PREDICTING LIFE INSURANCE APPLICANT RISK

University of Victoria
STAT 454: Topics in Applied Statistics - Machine Learning
Rai Goyal & Andrew Muth

Overview of the Competition

Business Problem

Life insurance is a financial product commonly used by consumers to transfer the financial risk associated with death to an insurance company. Typically, the insured pays a monthly premium and the insurance company pays a lump sum benefit to the designated beneficiary upon the death of the insured. Premiums vary based on the risk classification of the insured, which is impacted by factors such as age, medical history, lifestyle, etc.

To apply for life insurance, a prospective insured must fill out an insurance application and undergo medical exams. The process has not changed much over the last few decades despite technological advances. On average, it takes 30 days for the insurance application process. As a result, only 40% of US households own life insurance (though about 65% of Canadians have some life insurance coverage).

Prudential, one of the largest life insurance companies in the US, is trying to streamline the application process to make it easier for consumers. In order to do this, the company would like to understand the prediction power behind the information collected in the existing application. Using over one hundred life insurance application attributes, the goal is to predict the **risk classification for each individual (ordinal measure of risk with eight levels).**

Evaluation Metric

The evaluation metric used in this competition is quadratic weighted kappa, which measures the degree of agreement between two ratings (typically ranges from 0 (random agreement) to 1 (complete agreement), though it may fall below 0 if the rating is worse than random guessing).

The metric is calculated as follows. Firstly, given a vector of actual and predicted ratings, an 8 x 8 histogram matrix *O* is calculated (each entry O_{ij} corresponds to the number of individuals with actual rating *i* and predicted rating *j*). Next, a weight matrix *w* is calculated based on the difference between the ratings:

$$w_{ij} = \frac{(i-j)^2}{(8-1)^2}$$

Finally, an 8 x 8 expected histogram matrix *E* of expected ratings is calculated (outer product of the histogram vector of the actual and predicted ratings). The evaluation metric is computed by:

$$\kappa = 1 - \frac{\sum_{i,j} w_{ij} O_{ij}}{\sum_{i,j} w_{ij} E_{ij}}$$

In R, this is easily computed using the *ScoreQuadraticWeightedKappa* function in the *metrics* package.

The Data

Data Overview

The default data provided consists of a training set (59,381 observations) and a test set (19,765 observations). In total, there are 127 features, with the following description ("Prudential Life Insurance Assessment," 2015):

Table 1: Description of default data features

Feature	**Description**
Id	A unique identifier associated with an application
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
Wt	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
InsuredInfo_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

Among the variables, there is a good mix of categorical (nominal), continuous, and discrete variables.

The following variables are continuous: *Product_Info_4, Ins_Age, Ht, Wt, BMI, Employment_Info_1, Employment_Info_4, Employment_Info_6, Insurance_History_5, Family_Hist_2, Family_Hist_3, Family_Hist_4, Family_Hist_5.*

The following variables are discrete: *Medical_History_1, Medical_History_10, Medical_History_15, Medical_History_24, Medical_History_32.*

Medical_Keyword_1-48 are dummy variables.

All other variables are categorical.

Data Issues

The data had several issues that made it difficult to work with. The first was that all “normalized” continuous variables are confined to the unit interval. It was suspected that this was the results of min-max scaling on the part of Prudential in their attempt to anonymize the data. A consequence of this type of scaling is that when outliers are present the majority of observations are actually mapped into a much smaller interval than from 0 to 1. This motivated the use of regression trees which primarily rely on the order statistics of the covariates, which are unaffected by min-max scaling. The second issue present in the data was that many of the categorical variables had over 100 levels, often labeled with the positive integers. A consequence of this was that not all levels were present in the training set. Rather than remove these variables, they were treated as continuous. This was the approach used by the majority of competitors.

Another major issue was missing data. Every observation contained multiple data fields with missing values. The table below shows the distribution of the number of missing data fields for each observation.

Table 2: Distribution of the number of missing data fields for each observation

Min	**1st Qu.**	**Median**	**Mean**	**3rd Qu.**	**Max**
2.000	6.000	7.000	6.619	7.000	12.000

As shown in the table, 100% of the data has at least 2 missing data fields and 75% of the data has at least 6 missing data fields, so dropping observations was not a feasible method. Missing data for the continuous variables was imputed using the sample median. Because the training and testing data sets are large enough, the medians are likely to be similar and so this is a suitable assumption. Other methods such as *bagImpute* were investigated, but due to computational restrictions, the median impute method was preferred. In addition, we added a feature called *num_NA* which counts the number of missing data fields for each observation.

Due to the high dimensional nature of the data and computational limitations, feature engineering of the data was limited in scope. Interaction features were added within similar categories of continuous variables, including health factors (*Ins_Age, Ht, Wt, BMI*), employment factors (*Employment_Info_1, Employment_Info_4, Employment_Info_6*), family history factors (*Family_Hist_2, Family_Hist_3, Family_Hist_4, Family_Hist_5*), and product info / insurance history factors (*Product_Info_4, Insurance_History_5*). We had also initially planned to add interaction features for the medical keywords factors. However, subsequent model fitting was error prone as a sizable majority of the interaction terms did not have at least one instance in either the testing or training data. In particular, these features made it impossible to fit an LDA model, and many were removed by *removeCorrelatedFeatures,* as described in the *Methodology* section, subsection *Step 1: Obtaining the Subspace*. Hence, the medical keyword interaction terms were not included. With the addition of the *num_NA* feature and the continuous interaction terms, the total number of features was increased from 127 to 161.

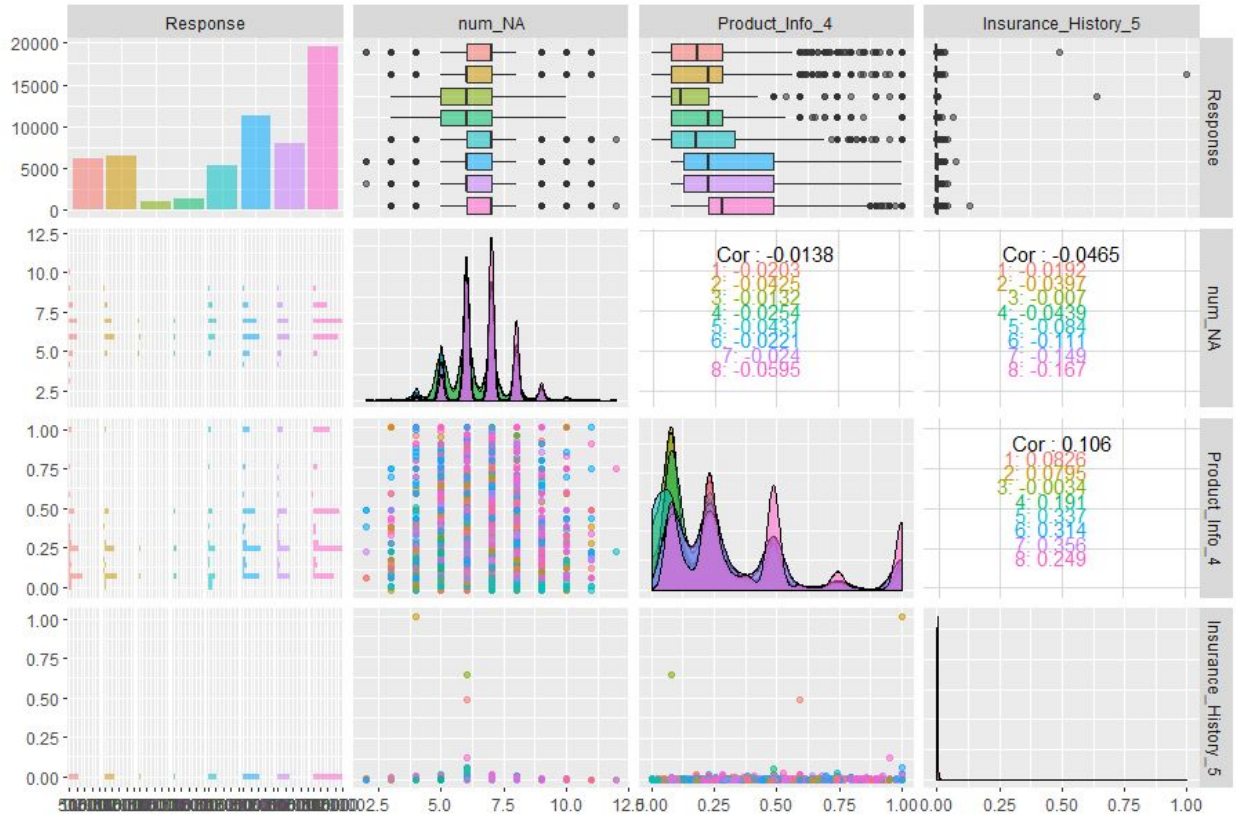
Exploratory Data Analysis

A summary of the data (grouped by continuous and discrete variables) is presented in the Appendix using the *tableone* package in R. The tables show the mean and standard deviation of each feature within each of the eight classes. In addition, the table shows the p-value for the appropriate hypothesis test. For the continuous variables, the hypothesis test is a t-test for equal means between all groups. Note that this test is not entirely appropriate as the observations have been scaled to the unit interval, although we believe it is still informative as to any difference in between group means. For the discrete variables, the test is a Pearson Chi-square test of independence for each variable with respect to the response category.

As we can see from the tables, nearly all variables have a p-value < 0.05, and so we would reject the null hypothesis of equality of means and conclude that the sample mean is different for at least one class (in the case of continuous variables) or that the variable is dependent on the response category (in the case of discrete variables). This validates our belief that almost all features should be included in the model by default. Since only a few variables out of 100+ variables have p-values higher than 0.05, we opted not to remove any features (apart from removing “non-constant” and correlated features for LDA, as explained in the *Methodology* section, subsection *Step 1: Obtaining the Subspace*).

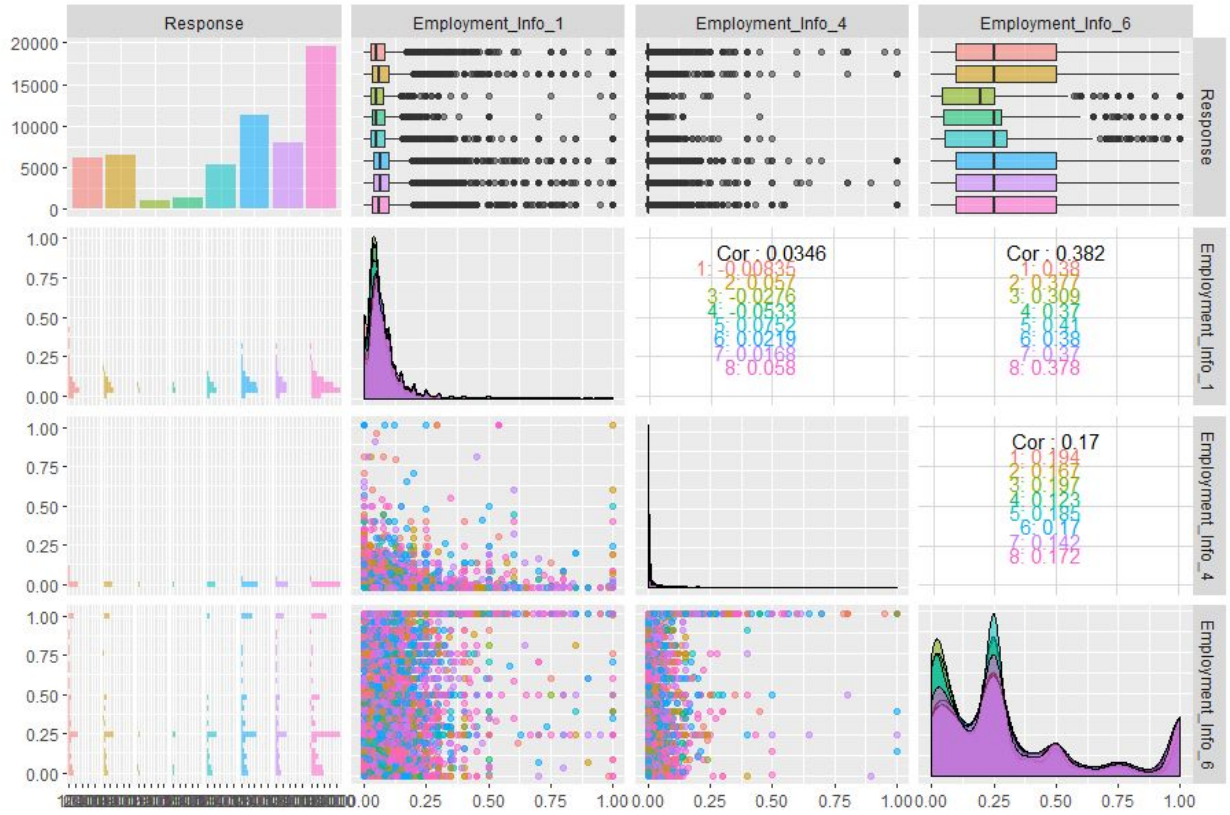
Below, we plot the response versus the continuous variables (in four categories for the purpose of interpretability) using the *ggpairs* function in the *GGally* package.

Figure 1: Response versus miscellaneous variables



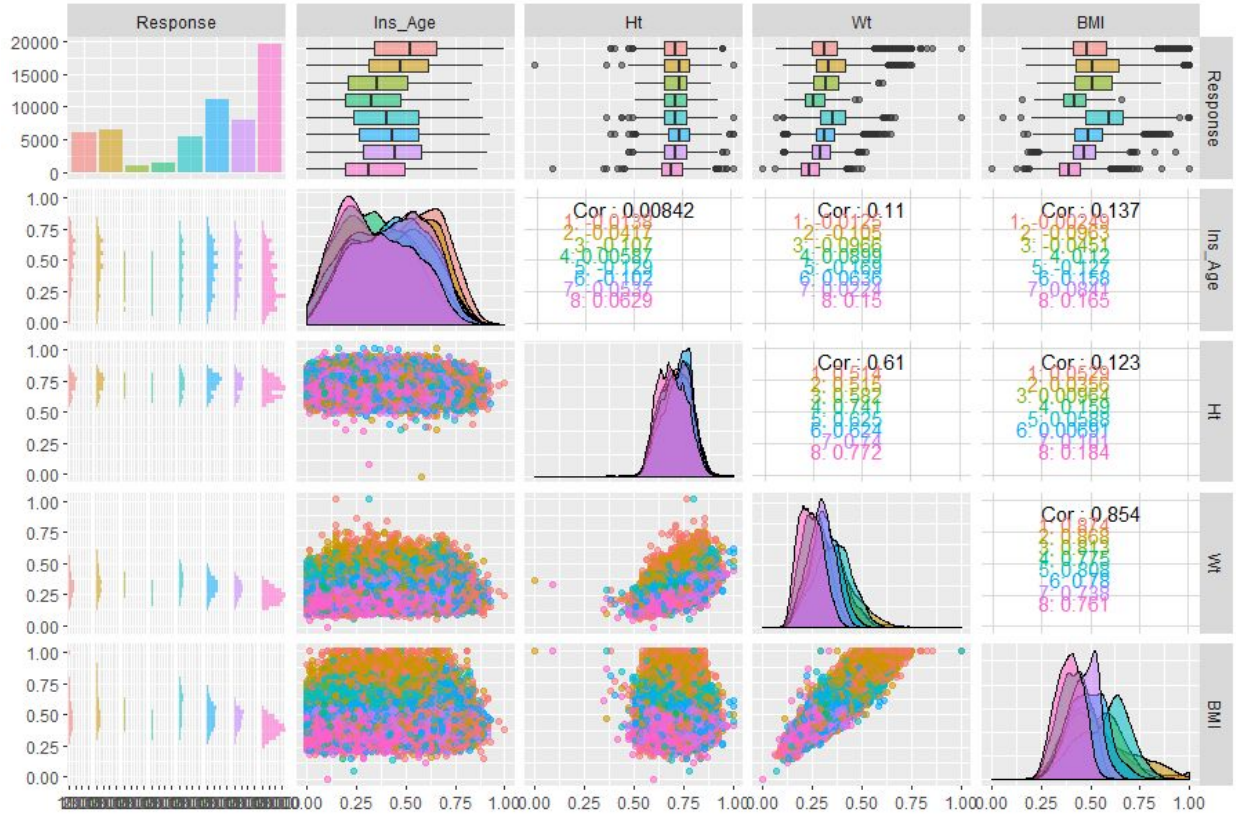
We see that classes are implanced and that when it comes time to fit models weighting techniques should be considered to try assign each class the same total weight. Values for *num_NA* and *Product_Info_4* occur in groups offering little in the way of classification, although this may change in higher dimensions. The values for *Insurance_History_5* are tightly clustered around zero, this is further evidence that min-max scaling was used. No significant correlations are present.

Figure 2: Response versus employment variables



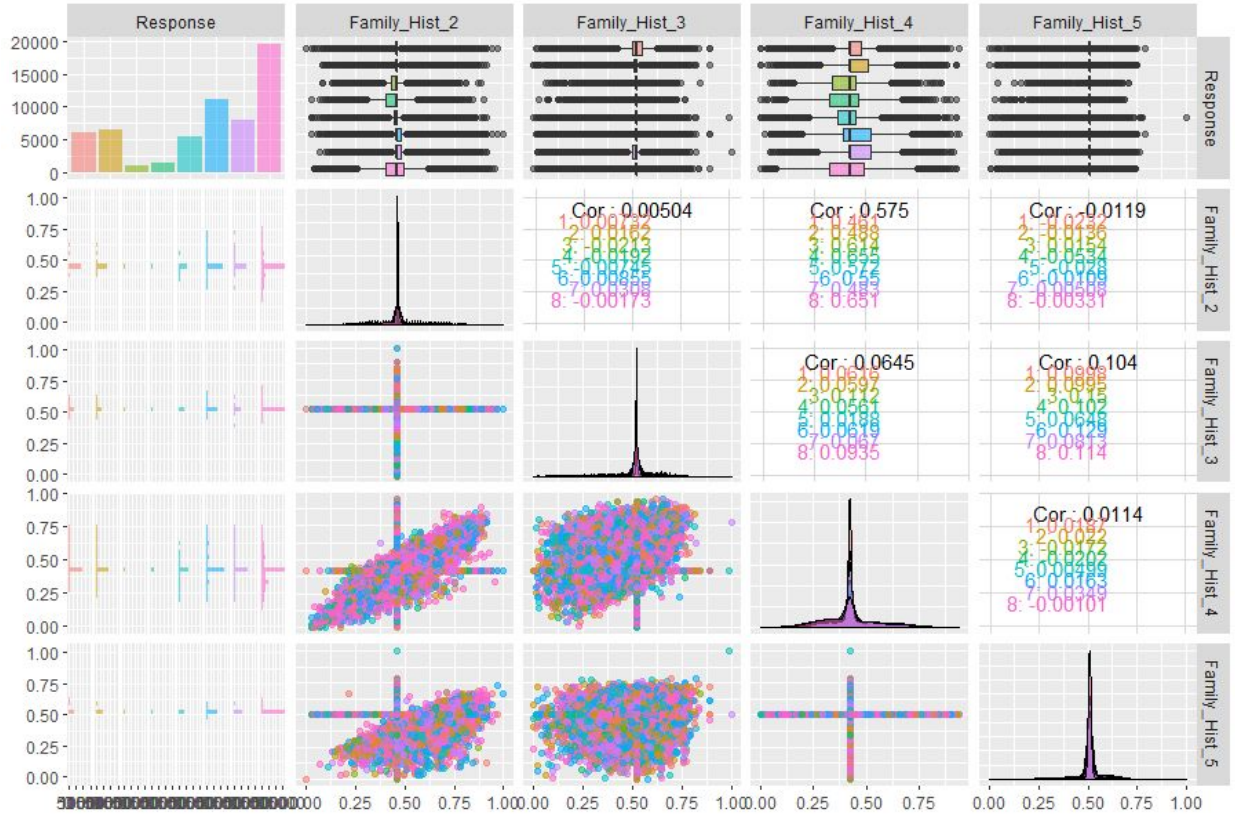
All three employment variables are right skewed which is expected given what we are fairly sure at this point that min-max scaling was used to anonymize the data. The variables are marginally correlated which is not unsurprising as they all relate to the employment history of the applicant.

Figure 3: Response versus physiological variables



Here we begin to see the first real class separation with respect to any continuous variables. Moreover, each of the physiological variables is approximately symmetrically distributed and the classes appear to have different modes. Of these variables it appears that *BMI* is the most indicative of the response level. Correlations between *Ht, Wt,* and *BMI* are present and unsurprising.

Figure 4: Response versus family history variables

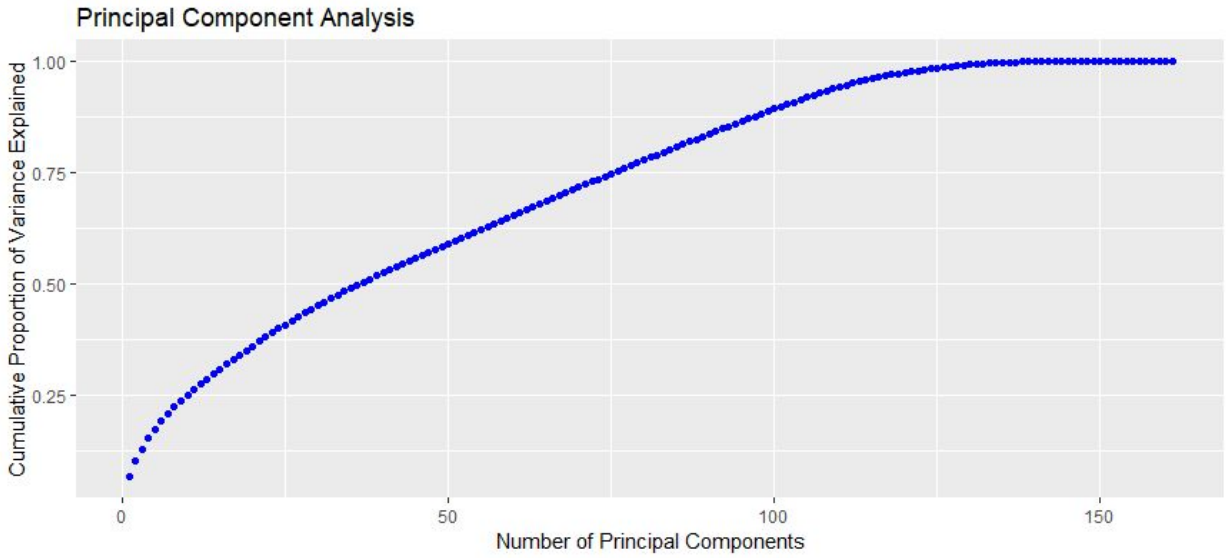


The family history variables are tightly clustered around 0.5 and do not appear to differ much between response levels. Only *Family_Hist_3* and *Family_Hist_4* share any sort of meaningful correlation, suggesting that this part of the survey covers a wide range of issues.

These plots of the continuous variables give us an idea of how the data is distributed. It appears that the physiological variables contain the most information as to the level of the response variable whereas the distributions for each level of the response variable for the other continuous variables appear very similar. However, visual differences may arises in higher dimensions. Moreover, given the results of the *tableone* in the appendix, one should not be too quick to discount the importance of any of the continuous variables in predicting class membership.

To gain a better idea of how information is distributed throughout the data set and whether dimensionality reduction should be should be meaningfully considered, we performed a principal component analysis (PCA).

Figure 5: Principal component analysis



As shown in the plot, initially, adding a few components explains more variance. However, subsequently, the proportion of variance explained increases linearly before plateauing as the maximum number of features (161) is approached. That is, there is no “elbow” in the graph at a low number of components and so any benefits from dimensionality reduction are likely to be modest and more easily attained through the use of methods which make use of regularization.

In addition, 7 components explain ~21% of the variance in the data, and approximately 37 components explain about 50% of the variance in the data. The fact that 7 components only explain a small amount of the variance is critical as LDA can be used easily to reduce the dimension of the data to seven (number of classes - 1) while maximizing the inter-class separation of the response variable whereas PCA maximizes the explained variance of the covariates. Thus, this confirms our view that PCA is not appropriate for dimensionality reduction, and so we examine LDA and QDA instead. Note that this will be made more clear in the *Methodology* section.

Methodology

Overview

Classification/regression trees were by far the most popular approach in this competition. These methods are well justified by their own merit, but are also likely to perform better than traditional regression methods given the large number of discrete variables and the min-max scaling of continuous variables discussed earlier. The package *XGBoost* was used to fit all boosted

classification and regression trees while the *mlr* wrapper package was used to facilitate easier training and tuning of these models.

Recall that the target variable is an ordinal variable with eight levels. As such there are several ways to approach this problem using classification/regression trees. The regression approach is to treat the target variable as continuous as then determine classification thresholds using cross validation. This approach was motivated by Kaggle user Giuseppe Casalicchio (2015). The two classification approaches we used were eight 1-vs-all classification trees (one for each level of the target variable) and a simple approach where a classification tree was fit on all eight levels. Note that these two classification approaches lend themselves easily to weighting the observations. While there are many ways to weight observations we used the simple approach of weighting each class by the sum of the negative instances divided by the sum of the positive instances. This ensured that the total weight of each class in the training data was equal.

However, it would be boring if we stopped here. Motivated by local linear regression we explored methods to both improve this method and extend it to other methods, which in our case is boosted trees. Recall that local linear regression involves partitioning the covariate space and then fitting linear regression models to each partition. Predictions from new observations are then obtained from the linear model fit on the partition to which the new observations belongs. One of the main issues with this method is that there is no clear way of partitioning the covariate space. To overcome this we propose using clustering algorithms, not to partition the sample space, but to fit one model for each cluster using the distance of each observation from that cluster as case weights (we will refer to this as a soft cluster-based partition). Note that it is then necessary that the clustering algorithm must emit a vector of distances (or perhaps probabilities) for each observation. New observations are then classified according to a weighted average, based on their proximity to each cluster (note again probabilities could be used in lieu of distances).

To our knowledge this approach of extending local linear regression to other models by using what we refer to as a soft clustered-based partition is largely if not completely absent from the literature. In the case of boosted trees we propose the following two theoretical justifications for why this should improve model performance. First, by fitting weighted boosted trees to each cluster we allow for the possibility of variable importance to vary throughout the covariate space. Second, by fitting a boosted tree to each cluster we are focusing on efficient partitions in the densest part of the covariate space whereas a single boosted tree fit on the entire sample space will greedly partition the entire sample space which is more likely to lead to overfitting/poor partitions in these dense clusters of observations. Finally, in the case of any model employed using a clustered-based partition one is likely to obtain several models which are not perfectly correlated in which case averaging their predictions is employing a simple ensemble model which is likely to improve upon the individual models.

The dataset used in this competition is diverse in categorical, discrete, and continuous variables. Somewhat ironically this diversity makes it quite difficult to discern an appropriate

distance metric. For example if one were to use any of the L^n-norms the min-max scaling of the continuous variables would mean that all else being equal the distance between the lightest and heaviest individuals would be the same as the distance between two individuals whose only difference was a single medical keyword, which hardly seems appropriate. This problem of how to account for both discrete and continuous variables in unsupervised clustering is well documented and often dealt with through a Mahalanobis distance metric. However, we suggest a different approach using discriminant analysis to obtain a distance metric. Recall that discriminant analysis projects the data onto a C-1 dimensional subspace (where C is the number of classes of the response variable) moreover this subspace maximizes the inter-class separation. Since each dimension of this subspace is a linear combination of the original variables it may now, without loss of generality, be considered a continuous variable and thus any of the L^n-norms are now appropriate. An astute reader will note that the data is not gaussian and may therefore argue that LDA is not appropriate. However, Hastie et al. (2017, p. 110) show that it is possible to derive the directions in discriminant analysis via a least squares without assuming normality. As we are only interested in the subspace discriminant analysis emits the assumption of normality is not needed.

Having overcome the difficulties in obtaining a distance metric we now summarize how we will use a soft cluster-based partition to fit boosted trees to the data. This will be a three step process and the training data was partitioned twice to allow for nested cross validation. At each step the partition was a ⅝-⅛ split for the training and testing data respectively. The evaluation metric throughout was the quadratic weighted kappa. The three steps were as follows:

1. Use cross validation to choose between linear discriminant analysis and quadratic discriminant analysis to obtain a subspace to be used by the clustering algorithm.
2. Fit a k-means clustering algorithm to the data projected onto the subspace returned from the previous step.
3. Fit boosted trees to each cluster using weights from the previous step.

Figure 6: Model fitting methodology and framework

Step 1: Obtaining the Subspace

As with most machine learning algorithms, it becomes difficult to implement discriminant analysis when the data exhibits a high degree of multicollinearity between features (even with high but not perfect correlation, there is numerical instability in terms of inverting the covariance matrix). In addition, since the data is high dimensional, these issues are naturally present and must be resolved somehow. While it is preferable to perform LDA on the entirety of the dataset, when initially running LDA on the cleaned data, R outputted errors such as “Variables are collinear” and “Variable x appears to be constant within groups”. To mitigate these errors, we created the *removeCorrelatedFeatures* function (in *codeBank.R*). This function prepares the cleaned data for LDA by performing the following steps:

- In the entire data set:
    - Removes “near-constant” features for which less than 0.5% of the observations differ from the mode value
    - Removes highly correlated features for which the pairwise absolute correlation exceeds 0.85
- Looping through each of the eight classes:
    - Removes “near-constant” and highly correlated features using the methodology as above within each class

After applying this function, the number of features was reduced from 161 to 98. About one-third of the medical history and medical keyword features were removed, along with some of the

interaction features as well. Although the number of features is reduced by ~39%, this ensures that an LDA model can be fit on the data.

To compare between LDA and QDA, both models were fit on the *train_lda* split and evaluated on the *test_lda* split. The table below shows the model performance on the *test_lda* split using the evaluation metric quadratic weighted kappa.

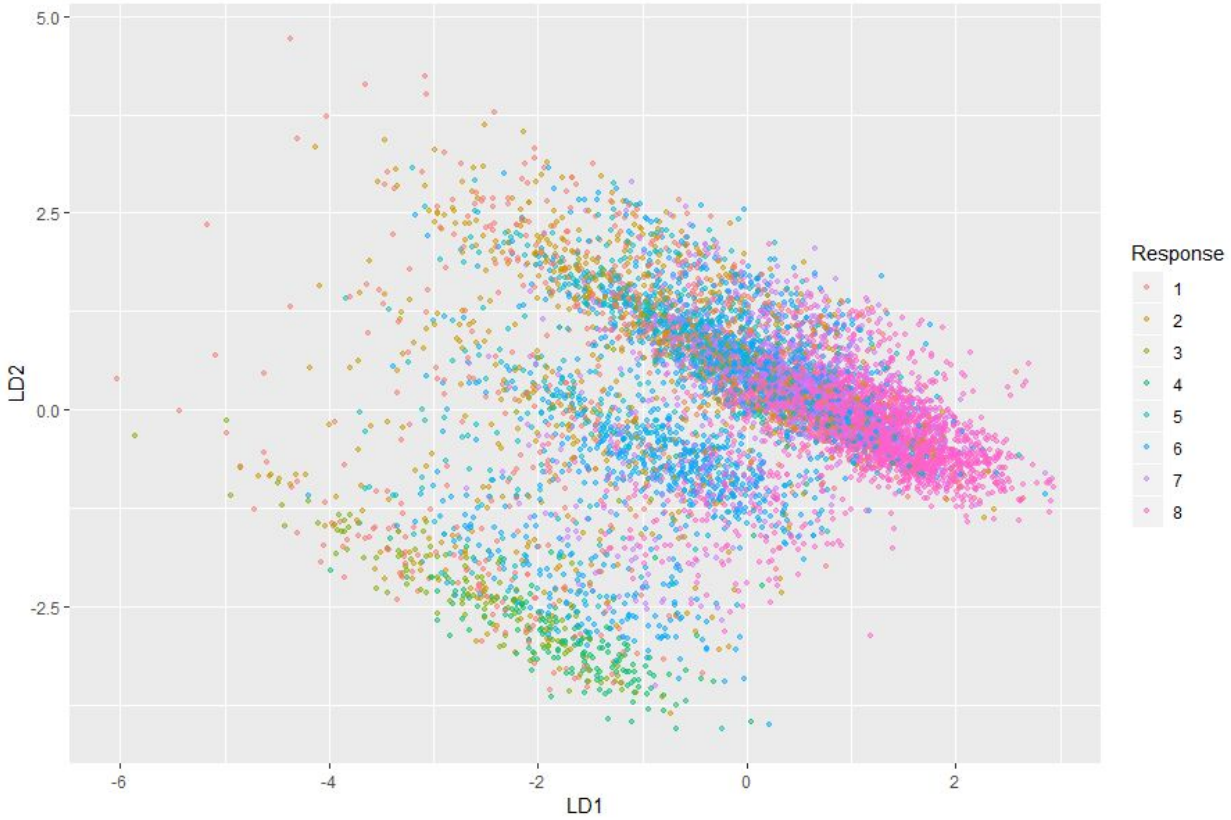
Table 3: LDA and QDA model performance comparison

Method	**Model performance on *test_lda***
LDA	0.4549253
QDA	0.3694868

Thus, LDA offers superior performance compared to QDA on the *test_lda* split, and so LDA was selected over QDA. Henceforth, all clustering models are subsequently trained using the projection onto the LDA subspace.

Although the LDA projection is a 7-dimensional subspace, for visualization purposes, we project the data onto the 2-dimensional subspace defined by the first and second linear discriminants. Note that these are the two linear combinations of the original covariates which best separate the levels of the response variable.

Figure 7: 2-dimensional LDA subspace projection



As shown in the figure, we can see the clustering and “pockets” of each class (for example, the cluster of class 8 in the middle-right, the cluster of class 4 in the bottom-left, etc.). This demonstrates our idea that the LDA projection helps not only with class separation, but with dimensionality reduction as well, which is an incredibly useful tool for improving the speed and performance of the machine learning algorithms applied subsequently (as detailed in later sections).

Regularized discriminant analysis (RDA) was also considered (essentially a penalized version of LDA and QDA) as an option for this step. However, using the *rda* function in the *klaR* package, there is no way (by default) to extract the change of basis matrix for the RDA subspace projection. To obtain the RDA subspace projection, we would need to examine the source code and determine how to get the pooled covariance matrix and then perform an eigen-decomposition on the pooled covariance matrix to get the change of basis matrix. This was thought to fall both outside of our area of expertise and the scope of this project. However, considering that our results are adequate using LDA (as seen in future sections), it is worth noting that they could be optimized further by using RDA, as long as we were able to obtain the pooled covariance matrix returned by RDA.

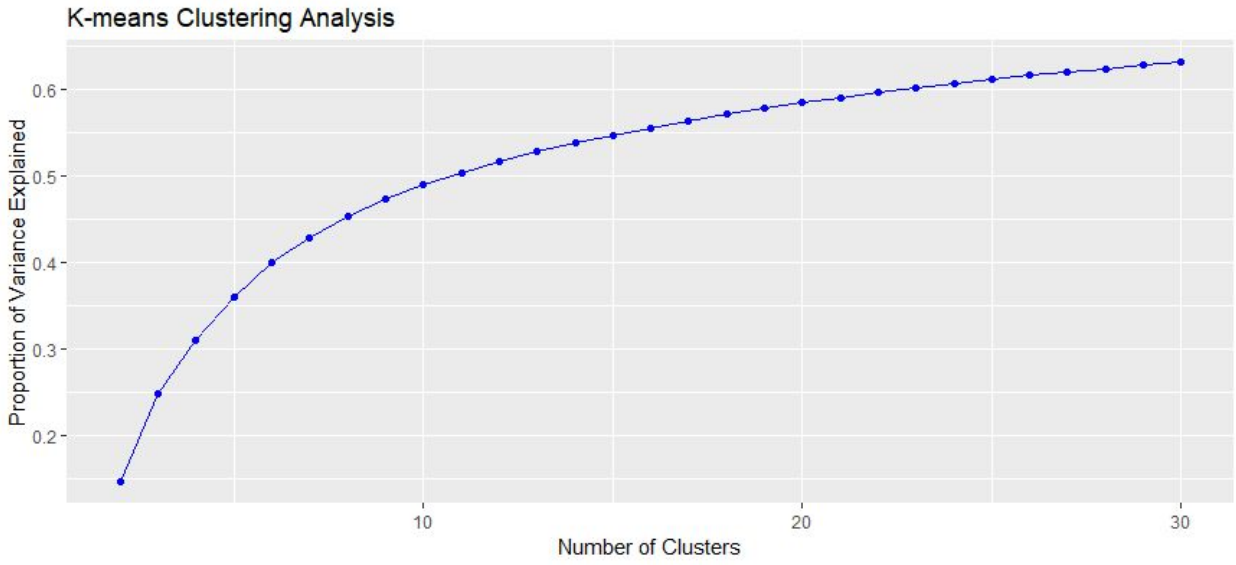
Step 2: Clustering

The next step in our approach is to apply k-means clustering for the soft partition method. To make sure that the LDA subspace projections are consistent for the clustering among the training and testing data sets, the methodology for the k-means clustering algorithm is as follows.

1. Fit an LDA model on all the training data, and then project the training and testing data onto the fitted 7-dimensional LDA subspace .
2. Fit a k-means clustering model (using the Euclidean distance) on the combined projection of the training and testing data from step 1, for different values of k.
3. Choose the optimal value of k using an appropriate method.
4. Retrain the k-means clustering model on the combined data using the optimal k from step 3.
5. Compute distances to each cluster for each data point in the combined data.
6. Save distances from step 5 to be used as weights for the training and testing data.

To determine the optimal number of clusters, we first investigated the “elbow” method. The plot below shows the proportion of variance explained versus the number of clusters for k-means clustering.

Figure 8: K-means clustering analysis



From the plot above, there is no clear “elbow”. The proportion of variance explained increases sharply at first, but then the marginal increase in variance explained as another cluster is added begins to decrease once 10 clusters are present. In addition, 10 clusters explains ~49% of the variance, while doubling the number of clusters only adds an additional 10%. Hence, keeping in

mind the purpose of the clustering (soft partition method for trees) and weighing the computational restrictions, we opted to use 10 clusters. Increasing the number of clusters would likely benefit our subsequent tree-based methods (though they may also be prone to overfitting if too many clusters are utilized), but due to the computational cost, we decided it would be better to use a smaller number of clusters and fit the models in a feasible amount of time to properly demonstrate the application of our technique.

In addition, we attempted to use the *pamk* function in the *fpc* package: partitioning around medoids to estimate the optimal number of clusters using average silhouette width. Essentially, this metric measures how similar an observation is to its own cluster compared to other clusters (using an appropriate distance metric such as the Euclidean distance). This can be used to generate a silhouette plot (average silhouette width for each number of clusters) and choose the optimal number of clusters (the one with the highest average silhouette width). However, due to computational restrictions, we were not able to utilize this method as it requires calculating the pairwise distance between all points (in R, given the size of our data, this requires more than 12GB of RAM!). Hence, we had to rely on the naive plotting method to select the number of clusters. If we had more computing resources, we could have further optimized the number of clusters, but this is outside the scope of the project. Nevertheless, it is important to assess the possibility of other techniques which differ from our methodology.

Step 3: Fitting Boosted Trees

Having obtained our soft cluster-based partition we now proceed to fit weighted boosted trees to each cluster. We will also fit boosted trees on the entire sample space for comparison. All boosted trees were fit using three fold cross validation over a four dimensional grid with a resolution of 4. During cross validation only 30 rounds of boosting was used. The evaluation metric was again the quadratic weighted Kappa. This is not a default metric in either the *XGBoost* or *mlr* packages and had to be implemented as a custom evaluation metric through both packages. The code for this implementation may be found in *codeBank.R*. The optimal parameters from cross validation were then used to fit a boosted tree using 100 rounds of boosting.

We acknowledge that the cross validation protocol described above is not ideal. However, it was necessary to impose constraints on the cross validation protocol as each model took over 12 hours to fit using this protocol. This is largely due to the time it takes to complete an exhaustive search over a four dimensional grid with a resolution of four. There are two potential remedies to this problem. The first is to use a random hyperparameter search with a fixed number of iterations. This would allow for a finer search of the grid but would at best achieve a local optimum. The second is to run *XGBoost* on a gpu which would speed up model fitting times by running the cross validation in parallel. However, to do this one must compile *XGBoost* with gpu support from the command line which (as we learned) is well out of our domain of expertise.

As previously mentioned three types of boosted trees were considered for analysis. Unfortunately, due to the model fitting times outlined above it was not feasible to fit a 1-vs-all model for each level of the target variable using a soft cluster-based partition.

Prior to assessing the performance of the soft cluster-based partition approach we provide the results from the non-partition based approach for comparison. Classification for the 1-vs-all models was determined according to the class with the largest probability after the predicted probabilities from each model were normalized using the softmax function.

Table 4: Quadratic weighted kappa for non-partition methods

	Regression Approach	**8 Class Levels Approach**	**1-vs-All Approach**
No Class Weights	0.6234278	0.5477858	0.5268869
Class Weights	NA	0.5550022	0.4801891

The regression approach is clearly the superior approach. This is likely because target variable is ordinal. That is while it is categorical nature there is an ordering to the outcomes and the regression approach is the only one out of the three that makes use of this relationship. We also note that is is not necessary to redo the regression approach using class weights as all that should change is the classification thresholds determined by cross validation not the actual performance of the model. Modeling all class levels at once is superior to the 1-vs-all approach regardless of class weights. It is unclear what effect class weights have over all and it appears to be dependant on the approach used.

Table 5: Quadratic weighted kappa for cluster-based partition methods

	Regression Approach	**8 Class Levels Approach**
No Class Weights	0.6451581	0.601293
Class Weights	NA	0.5931735

Again the Regression approach is clearly the superior approach, likely for the same reasons as outlined above. What is more interesting is that implementing these methods through a cluster-based partition approach has significantly improved their performance ranging from 0.02 to 0.05 in absolute improvement in quadratic weighted Kappa and from 3%-9% improvement in relative terms.

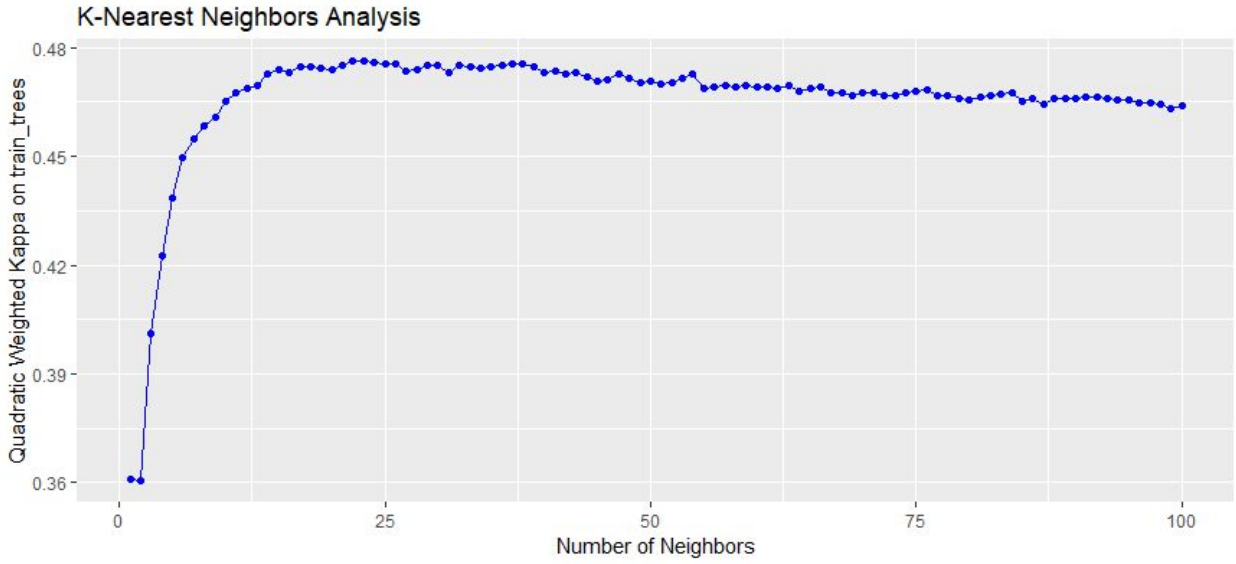
Alternative Method: K-Nearest Neighbors

To understand how effective the modeling approach we used was, we fit a k-nearest neighbours (KNN) model to the same data as the XGBoost models. We then compared the performance of

the KNN model to the classification/regression tree models discussed in the previous section. Since KNN does not take the ordinality of the response variable into account, we would expect this method to perform worse than the more sophisticated models outlined in the last section. However, this method is examined to further demonstrate our approach of dimensionality reduction using LDA.

The model is trained and evaluated on the *train_trees* split using 3-fold cross-validation (again 3-fold was chosen instead of 10-fold due to computational restrictions) to select the optimal number of neighbours. Below, we plot the quadratic weighted kappa for the *train_trees* split for different numbers of neighbours.

Figure 9: K-nearest neighbors analysis



From the plot, we can see that initially, the model performance improves substantially as the number of neighbors is increased. However, the quadratic weighted kappa reaches a maximum value of 0.4764164 when k = 22, and then decreases steadily as k increases.

Therefore, k = 22 is selected as the optimal number of neighbours, and the model is re-trained on the *train_trees* split and evaluated on the *test_trees* split. The quadratic weighted kappa score for the KNN model with k = 22 trained on the *train_trees* split and evaluated on the *test_trees* split is 0.4712069. Since this score is much lower than the score for the boosted trees, we decided not to pursue this model further (we did not refit the model on the training data and submit predictions to Kaggle for the testing data). This result shows that more sophisticated methods outlined in the previous section well outperform a naive approach.

Final Model

Each approach to predicting the target variable was improved upon when cluster-based partition methods were used. Additionally, treating the target variable as a continuous variable and then using cross validation to determine the optimal classification thresholds proved to be the superior method both when implemented on the data as a whole and using cluster-based partitioning. As such the regression approach using cluster-based partitions was retrained on the entire training set. Predictions on the test set were then submitted to Kaggle and the results from the private and public leaderboards are presented below.

Table 6: Private and public leaderboard results

	Quadratic Weighted Kappa	**Winning Score**	**Leaderboard Rank**	**Percentile**
Private Leaderboard (70% of Data)	0.66225	0.67938	1114	57^{th}
Public Leaderboard (30% of Data)	0.65826	0.68325	1159	56^{th}

First we note that the model has improved substantially from being trained on the additional data and is only 0.015 points worse in quadratic weighted kappa from the model that won this competition. Of course this only places us in the top half of competitors, but alas this kind of clustering among competitors is fairly common in Kaggle competitions.

Conclusion

While our model did not top the leaderboard, it was a respectable first attempt as most competitors ahead of us submitted tens if not hundreds of predictions. We have shown that, at least in this case, a cluster-based partition to regression and classification trees improves their performance with respect to quadratic weighted kappa. Moreover, as this metric explicitly accounts for random agreement between the model and the true class labels we are confident that these results will hold up for other classification metrics such as accuracy or AUC. We have hypothesised theoretical justifications for this improvement such as encouraging the greedy algorithms employed by regression and classification trees to make splits where the data is densest.

It is our hope that the methods described in this paper will pique the interest of those better equipped to research them.

References

Casalicchio, G. (2015). Use the mlr package (scores 0.649). Retrieved November 15, 2018, from https://www.kaggle.com/casalicchio/use-the-mlr-package-scores-0-649

Hastie, T., Tibshirani, R., & Friedman, J. H. (2017). *The elements of statistical learning: Data mining, inference, and prediction*. New York, NY: Springer.

Prudential life insurance assessment. (2015). Retrieved November 15, 2018, from https://www.kaggle.com/c/prudential-life-insurance-assessment

Appendix 1: Tableones

Table 7: Tableone of continuous variables

Response	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**p-value**
Count	6207	6552	1013	1428	5432	11233	8027	19489	
Product_Info_4 (mean (sd))	0.25 (0.25)	0.25 (0.24)	0.21 (0.23)	0.26 (0.23)	0.25 (0.26)	0.34 (0.28)	0.35 (0.28)	0.40 (0.30)	<0.001
Ins_Age (mean (sd))	0.49 (0.20)	0.46 (0.19)	0.36 (0.19)	0.34 (0.17)	0.40 (0.20)	0.43 (0.19)	0.43 (0.19)	0.34 (0.19)	<0.001
Ht (mean (sd))	0.71 (0.07)	0.72 (0.08)	0.71 (0.07)	0.71 (0.07)	0.71 (0.08)	0.72 (0.07)	0.71 (0.07)	0.69 (0.07)	<0.001
Wt (mean (sd))	0.32 (0.10)	0.34 (0.11)	0.32 (0.09)	0.26 (0.06)	0.35 (0.09)	0.31 (0.07)	0.29 (0.07)	0.24 (0.06)	<0.001
BMI (mean (sd))	0.51 (0.15)	0.55 (0.16)	0.52 (0.12)	0.42 (0.08)	0.57 (0.13)	0.49 (0.10)	0.46 (0.08)	0.39 (0.07)	<0.001
Employment_Info_1 (mean (sd))	0.07 (0.08)	0.08 (0.09)	0.06 (0.07)	0.06 (0.06)	0.07 (0.07)	0.08 (0.08)	0.09 (0.09)	0.08 (0.08)	<0.001
Employment_Info_4 (mean (sd))	0.01 (0.04)	0.01 (0.03)	0.00 (0.02)	0.00 (0.02)	0.00 (0.02)	0.01 (0.03)	0.01 (0.03)	0.00 (0.03)	<0.001
Employment_Info_6 (mean (sd))	0.34 (0.32)	0.35 (0.32)	0.24 (0.26)	0.26 (0.27)	0.28 (0.28)	0.36 (0.33)	0.37 (0.33)	0.34 (0.32)	<0.001
Insurance_History_5 (mean (sd))	0.00 (0.01)	0.00 (0.01)	0.00 (0.02)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	<0.001
Family_Hist_2 (mean (sd))	0.48 (0.10)	0.48 (0.10)	0.45 (0.11)	0.45 (0.12)	0.46 (0.11)	0.48 (0.11)	0.48 (0.11)	0.46 (0.12)	<0.001
Family_Hist_3 (mean (sd))	0.51 (0.11)	0.51 (0.10)	0.50 (0.09)	0.50 (0.08)	0.51 (0.09)	0.51 (0.09)	0.50 (0.10)	0.52 (0.08)	<0.001
Family_Hist_4 (mean (sd))	0.45 (0.13)	0.45 (0.13)	0.41 (0.13)	0.41 (0.14)	0.43 (0.13)	0.45 (0.13)	0.46 (0.13)	0.42 (0.14)	<0.001
Family_Hist_5 (mean (sd))	0.50 (0.09)	0.50 (0.08)	0.50 (0.07)	0.49 (0.07)	0.50 (0.08)	0.50 (0.07)	0.50 (0.08)	0.50 (0.06)	<0.001

Table 8: Tableone of discrete variables

Response	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**p-value**
Count	6207	6552	1013	1428	5432	11233	8027	19489	
Medical_History_1 (mean (sd))	4.93 (8.59)	5.86 (10.29)	7.89 (14.11)	7.70 (12.51)	7.19 (12.56)	7.68 (13.12)	7.31 (11.53)	8.50 (12.77)	<0.001
Medical_History_10 (mean (sd))	227.92 (15.63)	227.81 (16.01)	227.30 (19.33)	228.23 (12.50)	228.13 (13.75)	226.90 (21.48)	228.79 (6.54)	228.92 (4.15)	<0.001
Medical_History_15 (mean (sd))	117.11 (61.41)	119.84 (55.66)	19.45 (45.49)	20.74 (44.79)	123.81 (42.51)	120.00 (48.82)	127.77 (40.88)	125.20 (32.02)	<0.001
Medical_History_24 (mean (sd))	12.20 (28.82)	11.63 (26.25)	9.03 (12.85)	8.86 (12.24)	10.78 (22.23)	10.45 (21.61)	11.41 (24.39)	10.05 (18.79)	<0.001
Medical_History_32 (mean (sd))	0.19 (4.21)	0.33 (6.72)	0.19 (2.17)	0.07 (0.83)	0.23 (5.73)	0.20 (3.71)	0.36 (7.88)	0.17 (5.41)	0.123
Medical_Keyword_1 (mean (sd))	0.08 (0.27)	0.07 (0.25)	0.04 (0.19)	0.02 (0.15)	0.05 (0.22)	0.05 (0.21)	0.04 (0.19)	0.02 (0.14)	<0.001
Medical_Keyword_2 (mean (sd))	0.01 (0.10)	0.01 (0.10)	0.01 (0.09)	0.01 (0.12)	0.01 (0.10)	0.01 (0.09)	0.01 (0.09)	0.01 (0.09)	0.013
Medical_Keyword_3 (mean (sd))	0.15 (0.35)	0.17 (0.37)	0.03 (0.17)	0.00 (0.05)	0.06 (0.23)	0.05 (0.22)	0.00 (0.04)	0.00 (0.04)	<0.001
Medical_Keyword_4 (mean (sd))	0.02 (0.15)	0.02 (0.14)	0.01 (0.10)	0.01 (0.11)	0.02 (0.14)	0.01 (0.12)	0.02 (0.13)	0.01 (0.09)	<0.001
Medical_Keyword_5 (mean (sd))	0.01 (0.10)	0.01 (0.10)	0.01 (0.10)	0.01 (0.12)	0.01 (0.10)	0.01 (0.09)	0.01 (0.10)	0.01 (0.08)	<0.001
Medical_Keyword_6 (mean (sd))	0.01 (0.11)	0.01 (0.11)	0.01 (0.09)	0.00 (0.07)	0.01 (0.11)	0.01 (0.10)	0.01 (0.12)	0.01 (0.12)	0.002
Medical_Keyword_7 (mean (sd))	0.02 (0.13)	0.02 (0.13)	0.01 (0.12)	0.01 (0.09)	0.02 (0.14)	0.02 (0.13)	0.01 (0.11)	0.01 (0.10)	<0.001
Medical_Keyword_8 (mean (sd))	0.01 (0.10)	0.01 (0.09)	0.01 (0.12)	0.02 (0.13)	0.01 (0.11)	0.01 (0.08)	0.01 (0.09)	0.01 (0.11)	<0.001
Medical_Keyword_9 (mean (sd))	0.02 (0.12)	0.01 (0.10)	0.00 (0.07)	0.00 (0.07)	0.01 (0.09)	0.01 (0.08)	0.00 (0.06)	0.00 (0.06)	<0.001
Medical_Keyword_10 (mean (sd))	0.05 (0.22)	0.05 (0.22)	0.03 (0.18)	0.01 (0.12)	0.05 (0.21)	0.04 (0.20)	0.04 (0.19)	0.02 (0.15)	<0.001

Medical_Keyword_11 (mean (sd))	0.07 (0.26)	0.07 (0.25)	0.05 (0.22)	0.05 (0.22)	0.07 (0.25)	0.05 (0.23)	0.05 (0.22)	0.05 (0.22)	<0.001
Medical_Keyword_12 (mean (sd))	0.02 (0.13)	0.02 (0.13)	0.01 (0.10)	0.01 (0.10)	0.01 (0.10)	0.01 (0.09)	0.01 (0.08)	0.01 (0.08)	<0.001
Medical_Keyword_13 (mean (sd))	0.01 (0.10)	0.01 (0.09)	0.01 (0.09)	0.01 (0.08)	0.01 (0.09)	0.01 (0.09)	0.01 (0.08)	0.00 (0.05)	<0.001
Medical_Keyword_14 (mean (sd))	0.01 (0.11)	0.01 (0.09)	0.00 (0.06)	0.00 (0.04)	0.01 (0.10)	0.01 (0.09)	0.01 (0.10)	0.00 (0.07)	<0.001
Medical_Keyword_15 (mean (sd))	0.36 (0.48)	0.33 (0.47)	0.19 (0.39)	0.10 (0.30)	0.25 (0.43)	0.22 (0.42)	0.31 (0.46)	0.01 (0.11)	<0.001
Medical_Keyword_16 (mean (sd))	0.03 (0.16)	0.02 (0.15)	0.01 (0.08)	0.01 (0.08)	0.01 (0.11)	0.01 (0.11)	0.01 (0.10)	0.01 (0.09)	<0.001
Medical_Keyword_17 (mean (sd))	0.01 (0.12)	0.01 (0.09)	0.00 (0.07)	0.01 (0.11)	0.01 (0.09)	0.01 (0.08)	0.01 (0.09)	0.01 (0.10)	<0.001
Medical_Keyword_18 (mean (sd))	0.02 (0.12)	0.01 (0.11)	0.01 (0.10)	0.00 (0.06)	0.01 (0.08)	0.01 (0.08)	0.01 (0.08)	0.00 (0.07)	<0.001
Medical_Keyword_19 (mean (sd))	0.02 (0.13)	0.01 (0.12)	0.00 (0.07)	0.01 (0.11)	0.01 (0.10)	0.01 (0.10)	0.01 (0.09)	0.00 (0.07)	<0.001
Medical_Keyword_20 (mean (sd))	0.01 (0.08)	0.01 (0.09)	0.00 (0.03)	0.01 (0.08)	0.01 (0.08)	0.01 (0.09)	0.01 (0.10)	0.01 (0.09)	0.045
Medical_Keyword_21 (mean (sd))	0.02 (0.15)	0.02 (0.12)	0.01 (0.11)	0.02 (0.13)	0.01 (0.12)	0.02 (0.12)	0.02 (0.12)	0.01 (0.11)	<0.001
Medical_Keyword_22 (mean (sd))	0.05 (0.22)	0.05 (0.22)	0.03 (0.17)	0.02 (0.12)	0.04 (0.20)	0.04 (0.20)	0.05 (0.21)	0.02 (0.15)	<0.001
Medical_Keyword_23 (mean (sd))	0.16 (0.37)	0.12 (0.32)	0.25 (0.43)	0.14 (0.35)	0.08 (0.27)	0.13 (0.34)	0.08 (0.26)	0.05 (0.23)	<0.001
Medical_Keyword_24 (mean (sd))	0.04 (0.19)	0.03 (0.18)	0.01 (0.11)	0.00 (0.06)	0.02 (0.15)	0.02 (0.15)	0.02 (0.14)	0.01 (0.07)	<0.001
Medical_Keyword_25 (mean (sd))	0.13 (0.34)	0.13 (0.34)	0.08 (0.27)	0.06 (0.23)	0.10 (0.31)	0.11 (0.31)	0.11 (0.32)	0.04 (0.19)	<0.001
Medical_Keyword_26 (mean (sd))	0.01 (0.11)	0.02 (0.13)	0.01 (0.11)	0.01 (0.12)	0.01 (0.12)	0.02 (0.13)	0.01 (0.11)	0.01 (0.10)	<0.001
Medical_Keyword_27 (mean (sd))	0.02 (0.15)	0.02 (0.14)	0.01 (0.08)	0.01 (0.07)	0.01 (0.12)	0.01 (0.11)	0.01 (0.11)	0.01 (0.08)	<0.001

Medical_Keyword_28 (mean (sd))	0.02 (0.14)	0.02 (0.14)	0.01 (0.10)	0.01 (0.11)	0.01 (0.12)	0.02 (0.13)	0.02 (0.13)	0.01 (0.10)	<0.001
Medical_Keyword_29 (mean (sd))	0.01 (0.12)	0.02 (0.12)	0.01 (0.09)	0.01 (0.09)	0.01 (0.12)	0.01 (0.10)	0.01 (0.11)	0.01 (0.10)	<0.001
Medical_Keyword_30 (mean (sd))	0.04 (0.20)	0.03 (0.17)	0.02 (0.12)	0.01 (0.12)	0.02 (0.14)	0.02 (0.13)	0.02 (0.14)	0.03 (0.16)	<0.001
Medical_Keyword_31 (mean (sd))	0.02 (0.15)	0.02 (0.13)	0.01 (0.12)	0.01 (0.10)	0.01 (0.11)	0.01 (0.11)	0.01 (0.09)	0.01 (0.07)	<0.001
Medical_Keyword_32 (mean (sd))	0.02 (0.15)	0.02 (0.14)	0.03 (0.17)	0.03 (0.16)	0.02 (0.13)	0.02 (0.14)	0.02 (0.14)	0.02 (0.14)	0.065
Medical_Keyword_33 (mean (sd))	0.03 (0.16)	0.03 (0.18)	0.02 (0.13)	0.01 (0.11)	0.02 (0.15)	0.03 (0.17)	0.03 (0.17)	0.01 (0.11)	<0.001
Medical_Keyword_34 (mean (sd))	0.03 (0.16)	0.03 (0.18)	0.02 (0.14)	0.01 (0.11)	0.02 (0.15)	0.02 (0.13)	0.02 (0.14)	0.02 (0.13)	<0.001
Medical_Keyword_35 (mean (sd))	0.01 (0.10)	0.01 (0.11)	0.01 (0.09)	0.00 (0.06)	0.01 (0.09)	0.01 (0.09)	0.01 (0.11)	0.00 (0.03)	<0.001
Medical_Keyword_36 (mean (sd))	0.02 (0.13)	0.02 (0.12)	0.01 (0.09)	0.01 (0.10)	0.01 (0.11)	0.01 (0.10)	0.01 (0.10)	0.01 (0.08)	<0.001
Medical_Keyword_37 (mean (sd))	0.12 (0.32)	0.09 (0.29)	0.08 (0.27)	0.04 (0.19)	0.07 (0.25)	0.06 (0.24)	0.06 (0.24)	0.05 (0.22)	<0.001
Medical_Keyword_38 (mean (sd))	0.03 (0.17)	0.01 (0.11)	0.00 (0.07)	0.00 (0.00)	0.00 (0.06)	0.01 (0.10)	0.00 (0.00)	0.00 (0.01)	<0.001
Medical_Keyword_39 (mean (sd))	0.01 (0.12)	0.01 (0.11)	0.01 (0.08)	0.02 (0.13)	0.01 (0.11)	0.01 (0.11)	0.01 (0.11)	0.02 (0.13)	0.001
Medical_Keyword_40 (mean (sd))	0.09 (0.29)	0.09 (0.28)	0.07 (0.25)	0.08 (0.27)	0.06 (0.24)	0.05 (0.22)	0.05 (0.21)	0.04 (0.20)	<0.001
Medical_Keyword_41 (mean (sd))	0.01 (0.11)	0.01 (0.11)	0.00 (0.05)	0.00 (0.05)	0.01 (0.08)	0.01 (0.11)	0.01 (0.11)	0.01 (0.09)	<0.001
Medical_Keyword_42 (mean (sd))	0.09 (0.29)	0.07 (0.25)	0.06 (0.24)	0.04 (0.20)	0.05 (0.22)	0.04 (0.19)	0.03 (0.18)	0.03 (0.17)	<0.001
Medical_Keyword_43 (mean (sd))	0.03 (0.16)	0.02 (0.12)	0.01 (0.08)	0.00 (0.05)	0.01 (0.10)	0.01 (0.10)	0.01 (0.10)	0.01 (0.07)	<0.001
Medical_Keyword_44 (mean (sd))	0.01 (0.09)	0.01 (0.11)	0.01 (0.08)	0.01 (0.10)	0.01 (0.09)	0.01 (0.09)	0.01 (0.09)	0.01 (0.07)	<0.001

Medical_Keyword_45 (mean (sd))	0.01 (0.12)	0.01 (0.11)	0.01 (0.08)	0.01 (0.11)	0.01 (0.12)	0.01 (0.11)	0.02 (0.12)	0.01 (0.12)	0.195
Medical_Keyword_46 (mean (sd))	0.02 (0.13)	0.02 (0.12)	0.01 (0.09)	0.00 (0.05)	0.01 (0.12)	0.01 (0.10)	0.01 (0.08)	0.00 (0.05)	<0.001
Medical_Keyword_47 (mean (sd))	0.04 (0.20)	0.03 (0.18)	0.02 (0.13)	0.01 (0.12)	0.02 (0.13)	0.02 (0.13)	0.02 (0.14)	0.01 (0.11)	<0.001
Medical_Keyword_48 (mean (sd))	0.11 (0.32)	0.12 (0.33)	0.05 (0.22)	0.02 (0.12)	0.08 (0.27)	0.06 (0.24)	0.04 (0.20)	0.01 (0.12)	<0.001