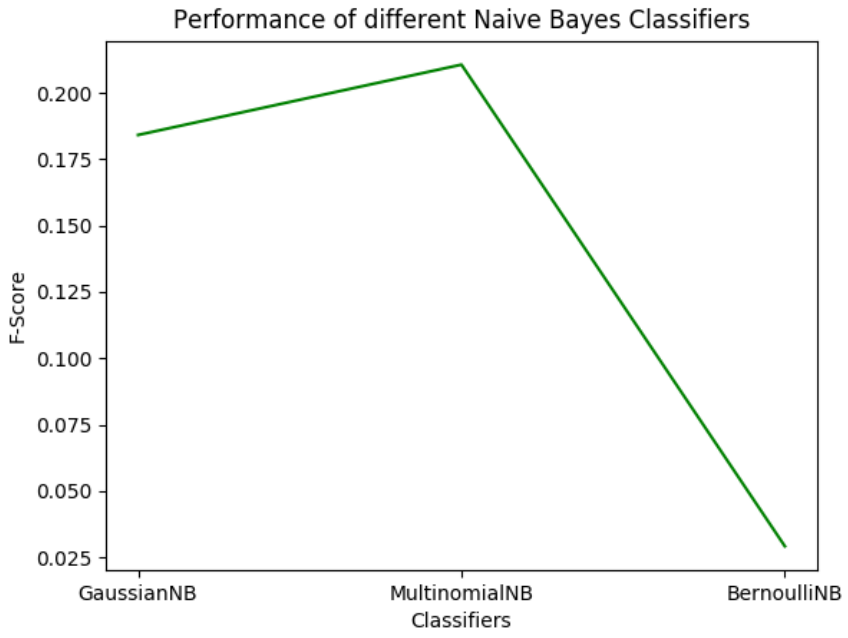
We decided to perform majority voting with the top 3 classifiers we obtained sofar-SVM with guassian kernel,XGBoost,Random Forests.We gave a voting weight proportional to their F-scores whent they were run individually(0.395-for SVM,0.32-XgBoost,0.30-for random forest) and performed majority vote using these 3 classifiers.Interestingly we obtained F-Score-0.389 slightly lower than SVM when ran individually.We felt that the weights made good sense of what is their their trust in majority voting and therefor a good heuristic.We are yet to figure out the real reason for decrease in F-scores.

0.3.7 Adaboost with Random Forest as base estimators

We decided to combine bagging with boosting(sounds wierd),just with a hope that we would land up in between under-fitting and overfitting.It actually increased to F-Score of random forests slightly,from 0.30 to 0.309 after which it started to saturate.

0.3.8 NaiveBayes

We then decided to try out Naive Bayes classifiers because we know that Naive Bayes Classifiers are quite successful when it comes to data with a large feature space such as text. We tried all the available Naive Bayes classifiers in sklearn package with their default values and the results obtained(as shown in Fig below) were not up to the mark.



0.3.9 Stacking

We finally tried stacking of algorithms. We implemented a simple stacking algorithm that stacks 4 of our best performing classifiers(XGBoost, RandomForest

Classifier,ExtraTrees Classifier and Support Vector Machine) and uses the predictions from these classifiers as an input to another classifier (Decision Tree Classifier) which then makes our final prediction.