

# Machine Learning and Data Mining

COMP9417 (19T2)

## Project Report

### Team – Treadstone

Ajinkya Biswas	z5196315
Daniel Hocking	z5184128
Ghazali Hali	z5220911

**Revision History**

Date	Author	Version	Change reference
10/08/2019	Ajinkya Biswas Daniel Hocking Ghazali Hali	1.0	Final Version

## Contents

Framing the Problem .....	3
Solution Description .....	3
Exploratory Data Analysis .....	4
1. Data is clean.....	4
2. Missing Data .....	4
3. Data is not balanced.....	4
4. Correlation.....	5
5. Medical Keywords.....	5
Feature Selection and Engineering.....	5
Model Selection.....	6
1. Bias-Variance Trade-off.....	6
2. Receiver Operating Characteristic (ROC) curves.....	7
Parameter Tuning.....	8
Model Training .....	8
Results.....	9
1. Support Vector Classifier (SVC):.....	9
2. Logistic Regression:.....	9
Conclusion.....	10
Appendix.....	11
A. Correlation of key features and Response .....	11
B. Missing Data .....	12
C. Data is not Balanced.....	12
D. Correlation.....	13
E. Bias-Variance Trade-off B70.csv .....	13
F. Final Result .....	13
Acknowledgement.....	14
References .....	14

## Framing the Problem

Alongside the ever-increasing cost of healthcare, another trend is the surging volumes of data being kept as part of our medical records. This presents a problem, and also an opportunity, by using this data to make smarter decisions about how to distribute healthcare efficiently and fairly. Prudential are an insurance company and one of the largest providers of life insurance in the US, they have released an anonymised dataset to Kaggle that includes a detailed set of over 100 features that represents an applicant's medical history along with the risk classification they present as a customer. Using this data that contains approximately 60,000 labelled training samples, containing continuous/discrete and categorical features, the expected response is a classification between one of 8 possible levels of risk associated with providing insurance.

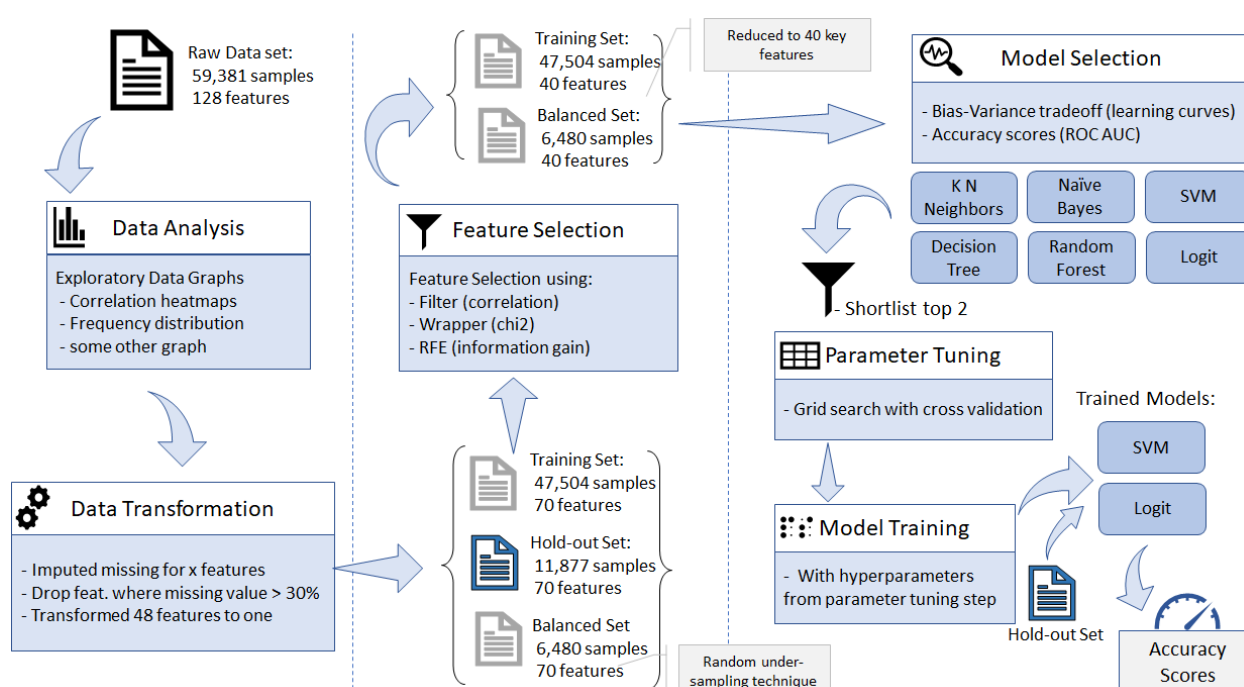
Based on initial exploration of the data it was clear that the dataset:

- Had been fully anonymised so no useful information could be gained from knowing that a feature was categorised as one thing versus another, only in the statistical trends observed in the data
- Was clean but did contain missing values in some features
- Was unbalanced, there were a lot more of some responses than others
- Was easy to overfit, this was based on discussion surrounding the Kaggle competition and also the leaderboards demonstrated a trend of significant difference between winners of the public and private leaderboards

This guided us in our choice of machine learning approach as the solution would be for a supervised classification problem, we would need to identify key features to reduce dimensionality, handle missing values, rebalance the data, and deal with overfitting.

## Solution Description

The process followed for implementing this project is depicted in the process diagram, it highlights the main methods used and the classification algorithms.



Raw data was downloaded from Kaggle's website which had 59,381 samples and 127 features. After data exploration and transformation, which are detailed in the relevant sections of this document, we formed the following sets for training and one hold-out set for final evaluation score.

File Name	Description	Samples	Features	Comment
train.csv	Raw Data	59,381	127	The original data from Kaggle website.
F70_train.csv	Training Set 70	47,504	70	Reduced set of features after transformation and discarding those with more than 30% missing values
F40.csv	Training Set 40	47,504	40	Shortlisted features based on multiple feature selection techniques.
B70.csv	Balanced Training set 70	6,480	70	The data set is skewed, and a few target classes are not well represented in the original data. Balanced dataset using random under sampling.
B40.csv	Balanced Training set 40	6,480	40	Random under-sampling technique has been used to reduce the training data
F70_test.csv	Hold Out set/ Validation set	11,877	70	We kept 20% of the training data separate to test at the end of Model training

### Software and libraries:

The project is implemented using Python Jupyter Notebook and relies heavily on Sci-kit Learn (sklearn) for the machine learning algorithms. For data manipulation PANDAS and NUMPY have been used. Visualizations are constructed using the PYPLOT and SEABORN libraries. Additionally, the YELLOWBRICK library has been used to visualize a few aspects of model selection and tuning.

## Exploratory Data Analysis

After we went through the data description and explored the training and test datasets, a few things stood out from our analysis/visualizations.

### 1. Data is clean

Both training and test datasets are very clean, i.e. there is no special characters or any unexpected values anywhere. Both train and test datasets are very similar, i.e. data distribution of each feature is very similar. (Appendix - A)

### 2. Missing Data

We found that some of the features have a large number of missing values. And values for these features are missing in the same ratio in both train and test datasets. (Appendix - B)

### 3. Data is not balanced

One of the key things we noticed in the dataset is that the dataset is not balanced, i.e. there are total 59381 data points in training set and out of those for response class 8 has the highest count (32.82%) and response class has the lowest count (1.71 %). (Appendix - C)

## 4. Correlation

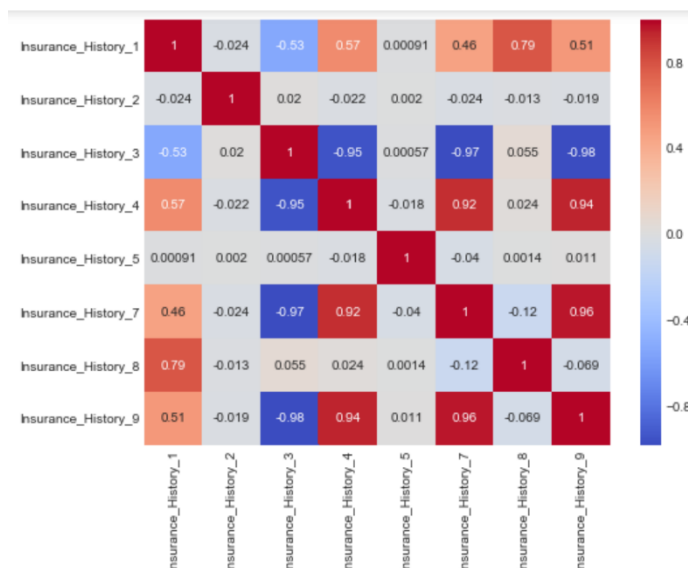
We did some additional feature analysis by grouping the features in eight different categories (as it is in the data description) and found that most of the features have very low correlations.

Except Insurance\_History\_1 and Insurance\_History\_8, correlation = 0.79,

Insurance\_History\_4 and Insurance\_History\_7, correlation = 0.92,

Insurance\_History\_4 and Insurance\_History\_9, correlation = 0.94,

Insurance\_History\_7 and Insurance\_History\_9, correlation = 0.96.



We also found Weight and BMI, correlation = 0.85. We noticed that Family\_Hist\_2 and Family\_Hist\_4 has high correlation (0.93), but both of these features have high missing values (48.25% and 32.31% respectively) and we later decided not to use features which have more than 30% missing values. (Appendix - D)

## 5. Medical Keywords

There are 48 medical keywords features which are dummy variables, values are [0, 1]. To analyse these features, we looked through some insurance forms online and tried to figure out what these features might be. These are most probably medial diseases which is normally the last section of an insurance application form and checkboxes (1 means someone had this disease, and 0 otherwise). As a general intuition, if someone had a higher count(sum) of diseases would probably be in different risk classification and eligibility than someone who had lower or zero count of these features. Later in feature engineering section, we used this intuition.

## Feature Selection and Engineering

The following steps are taken to prepare the data for modelling:

- Remove features that have > 30% missing values as these features do not contain as much information and there is a great deal of uncertainty in what values are missing
- For the remaining features that contain missing values impute the missing values using mode for nominal features and mean for continuous features
- Use [sklearn LabelEncoder](#) to take categorical labels for Product\_Info\_2 and convert to numeric as this is required by the models
- Create new Medical\_Keyword\_SUM feature that is the sum of all 48 Medical\_Keyword columns and remove the individual columns, the Medical\_Keyword features are binary values indicating if an option on a medical history questionnaire has been selected, a greater number selected indicates an increased risk so combining them into a single feature doesn't reduce information

- Train/test split: create an 80/20 train test split using [sklearn train\\_test\\_split](#) and use the stratify option, this option is required due to the unbalanced nature of the dataset. The result of this is saved to file F70\_train.csv and F70\_test.csv
- Use random under sampling to create a balanced set from the full train set (F70\_train.csv), balanced in the sense that it has equal numbers taken from each class. The result of this is saved to file B70.csv
- At this point there are 70 features remaining, of which 60 are nominal, 8 are continuous, and 2 are discrete
- We first look at chi-square score of discrete features, however due to only 2 features being left we chose to include them all
- A mutual information classifier is used to choose the top 30 out of 60 features
- Finally use recursive feature elimination using logistic regression for continuous features, again only 8 features are present, so we chose all of them
- The top k feature set now contains 40 features. These features are then used to create F40.csv and B40.csv.
- We also scaled our datasets to ensure all features are in the same scale

## Model Selection

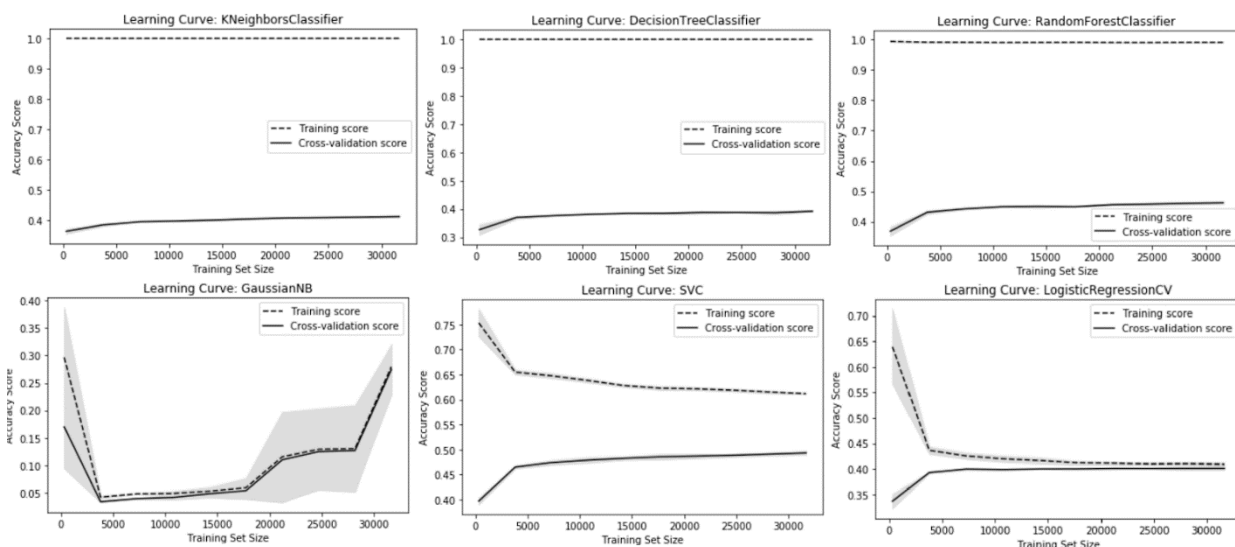
For model selection we opted to test 6 classification algorithms (KNN, Decision Tree, Random Forest, Naive Baise, SVM, Logistic Regression).

We plotted learning curves and ROC curves to gauge the performance of the algorithms on our data set.

### 1. Bias-Variance Trade-off

Simplifying assumptions give bias to a model. The more erroneous the assumptions with respect to the true relationship, the higher the bias, and vice-versa. To find the bias-variance trade-off, we have opted to plot learning curves for each of the algorithms with the following considerations:

1. Using the [learning\\_curve\(\)](#) function from the [scikit-learn](#) library
2. We used training sets of ten varying sizes for generating the learning curves. We had experimented with fifty sets, but execution time for SVC and logit was prohibitive.
3. Evaluation metric used is classification accuracy
4. Cross validation is used with 5 folds.



The chart above is based on the data set with 70 features (F70\_train.csv). Another iteration was run on the balanced data set (B70.csv), its learning curves were similar. It has been included in the Appendix - E.

The table lists the findings for each of the learning algorithms.

Classifier	Bias	Variance	Comments
KNeighbors	high	high	This model is underfitting the test data. Training accuracy is high and validation accuracy is very low. The gap between the curves is also large indicating high variance. Overall accuracy is the lowest of the set.
DecisionTree	high	high	This mode is also underfitting and shows high bias and variance. Difference between the train and test score is also very high.
RandomForest	high	high	Accuracy is better than that of decision tree however this model also has high bias and variance. Difference between the train and test score is also very high.
Naive Bayes	high	low	Gap between training and validation accuracy is small. However, since the overall accuracy is very low this indicates high bias
SVM	low	low	This model has the best accuracy from the set. Although there is a large gap between the training and validation curves, it is much smaller than that of the other underfitting algorithms
Logistic Regression	low	low	This model demonstrates the second highest accuracy score in the set. Judging by the size of the gap variance is also low

To go beyond classification accuracy, we will next employ receiver operating characteristic (ROC) for visualizing and ranking classifiers based on their performance using the F-measure which is based on precision and recall.

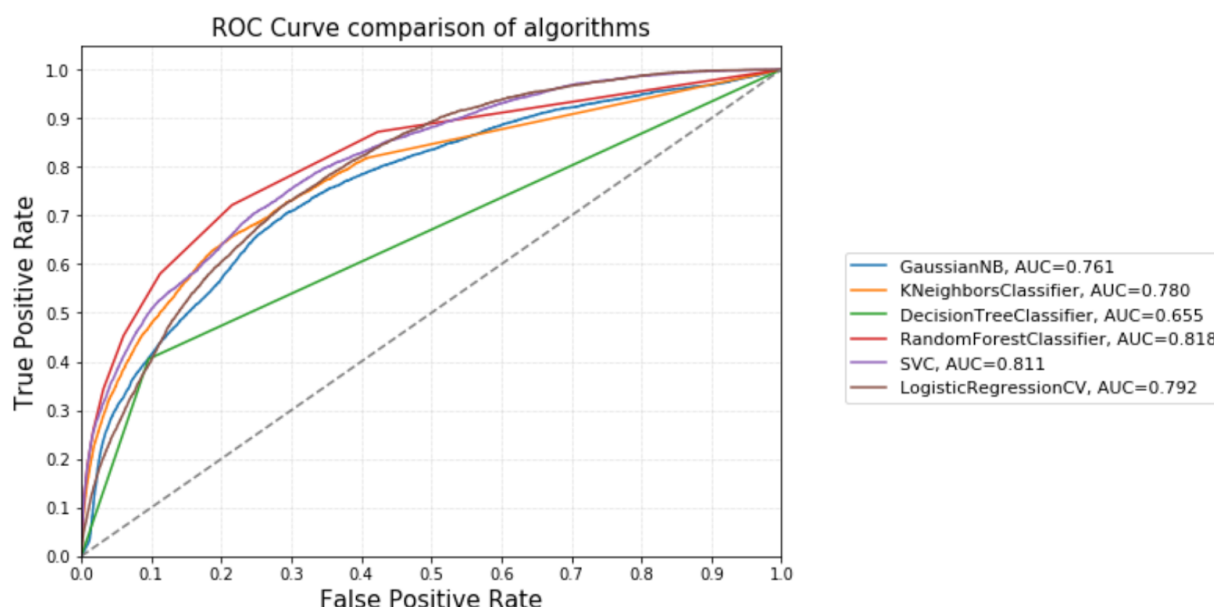
## 2. Receiver Operating Characteristic (ROC) curves

The ROC curve plots the true positive rate against the false positive rate. Instead of a confusion matrix for each classifier, we opted for ROC curves as it provides a convenient method to compare the performance of the classifiers chosen.

Since ROC curves are typically used in binary classification, it was extended for use in our multi-class problem by:

1. Binarizing the target.
2. Using micro-average for each classifier. Since we have an imbalanced (skewed) data set, the micro-average will aggregate the contributions of all classes to compute the average.
3. One-versus-the-rest strategy is used. The decision score from each classifier and the class whose classifier outputs the highest score.





This chart is based on the balanced data set and the top 40 shortlisted features. From the chart above we can see that Random Forest classifier has the highest AUC score followed by SVC and Logistic Regression.

We have shortlisted Support Vector Classifier (SVC) and Logistic Regression (LR) for further analysis and hyperparameter tuning to find the best performing classifier. Random Forest classifier is not included as for our dataset we have seen it to be highly biased.

## Parameter Tuning

To discover the optimal parameters for SVC and LR we made use of a grid search with stratified k-fold cross validation:

Classifier	Parameters	Optimal parameters
Logistic Regression	Penalty: l1, l2 C: 0.0001 – 10,000 (log scale, 10 intervals)	F70_train.csv: penalty=l2, C=10000 B70.csv: penalty=l2, C=166.81
SVC	Kernel: poly, rbf, sigmoid   gamma: 0.000001 – 0.1 (log scale, 5 intervals)   C: 0.0001 – 10,000 (log scale, 10 intervals)	F70_train.csv/B70.csv: kernel=rbf, gamma=0.0056, C=2.78

## Model Training

With parameter values from grid search we trained logistic regression and SVC with two sets of training data (4 models). The second set is the one created from random under sampling described earlier in the report.

1. Training set 1: 47,504 samples and 70 features (Full Dataset) (F70\_train.csv)
2. Training set 2: 6,480 samples and 70 features (Balanced/Under sampled) (B70.csv)

These 4 models are then tested against the hold-out set which was set aside earlier in the process. This test set consists of 11,877 samples and 70 features. The results of the testing are described in the following section.

## Results

The classification report shows a representation of the main classification metrics (precision, recall and F1) on a per-class basis. Precision is the ability of a classifier not to label an instance positive that is actually negative. Recall is the ability of a classifier to find all positive instances. The weighted average of F1 should be used to compare classifier models.

### 1. Support Vector Classifier (SVC):

The result tables show a row for each of the target class. Target class 3 and 4 were the most under-represented in the data set, it is interesting to note that recall on these classes is indeed higher for the model that was trained on the balanced data set. However, class 6 which is well represented in the original data set suffers with a low recall score with the model trained on balanced data set.

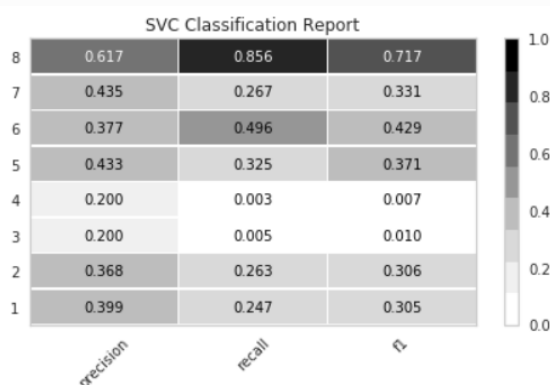


Fig 1: Training set: with 70 features

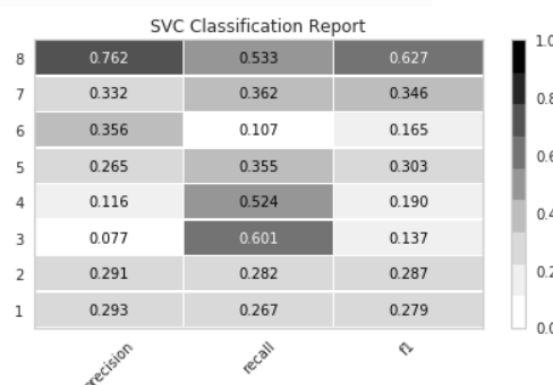


Fig 2: Balanced training

To further compare these two models, we computed the macro-averaged F1-score and the accuracy score:

Training set	Accuracy	Macro-averaged F1-score
F70_train.csv	0.49541	0.3095
B70.csv	0.35833	0.2917

We can see that the model trained on the full data set (47,504 x 70) has a higher F1 and accuracy score.

### 2. Logistic Regression:

Logistic Regression has much better recall scores on the under-represented classes as compared to SVC. It is also interesting to note that the choice of the training set (balanced vs unbalanced) did not affect a major effect on the precision and recall scores. However, the accuracy of the model trained on F70\_train.csv is indeed higher.



Fig 1: Training set: with 70 features

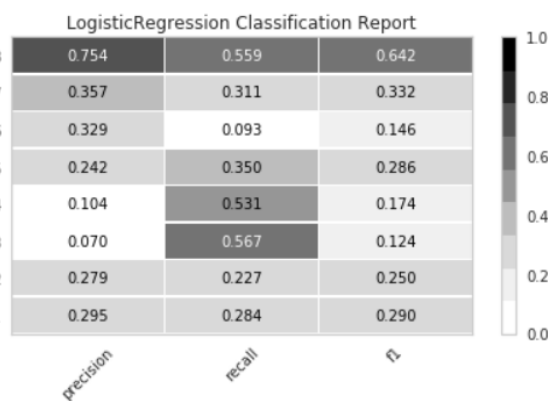


Fig 2: Balanced training

Training set	Accuracy	Macro-averaged F1-score
F70_train.csv	0.46703	0.293
B70.csv	0.39445	0.293

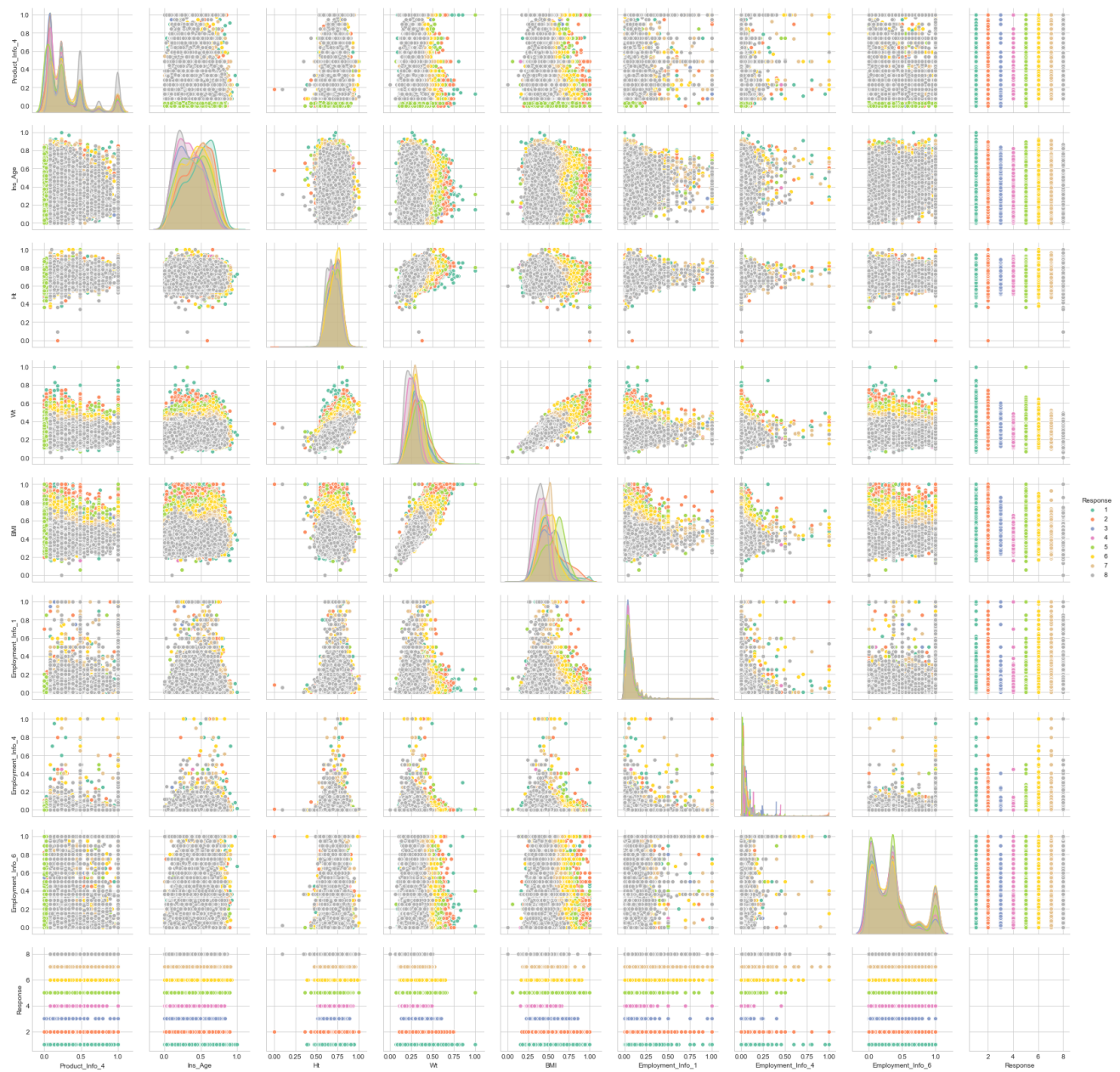
## Conclusion

Based on the higher accuracy score and higher macro-average F1-Score Support Vector Classifier trained on 47,504 samples and 70 features is the better model to proceed with.

At the end, we ran SVC and Logistic models on all our four training datasets (F70, F40, B70, B40) and predicted on both our validation dataset and original test dataset to generate final results. Based on the performance/accuracy, we chose to make the submission file predicted by SVC model. We achieved 0.50498 on Kaggle(Appendix - F).

# Appendix

## A. Correlation of key features and Response

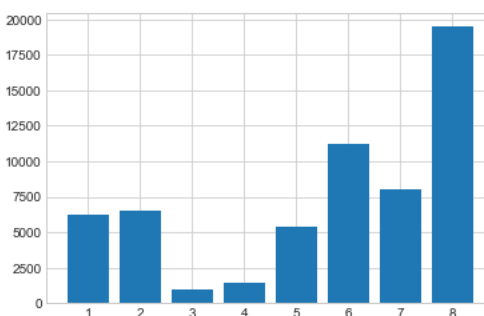


## B. Missing Data

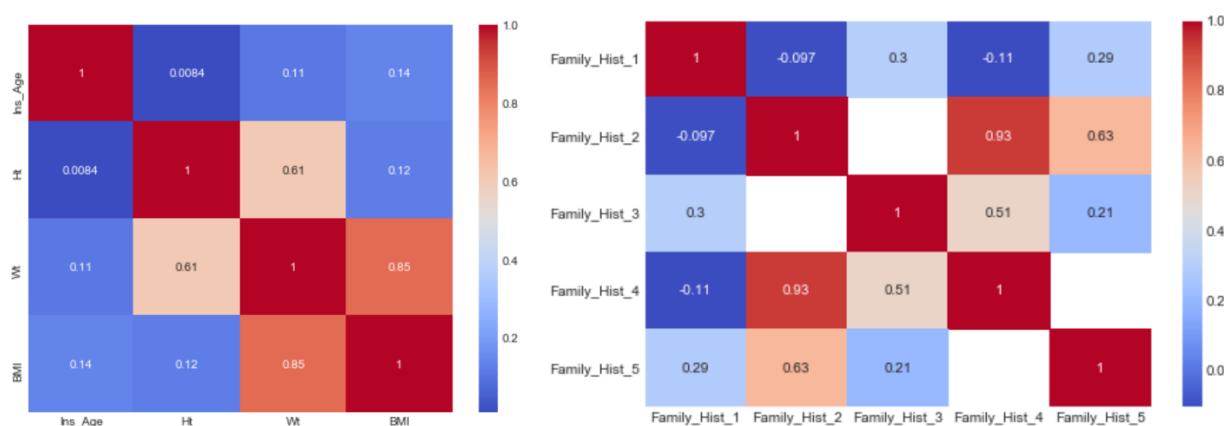
Column_Name	Training_Missing %	Test_Missing %
Employment_Info_1	0.031997	0.015178
Employment_Info_4	11.416110	10.812041
Medical_History_1	14.969435	15.036681
Employment_Info_6	18.278574	19.160132
Family_Hist_4	32.306630	33.781938
Insurance_History_5	42.767889	41.006830
Family_Hist_2	48.257860	49.987351
Family_Hist_3	57.663226	55.977738
Family_Hist_5	70.411411	68.929927
Medical_History_15	75.101463	75.203643
Medical_History_24	93.598963	94.029851
Medical_History_32	98.135767	98.224134
Medical_History_10	99.061990	98.983051

## C. Data is not Balanced

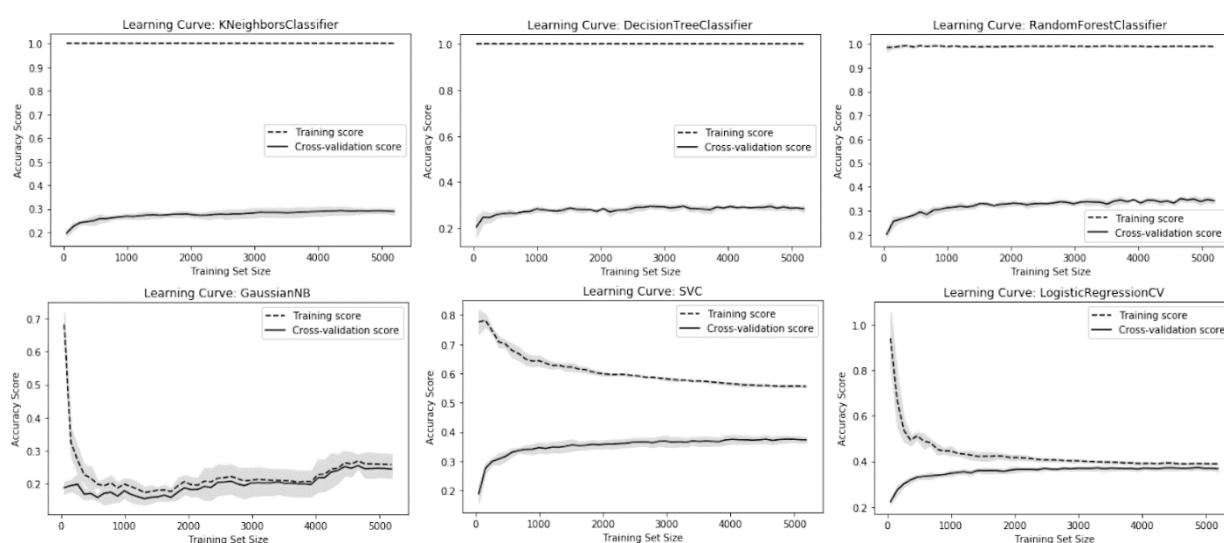
Response	Data Count	Percentage
1	6207	10.4528
2	6552	11.0338
3	1013	1.7059
4	1428	2.4048
5	5432	9.1477
6	11233	18.9168
7	8027	13.5178
8	19489	32.8203
Total	59381	100



## D. Correlation




## E. Bias-Variance Trade-off B70.csv



## F. Final Result

### Prudential Life Insurance Assessment



Can you make buying life insurance easier?  
 \$30,000 · 2,618 teams · 3 years ago

Overview
Data
Kernels
Discussion
Leaderboard
Rules
Team
My Submissions
Late Submission

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
submission.csv	just now	0 seconds	0 seconds	0.50498

Complete

[Jump to your position on the leaderboard](#) ▼

## Acknowledgement

As a completed competition there are many well thought out solutions available:

1st place used an XGBoost based solution: <https://www.kaggle.com/c/prudential-life-insurance-assessment/discussion/19010#latest-475517>

2nd place used Linear regression: <https://www.kaggle.com/c/prudential-life-insurance-assessment/discussion/19003#latest-229720>

3rd/4th place both used an ensembling technique: <https://www.kaggle.com/c/prudential-life-insurance-assessment/discussion/19016#latest-108684>

## References

The Kaggle competition: <https://www.kaggle.com/c/prudential-life-insurance-assessment/overview>  
<https://people.inf.elte.hu/kiss/13dwhdm/roc.pdf>

Sklearn LabelEncoder (<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html>)

Sklearn train\_test\_split ([https://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.train\\_test\\_split.html](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html))