Prudential Insurance Data Science Report

Project Description

The prudential data set is taken from the current Kaggle Competition. The detailed description about the data set can be found at https://www.kaggle.com/c/prudential-life-insurance-assessment

Features Description

Data set consists of 127 features and 59381 observations. The features are divided into broadly 4 categories namely, classification, discrete, dummy and continuous features. Response feature is the feature to be predicted and Id gives a unique id to each observation.

Below is a small description of the kinds of variables

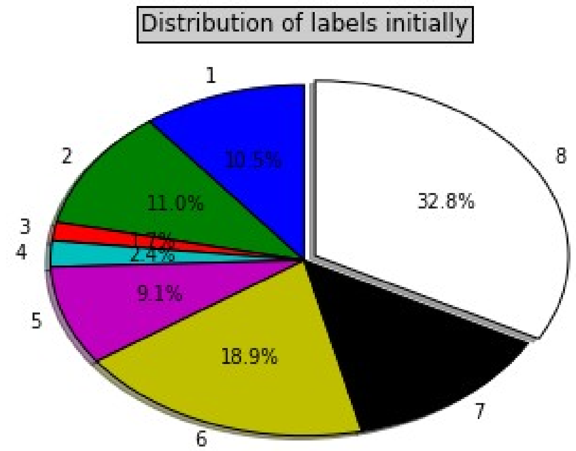
Features	Description
Product_Info_1-7	A set of normalized variables relating to the product applied for
Ins_Age	Normalized age of applicant
Ht	Normalized height of applicant
Bt	Normalized weight of applicant
BMI	Normalized BMI of applicant
Employment_Info_1-6	A set of normalized variables relating to the employment history of the applicant.
Insured Info_1-6	A set of normalized variables providing information about the applicant.
Insurance_History_1-9	A set of normalized variables relating to the insurance history of the applicant.
Family_Hist_1-5	A set of normalized variables relating to the family history of the applicant.
Medical_History_1-41	A set of normalized variables relating to the medical history of the applicant.
Medical_Keyword_1-48	A set of dummy variables relating to the presence of/absence of a medical keyword being associated with the application.
Response	This is the target variable, an ordinal variable relating to the final decision associated with an application

Description of features of the DataSet

Preliminary Data Exploration

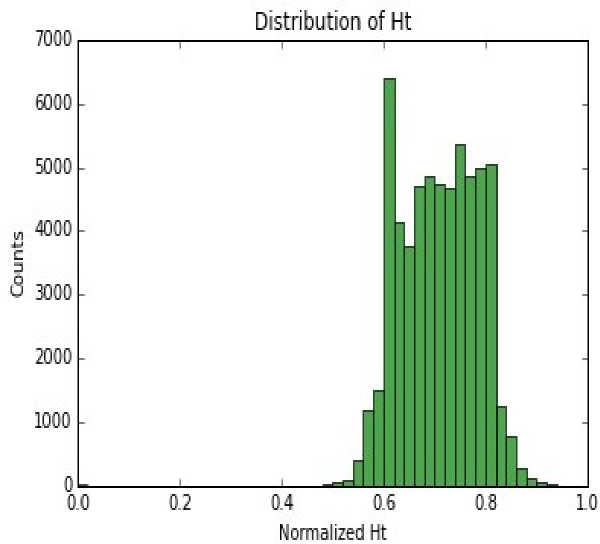
1. Label/Target Feature Description

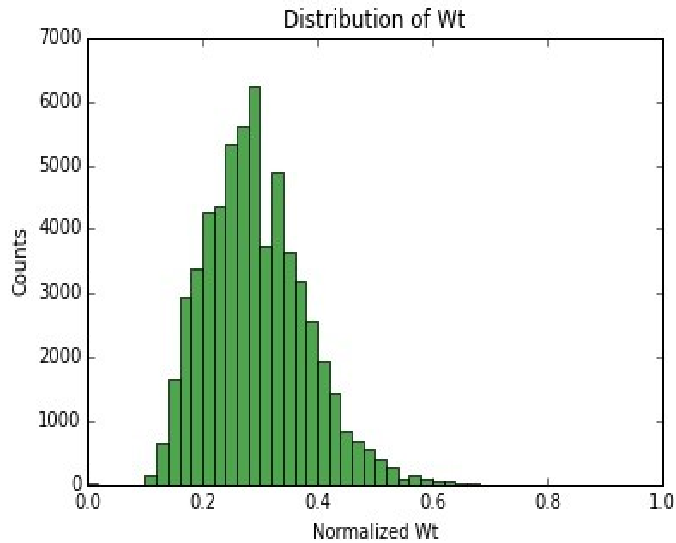
Response variable has 8 categories. Distribution of response variable in each of these categories is show below in the pie chart.

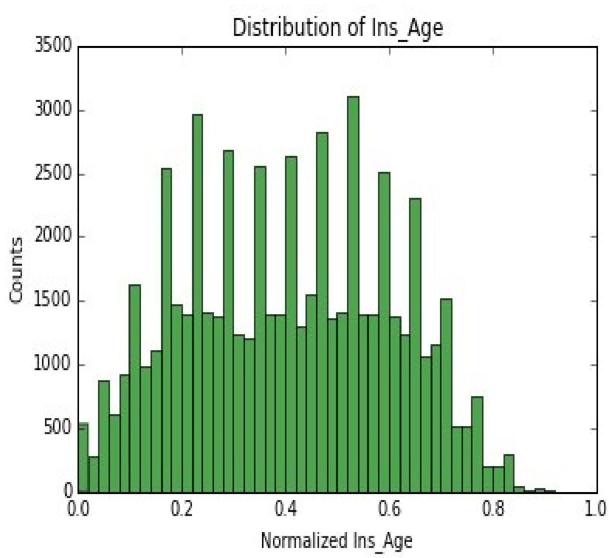


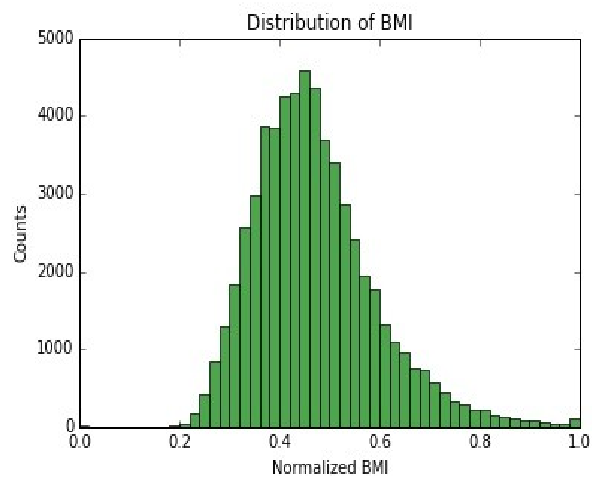
As can be seen in the pie chart representation above distribution amongst the labels is not uniform. Label 8 covers roughly 33% of the distribution whereas label 3 has only 1% of the distribution. We will expect our model to capture a similar trend during the cross validation. However, exact capturing of the trend may lead to over fitting as can be seen in the descriptions of methods below where the method kind of over-fits with the decision tree and thus gives a lower score than gradient boosting.

2. Exploring Height,Weight,BMI and Insurance Age









The above plots show the distribution of the features normalized height, weight, BMI and Insurance Age across the data set. As can be seen from the figure apart from Insurance Age all other features follow a normal distribution

3. Description of Product_Info Variables

	Product_Info_1	Product_Info_3	Product_Info_5	Product_Info_6
count	59381.000000	59381.000000	59381.000000	59381.000000
mean	1.026355	24.415655	2.006955	2.673599
std	0.160191	5.072885	0.083107	0.739103
min	1.000000	1.000000	2.000000	1.000000
25%	1.000000	26.000000	2.000000	3.000000
50%	1.000000	26.000000	2.000000	3.000000
75%	1.000000	26.000000	2.000000	3.000000
max	2.000000	38.000000	3.000000	3.000000

	Product_Info_7
count	59381.000000
mean	1.043583
std	0.291949
min	1.000000
25%	1.000000
50%	1.000000
75%	1.000000
max	3.000000

4. Description of Employment Info Variables

	Employment_Info_1	Employment_Info_2	Employment_Info_3
count	59381.000000	59381.000000	59381.000000
mean	8.641821	1.300904	2.142958
std	4.227082	0.715034	0.350033
min	1.000000	1.000000	2.000000
25%	9.000000	1.000000	2.000000
50%	9.000000	1.000000	2.000000
75%	9.000000	1.000000	2.000000
max	38.000000	3.000000	3.000000

5. Description of Insured Info Variables

	Insured_Info_1	Insured_Info_2	Insured_Info_3	Insured_Info_4
count	59381.000000	59381.000000	59381.000000	59381.000000
mean	1.209326	2.007427	5.835840	2.883666
std	0.417939	0.085858	2.674536	0.320627
min	1.000000	2.000000	1.000000	2.000000
25%	1.000000	2.000000	3.000000	3.000000
50%	1.000000	2.000000	6.000000	3.000000
75%	1.000000	2.000000	8.000000	3.000000
max	3.000000	3.000000	11.000000	3.000000

	Insured_Info_5	Insured_Info_6	Insured_Info_7
count	59381.000000	59381.000000	59381.000000
mean	1.027180	1.409188	1.038531
std	0.231566	0.491688	0.274915
min	1.000000	1.000000	1.000000
25%	1.000000	1.000000	1.000000
50%	1.000000	1.000000	1.000000
75%	1.000000	2.000000	1.000000
max	3.000000	2.000000	3.000000

6. Description of Insurance History Variables

	Insurance_History_1	Insurance_History_2	Insurance_History_3
count	59381.000000	59381.000000	59381.000000
mean	1.727606	1.055792	2.146983
std	0.445195	0.329328	0.989139
min	1.000000	1.000000	1.000000
25%	1.000000	1.000000	1.000000
50%	2.000000	1.000000	3.000000
75%	2.000000	1.000000	3.000000
max	2.000000	3.000000	3.000000

	Insurance_History_4	Insurance_History_7	Insurance_History_8
count	59381.000000	59381.000000	59381.000000
mean	1.958707	1.901989	2.048484
std	0.945739	0.971223	0.755149
min	1.000000	1.000000	1.000000
25%	1.000000	1.000000	1.000000
50%	2.000000	1.000000	2.000000
75%	3.000000	3.000000	3.000000
max	3.000000	3.000000	3.000000

	Insurance_History_9
count	59381.000000
mean	2.419360
std	0.509577
min	1.000000
25%	2.000000
50%	2.000000
75%	3.000000
max	3.000000

7. Description of Medical History Variables

	Medical_History_2	Medical_History_3	Medical_History_4
count	59381.000000	59381.000000	59381.000000
mean	253.987100	2.102171	1.654873
std	178.621154	0.303098	0.475414
min	1.000000	1.000000	1.000000
25%	112.000000	2.000000	1.000000
50%	162.000000	2.000000	2.000000
75%	418.000000	2.000000	2.000000
max	648.000000	3.000000	2.000000

	Medical_History_5	Medical_History_6	Medical_History_7
count	59381.000000	59381.000000	59381.000000
mean	1.007359	2.889897	2.012277
std	0.085864	0.456128	0.172360
min	1.000000	1.000000	1.000000
25%	1.000000	3.000000	2.000000
50%	1.000000	3.000000	2.000000
75%	1.000000	3.000000	2.000000
max	3.000000	3.000000	3.000000

	Medical_History_8	Medical_History_9	Medical_History_10
count	59381.000000	59381.000000	557.000000
mean	2.044088	1.769943	141.118492
std	0.291353	0.421032	107.759559
min	1.000000	1.000000	0.000000
25%	2.000000	2.000000	8.000000
50%	2.000000	2.000000	229.000000
75%	2.000000	2.000000	240.000000
max	3.000000	3.000000	240.000000

	Medical_History_11	...	Medical_History_31 \
count	59381.000000	...	59381.000000
mean	2.993836	...	2.985265
std	0.095340	...	0.170989
min	1.000000	...	1.000000
25%	3.000000	...	3.000000
50%	3.000000	...	3.000000
75%	3.000000	...	3.000000
max	3.000000	...	3.000000

	Medical_History_33	Medical_History_34	Medical_History_35 \
count	59381.000000	59381.000000	59381.000000
mean	2.804618	2.689076	1.002055
std	0.593798	0.724661	0.063806
min	1.000000	1.000000	1.000000
25%	3.000000	3.000000	1.000000
50%	3.000000	3.000000	1.000000
75%	3.000000	3.000000	1.000000
max	3.000000	3.000000	3.000000

	Medical_History_36	Medical_History_37	Medical_History_38 \
count	59381.000000	59381.000000	59381.000000
mean	2.179468	1.938398	1.004850
std	0.412633	0.240574	0.069474
min	1.000000	1.000000	1.000000
25%	2.000000	2.000000	1.000000
50%	2.000000	2.000000	1.000000
75%	2.000000	2.000000	1.000000
max	3.000000	3.000000	2.000000

	Medical_History_39	Medical_History_40	Medical_History_41
count	59381.000000	59381.000000	59381.000000
mean	2.830720	2.967599	1.641064
std	0.556665	0.252427	0.933361
min	1.000000	1.000000	1.000000
25%	3.000000	3.000000	1.000000
50%	3.000000	3.000000	1.000000
75%	3.000000	3.000000	3.000000
max	3.000000	3.000000	3.000000

Methods Tried

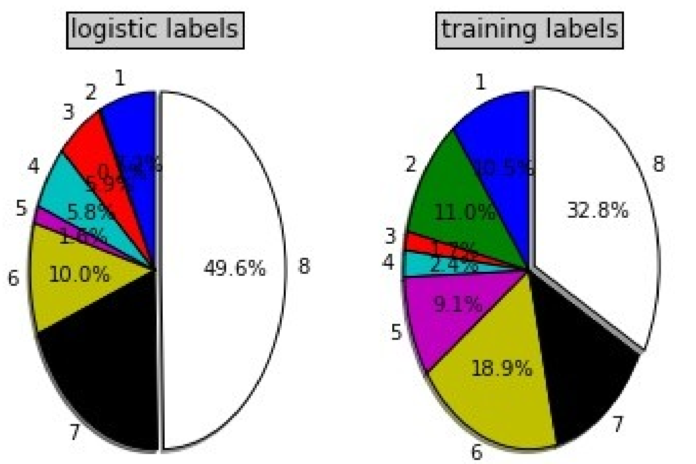
All these methods have been tried by taking 80% of the randomly selected training set for training and then predicting on remaining 20% of the set taking it as test set

Logistic Regression

Logistic Regression is one of the most prominent methods used in for classification problems. Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution. The outcome is a probability to be true for each of the possible outcomes/labels. The label with highest probability is choses as the final result.

The accuracy achieved with this method was **0.5037 or 50.37%** and the **f1Score was 0.4653**.

Training Labels vs Predicted Test Labels Distribution with Logistic Regression



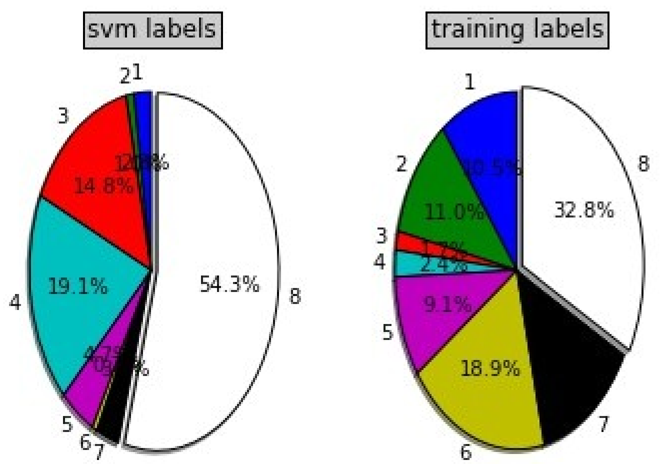
The above diagram shows the distribution of labels on the training data set vs distribution of labels predicted on the test data set as predicted by the logistic regression algorithm.

Support Vector Machines

In machine learning , support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non probabilistic classifier . An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

The accuracy achieved with this method was **0.4014** or **40.14%** and the **f1Score was 0.3254** on using SVM with a linear kernel.

Training Labels vs Predicted Test Labels Distribution with SVM



The above diagram shows the distribution of labels on the training data set vs distribution of labels predicted on the test data set as predicted by the SVM algorithm.

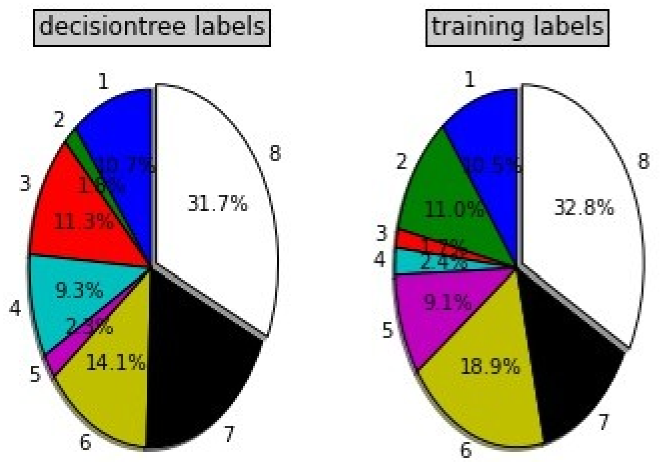
Decision Trees

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represents classification rules.

Each feature is ranked upon by using Information Gain as a parameter and the data set is then trained upon by classifying it on these features in order of the values of Information Gain for these features.

The accuracy achieved with this method was **0.4503** or **45.03%** and the **f1Score was 0.4508**.

Training Labels vs Predicted Test Labels Distribution with Decision Trees



The above diagram shows the distribution of labels on the training data set vs distribution of labels predicted on the test data set as predicted by the decision tree algorithm.

Random Forest

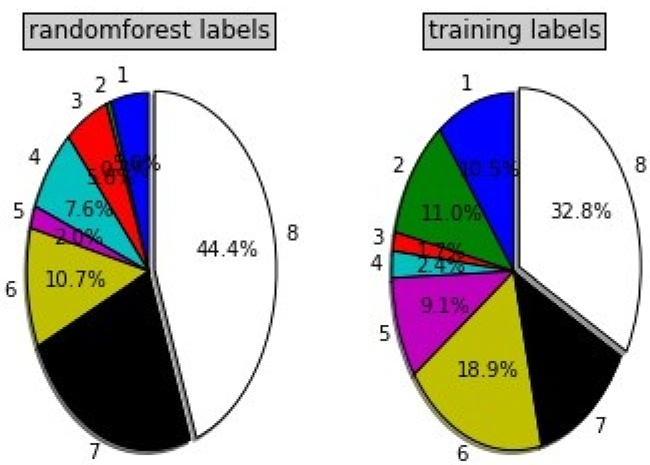
Random forests is a notion of the general technique of random decision forests, that are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

Variation in Accuracy,F1score vs Depth in Random Forest Classifier

Serial No,	Depth	Accuracy	F1score
1.	6	0.4613	0.3886
2.	10	0.5240	0.4718
3.	14	0.5574	0.5196
4.	18	0.5696	0.5365
5.	22	0.5701	0.5389
6.	**26**	**0.5707**	**0.5400**

The accuracy achieved with this method was **0.5707** or **57.07%** and the **f1Score was 0.5400**

Training Labels vs Predicted Test Labels Distribution with Random Forest



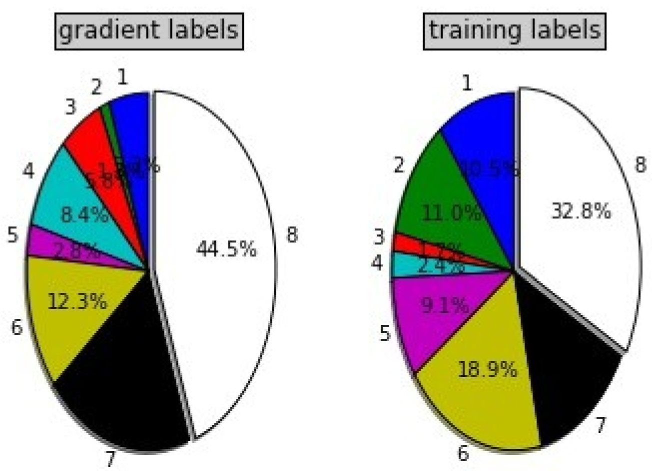
The above diagram shows the distribution of labels on the training data set vs distribution of labels predicted on the test data set as predicted by the random forest algorithm.

Gradient Boosting

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

The accuracy achieved with this method was **0.5913** or **59.13%** and the **f1Score was 0.5643**

Training Labels vs Predicted Test Labels Distribution with GBM



The above diagram shows the distribution of labels on the training data set vs distribution of labels predicted on the test data set as predicted by the gradient boosting algorithm.

XGBoost

XGBoost is short for “Extreme Gradient Boosting”, where the term “Gradient Boosting” is proposed in the paper *Greedy Function Approximation: A Gradient Boosting Machine*, by Friedman. XGBoost is based on this original model. The algorithm made its debut during the Higgs Boson competition at Kaggle and has been one of the most successful algorithms since then.

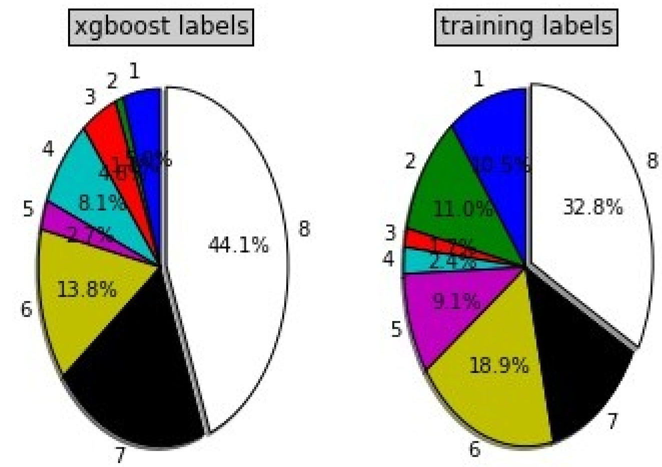
Different combinations of parameters were tried with XGBoost algorithm on the training data to to obtain suitable combination. A summary of those combinations is given below in the table.

Results obtained with different variations in XGBoost

Serial No.	ETA	DEPTH	Accuracy	F1Score
1.	**0.1**	6	0.5908	0.5667
2.		9	0.5909	0.5660
3.		**12**	**0.5915**	**0.5668**
4.		15	0.5892	0.5632
5.	0.2	6	0.5891	0.5657
6.		9	0.5849	0.5596
7.		12	0.5816	0.5583
8.		15	0.5773	0.5529
9.	0.3	6	0.5875	0.5635
10.		9	0.5833	0.5591
11.		12	0.5769	0.5536
12.		15	0.5704	0.5472
13.	0.4	6	0.5889	0.5658
14.		9	0.5774	0.5513
15.		12	0.5732	0.5509
16.		15	0.5634	0.5415

From the above table we can see that the best combination was with eta = 0.1 and depth =12 on the training data. Hence this model was used to predict results on the test data set and the accuracy achieved with this method was **0.5987** or **59.87%** and the **f1Score was 0.5706**

Training Labels vs Predicted Test Labels Distribution



The above diagram shows the distribution of labels on the training data set vs distribution of labels predicted on the test data set as predicted by the XGBoost algorithm.

Since the f1Score by gradient boosting and xgboost were quite close to each other hence models from these algorithms were used to predict on the data set. Out of the two models the one by gradient boosting scored a better score of 0.56370 whereas the one by xgboost scored 0.54124 on the test set on Kaggle.

Summary of Results

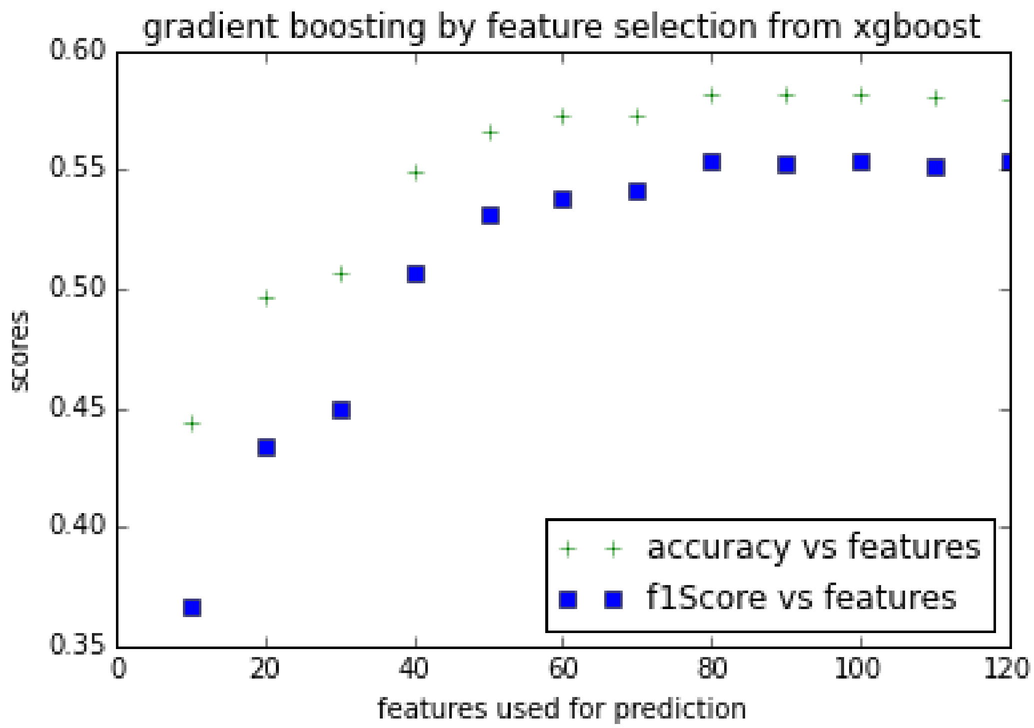
The table below gives a short summary of accuracy and weighted f1Score obtained for different tried methods on a data set. The method which performed best on the leader board has been highlighted.

	Method Name	**Accuracy**	**F1Score**
1.	Logistic Regression	0.5037	0.4653
2.	Support Vector Machines	0.4014	0.3254
3.	Decision Tree	0.4503	0.4508
4.	Random Forest	0.5760	0.5465
5.	**Gradient Boosting**	**0.5913**	**0.5643**
6.	XG Boost	0.5987	0.5706

Methods vs Accuracy vs f1Score table.

Improvements Tried

1. Feature Selection through XGBoost
XGBoost algorithm can also be used for feature selection since it provides a score to each of the feature being used for prediction. One of the obvious methods thus was to perform a feature ranking with the help of XGBoost and then using Top features for prediction using gradient boosting. Below attached is a plot to show the variation of accuracy and f1Score with the selected features.



As can be seen from the graph above calculated f1Score plateaus at a value of 0.556 and almost all features are used. Thus the idea of selecting features first through XGBoost and then using those features in gradient boosting did not yield any benegit over the previous method.

2. Prediction by leaving out the continuous variables

Another method which was tried was to predict the response variable by only taking into consideration the categorical , discrete and dummy variables. The intuition behind the method was that since the end task is to do multi-class classification, hence more value can be obtained from non-continuous variables. However, the method was not a success as the accuracy was **0.5637** and f1Score achieved was **0.5417** which was less than the best score

3. Feature selection through PCA

Another method tried was to perform a feature selection using PCA and then predicting using gradient boosting algorithm. The intuition behind the method was to if the best features can be obtained from PCA and then prediction is done using those features only it might lead to a better result. However, this method was also not a success. The method showed the same trend as was shown by feature selection in XG Boost method with best f1Score not been able to beat the previous best.

Conclusion

Amongst all the methods tried for the current data set gradient boosting methods performed best on the data set. The initial intuitive assumption of trying a data set of only categorical variables performing better than dataset with all the variables did not hold ground as the best results were obtaines when all the variables were used.

The best training accuracy and f1score obtained was 0.5913 and 0.5643. The best accuracy obtained on the Kaggle data set was 0.56816.

Future Work

The future work includes trying out the mlr package which is said to have an accuracy of 0.64 https://www.kaggle.com/casalicchio/prudential-life-insurance-assessment/use-the-mlr-package-scores-0-649

Also, a better way of feature engineering needs to be carried out so as to extract the best of information from the relevant data sets which will require a good statistical background.