PRUDENTIAL LIFE INSURANCE PROJECT

Tejas, Arnav

AIM

In a one-click shopping world with on-demand everything, the life insurance application process is antiquated. Customers provide extensive information to identify risk classification and eligibility, including scheduling medical exams, a process that takes an average of 30 days.

Our aim here is to develop a predictive model that accurately classifies risk using a more automated approach in a minimum amount of time.

The dataset provided has 122 features and 79146 observations.

PRE-PROCESSING

Initially we calculated how many missing values exist in each column and removed missing columns that had a ratio of more than 0.85 missing values. This reduced the number of features to 110 Interestingly, after re-evaluating, the remaining features had less than 0.1 ratio of missing values.

In this dataset, most of the column names weren't provided. We had to impute data logically and we found that, with the remaining columns imputing data with the highest frequent values in the categorical columns and with mean in the numerical columns made most sense.

INITIAL ATTEMPTS: BORUTA

Below, is the step wise working of Boruta algorithm:

A) Initially, it adds randomness to the given data by creating shuffled copies of all features (which are called shadow features).

B) Then, it trains a random forest classifier on the extended data set and applies a feature importance measure (the default is Mean Decrease Accuracy) to evaluate the importance of each feature where higher means more important.

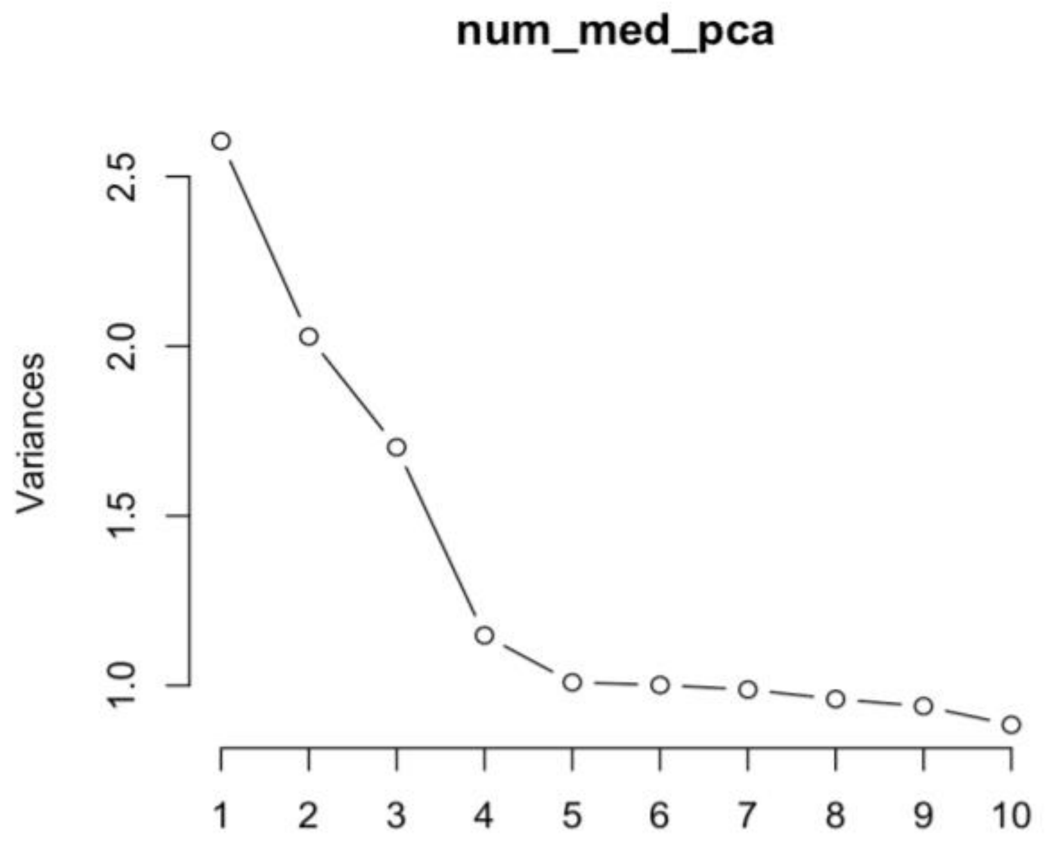
C) At every iteration, it checks whether a real feature has a higher importance than the best of its shadow features (i.e. where the feature has a higher Z score than the maximum Z score of its shadow features) and constantly removes features which are deemed highly unimportant.

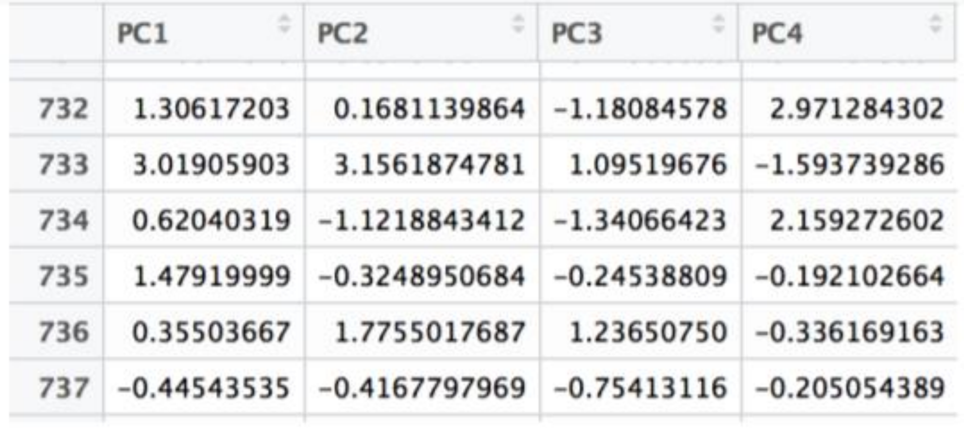
D) Finally, the algorithm stops either when all features get confirmed or rejected or it reaches a specified limit of random forest runs.

INITIAL ATTEMPTS: PCA

Using PCA, we reduced the number of variables in the data set by converting 13 numerical variables to 9.

Replaced with 4 PC's that explain maximum variance of the data available.





	PC1	PC2	PC3	PC4
732	1.30617203	0.1681139864	-1.18084578	2.971284302
733	3.01905903	3.1561874781	1.09519676	-1.593739286
734	0.62040319	-1.1218843412	-1.34066423	2.159272602
735	1.47919999	-0.3248950684	-0.24538809	-0.192102664
736	0.35503667	1.7755017687	1.23650750	-0.336169163
737	-0.44543535	-0.4167797969	-0.75413116	-0.205054389

INITIAL ATTEMPTS: OTHER METHODS

1) Using the random forest VarImpPlot to select important variables.

2) Plotting decision trees to see the important variables.

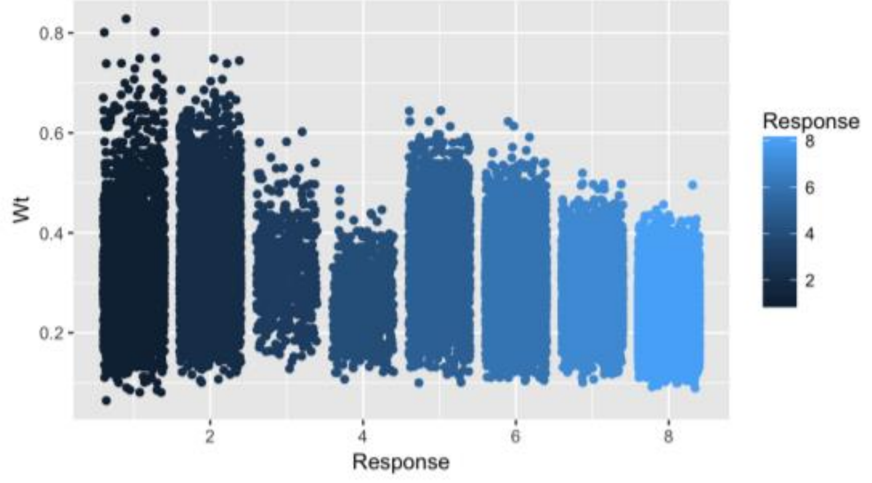
3) Stepwise regression.

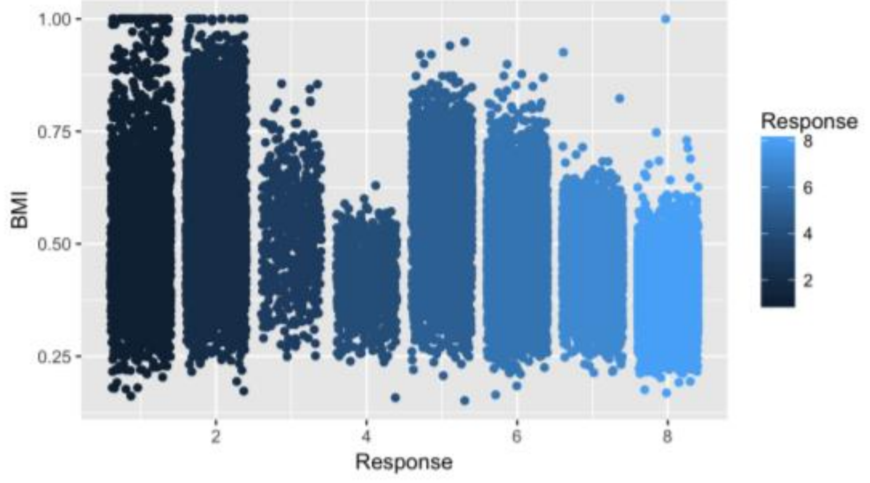
After our initial attempts, our best RMSE values were using different models were:

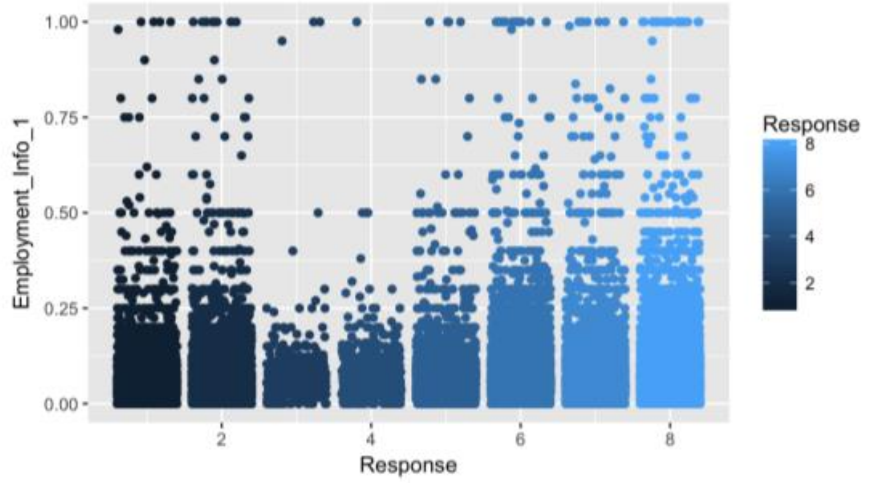
Linear regression	1.9735
Decision tree	2.206184
SVM	2.194313

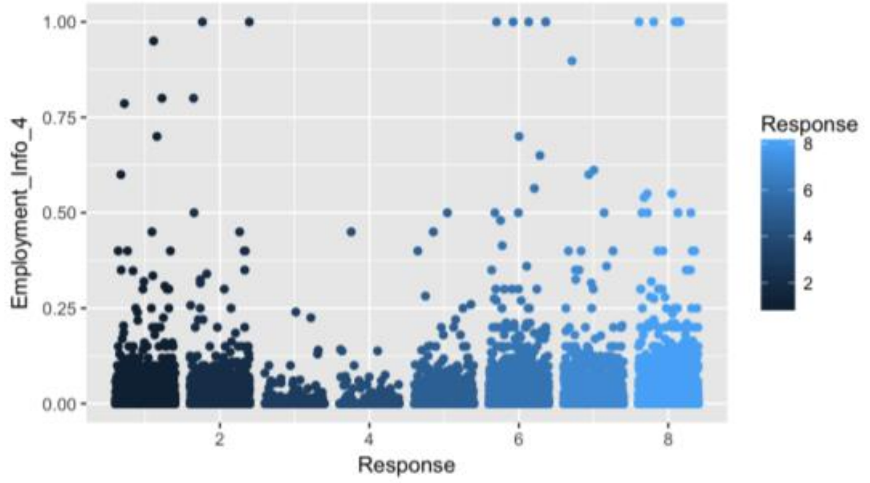
We then decided to switch to manual feature extraction.

REORDERING THE RESPONSE









When we plotted some variables with the response, we noticed a common pattern in each of them. Intuitively, the risk should increase with increase in weight and/or BMI. We concluded that the ordering of the risk was not exactly linear, so we made them linear by rearranging the risks.

After rearranging them, the graphs of the variables fall into order and make intuitive sense.

We switched the columns in the following way:-

Original ➡ Reordered

1 ➡ 1

2 ➡ 2

3 ➡ 7

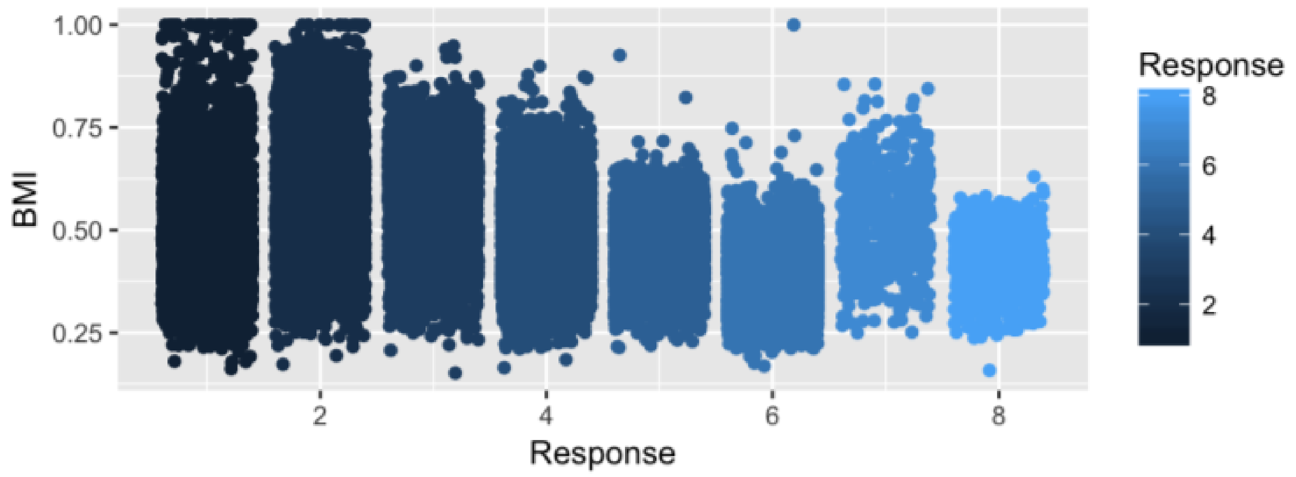
4 ➡ 8

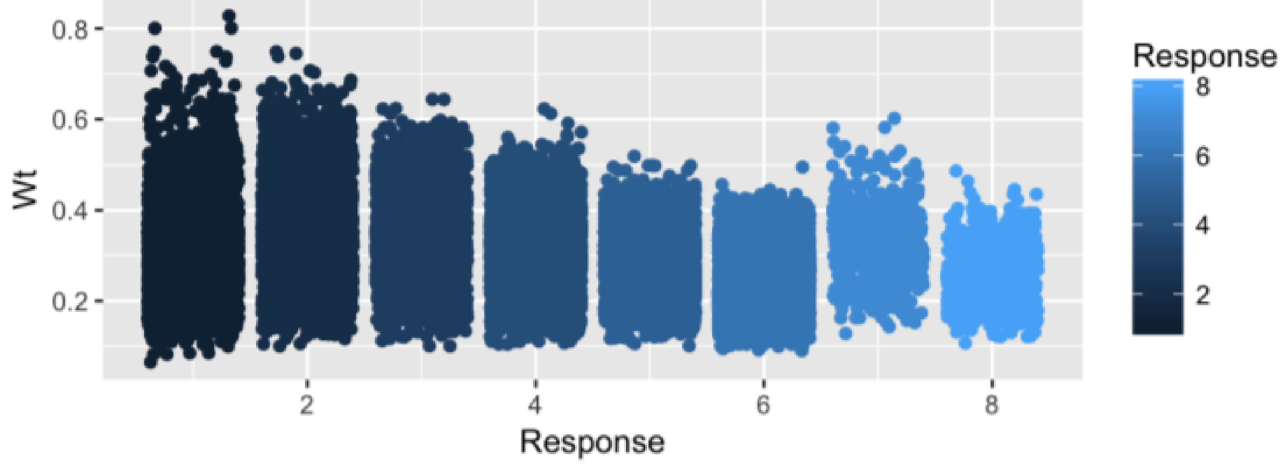
5 ➡ 3

6 ➡ 4

7 ➡ 5

8 ➡ 6





FEATURED ENGINEERING – NEW FEATURES

A ‘trans’ function we created works as kind of a poor man’s neural net with one layer of activation functions. The best activation function and the best inter-related nonlinear transformations are displayed by the function, and then that information can be used for feature extraction.

Created non linear transformations that could better fit the model.

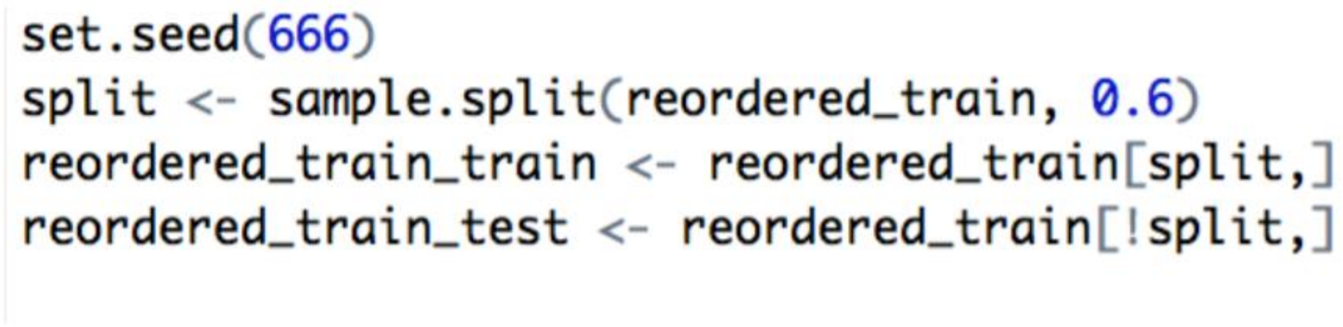
For e.g. – created a log of two columns and compared the fit of it against the response by comparing the Rsquared value.

We even added each column in the key word to create a new variable. (Sum_Keyword).

When we used rpart with only Product Info 4 as the predictor, we noticed that there was a very decisive split at 0.074. So, we decided to Convert that column to a binary column that indicates whether each observation is above 0.074 or below.

SPLITTING THE DATA INTO TRAIN AND TEST

We split the data into 60% train and 40% test.



```
set.seed(666)
split <- sample.split(reordered_train, 0.6)
reordered_train_train <- reordered_train[split,]
reordered_train_test <- reordered_train[!split,]
```

This train set is then further divided into training (70%) and validation (30%) set depending on the type of model.

SELECTING IMPORTANT COLUMNS FROM THE LINEAR MODEL

At this point we reduced our variables to 84 and used that in our linear model to predict the risk categories and used RMSE values to get the best models.

We know that regression coefficients with high values explain more variance of the data and would be more significant compared to others.

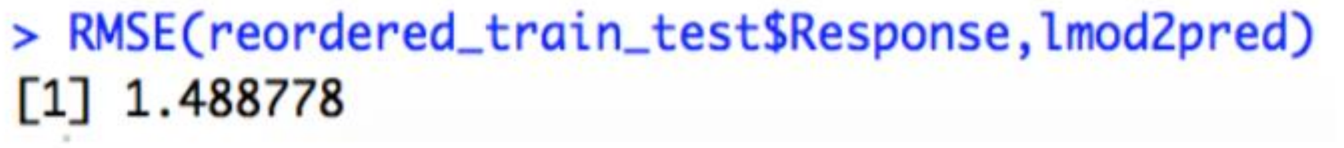
After checking p-value we got the top 64 variables with high coefficient values that would help make the model better.

LINEAR REGRESSION

Values we got less than 0, got converted to risk category 1 as that was the closest and made sense to merge it into that category.

For our Linear Model, we checked for multicollinearity by evaluating the vif values (all values were less than 5).

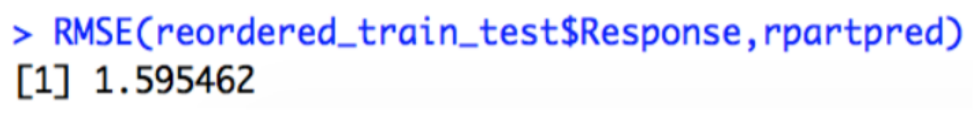
After training, validating and testing our model on the dataset. Our best model got us a value of RMSE: 1.488778



```
> RMSE(reordered_train_test$Response,lmod2pred)
[1] 1.488778
```

DECISION TREE(S)

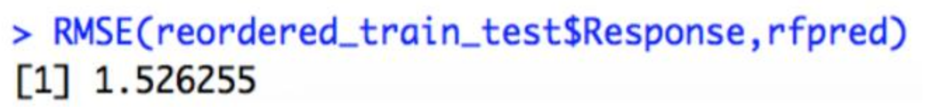
RPART (Recursive Partitioning and Regression Trees)



```
> RMSE(reordered_train_test$Response,rpartpred)
[1] 1.595462
```

Random Forest

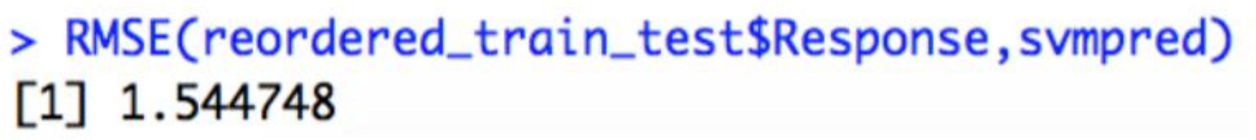
After tuning…



```
> RMSE(reordered_train_test$Response,rfpred)
[1] 1.526255
```

SVM

We used grid search technique to determine the value of cost and gamma, and found that cost = 8, gamma = 0 and kernel = “radial” provided the best results.



```
> RMSE(reordered_train_test$Response,svmpred)
[1] 1.544748
```

CONCLUSION

After extracting features and reordering the risk levels of the response, we saw a vast improvement in our RMSE values. Below is a table displaying this improvement:

	Linear Regression	Random Forest	SVM
Before FE and reordering	1.973	2.206	2.194
After FE and reordering	1.488	1.595	1.544