CSE 575 Class Project Report

BNP Personal Insurance – Filing Claims

By

Akshay Iyangar, 1208604211, aiyangar@asu.edu

Hector Flores, 1000872235, hector.flores@asu.edu

Jayaprakash Jayakumar, 1209340128, jjayaku1@asu.edu

Sudharsh Subbaraman, 1207464111, ssubbar2@asu.edu

Sudarshan Venkat Ram, 1209590886, svram@asu.edu

Neha Prasad, 1208662985, nhprasad@asu.edu

Contents

[1 Introduction 1](#_Toc450050574)

[1.1 Project Background 1](#_Toc450050575)

[1.2 Problem Description 1](#_Toc450050576)

[1.2.1 Class Imbalance Problem 2](#_Toc450050577)

[1.2.2 Feature Selection for High Dimensional Data 2](#_Toc450050578)

[1.2.3 Categorical Variables for Classification 3](#_Toc450050579)

[1.2.4 Sampling methods to create simpler models 3](#_Toc450050580)

[1.3 Project Plan 3](#_Toc450050581)

[2 General Methodology 3](#_Toc450050582)

[2.1 Pre-Processing 4](#_Toc450050583)

[2.1.1 Categorical variables 4](#_Toc450050584)

[2.1.2 Sampling 5](#_Toc450050585)

[2.2 Learning Phase 5](#_Toc450050586)

[2.3 Performance and Cross Validation 5](#_Toc450050587)

[3 Feature Selection Process 6](#_Toc450050588)

[3.1 Stepwise/Adaptive Lasso Approach 6](#_Toc450050589)

[3.2 Information Gain Approach 7](#_Toc450050590)

[4 Modeling Analysis 9](#_Toc450050591)

[4.1 Logistic Regression 9](#_Toc450050592)

[4.1.1 Approach 10](#_Toc450050593)

[4.1.2 Results 10](#_Toc450050594)

[4.2 Training the Logistic Regression model using Gradient Methods 10](#_Toc450050595)

[4.2.1 Stochastic Gradient Methods 10](#_Toc450050596)

[4.3 Support Vector Machine(SVM) 11](#_Toc450050597)

[4.3.1 Approach 12](#_Toc450050598)

[4.3.2 Results 13](#_Toc450050599)

[4.4 Random Forests 14](#_Toc450050600)

[4.4.1 Approach 15](#_Toc450050601)

[4.4.2 Results 15](#_Toc450050602)

[4.5 Decision Trees with Boosting 16](#_Toc450050603)

[4.5.1 Approach 16](#_Toc450050604)

[4.5.2 Results 16](#_Toc450050605)

[4.6 Neural Network 17](#_Toc450050606)

[4.6.1 Approach 17](#_Toc450050607)

[4.6.2 Results 19](#_Toc450050608)

[4.7 Overall Performance 21](#_Toc450050609)

[5 Conclusions and Final Discussion 22](#_Toc450050610)

[6 Team Member Contribution 23](#_Toc450050611)

[Bibliography 25](#_Toc450050612)

# Introduction

## Project Background

Our project was derived from a Kaggle competition in which the task was to accelerate the speed at which a claim is processed in a personal insurance company BNP Paribas Cardiff. Currently, the management process requires the document to reach different levels before a claim could be approved and a payment made. The task that we are addressing is, given a database with anonymous fields, predict which of the two potential classes a given claim belongs to: (1) claims for which approval could be accelerated, and (2) claims for which additional information would be required before they can be processed. The problem’s data set, has a high degree of missing values, 114,000 instances (observations), and 131 variables (features). We have made an attempt to develop a model that correctly identifies those new claims that should be expedited (or not).

This problem is very interesting to us because when we look at the submissions made in Kaggle, we find that the variance of the performance is low; not more than 0.6 log loss, which intrigued us. Another important aspect that made us choose this problem was, that the dataset, readily available had many kinds of inconsistencies, and the sheer volume of data was one we had never previously handled. We wanted to try our hands on such an extensive data set. The process was further complicated by the fact that the dataset was anonymized.

## Problem Description

At first glimpse, our problem may seem rather simplistic given that it is very intuitive and easy to understand. However, as we proceeded through the data exploration, feature selection, and model development process, we found that in fact it had various types of complications that made it difficult to solve. In this case, we identify 4 key problems that we had to deal with that also are consistent with classification problems within other environments: Class Imbalance, Feature Selection, Handling of Categorical Variables, and Sampling Methods (Figure 1). As it will be discussed in detail within this report, class imbalance refers to the fact that there are significantly more data points from one class than the other. Another problem that we dealt with was feature selection, which is a very common theme in any modeling analysis. Thirdly, we had to determine the best way to handle a combination between categorical and continuous variables, especially when there are variables with factor levels larger than 1,000. Lastly, given that there were more than 114,000 rows of data, we also had to deal with memory constraints. Thus, we had to resort to sampling from our data set to determine a subset that was still representative of the original and also provide the necessary information to create good predictive models.

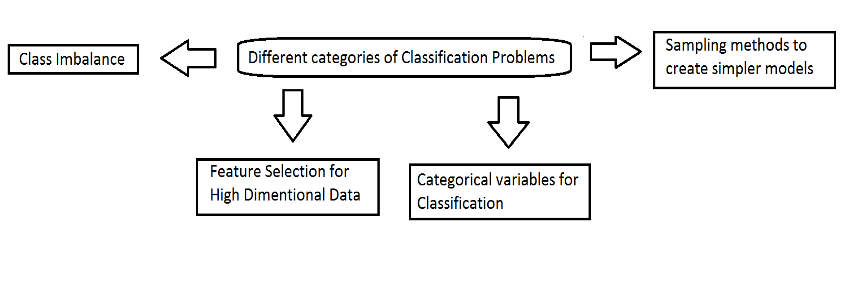
****

Figure : Classification problems

Next, we provide a quick overview of our research in identifying potential techniques that could be transferred to our own model development process. This includes a very brief summary of the domain of the problem and the models used.

### Class Imbalance Problem

After our initial investigation of the data, we recognized a problem of   
class imbalance in our data set. The class imbalance problem occurs when there are disproportionately many samples of one class as compared to the other. A common domain where this problem is also encountered is the area of fraud detection. It is much more likely for a fraud to not occur as compared to the likelihood that it will. Hence a sample dataset would usually have more occurrences of one class than the other. This often leads to a situation where the model becomes biased towards predicting the more frequently occurring class. This can lead to a misleadingly high accuracy score where the accuracy is merely representing the distribution of the underlying dataset. Some review of similar problems of class imbalance lead us to a few methods to get a truer picture of the robustness of any model's prediction.

Recognizing facial action units is another area which suffers from the problem of imbalanced data since facial expression data is highly skewed [1]. This skew may affect the metric which are used to evaluate performance of the model. So, along with accuracy, we need to measure scores such as F1-Score which will give us a more balanced view of the precision versus recall for a model. Random sampling to normalize skewed distributions can be used to reduce the bias but with some loss of information in the class being under-sampled.

### Feature Selection for High Dimensional Data

The high dimensionality of the dataset implied that we needed to select the features that were most relevant to our prediction objective and to eliminate the irrelevant features. We needed to optimally select the features which contributed the most to our objective query of interest with minimal loss of information. High dimensional data is a recurrent feature in domains such as bioinformatics and chemometrics [2]. Our review of the literature in these areas motivated us to do a deeper dive into Information Theory**.** We explored metrics such as Mallow’s, Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC) and the concept of information gain (IG).These approaches gave us a deeper understanding of the criteria for determining the relevant features in our data.

We looked into approaches to perform model comparison to detect outliers and feature selection. An approach of interest to us was one wherein random sampling was done of the dataset and models were generated based on those subsets and those models from the subsets were then analyzed to infer interesting properties of the dataset.

### Categorical Variables for Classification

A problem with our dataset is that it has a large number of categorical variables, which due to their non-numerical nature are harder to deal with. We looked at methods to models the categorical variables in a way that would be easier to merge into the modeling approaches we took. Categorical variables are often useful in research areas such as ecology and biometrics.

We looked into literature reviewing strategies for modeling a categorical variable with, say c, outcome categories and how they can be translated to a c-dimensional binary outcome response indicating if a category is chosen or not [3]. The strategies take into account logit models that consider the marginal distribution of each component as well as those containing randomly generated dependence between components. Marginal models were found to extract most information with least amount of assumption made.

### Sampling methods to create simpler models

The sheer volume of the training dataset added to the high dimension as mentioned before, made it near-impossible to train and test the entire data set due to the computational cost involved. We realized we needed to sample the data in order to be able to quickly train and test our models and derive meaningful results. To this effect we studied the various sampling techniques that could achieve this result while at the same time not causing too much information loss.

A similar nature of problem is encountered in the implementation of Monte Carlo methods, used for solving high-dimensional numerical problems. These calculations require high-dimensional probability distributions which are computationally costly. Hence they rely on sampling methods to achieve the desired computational efficiency [4]. Monte Carlo methods are used to generate samples from a target distribution and also for estimating the expectations of random variables. Generally, this is tackled by factorizing the distribution, if possible, into product of single dimensional conditional distributions. Uniform sampling is another simpler expectation maximization technique which can be used for data sampling. As mentioned before, other applicable sampling techniques are those which cater to the under-represented class.

## Project Plan

Over the next few sections, we discuss all of the activities performed as part of this project. First, in Section 2, we discuss the overall methodology plan for our project. In Section 3 we discuss the process of feature selection used to determine the best subset of variable to use. In Section 4, we detail the actual model development process, by first giving a very brief background into each model and then proceeding to explain our approach and results. Finally, in Section 5 we provide our final project conclusions and discussion on key takeaways.

# General Methodology

Our general methodology consisted of three components: data exploration/pre-processing, model learning, and performance. As it will be discussed in this report, in the first component, we first explored the characteristics of the data sets, including a pre-analysis of the kind of pre-processing needed to improve the chances of obtaining efficient algorithms with good predictive power. This included data imputation and sampling, feature reduction, and the ability to handle both categorical and continuous variables. The second component was the actual development of the models. In this case, we attempted 5 basic modeling techniques based on previous research that have good predictive performance under a classification setting: logistic regression, random forest, decision trees with boosting, support vector machines, and neural networks. As part of the model learning process, we gauged the structure of the model and identified “good” parameter structures that would enhance their predictive capability (e.g. neural network structure, kernel functions, etc.). Finally, the third component is in relation to the actual performance of the model. The goal of this phase was to check the overall predictive capability of each of the models. At the end, the main question to answer is whether the model was a good predictor. If in fact the model was a good predictor, then we used it to classify. If it was found that the model could not correctly classify the test set, then more information would be needed in either the data exploration/pre-processing or the actual model construction phase. Multiple iterations was attempted before finalizing each of the models.



Figure : General Methodlogy Pipeline

Next, we detail each of these steps in our methodology and provide examples of specific activities that were performed for each of these three component.

## Pre-Processing

### Categorical variables

We start with arguably the most important part of the project, getting the training data set ready. Starting off with the data exploration and pre-processing, we deal with imputing data, and categorical data. A categorical variable is a variable that can take only one of the limited values. [5]. We convert the categorical variables into a type of continuous by using the method called dummy coding. In this method, to represent a categorical variable with N levels, N-1 dummy variables are used. For instance, suppose a categorical variable has 3 levels A, B, C, two dummy variables X1, X2 are used to represent these variables.

Table : Dummy Coding

|  |  |  |
| --- | --- | --- |
|  | X1 | X2 |
| A | 1 | 0 |
| B | 0 | 1 |
| C | 0 | 0 |

We also imputed the missing values in the training data, using Multivariate Imputation by Chained Equations, available to us in the mice package in R. we use MICE as we assume that the missing data is missing at random [6]. Though this helps convert the categorical variables into the continuous type, it actually increases our problem as the number of features are already large. To reduce the time that will be spent on learning, we use random forests to identify the features that give maximum information gain. Only these features are selected to train the model for classification.

### Sampling

The training data set consists of 114,000 observations and 131 features, to speed up the process of training the model we sample the original dataset using random sampling to obtain a reduced training set, which we further split into training and testing data sets. We perform the splitting using *“sample.split”* function in R which belongs to the “*caTools”* package. At the end of all the above mentioned steps we arrive at the data set that is to be used to learn the model to classify. We use multiple data sets to train classifier and test performance of the different classifiers we have chosen to run on the data set.

Data preprocessing is followed by the learning the model to be used for classification. For the learning process we use different methods such as logistic regression, random forest, support vector machines, decision trees and neural networks. We train the model using the datasets we have created in the previous step, with slight modification depending on the model that is being learnt.

## Learning Phase

After the learning phase we do the classification based on the model learnt in the previous step. We performed k fold cross validation for each of these classifiers in the training dataset to choose the optimal set of parameters. To take an example, in the case of a soft margin support vector machines we chose good values of the cost parameter C and the gamma parameter for the radial kernel using k fold cross validation by splitting the training data set into k folds, using k-1 folds to train the classifier and the remaining one fold to test it. This procedure was repeated for all the k folds and the average accuracy for each set of these parameter values were determined. The best set of parameter values which gave maximum average accuracy over the k folds was then chosen.

## Performance and Cross Validation

After choosing the best models for each of these classifiers using k fold cross validation, test set performance are determined for these classifiers. We tested the model generated by these classifiers on the same test dataset in order to perform fair assessment of the performance. Performance metrics used were accuracy and F1 score. Accuracy is a measure of number of correct classifications to total number of samples, while F1 score is the harmonic mean of precision and recall.





For each of the models in this work, we provide a value for the F1 score and accuracy.

# Feature Selection Process

Feature selection is a common problem within any modeling framework. Now before describing our own methods, it is important to provide basic background into selection techniques considered, as well as the type of metrics used. In this case, we used three basic modeling metrics to decisions on variable selection, which includes Mallow’s , Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), information gain (IG) and entropy. The Mallow’s is a statistic that that adds a penalty to the training residual sums of squares in order to adjust for the fact that training error tends to under estimate the test error. Similarly, AIC is a statistic from a model’s goodness fit, while penalizing the number of variables in the model. BIC is a similar statistic but gives a higher penalty to additional model parameters. As it is used later, the most common error measurement associated with a decision tree framework is Information Gain (S,A) and Entropy(S), where A is the relative increase in “information” relative to the rest of the collection of variables in S. The definition of Entropy (S) is given by the following:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | where *c* is the number of classes and is the proportion of S belonging to |  |

Essentially, Entropy(S) is a measure of homogeneity of a set of examples and the goal of the algorithm is to split up the data set such that the entropy among the split data sets is high. The way this is accomplished is by looking for the split in any particular variable that gives the highest amount of separation among the data, or in other words, maximizes the information gain for the learning process of the algorithm. Therefore, the definition of Information Gain (S,A) measures the difference between the entropy in the original data set and the entropy after the set S is split using attribute *A*:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | where *Value(A)* is the set of all possible values for attribute *A* and is the subset of *S* for which feature *A* has value (i.e. |  |

The most basic techniques for feature selection (or variable selection) can be divided into three main methods: filter, wrapper, and embedded [7]; [8]; [9]). Next, we show the application of stepwise methods to our problem using , AIC, and BIC, as metrics. We also show the use of decision trees-based methods applied to the variable selection problem, in which information gain and entropy become the measures of variable importance.

## Stepwise/Adaptive Lasso Approach

A common approach to handle the variable selection problem is to select a subset of the total number of variables in the problem. The simplest method is to attempt all combination of variables and simply select the group that provides the “best” model based on certain modeling criterion. However, given the large number of variables in our problem, the computational complexity increases exponentially. Therefore, in this case, it is better to use a stepwise approach in determining the best subsets. However, another issue is being able to handle categorical variables, given that traditional regression models deal with continuous variables, and it becomes extremely hard to use stepwise regression techniques for the variable selection problem. In this case, one approach is to resort to more traditional statistical analysis tools using a generalized regression with binomial response variables using adaptive lasso techniques within the JMP statistical analysis software. Using hypothesis testing on the individual variable effects we were able to determine which variables were statistically significant.

|  |  |
| --- | --- |
|  |  |

Figure : BIC-based Selection of Features

Based on this analysis, we were able to significantly reduce the variable set and the number of factor levels analyzed. For example, in Figure 3 we show the resulting variable effect selection obtained from the logistic regression with adaptive lasso techniques. In this case, the variable subset that reduced BIC is given by only 9 of the original 131 variables from which a total of 102 factor levels can be removed. This resulted in a great simplification of the datasets that we constructed in our modeling development process.

## Information Gain Approach

The next set of approaches that we attempted are from decision tree approaches. In this case, the first thing attempted was to determine the information gain derived from the set of variables in our model. The information gain with regards to how well the features split the independent variable was measured using the entropy value. The information gain function from the R-package, FSelector, was used to determine the list of variables most important to the model. These values were estimated and ranked on the basis of importance and are shown Figure 4. Based on this graph, we can observe that the information gain in the variable set is only contained within roughly the first 20 variables. Furthermore, we can observe that the cumulative information gain (blue line) as one traverses the graph from the most important variables in the left to the right has a distinct elbow after which the marginal gains in information level off.



Figure : Variable Information Gain (Cumulative in Blue)

One of the main problems with our data set is that variable names are hidden and therefore there is no way of knowing variable relationships, which makes the problem that much harder to solve. Thus, we searched for methods that not only could potentially reduce the variable set but could also tell us something about variable relationship importance. Furthermore, one of the things noted from our data set was the use of random samples to create the models. Therefore, if only makes sense to use a randomized approach to the variable selection problem in order to reduce potential bias. In this case, we implemented an all-relevant feature method using an R-package developed in Kursa & Rudnicki [10] called Boruta algorithm. In this case, the importance value relates to a variation of the traditional Z Score distributed . Finally, we obtain a plot of the sampled importance ranges for each of the variables in our full variable set (Figure 5).

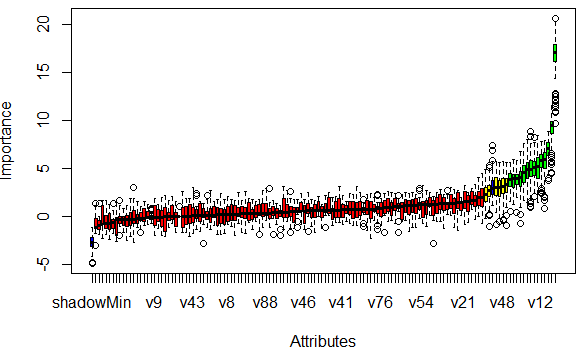


Figure : All-Relevant Feature Approach based on Boruta Algorithm

As we can observe from this figure, again, only the first few variables are deemed relevant (marked in green box plots) with one being the overall highest. The red box plots are those that have been rejected as being part of the model. Also, one of the things that we can potentially infer from this analysis is that the variables highlighted in green can be considered as an important classification group. Next, we finalize this section by comparing the feature selection techniques with the top 15 variables for each approach shown in Table 2. As one can observe from this table, most variables from the information gain and Boruta-based algorithm matched with the exception of v113. Interestingly, using adaptive lasso techniques under a logistic regression framework, only 4 out of the 9 variables match those selected within the other techniques. Moreover, only a few factors of these categorical variables (marked with an asterisk \*) were selected. Finally, from this process, we can conclude that most likely variables 10, 47, 50, and 56 have the most important predictive power. Variable 113 is the one anomaly, which needed further investigation.

Table : Selected Variables by Feature Selection Technique

|  |  |  |
| --- | --- | --- |
| Adaptive Lasso  (p-value) | Information Gain | Boruta (Green) |
| v10 (0.047) | v10 | v10 |
| v24 (0.002 | v12 | v12 |
| v30 (<0.001) | v14 | v14 |
| v47 (0.001)\* | v31 | v31 |
| v50 (<0.001) | v47 | v47 |
| v56 (<0.001)\* | v50 | v50 |
| v66 (<0.001) | v56 | v56 |
| v74 (0.008)\* | v58 | v58 |
| v113 (0.09) | v59 | v59 |
|  | v79 | v79 |
|  | v101 | v101 |
|  | v110 | v110 |
|  | **v113** | **v111** |
|  | v129 | v129 |

Throughout the rest of the modeling development process, this set of variables were given more importance. This allowed us to work with a smaller set of variables and greatly increased the speed of our model generation and prediction. We also applied factor analysis to determine additional variable groupings that could possibly help us further, using JMP statistical analysis software.

# Modeling Analysis

## Logistic Regression

Logistic regression is an efficient and powerful technique that works well in analyzing the effect of the independent variables on a binary outcome. Logistic regression is a discriminative classification model which takes multiple random variables as input and predicts the probability of the class [11]. This method uses a sigmoid function as objective function to learn the parameters, that will be used to estimate the probability.

This model also uses the sigmoid function to determine the conditional likelihood of the outcome. It uses the training data to learn the parameter (weights) vector that maximizes the conditional likelihood.

In logistic regression, the magnitude of the weight vector can potentially reach infinity, for example if the classes are linearly separable. To avoid this, we add a regularization parameter λ to the objective function, which modifies the weight update rule which helps prevent the magnitude of the weight vector from reaching infinity while improving the convergence rate of the algorithm. It is important to choose the λ value properly to achieve better predication accuracy [12].

Our problem is a binary classification problem in which we need to predict whether a particular insurance claim requires additional information or can be accelerated for approval leading to faster payment. Since it is a binary classification problem with random variables as input, Logistic regression model might perform well.

### Approach

The continuous variables are used as is and the categorical variables are recoded as mentioned and then used. We used regularized logistic regression method to train the classifier. For estimating the optimal value for the regularization parameter i.e. λ, we are using K fold cross validation technique. It is done by splitting the training set into different subsets and it is cross validated with one set while we train the classifier with other k-1 subsets. We implemented this model in R. We used the *“glmnet”* package for training the logistic regression model. This package trains the logistic regression model using the coordinate descent technique, which provides faster convergence rate. This trains the data with different lambda values and fits the data. It also provides the lambda value that provides the best fit for this data.

### Results

The accuracy of the model on the test data is 76.11%. The best λ value that we obtained for the training data is λ = 0.011. The F1 score that we achieved for the model was 86.3. As we expected the regularized logistic regression model performed well in our dataset and the convergence was fast.

Table : Logistic Regression sample result

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 33 | 2151 |
| 1 | 33 | 6929 |

We believe we should be able to achieve more accuracy by using a powerful machine and by including these categorical variables in the training set.

## Training the Logistic Regression model using Gradient Methods

### Stochastic Gradient Methods

As mentioned earlier, one of the problems with our dataset is its actual size. Thus, we searched for a stochastic-based approach to improve the convergence time of our model through the range of available R-packages but were unable to find a satisfactory algorithm that could be applied. Therefore, we searched for an existing gradient descent method that could be adapted to our dataset and could be modified to consider a stochastic component. In this case, we were able to find an existing GitHub code that used a basic gradient descent method [13], which we then modified to work with our own data set. These modifications are clearly highlighted in our coding set**Error! Reference source not found.**. To this framework, we added the stochastic component and used the final outputted s to estimate its testing error and overall performance.

Importantly, one of the advantages with working with the base components of the gradient descent methods was that we were able to observe firsthand the inner workings of this technique. For example, in order to incorporate the stochastic component to the model, it was necessary to understand all parts of the working algorithm. After converting our categorical data set into a dummy variable representation, a stochastic version of the gradient decent method was applied. An important aspect of this phase was that we were able to observe the impact that the different parameters have on the convergence patterns of the s. For example, using a sample size of 500 per iteration (as part of the stochastic version), with , the theta values failed to converge and started to oscillate back and forth. However, when we decrease the value of , the convergence pattern looked totally different and converged relatively quickly to its final value (Figure 6).

|  |  |
| --- | --- |
|  |  |

Figure : Convergence of Thetas per Iteration (Stochastic Gradient Method)

Finally, based on the matrix multiplication between the outputted theta values and the values in our test set, we obtain the predicted set and were able to obtain a relatively satisfactory performance (Table 4).

Table : Predictive Performance from Stochastic Gradient Methods

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 57 | 4560 |
| 1 | 105 | 14104 |

As one can observe, the accuracy measure of 75.4%, and an F1 score of 85.8 which is comparable against the results obtained using regularized logistic regression outputted from the *“glm”* function in R.

## Support Vector Machine(SVM)

Support Vector Machine, based on the given learning sample, constructs a model, which is something like a hyperplane or even a set of hyperplanes in a higher dimensional space. These models; hyperplanes are used for classification, regression and related tasks. Our goal is to find a good hyperplane which has the largest functional margin, i.e. largest distance from the nearest training sample point. The support vector machine aims to maximize this margin, because intuitively the larger the margin the lower the error of classification.

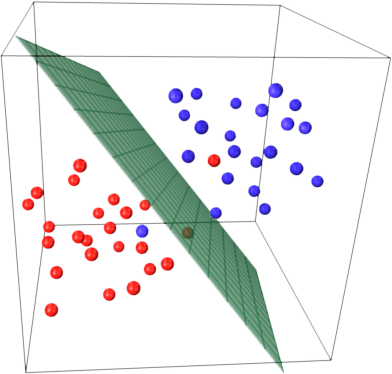


Figure : SVM hyperplane

As evident from the figure sometime it is not possible to completely separate two samples using a linear model, in these cases we use a non-linear kernels introduced by Boser [14][15], which maps the original problem defined in finite dimensional space into higher dimensions, making the separation easier to establish. The extended hyperplane is best defined as a set of points whose dot product with a vector (that defines the hyperplane) is constant. The vector defining the hyperplane is chosen based on *αi* which is a representation of *xi*, the feature vector. This mapping is based on the relation , where is the kernel function, which is larger the closer *x* is to *y*. Support Vector Machine also gives another important flexibility component, where the margin of classification can be either hard, where there is no room for any error, or soft [16], where there is some room for a sample portion of test samples to be wrongly classified. Figure 8 illustrates the difference between a soft margin and hard margin Support Vector Networks which is a version of SVM.

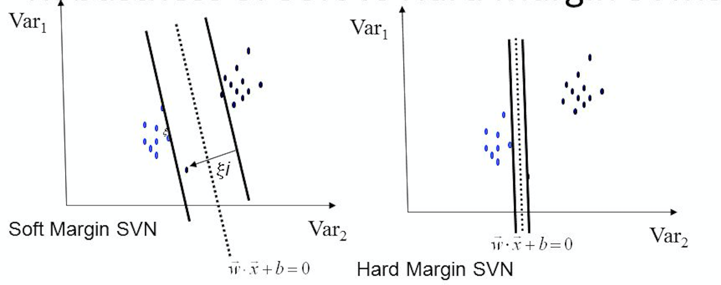


Figure : Soft and Hard Margin SVN

Our particular problem, we observed, is tailor made for SVM. Binary classification, with a large number of sparse observations is a condition the SVM thrives in. We believed that if we get a good support vector space, we can potentially speed up our classification process, especially with the anonymous dataset that we had obtained.

### Approach

For our particular problem we chose to go with two variants of support vectors, one was with linear kernel and the other one with radial kernel, with both of them being a soft margin classifier. The linear kernel was run to analyze the performance of the given data set for a linear classifier, while the radial provided a base to analyze the performance on a non-linear classification model. We choose the soft margin type as there are high chances that in a data set this big there may be many instance of misclassification in the training sample.

We use the implementation of SVM provided in the e1071 package of R, after doing many comparisons between the various available implementation and ease of use, thanks to Hornik [17]. This provides us with a flexible yet stable implementation.

*svm(target~., Train, kernel="radial", type="C-classification", cost=C, gamma=γ)*

This is the basic call to the svm function provided in the e1071 package to train the model on the given training set. We change the type between *“linear”* and *“radial”* as required. The cost and the gamma value determine the decision boundary of the classifier. The SVM implementation in this particular package uses the *coordinate descent* [18]approach to get a near optimal coefficient vector.

### Results

We ran a N-fold cross validation on the training set to determine the best possible soft margin parameters. We cross validated against 36 possible combinations of cost and gamma parameters to obtain the best result. This was obtained for the particular value of   
*C = 100, γ = 0.01* with a Radial kernel. When we ran this configuration for a particular data set we got a very impressive result as depicted in Table 5. The rows represent the actual class while the columns represent the predicted class.

Table : SVM sample result

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 617 | 1567 |
| 1 | 1290 | 5672 |

Based on the result obtained we get an accuracy of 68.7% and Precision of 78.4, with recall of 81.47 and F1 Score of 79.87.

We made some interesting observations while we performed the classification using SVM. Even with high cost value (greater than 10) and low gamma values (less than 0.01), we get a large number of support vectors, almost 94% of the observations were touted as support vectors. We performed closer inspection and proposed that the the reason for this phenomenon may be the fact that the observations are very closely spaced. To test our hypothesis, we decided to test the performance on a linear kernel with a lower cost and gamma value, what was surprising was that the number of support vectors did not decrease by much, it dropped to 89%, but we took a huge hit on the overall accuracy and precision, going down from 65.43 to 61.24 and from 75.62 to 71.72 respectively. After this round of results, we came to the conclusion that the classes were randomly spaced, for a linear kernel to correctly classify with high accuracy.

Figure : Accuracy vs Cost Value for different Gamma Values

Next step in our process was to move to the radial kernel, which in theory should give us a better result. Our problem was compounded by the fact that the data set is anonymous. This allowed us no room to maneuver. We selected only those features that gave the maximum information gain, and even then faced the problem of categorical variables, with multitude of possible values. The radial kernel helped improve the result. We got a very good result as show above, with a high cost value and a low gamma value, but even then we had an issue with the number of support vectors, around 82%. Our final conclusion was that SVM is not a great method for this particular data set with its skewed distribution where one class made up 75% of the recorded observation, even methods like Over sampling or under sampling really don’t help as outlined by Japkowicz [19].

## Random Forests

Decision tree methods classify by constructing a tree, it sorts the data from root to the leaf nodes, thus providing a classification. Each node in the decision tree corresponds to an attribute of the dataset and it represents one of all possible values of that attribute. Decision tree creates a classification rule by forming the disjunctions of conjunctions of constraints on the attribute values.

Random forests is a technique similar to the decision tree, where we construct many classification trees. Each classification tree gives a classification result which is termed as votes for the classification. The algorithm chooses the classification which has the most number of votes. At each node some *m* random variables are selected out of the training data and the best split on these *m* is used to split the nodes [20]. The error rate of the random forest depends on the correlation between any two trees in the forest and the strength of the individual tree. The selection of parameter *m* (split candidates) influences the error rate of the decision trees hugely. A higher value for *m* increases the correlation between the trees and the strength of the individual tree. It is important to choose the optimal value of *m* to achieve the better performance.

Random forests algorithm performs well in case of binary classification problems where we need to classify the data into two discrete classes. It works well with the data that consists of discrete values that are represented by attribute value pairs. Since Random forests algorithm constructs many decision trees by splitting the nodes based on random variables, it usually performs well on large datasets. Also since it uses all the variables present in the dataset, it minimizes the loss of information and increases the classification performance [21]. The random forest constructed can be stored and applied on new data for prediction. In some of the problems, the dataset will have large number of missing values and it acts as a major obstacle for many classification algorithms. But Random forests generally perform well on the datasets that contain missing values.

In our problem the dataset we have is a large dataset with 114,000 observations with many missing values. Many of the variables in the dataset are discrete, categorical variables, also the data set has a large number of variables (131). Random forests algorithm would be a better choice to fit the data. The classification problem we have is also binary where we need to classify the data as yes or no (0 or 1). The dataset has huge class imbalance as well, where around 70% of the data is towards the class 1, which becomes a challenging task to train the data set without overfitting. All these reasons contributed to our choice of Random forest as a classification approach.

### Approach

We have used the *“randomForests”* package available in R to implement the Random forest algorithm. We have implemented the K fold cross validation to determine the optimal value for the number of split candidates (*mtry*). The training data is split into three folds and we trained with two folds & cross validated with the other fold in each iteration. Then the average accuracy is used to determine the performance for a particular value of mtry. We tried for five values of mtry which are 5,10,25,50,100. The number of trees we used for training is 500.

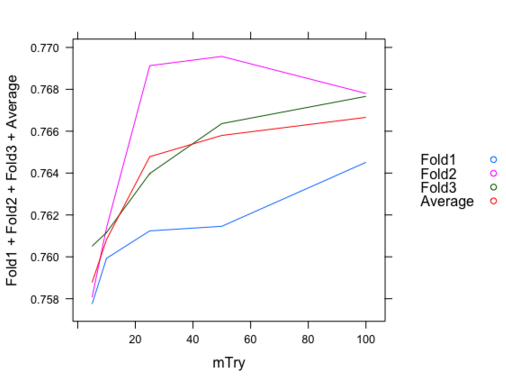
****

Figure : Random forest algorithm

### Results

When we tried for different values of *mtry* we identified that the accuracy was initially increasing as we increased the value of *mtry*. After a certain point the there was no major change in the performance. The higher accuracy was achieved for *mtry* value around 20. This result is shown in the graph in Figure 10. We selected the value of *mtry* as 10 as it gave better F1 score than other values.

Table : Random Forest sample results

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 78 | 2106 |
| 1 | 57 | 6905 |

This approach produced an accuracy of 76.4% and F1 score of 86.4 on the test dataset.

## Decision Trees with Boosting

The dataset in our problem has more missing values and huge class imbalance in it. Because of the class imbalance, the model was biased towards one class. Therefore, the use of boosting approach could potentially perform better. In this case, the general framework of the boosting depends on the iterative selection of wrongly classified data points. Similar to the random forest in its sampling methods for each of the decision tree constructed, boosting methods tends to add more weights to the training examples that were misclassified after a particular round. In other words, after a boosting round is complete, data points with larger weights have a higher probability of being selected for next boosting round. In this case, the final ensemble is obtained by aggregating the base classifiers obtained from each boosting round [22].

### Approach

We used the implementation of a generalized boosting algorithm proposed by Ridgeway [23] for training the data. The algorithm is based on the decision tree model. We used a package called *“gbm”* in R which is an implementation of Freund and Schapire's AdaBoost algorithm [24] and Friedman's gradient boosting machine. The package has implementations with the Adaboost’s exponential loss function [25] and the Friedman’s gradient descent algorithm [26]. With exponential loss, penalizing the misclassifications with large values would improve the performance.

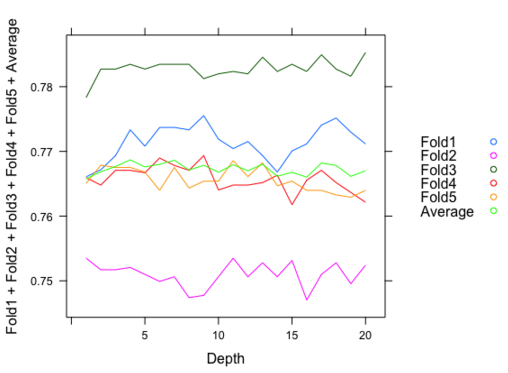


Figure : Decision tree with Boosting(GBM)

We used this *“gbm”* package and cross validated the training data to determine the optimal value for the depth parameter, which indicates the maximum depth of the variable interactions. We used 5 folds to train and cross validate the model. We trained with the number of trees as 1000. We have cross validated the model for 20 different depth values. The result we obtained in this method is shown in Figure 11.

### Results

The graph in the Figure 11 shows the performance of the gbm model on the different folds of training data for different values of interaction depth parameter. The accuracy was initially increasing as the depth value increases and then it decreased after a certain value. The accuracy was high with the depth value of 10.

Table : Decision Trees with Boosting sample results

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 251 | 1933 |
| 1 | 164 | 6798 |

We achieved an accuracy of 77.1% and F1 score of 86.5 on the test data. The boosting model gave the best performance when compared to the other algorithms as it was able to handle the class imbalance that was present in the data. The algorithm was also faster than other methods with a better performance overall.

## Neural Network

Neural networks are based generally on biological neuron systems. The simplest structure in the neural network system is the perceptron, which consists of two types of nodes: input nodes, which is used to represent the input attributes, and an output node, which is used to represent the model output [22]. The idea is that the perceptron computes its output value by performing a weighted sum on its inputs minus a bias factor. In our particular problem, each output node is assigned a function that ranges from 0 to 1 depending on the input value. In this case, the task is to determine how to assign weights to each individual node, such that at the end, the model is able to correctly classify a given data set. As we worked on the model development aspect of neural networks, we attempted to exploit the fact that our dataset consists of both categorical and continuous variables, the latter of which has multi-modal distribution. We also attempt to find an adequate neural network structure and activation functions for its nodes.

### Approach

The Architecture used for Neural Networks to train was a 3-layer neural network. The neural network consisted on 1 hidden layer 1 input layer and 1 output layer. The input to Neural Network consisted 118 units as there were 118 features, after we removed about 13 features, as they do not contribute towards the classification (Figure 12). This was then passed to the Hidden Layer of the input Neural Network. Each of the Neural Network activation unit was a logistic sigmoid function and we used 100 such activation units to train the neural network. The reason we used logistic sigmoid as the activation unit and not *tanh* a.k.a hyperbolic tangent activation was because we wanted the output to be ranged from [-1,1]. This would make it easier for the output not to have any bias parameter and simply use 0 as the threshold for classification. Also logistic sigmoid is just another way of representing the softmax function and we wanted a “soft thresholding” as compared to *tanh* function which is very abrupt. The lower values of *tanh* are linear and thus the output would be significantly constant even if there was a large difference in the input for negative values. Furthermore, 60 epochs were used, which represents the number of times our particular set of training vectors were used to update the weights of the neural network.

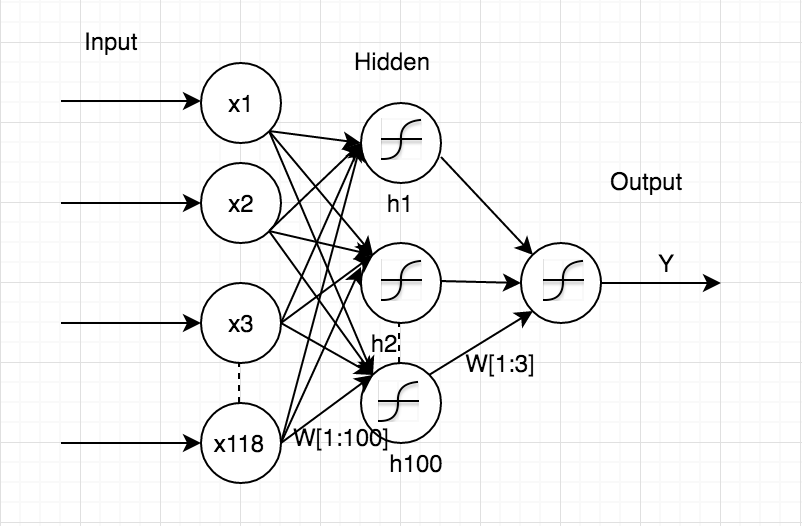


Figure : Model for Neural Networks

The Neural Net model with features where *t* ranges from 1:118, using a random variable which indicates whether the claim could or could not be accelerated. The Neural Network, with one layer of sigmoidal hidden units had 100 units. The hidden layer will have a set of weights mapping from its inputs to its hidden units. The output layer that predicts the output where the parameters of our model are Θ = {W [1], W [2]}. The hidden unit forms output by a logistic sigmoid over its weighted input.

This is the equation for every unit in the hidden layer.

The output of this equation is given to the output layer which decides the class based on the following equation

The Mean square error is calculated by where is the expected output of the data-sample and is the predicted output for the given data-sample.

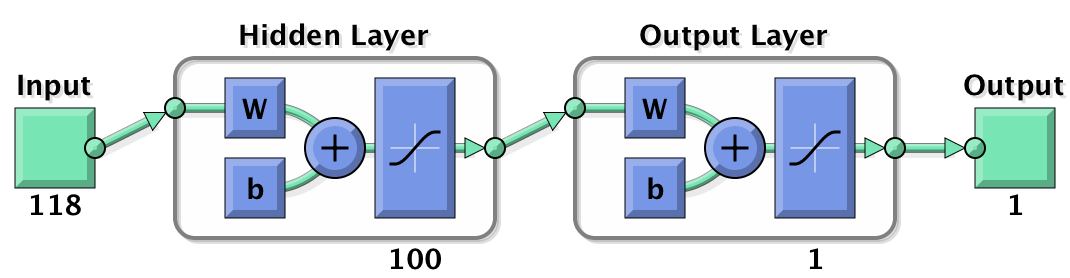


Figure : Neural Network Architecture for training

The Network used was the classic Feed-forward back propagation network. The algorithms used to train the neural network was Levenburg-Marquardt. This algorithm basically predicts the least curve fitting model by finding the MSE which is given by

### Results

The data was split into 60-40 where 60% of the data at random was used to train the Neural Network and 40% of the data was split to [20-20] for testing and cross-validating the dataset. The network did not have any bias term added to it and there was no drop out performed while training the model. Output was measured using mean square error, training of the model took 6 hour 30 minutes to complete.

The change in the gradient shows that the Neural indeed converged to the estimated minimum of and also that, there are not many oscillations while trying to converge. The validation check confirms that at every iteration the gradient was descending and trying to converge. Figure 14 explains the whole process of the Network for all the 60 epochs, at which the Neural Network converged.

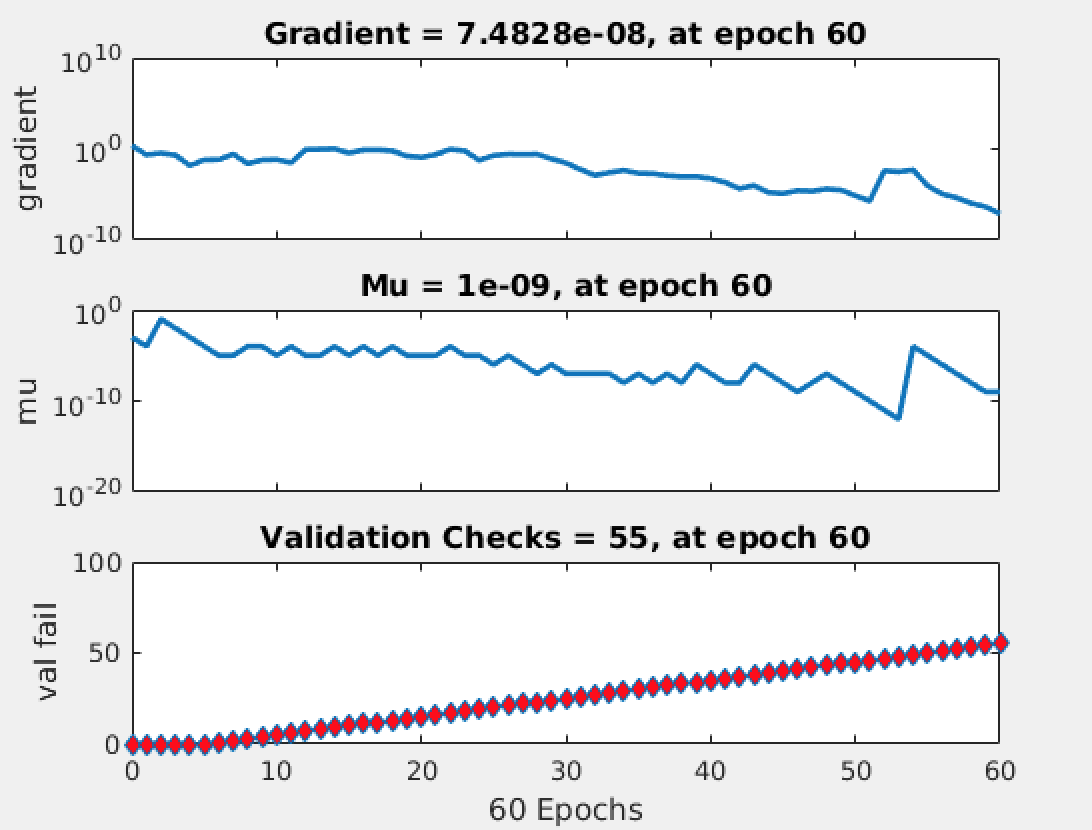


Figure : Gradient Parameters for the Neural Network

The Regression test shows that the global minimum of gradient descent did not fit the model well. However, the model fitted during the training phase behaved similarly to the one in test and validation, as shown below. The R value, which is a statistic measure of goodness of fit (mean squared error between observed value and fitted value), is similar in all four cases.

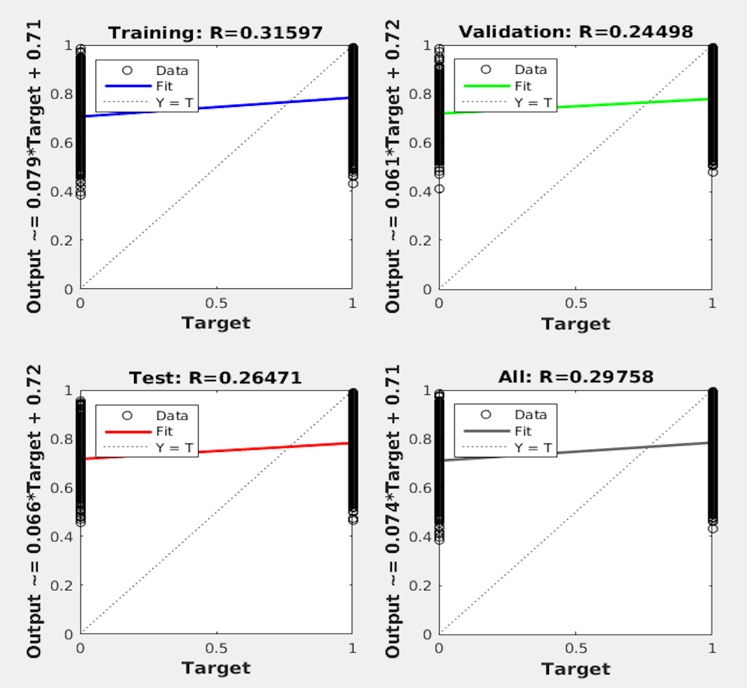


Figure : Regression Analysis of the Model

Performance testing here shows that the MSE was minimum for training and testing till the 5th epoch and deviated from there roughly by 0.23 percent, as shown in Figure 16.

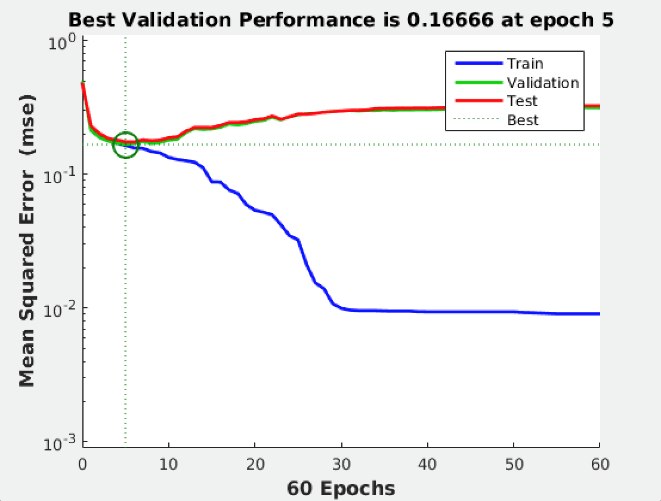


Figure : Performance of the model at each epoch

Finally, the classification results from our neural network were consistent with the results of the previously discussed. The results are shown in the table given below.

Table : Neural Networks sample results

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 523 | 1661 |
| 1 | 516 | 6446 |

In this case, the accuracy for the neural network algorithm was estimated to be 76.19%, while the F1 score was 85.4. Again, this is comparable to the results for the rest of the modeling approaches.

## Overall Performance

The performance of our dataset was deemed satisfactory. As mentioned in the introduction, the difference in log loss between the top Kaggle performer and the median was approximately 0.05. Thus, we knew that any marginal gain in performance for the accuracy measure over the testing set would be important. In this case, we observed that class imbalance is an important characteristic of our data set, since much fewer data points were selected for additional information (i.e. 0’s), which resulted in some difficulty being able to construct the models and having enough data points to correctly determine the parameters needed. Furthermore, given that in order to be able to run some of the data models (given CPU memory constraints), samples were taken from the original data set. In this case, we hypothesize that models that could potentially reduce the amount of bias would perform best. Therefore, initially we thought that ensemble-based methods would give the best results, such the combination Friedman Gradient Boosting/AdaBoost and Random Forest algorithms.

After model development and tuning, we found that our initial hypothesis was not far off. Based on the result summary presented in Table 9, we find that the best performer, in terms of accuracy over the testing set, was classification decision trees with boosting with an accuracy of 77.1%. This was almost one full percent over the second-best performer, Random Forest, with an accuracy of 76.4%. Interestingly, the worst performer over the testing set was support vector machines with an accuracy measure of 68.7%. Now, if we observe the F1 scores which also considers the recall measure, Random Forest was actually the best performer with a score of 86.5 and was followed closely by our boosting algorithm and logistic regression with scores of 86.4 and 86.3, respectively. Again, our support vector machine model did not produce good results with a score of 79.9. Finally, based on these results, our recommended model for this particular problem is classification decision trees with boosting given that it has both a high accuracy, as well as F1 score.

Table : Model Metrics Comparison

|  |  |  |
| --- | --- | --- |
|  | Accuracy | F1-Score |
| SVM (C=100, gamma=0.01) | 68.7% | 79.9 |
| Classification Decision Trees w/Boosting (depth=10) | 77.1% | 86.4 |
| Random Forest (mtry = 10, Trees = 500) | 76.4% | 86.5 |
| Logistic Regression | 76.1% | 86.3 |
| Logistic Regression (using Stochastic Gradient) | 75.4% | 85.8 |
| NN (Hidden Layer(Units) = 1(100), Activation function = logsigmoid, Epochs=60) | 76.2% | 85.4 |

Next, we finalize this report by discussing additional insights into our modeling framework and possible extensions that could have been taken to improve our predictive performance. We also discuss general concepts that our team learned through this process and what other teams could potentially learn from our experience.

# Conclusions and Final Discussion

As explained in our introduction, this project involved several different problems that we had to deal with that ranged from feature selection to the construction of adequate predictive models. In this case, we believe that our approach was appropriate with respect to the needs of the project. In the feature selection phase, we took two main approaches for selecting an optimal subset of variables that had the most predictive power. From the statistical perspective, we were able to reduce the number of features to roughly 9. Most importantly, using an adaptive lasso technique under a logistic regression framework available in JMP Statistical Software, we were able to identify the specific factor lcevels most important to the model, which in some cases improved our predictions. The second approach was more information theory related techniques which included information gain and the Boruta algorithm. Based on these methods, we were able to corroborate some of the findings from our first approach, while also gaining additional insights into other variables that may have predictive power. Although, we believe that our approach was appropriate, there are many other things that could have been applied to this phase of the problem, given that the field is so extensive. This includes the potential application of meta-heuristic technique (e.g. swarm and ant colony optimization) that could have given other type of results and could have reduced algorithm running times.

In the model development process, all team members gained a greater appreciation for creating models that are both predictive but are also efficient. Given that our data set was large and the number of variables expanded (when recoding from categories to indicators), we had to find ways to improve the convergence times of our models. We also wanted to compare the predictive power of different modeling techniques so that we had several options when selecting the final model to use. Again, although, we believe that our approach was adequate we could also have expanded our list of options. For example, we could have considered using a stochastic version of support vector machines, sequential minimal optimization, to be able to make the model converge using a full data set. However, given that the accuracy was so low for the full model approach, we did not pursue this path. Finally, there are several aspects of our modeling process that could be of use to people working on similar type projects. This includes our general methodology framework of data exploration, imputation, feature selection, and modeling framework analyses. Since we wanted to show that ensemble methods are actually the best choice for this kind of problem, interested readers could also look at our specific problem and could potentially decide to strictly expand on the use of ensemble methods to improve on the predictive performance of our models. Furthermore, we stress the importance for considering statistical analysis in the sampling of data points, feature selection, and model development.

# Team Member Contribution

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | ASU ID | Email | Specific Contribution | (%) |
| Hector Flores | 1000872235 | [hector.flores@asu.edu](mailto:hector.flores@asu.edu) | * Modified existing gradient descent method for our data set (expanded for stochastic components) for logistic regression * Feature Selection Process, including adaptive lasso techniques, information gain, and Boruta algorithms * Worked on SVM, Dtrees, and random forests * Report: Sections 3, 4.2.1, 4.7, and 5, report structuring, proofread | 16.67% |
| Jayaprakash Jayakumar | 1209340128 | [jjayaku1@asu.edu](mailto:jjayaku1@asu.edu) | * Worked on regularized logistic regression model * Worked on Random Forest models * Decision trees with boosting * Cross-Validation for SVM * Report: Sections 4.1, 4.4, 4.5, report structuring * Cross-Validation for SVM | 16.67% |
| Sudarshan Venkat Ram | 1209590886 | [svram@asu.edu](mailto:svram@asu.edu) | * Data imputation, Sampling * Worked on Random Forest models * Boosting models * SVM models | 16.67% |
| Sudharsh Subbaraman | 1207464111 | [ssubbar2@asu.edu](mailto:ssubbar2@asu.edu) | * SVM models * Cross-Validation for Boosting and SVM models * Logistic regression model * Report: Sections 4.3, contributions to section 1.1, 2, structuring and proofreading | 16.67% |
| Akshay Iyangar | 1208604211 | [aiyangar@asu.edu](mailto:aiyangar@asu.edu) | * Neural network models * Data imputation * SVM models | 16.67% |
| Neha Prasad | 1208662958 | [nhprasad@asu.edu](mailto:nhprasad@asu.edu) | * Neural network models * Handling of categorical variables * Data analysis | 16.67% |

# Bibliography

[1] L. A. Jeni, J. F. Cohn, and F. D. L. Torre, “Facing Imbalanced Data–Recommendations for the Use of Performance Metrics,” in *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction (ACII)*, 2013, pp. 245–251.

[2] M. L. Bermingham, R. Pong-Wong, A. Spiliopoulou, C. Hayward, I. Rudan, H. Campbell, A. F. Wright, J. F. Wilson, F. Agakov, P. Navarro, and C. S. Haley, “Application of high-dimensional feature selection: evaluation for genomic prediction in man,” *Sci. Rep.*, vol. 5, p. 10312, May 2015.

[3] A. Agresti and I. Liu, “Strategies for Modeling a Categorical Variable Allowing Multiple Category Choices,” *Sociol. Methods Res.*, vol. 29, no. 4, pp. 403–434, May 2001.

[4] W. K. Hastings, “Monte Carlo sampling methods using Markov chains and their applications,” *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[5] D. Yates, D. S. Moore, and D. S. Starnes, *The Practice of Statistics: TI-83/89 Graphing Calculator Enhanced*. Macmillan, 2002.

[6] S. van Buuren, “Multiple imputation of discrete and continuous data by fully conditional specification,” *Stat. Methods Med. Res.*, vol. 16, no. 3, pp. 219–242, Jun. 2007.

[7] R. Kohavi and G. H. John, “Wrappers for feature subset selection,” *Artif. Intell.*, vol. 97, no. 1–2, pp. 273–324, Dec. 1997.

[8] Y. Saeys, I. Inza, and P. Larrañaga, “A review of feature selection techniques in bioinformatics,” *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, Oct. 2007.

[9] T. Mehmood, K. H. Liland, L. Snipen, and S. Sæbø, “A review of variable selection methods in Partial Least Squares Regression,” *Chemom. Intell. Lab. Syst.*, vol. 118, pp. 62–69, Aug. 2012.

[10] M. B. Kursa and W. R. Rudnicki, “Feature Selection with the Boruta Package,” *J. Stat. Softw.*, vol. 36, no. 11, Sep. 2010.

[11] D. G. Kleinbaum and M. Klein, *Logistic Regression: A Self-Learning Text*. Springer Science & Business Media, 2010.

[12] S. Domínguez-Almendros, N. Benítez-Parejo, and A. R. Gonzalez-Ramirez, “Logistic regression models,” *Allergol. Immunopathol. (Madr.)*, vol. 39, no. 5, pp. 295–305, Sep. 2011.

[13] smc77, “Logistic Regression with Gradient Descent.” 2011.

[14] B. E. Boser, I. M. Guyon, and V. N. Vapnik, “A Training Algorithm for Optimal Margin Classifiers,” in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, New York, NY, USA, 1992, pp. 144–152.

[15] W. H. Press, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.

[16] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[17] A. Karatzoglou, D. Meyer, and K. Hornik, “Support Vector Machines in R,” 2005. [Online]. Available: http://epub.wu.ac.at/1500/. [Accessed: 02-May-2016].

[18] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan, “A Dual Coordinate Descent Method for Large-scale Linear SVM,” in *Proceedings of the 25th International Conference on Machine Learning*, New York, NY, USA, 2008, pp. 408–415.

[19] N. Japkowicz, “The class imbalance problem: Significance and strategies,” *Proc Int’l Conf Artif. Intell.*, 2000.

[20] A. Liaw and M. Wiener, “Classification and regression by randomForest,” *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[21] K. J. Archer and R. V. Kimes, “Empirical characterization of random forest variable importance measures,” *Comput. Stat. Data Anal.*, vol. 52, no. 4, pp. 2249–2260, Jan. 2008.

[22] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

[23] G. Ridgeway, “gbm: Generalized boosted regression models,” *R Package Version*, vol. 1, no. 3, 2006.

[24] Y. Freund, R. E. Schapire, and others, “Experiments with a new boosting algorithm,” *ICML*, vol. 96, pp. 148–156, 1996.

[25] J. Elith, J. R. Leathwick, and T. Hastie, “A working guide to boosted regression trees,” *J. Anim. Ecol.*, vol. 77, no. 4, pp. 802–813, Jul. 2008.

[26] J. H. Friedman, “Stochastic gradient boosting,” *Comput. Stat. Data Anal.*, vol. 38, no. 4, pp. 367–378, Feb. 2002.