Insurance Risk Evaluation through Data Mining

Yousaf Khan
Department of Computer Science, DePauw University
Greencastle, IN 46135, U.S.A.
Yousafkhan_2022@depauw.edu

Abstract

Today, most applicants apply for insurance policies online. Consequently, data science plays an important role in the insurance world. An insurance company's success is greatly impacted by its ability to accurately and reliably assess risk associated with a particular policy, based on the specific applicant's information and data.

This project uses an ensemble and stacking of machine learning algorithms to solve a complex multi-classification problem on a life insurance data set. Knowledge acquired in the Data Mining course at DePauw is used as a foundation to navigate and tackle a real-life normalized data set.

Keywords: Multi-Class Classification, Stacking, Extreme Gradient Boosting, Logistic Regression, Grid Optimization, Hyperopt.

1 Running the Product

This project can be run in the same way we were taught to run our code in the Data Mining course - relevant methods are executed through the 'main' function. The methods of 'pd.read_csv' to read the data and begin processing will always want to be run (I believe), and are included in the main function of the project submission. Other important methods are 'stackertest1' method, that runs the best performing model. Additionally, the 'doKaggleTest' method is an implementation of the same method for Kaggle result.

2 How to Use the Product

Using the product is fairly simple. The main submission file included contains a plethora of methods that test alternative strategies. I made sure to remove redundant 'trial' code, and the code included is primarily examples of different kinds of techniques that I implemented, which can be tested out by including the relative method in the main function.

Some of the code in the project submission relies on unconventional libraries that might need to be installed on your computer. However, in most cases, these special libraries only pertain to very specific methods that might not necessarily need to be run and thus these libraries could be commented out for simplicity - though this will probably show warnings in your IDE of choice as it would not recognize some methods in some particular functions.

3 Project Significance

While working as in the corporate treasury department at the headquarters of a large multi-national corporation, one of the projects pertained to foreign exchange trade data. I was given this data with the task of extracting meaningful insight and deciphering the causes of over exposure to currency volatility. As a junior in college, I did not have the necessary skills to meaningfully interpret the information. Additionally, my analysis was restricted to Microsoft Excel. As I worked on this project, I was concurrently taking Data Mining at DePauw which led me to wonder what further insights I could unlock if I used a powerful tool like Python.

Right after this internship, I spent the summer working on a government contract as an actuarial intern. Here again, I was working with healthcare data sets with little understanding of the data itself and thus my ability to do meaningful work on the set was greatly limited to what statistical methods I had to interpret the relationship between the predictors and variables. Additionally, I worked on the data with tools built in Microsoft Excel and Access - these tools are limited in their usefulness when working with data that is difficult to interpret.

Having to work with data with limited understanding of the significance and theory behind the variables has been a repeating theme in corporate environments lacking a specific data science focus. Through this project, I want to explore the possibility of extracting meaningful insight from data that is hard to interpret theoretically. Consequently, I picked a real world data set that is normalized for the purpose of confidentiality. This project seeks to explore how to work with such data given the aforementioned constraints.

Throughout my education at DePauw, I have worked with data in the Data Mining and Regression and Simulation courses. In both of these classes, we primarily worked with regression problems. I wanted to use this experience to work on a multi-class classification problem. The chosen data set, with 8 separate classifications for the response variable, is an incredibly difficult problem to navigate and most regression techniques simply fail to provide significant results from a machine learning perspective.

4 Data Description

The data set under consideration is a real life data set provided by a life insurance company. Thus, the data is particularly challenging to work on as it contains data that is normalized to maintain confidentiality. In fact, this is a prime reason why this data set was chosen for the purpose of this paper, as it proposes a problem that is difficult to decipher theoretically since the precise meaning behind the relative values assigned to variables is unknown and thus up for interpretation.

The data set is comprised of 128 variables that pertain to attributes of life insurance applicants, and only 4 of these variables have a clear interpretation: Age, BMI, Height, and Weight. Additionally, the data set includes a variety of different types of variables:

- Over 50 categorical variables pertaining to information like the medical history of the applicant.
- More than 10 continuous variables containing information such as height and weight of the applicant.
- Many discrete variables: over 40 of these discrete variables are dummy variables regarding the existence of a 'Medical_Keyword' in the applicant's application.

In addition to the high dimensionality of the data set, there are approximately 60,000 entries or observations which make the problem very computationally expensive. The task is to create a model that accurately predicts the 'Response' variable for each ID/individual in the data set. The target variable is an ordinal measure of 'risk' on a scale from 1-8, and thus presents a high dimensional multi-class classification problem.

5 Experiment

5.1 Computational Constraints

Working on a data set with so many attributes and entries was inevitably computationally expensive. Running models on the most powerful computers available to me would take over 12 hours sometimes to run algorithms. Consequently, I coped with this by running code remotely on DePauw servers through an SSH connection.

This entailed learning how to connect through SSH, transfer files through SCP, operate a command line, and understanding Linux to operate the machines. Additionally, this also meant that I would need a way to track progress on code that is running remotely. I achieved this using the 'logging' library to save output and warnings to a CSV file.

As computational times became unreasonably extensive by the end of the project, I investigated alternative techniques to help with this:

- Using google colab to allow code to utilize GPU for potentially faster computation.
- Using the Rapids Machine Learning library (includes cuML etc.) to speed up running time for the code.

5.2 Exploratory Analysis

The project begins with an exploration of the key characteristics of the data set. I used the helper functions to identify that the data set has non-numeric attributes as well as missing values. Further, I plotted the missing value percentages for the relevant predictors. This can be seen in figure 1.

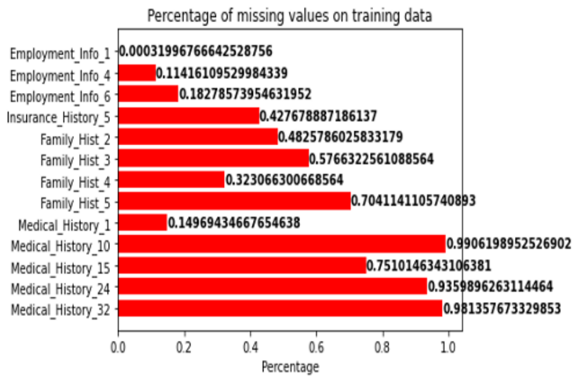


Figure 1: Percentage of missing values in attributes with missing values

Since this is a classification problem, it is important to see how the observations in the data set are classified. As is visible in figure 2, there is significant classification imbalance in the data set which can potentially be of concern.

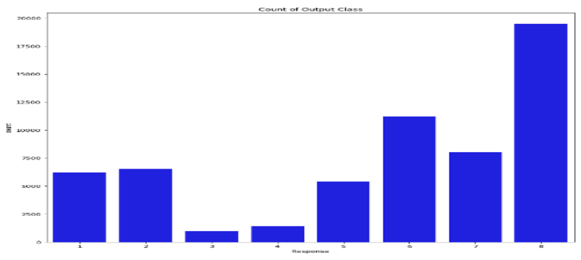
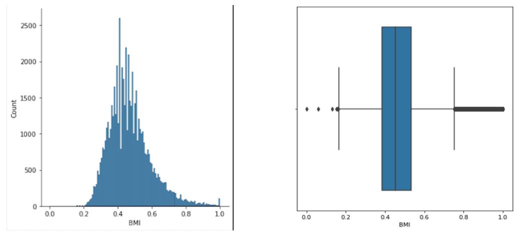


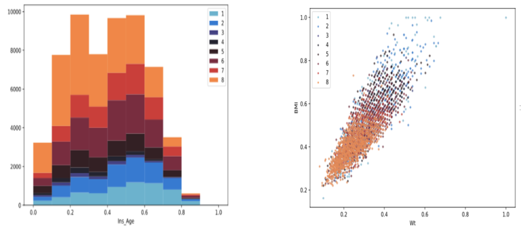
Figure 2: Classification imbalance in the data set

Due to the difficulties in interpretation of the predictor variables, I spent significant time visualizing key variables in the data set. I began by plotting simple histograms and box plots (figure 3). However, since this is a classification problem, simple graphics were not useful; more advanced visualizations that plot the classification alongside the variables were needed. In order to do this, I plotted histograms with classification hues and scatter plots (figure 4).



(a) Histogram (b) Boxplot

Figure 3: Examples of simple Visualizations



(a) Histogram (b) Scatter Plot

Figure 4: Examples of Advanced Visualizations

As I am working with over 100 variables, plotting each variable one after the other was highly time consuming. Therefore, I learned and implemented helped functions to allow for plotting of multiple variables concurrently, as well as saving the visualizations in a neat format in a PDF file.

Though visualizations for key variables like Weight and Age were helpful, it quickly became apparent that other attributes are very hard to decipher for trends in regards to predicting the output variable. Instead, it seems that all of the variables come together for the final prediction. This idea was further explored by calculating the correlation between the response variable and the predictors. The results of the correlation test can be seen in figure 5. The highest correlation to the target is held by the 'BMI' variable, and this is merely a 0.382 correlation. Most other predictors have much lower correlations, further indicating the importance of the high dimensions of the data set.

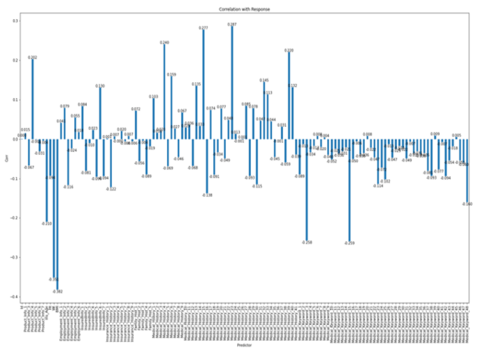


Figure 5: Correlation of attributes with the response variable

Another interesting area of the data set is the 36 'Medical_Keyword' dummy variables. These variables greatly increase the dimensions of the data set and thus are an area of interest in terms of pre-processing. I made a correlation heatmap to visualize relationships between the variables (figure 6). Here again, I was surprised by the difficulty of interpreting the predictors. The highest correlation is 0.25, and almost the entirety of other medical keyword variables have little to no correlation amongst them.

5.3 Pre-Processing

Firstly, we must clean the data set. Since the data is already normalized, standardization and such is not needed. Next, we clean the missing values. Since all of the missing variables are either continuous or discrete, we can easily replace them. First and foremost, I dropped three of the attributes with missing variables since they had greater than 90% missing values. In order to determine how to fill the rest of the missing values, I relied on correlation with the response variable; I calculated correlation before and after filling the missing values. My tests showed that predictors with greater than 20% missing values do well when the missing values are replaced with 0. For lower percentage

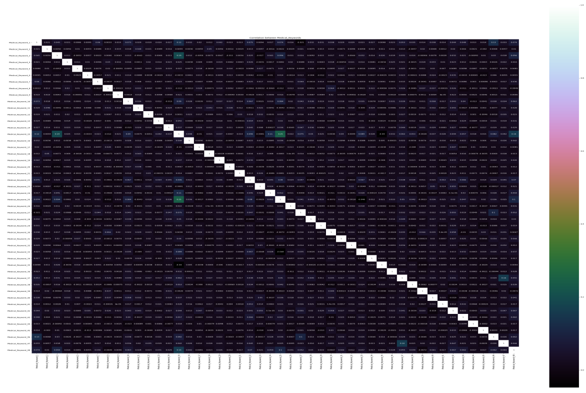


Figure 6: Correlation heat map between Medical Keyword dummy variables

of missing values, I used the mean of the existing values.

Through visualization and trial and error, I created 3 new variables using the BMI, Weight, Height and Age predictors in order to help track high risk cases. For instance, one of the new variables is a dummy variable 'Agehigh', which is calculated in the following manner:

$$Agehigh = 1, if' Ins_Age' > 0.7$$

The helper function identified that 'Product_Info_2' variable is a non-numeric attribute. In fact, it is comprised of a number and a character. I split the number from the character and created two new variables by applying label encoding to the two variables.

Since a major chunk of our data is categorical, it is important to encode it properly. Since the categorical data is numerical, a theoretical approach to encoding is nearly impossible. Therefore, I used an alternative approach: target mean encoding. Target encoding is a Bayesian encoding technique as it uses information from the target or dependent variable to encode the categorical data.

I created a helper function that calculates the mean of the target variable for each category and replaces the particular category with the mean value of the target. In the case of the categorical target variables, the posterior probability of the target replaces each category. A major pitfall of target encoding is its tendency to overfit the data. In order to avoid this, my encoding function uses 10 fold average cross-validation.

Working with the dummy variables proved especially challenging considering the correlation heat map. I engineered a number of features regarding the binary dummy variables:

- A new variable for the sum of the binary variable values for each entry in the data set.
- Created a helper function similar to the target mean encoding method, so that it calculates the mean of the target for each of the dummy variables.
- These target means are then used to introduced three new features that contain the average, maximum, and minimum of the respective encoding.

Since correlation itself was not a good indicator for feature selection, I used three separate feature selection methods in conjunction in an attempt to holistically drop variables:

- A filter feature selection method, Pearson's Chi Squared Statistical test, that test for independence between categorical variables.
- Another filter feature selection method, Mutual Information Statistical test, that measures the reduction in uncertainty for one variable given the value of another variable.
- A wrapper feature selection method, Recursive Feature Elimination, that utilizes an algorithm (RandomForestClassifier()).

In the end, I did not drop these variables as removing them did not contribute to improving my model's accuracy.

Lastly, 8 variables are added to attempt a 'binarization' of the data. Since there are eight classes, we make eight variables where each added variable contains the probability that the particular observation belongs to a specific class. In essence, this is done to condense the difficult multi classification task to a more simpler 'one vs rest' binary problem that is easier to predict. These eight attributes should help as inputs to a model. Again, overfitting is combated by using a stratified K-Fold with 5 splits.

Despite attempting many pre-processing techniques, my efforts proved very challenging as it did not improve the accuracy of the model. It seems that the data set relies more on robust models than intensive pre-processing.

5.4 Algorithm and Hyperparameterization

5.4.1 Algorithm

I tested a plethora of different machine learning and deep learning models in an attempt to discover what works best. I began my analysis by considering regression models to see if that would bring any benefit. This first test included OLS Linear Regression, Gradient Descent Boosting, Ridge Regression, Lasso Regression, and Bagging meta-estimator.

OLS Linear Regressions uses the ordinary least squares to find a hypothesis function that best predicts unseen values of the dependent variable: in this case, 'SalePrice'. The hypothesis function has an erorr surface, and we want to find the inputs that minimizes the error.

Gradient Descent Boosting is a type of machine learning boosting that essentially minimizes the prediction error by looking at the next best possible model. This is because combining the next best possible model with previous models decreases overall prediction error. To do so, it must re-set the target outcomes for successive models. In order to find the lowest point on the error surface, the target outcome for each case is calculated by measuring the change in overall prediction error by that case's prediction. This process is called gradient boosting because target outcomes are determined by the gradient of the error with regard to the prediction. Boosting works well with weak models.

Ridge regression is very similar to the previously discussed OLS Linear Regression. However, it improves on the former because ridge coefficients have a penalty on their size, as opposed to OLS coefficients.

Lasso Regression is a linear model that estimates sparse coefficients. It reduces the number of features that the solution is dependant on, and thus prefers solutions with non-zero coefficients.

Lastly, Bagging meta-estimator is an ensemble method that builds black-box estimators on randomized subsets of the total dataset. After making its predictions on the individuals, it combines them together to give a final prediction. Bagging is ideal for complex models since they reduce overfitting.

All of these models led to poor results, so I moved onto more considering a mix of classification and regression models. Some of the algorithms considered here include the following:

- K Nearest Neighbors
- Extra Trees Regressor
- Extra Trees Classifier
- Random Forest Regressor
- Random Forest Classifier
- XGB Regressor
- XGB Classifier
- SVC + SVM

Additionally, many stacking and ensembling configurations of these algorithms were considered which mostly yielded in improvement in model accuracy. These models included stacking, voting classifiers, ensembles etc.

A binary model implementation was also considered, where first classifiers were used to calculate probabilities of each class and then subsequent models worked on just these 8 categories to reduce the dimensions.

Lastly, I explored some Deep Learning techniques. This began with the sklearn implementation of the MLP-Classifier and Regressor. I furthered this effort by attempting customized multi-layer perceptron models in Tensor Flow.

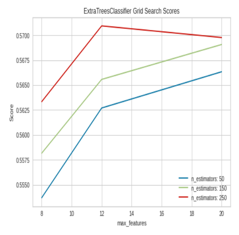
Here, I put significant effort to learning and implementing a convulation neural network as well. These models are designed to work on image input, so I used a library to convert my text tabular data to images - the theory for this was that it would allow for the position of the particular variable to be considered as well. I thought this would be useful to especially extract information from the multiple dummy varibles that my perviously results showed were not correlated to each other.

5.4.2 Hyperparameterization

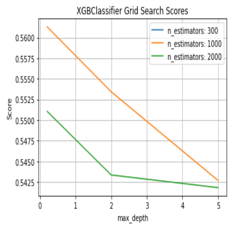
Hyperparameter optimization proved to be especially difficult, as the model run times were significant, and thus multiple attempt to evaluate different parameters were taking extremely long times.

As such, I inclined towards Grid Search CV to attempt hyperparameterization of the most import hyper parameters - often times, I had to interpolate from the results to make ’reasonable’ assumptions for the best parameters for the models.

After testing a range of values, I plotted the results of the grid optimizations. Some examples of the parametrization visualizations can be seen in figure 7.



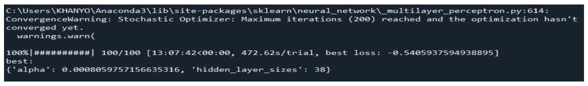
(a) Extra Trees Classifier



(b) XGB Classifier

Figure 7: Examples of GridSearch Visualizations

Another techniques considered here was made avalaible through the HyperOpt, which is a kind of Bayesian optimization for parameter tuning. The reason for using this was to allow for tuning within a defined range for the extremely large data set in an efficient manner. This was also particularly useful for the Neural learning and XGB algorithms, as my lower understanding of these models resulted in grid searches that did not yield meaningful results. An example of the results of HyperOpt optimization can be seen in figure 8.



```
C:\Users\KHANYO\Anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:614:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't
converged yet.
  warnings.warn(

100%|##########| 100/100 [13:07:42<00:00, 472.62s/trial, best loss: -0.5405937594938895]
best:
{'alpha': 0.0008059757156635316, 'hidden_layer_sizes': 38}
```

Figure 8: MLP HyperOpt hyperparametrization

5.5 Results

Some summary results of key models are included in table 1.

5.6 Analysis

I used a plethora of model evalutation tactics to determine which model works best. This included the following techniques:

- Cohen Kappa

Table 1: Before and After Analysis

Model	Accuracy
Gradient Boosting Regressor	0.3641
Ridge Regression	0.3980
Lasso Regression	0.3580
K Nearest Neighbors	0.3829
Random Forest Regressor	0.3929
Extra Trees Classifier	0.5214
MLP Classifier	0.5351
CNN Classifier	0.4591
XGB Regressor	0.4135
Varied Stacking Classifier	0.5568
Voting Ensemble Classifier	0.5727
Binary Model	0.5584
Optimized Stacking Classifier	0.5751
Kaggle (optimized Stacking Classifier)	0.57324

- Hamming Loss
- Confusion Matrix
- Prediction Error
- ROC_AUC Curve
- Pecision
- Recall
- F1 score

The most interesting observation was the fact that some models that had a higher accuracy score were actually not performing as well as models with lower accuracy, since sometimes models with lower accuracies can actually better classify lower occurring classifications in an unbalanced data set. I took this into consideration when deciding on the final model.

6 Future Improvements

One of the most difficult aspects of Data Mining projects is the sheer number of techniques one can try and evaluate on the data set - this was especially the case with this project as the predictors in this data set are essentially up for interpretation in regards to their relation with each other and the response variable. Therefore, as I complete this project there are many ideas I have regarding pre-processing techniques that might yield better results than the ones I have obtained:

- While conducting exploratory analysis on the data, I was unable to identify significant indications for dropping variables. Therefore, I chose to maintain the high dimensionality of the data set and worked to make robust models that can make sense of this abundance of predictors. It might be worth spending significant time to identify relationships between predictors and thus dropping predictors that negatively impact the accuracy of algorithms.
- While this project identified the classification imbalance in the data set, there was not much done regarding it. I chose to maintain the imbalance because it theoretically makes sense. A higher number of '8' classification in the response variable means that most individuals in the data set have lower risk, while much fewer individuals have a 'high risk' classification like '1'. In the real world, it is true that a smaller group of individuals, who can be considered outliers, have especially high risk associated with the policies; this is how insurance companies remain profitable. I chose to avoid balancing the classifications so that the model does not overfit the data. Instead, more work is put in to make a model that should be able to identify the few cases where risk is high. Though all of this makes sense in theory, it would be interesting to see the results of a pre-processing that seeks to fix the class imbalance - what technique works best for this would also be a fascinating question.
- This project attempted to deal with the multiple classes in the response variable by adding 8 predictors to the data set that capture probabilities. An alternative approach could entail creating an ensemble of 8 separate models, where each model focuses on finding the probability of the predictors resulting in each of the 8 different response variable classifications.
- An alternative implementation might be a base model that first determines the probability of the response variable to be greater or less than 4 (the mid level classification). Based on the determination of the first model, the particular observation can then be handled by one of two further models, where each model focuses on different subsets of response variables: 1-4 and 4-8.
- Running models on a graphics card.