ECON 412 Project2 (Final)

Alexander Ramos (ID:605657325)
Anshika Sharma (ID:305488635) Cristian Martinez (ID:205642760)

05/26/2021

Contents

I. Introduction

The purpose of this project is to analyze two separate datasets using a multitude of Machine Learning algorithms that are classified as classification and regularization algorithms. The dataset that was selected to be used for the classification models is an insurance premium dataset where we are given customer characteristics and are tasked to predict/classify how much their insurance premium expenses will be. For the dataset that will be used with the regularization machine learning algorithms, the group elected to use 2007 ACS data to predict per capita income for each county based on characteristics given. After fitting each model with the selected datasets, accuracy and cross validation results are reported for the models that used a classification framework, along with determining whether a linear or non-linear model is appropriate for our data and bias-variance tradeoff metrics are reported for the regularization models.

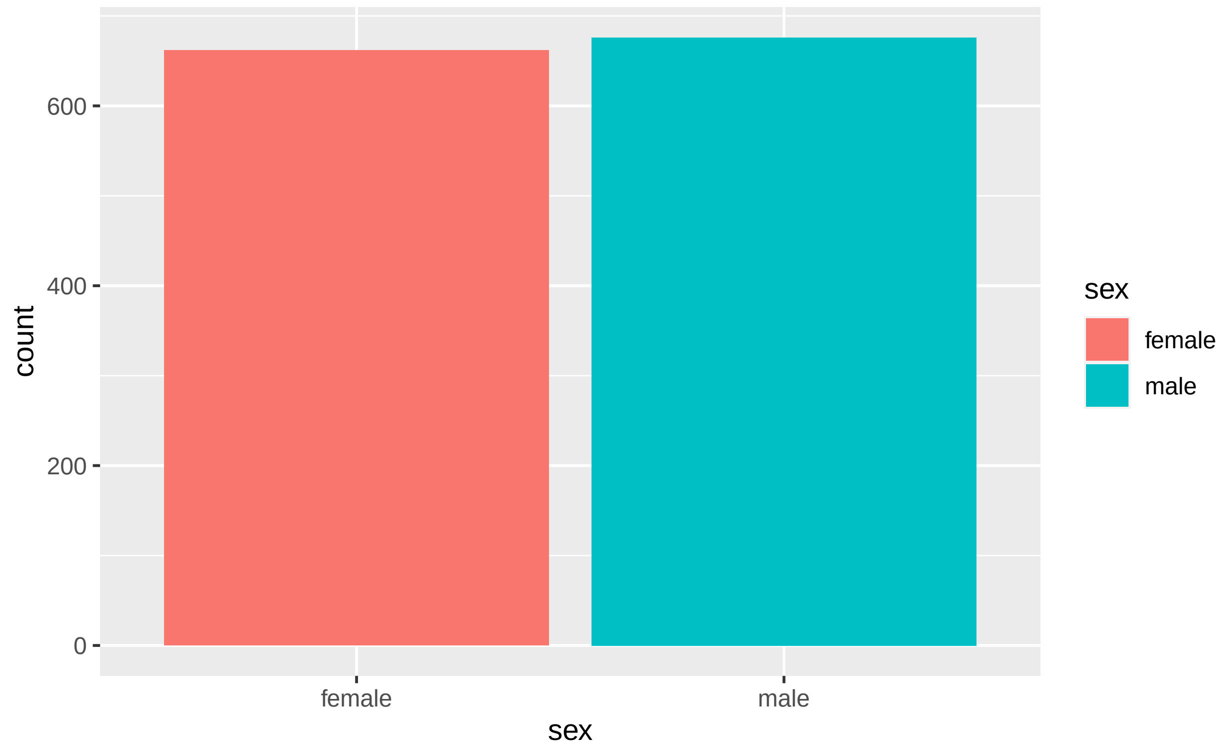
II. Classification (Part 1)

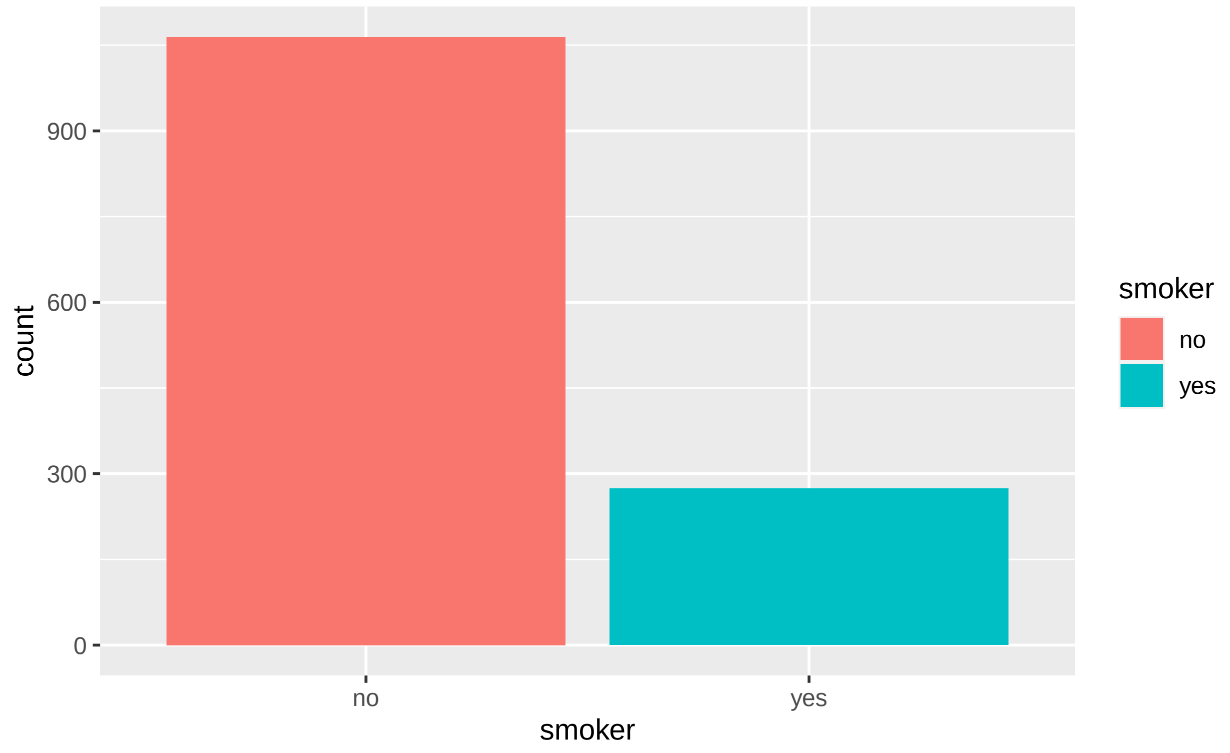
We opted to create artificial groupings for the insurance expenses and classify them at three levels: "Low", "Medium", and "High". We performed this modification on our dependent variable to create a 3 factor variable that will be used in our analysis so that we could be able to create predictions that would act as if were tasked to classify individuals into cost groups for insurance premium purposes. We also opted to segment the age variable into artificial groups because we felt that insurance firms in reality perform the same grouping of ages when determining insurance premiums given to customers.

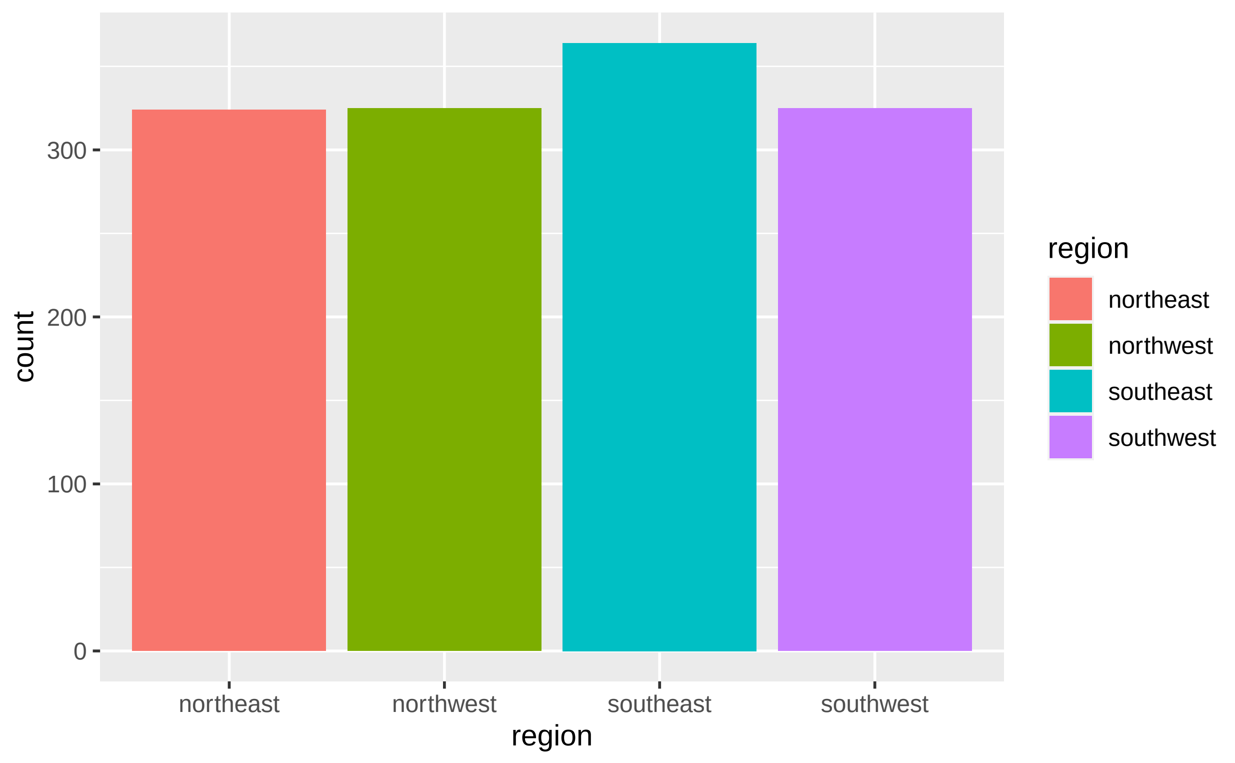
```
##            age          sex            bmi           children       smoker
##  18-28       :390   female:662   Min.   :16.00   Min.   :0.000   no :1064
##  29-39       :284   male  :676   1st Qu.:26.30   1st Qu.:0.000   yes: 274
##  40-50       :308                Median :30.40   Median :1.000
##  51-60       :265                Mean   :30.67   Mean   :1.095
##  60 and above: 91                3rd Qu.:34.70   3rd Qu.:2.000
##                                  Max.   :53.10   Max.   :5.000
##        region      expenses
##  northeast:324   high  : 133
##  northwest:325   low   :1065
##  southeast:364   medium: 140
##  southwest:325
##
##
```

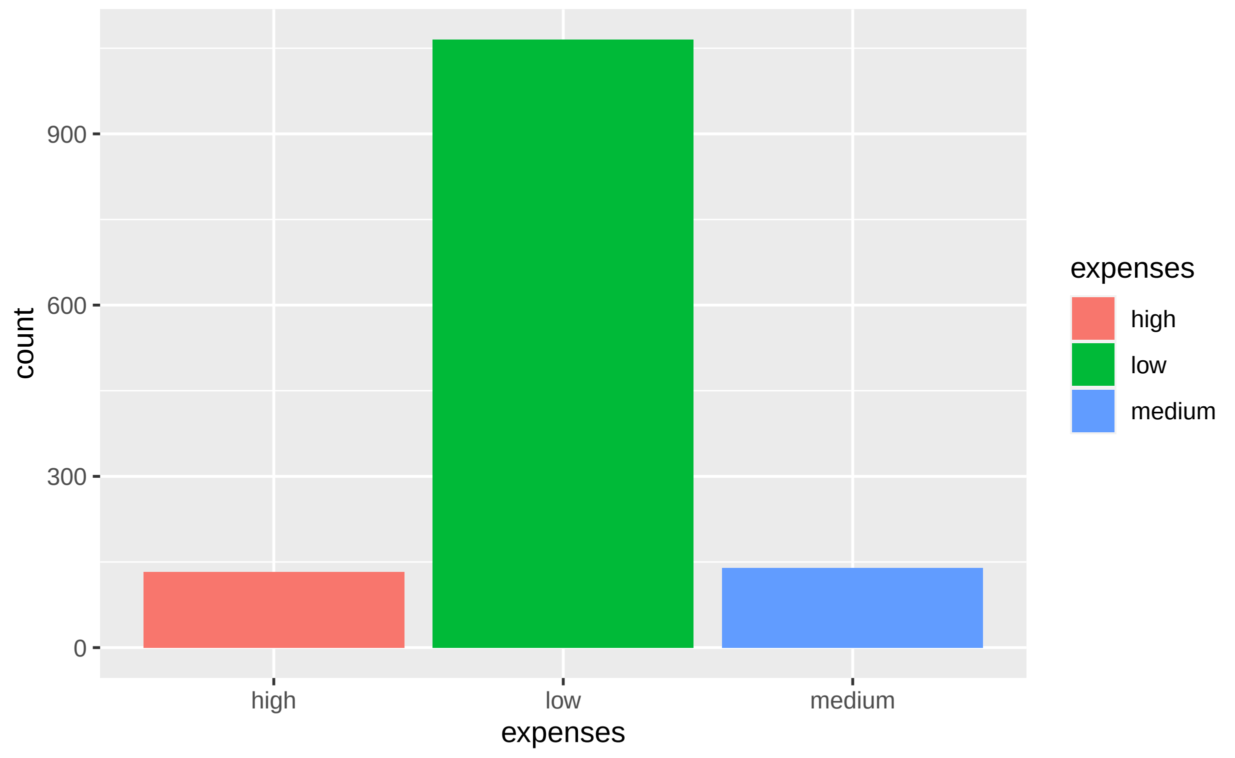
Now we will transition to viewing and analyzing the summary statistics for our dataset after we performed the artificial groupings above. As we can see, there are a total of 1,338 observations and for the most, there is a good distribution for observations in the geographical regions along with males and females. As for age, there is a good distribution for age groups 18-60, but the observation count decreases to 91 for individuals 60 and above, but it shouldn't pose any problems for our analysis. BMI reports a mean of 30.67 and 1st and 3rd quantile values of 26.30 and 34.70. As for number of children, the mean is 1.095 and the 1st and 3rd quantile values are 0 and 2 children. Our dependent variable, which again is insurance premium expenses, has a skewness towards low premium expenses of 1065 observations, but it shouldn't pose a problem for our analysis because our models should be able to compensate for the dispersion in observations.

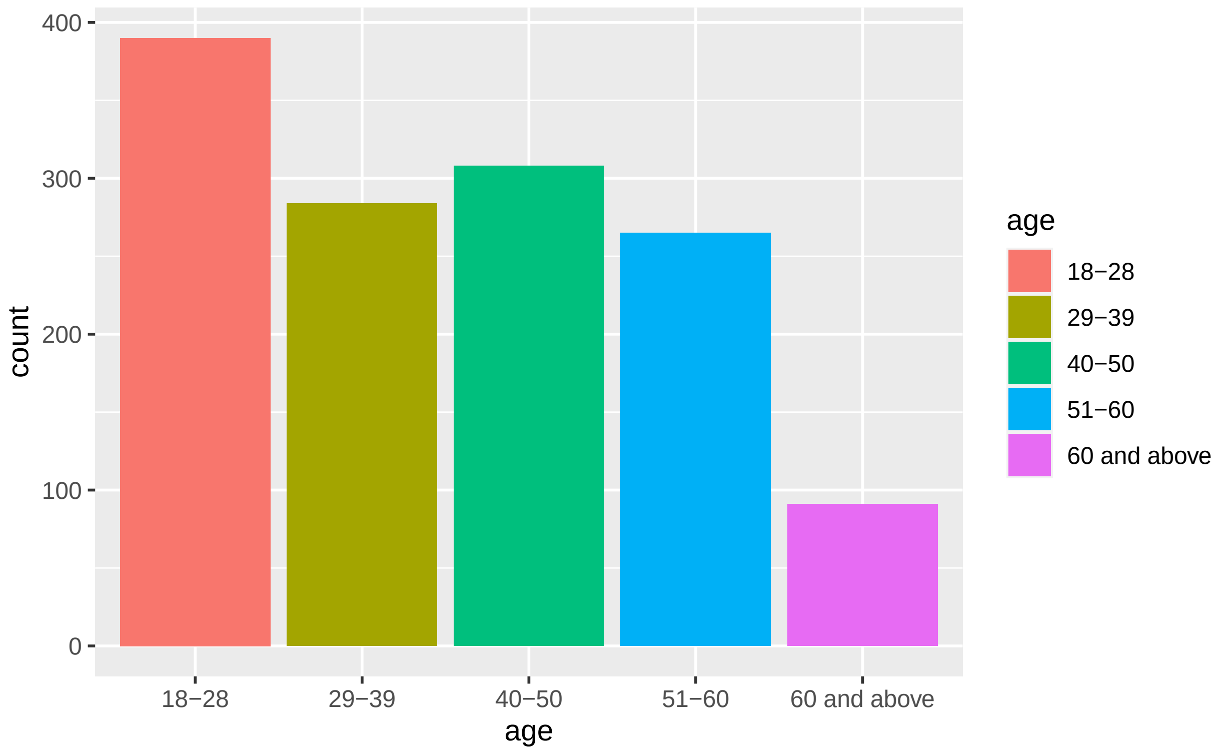
Exploratory data analysis






count
300
200
100
0
northeast
northwest
southeast
southwest
region
region
northeast
northwest
southeast
southwest


count
900
600
300
0
high
low
medium
expenses
expenses
high
low
medium



The plots above provide a visual representation of the data. As can be seen from the gender barplot, the number of males and females are almost the same which implies that the data does not suffer from gender bias. The second graph depicts that the frequency of non-smokers is higher than that of smokers by about 600 people.The third graph provides a visual representation of the regional distribution of people. The fourth barplot suggests that most of the people in our data set incur low insurance expenses, followed by medium and then high expenses. The last plot shows the distribution of people based on age. Most of the people lie in the age group of 18-28 years.

We begin by partitioning the data into a 60-40 framework where 60% of the data will be the training parition and the remaining 40% of the data will be used as the testing data.

Multinomial Logistic Regression

The Multinomial Logistic Regression is an extension of the classic Logistic Regression model, but is used when the dependent variable has more than two possible discrete outcomes. For the purpose of our analysis, our dependent variable has three possible outcomes. The assumption that is made for the Multinomial Logistic Regression model to be used is that the data that is used is case-specific, meaning each independent variable has a single value for each case. As for the choice for predictors to be used in the model, it is assumed that the dependent variable is independent, which in other words means that one predictor is not related to another. Thus, we felt our data contains characteristics that we feel are independent thus, Multinomial Logistic Regression can be used.

We will now fit a Multinomial Logistic model to our data.

```
## # weights:  39 (24 variable)
## initial  value 882.185668
## iter  10 value 350.386078
## iter  20 value 249.556317
## iter  30 value 225.518834
## iter  40 value 224.744336
## final  value 224.744328
## converged

## Call:
## multinom(formula = expenses ~ ., data = train_df)
##
## Coefficients:
##        (Intercept)  age29-39  age40-50  age51-60 age60 and above    sexmale
## low       29.00985 -1.317095 -2.762951 -3.455413       -3.198107 0.06166665
## medium    23.80808 -1.679482 -1.516806 -1.528949       -1.442325 0.22562458
##               bmi   children  smokeryes regionnorthwest regionsoutheast
## low    -0.5282925 -0.1352492 -12.381045       0.1363132       0.3351219
## medium -0.4895635 -0.1264704  -8.129545       0.3217277      -0.2803548
##        regionsouthwest
## low          0.3294505
## medium      -0.3319498
##
## Residual Deviance: 449.4887
## AIC: 497.4887

##         (Intercept)  age29-39   age40-50   age51-60 age60 and above  sexmale
## low    3.970256e+12 0.2679124 0.06310526 0.03157426      0.04083942 1.063608
## medium 2.186341e+10 0.1864706 0.21941163 0.21676331      0.23637751 1.253105
##              bmi  children    smokeryes regionnorthwest regionsoutheast
## low    0.5896108 0.8734982 4.197402e-06        1.146041       1.3981109
## medium 0.6128939 0.8812002 2.947022e-04        1.379509       0.7555156
##        regionsouthwest
## low          1.3902040
## medium       0.7175233
```

We take the exponential of the coefficients because the Multinomial Logistic Regression model reports the coefficients, but in log outcomes, thus, this mathematical operation provides coefficients that can be interpreted.

As you can see, the Multinomial Logistic Regression reports the coefficients for each predictor, but we understand that these coefficients won't matter if the model reports poor accuracies, thus, we will transition to observe how the model performs when we train it and measure its accuracy against the testing data.

```
##
## Multi_Logistic.pred_test high low medium
##                   high     51   0      5
##                   low       1 415     35
##                   medium    1  11     16

## [1] 90.09
```

After fitting a Multinomial Logistic model to our data, we wanted to observe the accuracy of the model, thus, using the established training and testing dataset we generated earlier, we used our trained model to predict the actual values of insurance cost premiums. We find that the model performs relatively well with a accuracy of 90.09%. However, we will transition to testing our Mulitnomial Logistic model using a 10-Fold Cross Validation approach because we are concerned about introducing bias due to the 60-40 data partition that was performed to observe how the model performed.

Below are the results:

```
## Penalized Multinomial Regression
##
## 1338 samples
##    6 predictor
##    3 classes: 'high', 'low', 'medium'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1205, 1205, 1204, 1205, 1204, 1204, ...
## Resampling results across tuning parameters:
##
##   decay  Accuracy   Kappa
##   0e+00  0.9073369  0.7078753
##   1e-04  0.9073369  0.7078753
##   1e-01  0.8961369  0.6791709
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was decay = 1e-04.
```

After performing the 10-Fold Cross Validation technique on the Multinomial Logistic Model, we find that the accuracy derived was 90.73%, which is a marginal improvement compared to 90.09% accruacy that was derived from our prior analysis, which seems to indicate that our model has a high accuracy rate.

We will now perform the same analysis by fitting our data to the Linear Discriminant Analysis (LDA) model.

Linear Discriminant Analysis (LDA)

Linear discriminant analysis (LDA) is a generalization of Fisher's linear discriminant to find a linear combination of features that characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification. In this case, LDA will be used for classification.

```
## Call:
## lda(expenses ~ ., data = train_df)
##
## Prior probabilities of groups:
##      high       low    medium
## 0.0996264 0.7957659 0.1046077
##
## Group means:
##         age29-39  age40-50  age51-60 age60 and above   sexmale      bmi
## high   0.2125000 0.2375000 0.1750000      0.15000000 0.6375000 34.74625
## low    0.2222222 0.2269171 0.1830986      0.06259781 0.4882629 30.39609
## medium 0.1071429 0.3095238 0.2380952      0.11904762 0.5238095 28.24286
##        children  smokeryes regionnorthwest regionsoutheast regionsouthwest
## high   1.287500 0.97500000       0.1500000       0.4000000       0.2625000
## low    1.115806 0.05946792       0.2644757       0.2535211       0.2550861
## medium 1.059524 0.65476190       0.3333333       0.2500000       0.1547619
##
## Coefficients of linear discriminants:
##                         LD1         LD2
## age29-39        -0.02859809 -0.33199666
## age40-50        -0.27199427  0.67731165
## age51-60        -0.54039424  0.61842519
## age60 and above -0.51362888  0.62331771
## sexmale         -0.13302204 -0.23153914
## bmi             -0.06075025 -0.14372382
## children        -0.02684285 -0.06922162
## smokeryes       -3.88747904  0.31253694
## regionnorthwest -0.02418811  0.29559816
## regionsoutheast  0.14808950 -0.20608578
## regionsouthwest  0.05901924 -0.62444412
##
## Proportion of trace:
##    LD1    LD2
## 0.9585 0.0415
```

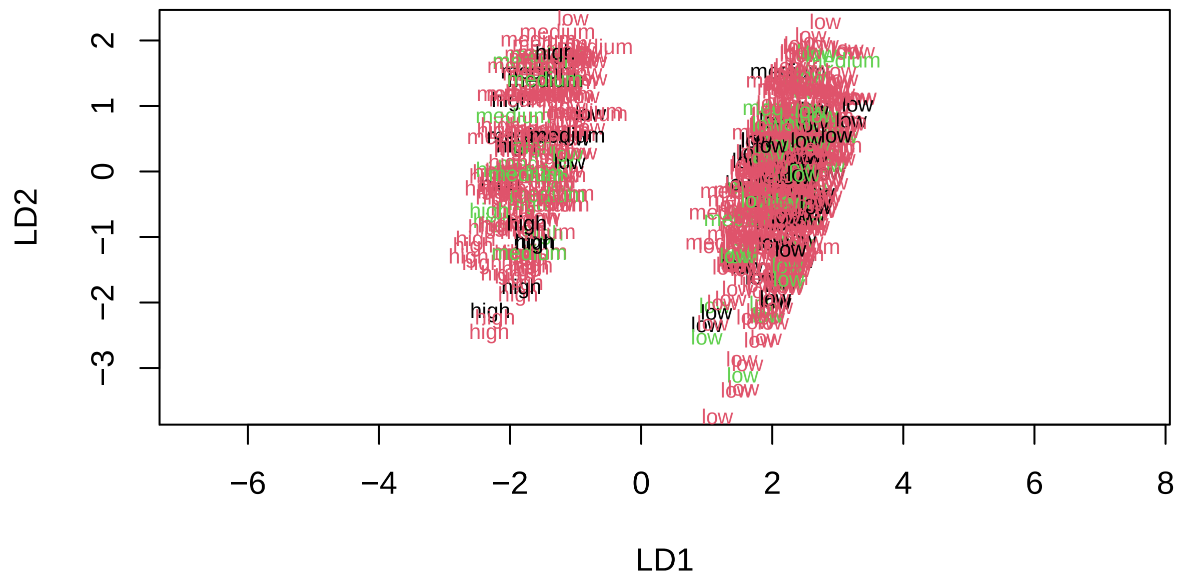
we first fit the LDA model on the training data set that was partitioned above. By looking at the prior probabilities, one can tell that maximum observations in our data set lie in the category of low expenses (79%), followed by medium and then high expenses. The coefficients of the linear discriminants define the weights that have been assigned to each of the predictor variables. The higher the absolute values of the weight, the better the predicting power. The proportion of the trace, LD1, is located at where the trace captures the most data variablity; then, trace LD2 is the perpendicular trace to LD1. As the above reslut indicates, our trace LD1 caputures roughly 96% of the data variability.

```
## [1] "class"     "posterior" "x"
```

```
##
## lda.class high low medium
##    high     52   9     15
##    low       1 402     29
##    medium    0  15     12
```

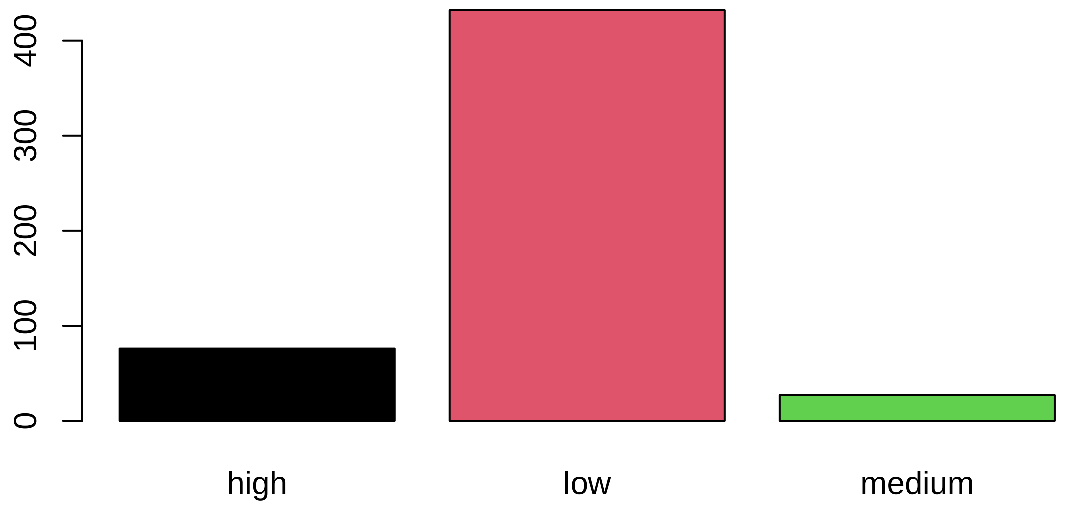
```
## [1] 0.871028
```

Next, we test the trained model's performance by running the LDA model on the testing data frame to predict the actual values of insurance cost premiums. The following confusion matrix is obtained. As can be seen, the model performs well with a high accuracy of 87.10%. We have to note that our data is highly imbalanced, and the LDA model is known to be extremely sensitive to the imbalance data. While our original data set comes with 8:1:1 data, low, medium, high respectively, we still improved our classification model with LDA by nearly 7% compared to a model just classifying every point into low. In addition to the special note of the imbalance data, the accuracy rate is marginally lower than the accuracy rate obtained in the Multinomial Logistic Regression Model.



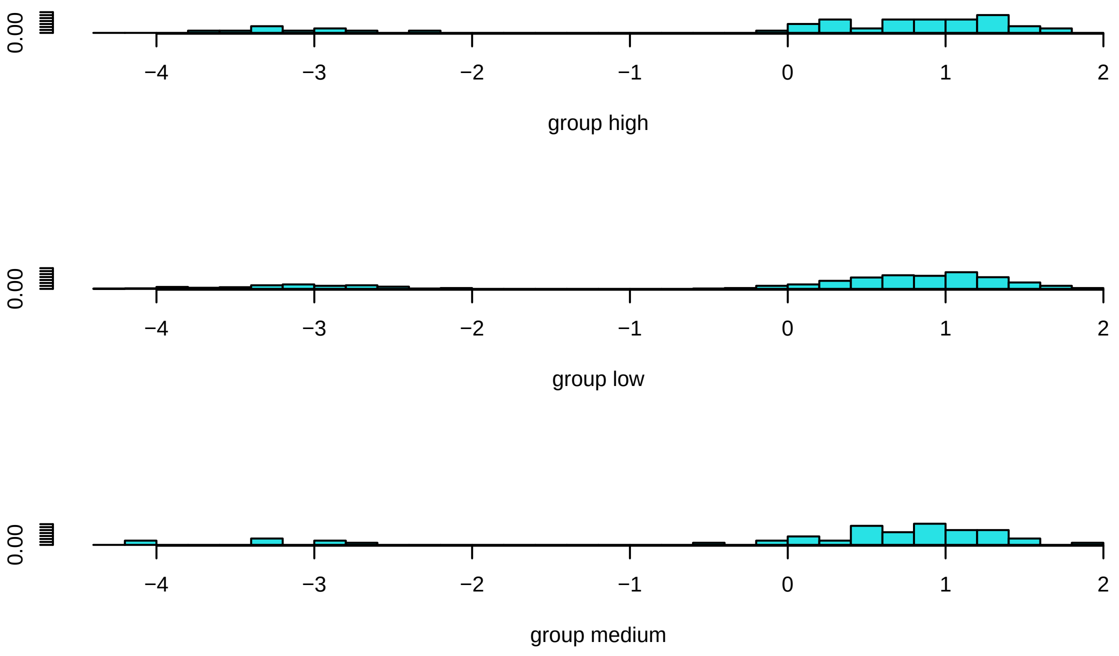
The above plot shows what LD1 and LD2 capture in terms of the data variation. As the prior analysis exhibits, LD1 captures about 96% of the data variability, whereas LD2 captures about 4% of the data variability. As mentioned before, the data is highly imbalanced, and that makes it hard to interpret how LDA is splitting the data points with the linear lines in the manner of classifying the data points into the different class from LD1 vs. LD2 plot.

```
## [1] "black"   "#DF536B" "#61D04F" "#2297E6" "#28E2E5" "#CD0BBC" "#F5C710"
## [8] "gray62"
```



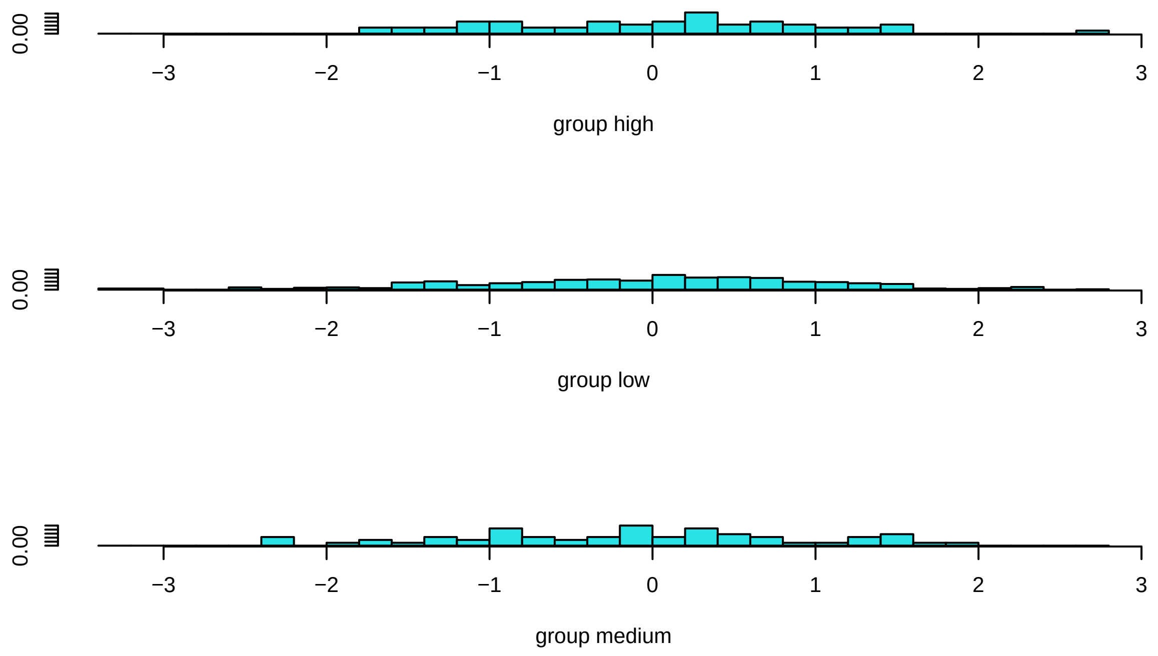
The above bar plot displays how many of the data points were classified into the each class by the LDA model. It is clear that our model classifies more data points as low than the other two classes, and that is consistent with the distribution of the original dataset.

Next, we are plotting our LD1 vs LD2 plot with histogram. It will allow us to show where the data is concentrated for the each class. We will start with displaying what LD1 plane has captured.



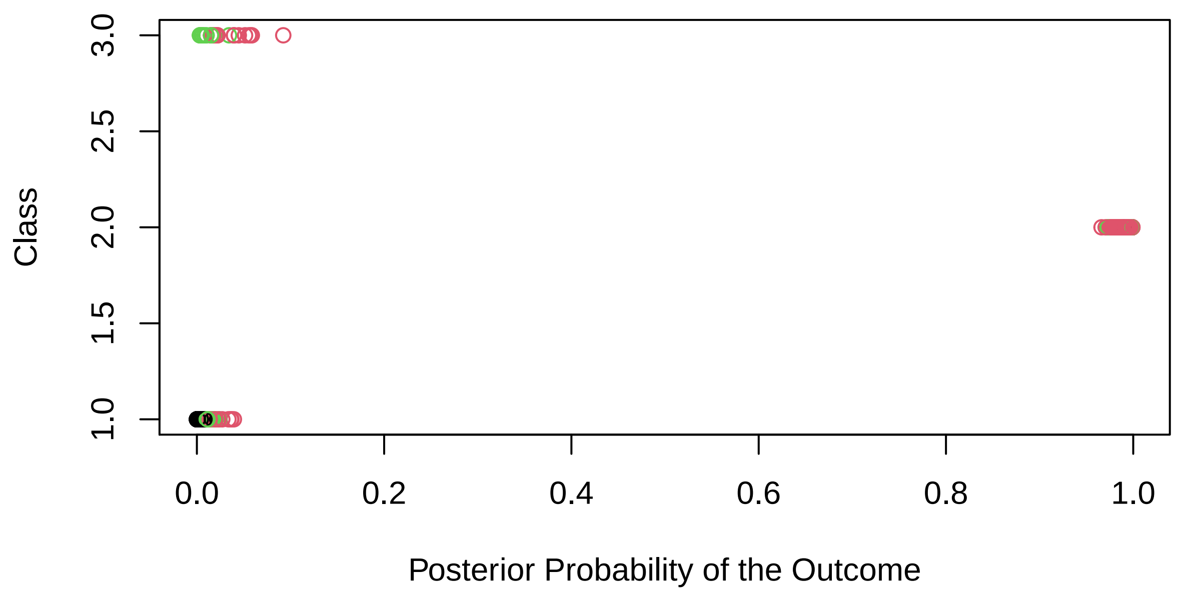
As seen in LD1 vs LD2 in the prior analysis, our data is concentrated into two areas, and that is reflected in the location of the data points in the histograms above. The histogram only shows where the data is, in terms of the data concentration in percentage wise. Therefore, the useful information that we can take from the above histogram is that the mode for the each class is not overlapping. That is because our model successfully classifies our testing data set. Meanwhile, the size of the histogram and its spread are displayed at the same size and nearly located, the actual number of the data is very imbalanced, as 8:1:1. Hence, interpreting the histogram in terms of the data concentration, as the percentage, is very crucial.

Similarly, we now display what LD2 has captured.



The logistic of interpreting the histogram above is the same as LD1, but the difference is that LD2 plane is the plane that is perpendicular to LD1. In other words, LD1 is optimized to capture as many data variations as possible while LD2 is just there to compliment LD1. Hence, the LD2 histogram tends to be not showing how the model is separating the classes as clear as the LD1 histogram, and that is what we have here. Yet, we can see the slight locational difference of the mode in the each class from the above LD2 plot.

To depict a better image of how the model classifys our testing dataset, we will apply dot plot.



The dot plot above has three colors, black, green, and red, as we have have three classes. Red is for low. Black is for high. Green is for medium. With the dot plot, we can see that the LDA model classified them very well as the separation of the all three colors are well depicted.

Now, we will test our Linear Discriminant Analysis model using a 10-Fold Cross Validation approach because we are concerned about introducing bias due to the 60-40 data partition that was performed to observe how the model performed.

```
## Linear Discriminant Analysis
##
## 1338 samples
##    6 predictor
##    3 classes: 'high', 'low', 'medium'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1203, 1205, 1204, 1205, 1205, 1204, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8737149  0.6352901
```

After performing the 10-Fold Cross Validation technique on the LDA model, we find that the accuracy derived is 87.37%, which is a marginal improvement compared to 87.1% accuracy that was derived from our prior analysis, which seems to indicate that our model has a fairly high accuracy rate.

We will now perform the same analysis by fitting our data to a Quadratic Discriminant Analysis (QDA) model.

Quadratic Discriminant Analysis (QDA)

LDA assumes that the observations within each class are drawn from a multivariate Gaussian distribution and the covariance of the predictor variables are common across all k levels of the response variable Y. Quadratic discriminant analysis (QDA), a variant of LDA, provides an alternative approach. QDA assumes that each class has its own covariance matrix. In other words, the predictor variables are not assumed to have common variance across each of the k levels in Y. QDA is particularly useful if there is prior knowledge that individual classes exhibit distinct covariances.

We first fit the QDA model on the training data set that was partitioned above.

Note: We attempted to correct for data imbalances since QDA does not work with imbalanced data, but the current grouping of our multi-class dependent variable was the only segmentation that QDA accepted. We understand that by having a more balanced dataset, our accuracies could be higher, but we are confident that the accuracies derived using the current segmentation were robust as they are.

```
## Call:
## qda(expenses ~ ., data = train_df)
##
## Prior probabilities of groups:
##      high       low    medium
## 0.0996264 0.7957659 0.1046077
##
## Group means:
##         age29-39  age40-50  age51-60 age60 and above   sexmale      bmi
## high   0.2125000 0.2375000 0.1750000      0.15000000 0.6375000 34.74625
## low    0.2222222 0.2269171 0.1830986      0.06259781 0.4882629 30.39609
## medium 0.1071429 0.3095238 0.2380952      0.11904762 0.5238095 28.24286
##        children  smokeryes regionnorthwest regionsoutheast regionsouthwest
## high   1.287500 0.97500000       0.1500000       0.4000000       0.2625000
## low    1.115806 0.05946792       0.2644757       0.2535211       0.2550861
## medium 1.059524 0.65476190       0.3333333       0.2500000       0.1547619
```

```
##
## qda.class high low medium
##    high     51   4     13
##    low       1 395     27
##    medium    1  27     16
```

```
## [1] 0.8635514
```

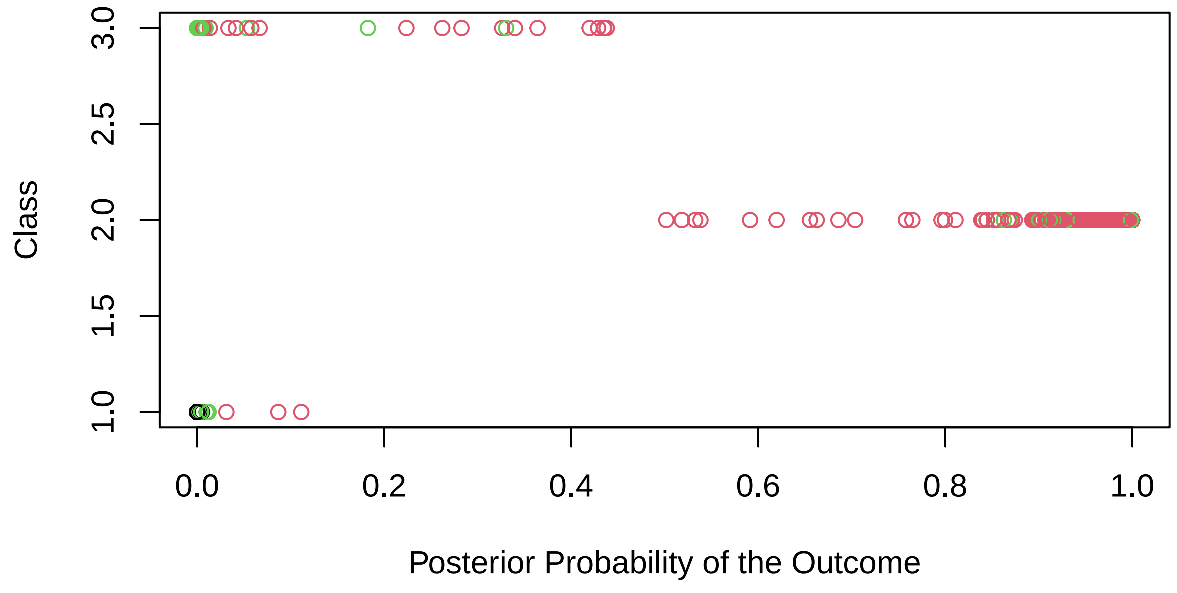
Next, we test the trained model's performance by running the QDA model on the testing data frame to predict the actual values of insurance cost premiums. The following confusion matrix is obtained. As can be seen, the model performs well with a high accuracy of 86.35%.

Now, we will transition to testing our Quadratic Discriminant Analysis model using a 10-Fold Cross Validation approach because we are concerned about introducing bias due to the 60-40 data partition that was performed to observe how the model performed.

```
## Quadratic Discriminant Analysis
##
## 1338 samples
##    6 predictor
##    3 classes: 'high', 'low', 'medium'
```

```
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1203, 1205, 1204, 1205, 1205, 1204, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8543113  0.5967008
```

After performing the 10-Fold Cross Validation technique on the Multinomial Logistic Model, we find that the accuracy derived is 85.43%, which is a marginal drop compared to 86.35% accruacy that was derived from our prior analysis.



The above figure shows how the test data has been classified using the QDA model. The plot has three colors, black, green, and red, as we have have three classes. Red is for low. Black is for high. Green is for medium. The Predicted low, medium and high expense categories has been colored with actual classification with red, green and black color. The mix of color in the three categories show the incorrect classification prediction.

Now, we will perform the same analysis for K-Neighbors.

K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a non-parametric method that is used for classification. How the KNN is performed is it uses nearby observations to classify if a new observation would be identified near similar observed observations. This is done by designating one parameter, K, which represents the number of nearby observation or "neighbors" that will be used to classify a new record or in other words observation. This approach of classifying the observation is data-driven and does not require assumptions about the data, which is the non-parametric element in place. Typically, the measured distance that is used when using the KNN is the Euclidean distance, thus, we will normalize our data to ensure that we are in compliance of the model specification. We perform this normalization of the data because we do not want certain predictor variables to dominate the others.

```
## 'data.frame':    1338 obs. of  7 variables:
##  $ age     : num  1 1 1 2 2 2 3 2 2 4 ...
##  $ sex     : num  1 2 2 2 2 1 1 1 2 1 ...
##  $ bmi     : num  27.9 33.8 33 22.7 28.9 25.7 33.4 27.7 29.8 25.8 ...
##  $ children: int  0 1 3 0 0 0 1 3 2 0 ...
##  $ smoker  : num  2 1 1 1 1 1 1 1 1 1 ...
##  $ region  : num  4 3 3 2 2 3 3 2 1 2 ...
##  $ expenses: Factor w/ 3 levels "high","low","medium": 2 2 2 3 2 2 2 2 2 3 ...
```

For KNN to work efficiently, we converted all our predictor variables to numeric variables to ensure the normalization of the data makes logical sense. We still hold onto the concept that was introduced earlier that we are grouping our dependent variable into three insurance cost premium groupings for consistency.

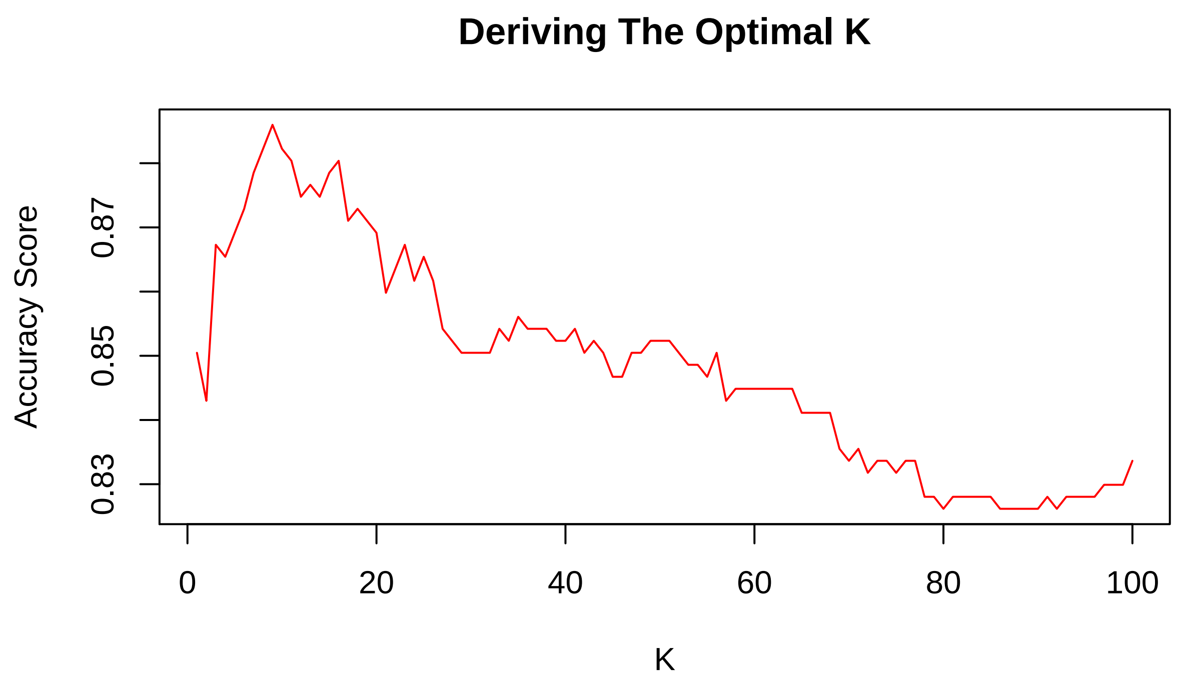
We again partition our normalized data into a 60-40 training and testing datasets to fit and train a KNN model with our data. We begin by fitting the KNN model with a value of 1 for K and observe the accuracy of the model.

```
## [1] 803   6
```

```
## NULL
```

As we can see, we derive an accuracy of 90.18%, which is similar to the accuracy we derived from the Mulitnomial Logistic Regression. However, we simply just used 1 as our K and we are concerned that perhaps there exists an optimal K that would provide us with a better result.

```
## [1] 0.8859813
```



After trying 100 different values of K, we find the optimal K to be 9 with an associated accuracy of 88.59%. As we can see from the plot, as we increase parameter K, the accuracy begins to decrease, which one can argue that overfitting is present.

We will now transition to performing a 10-Fold Cross Validation because again we are concerned that by partitioning our data, we are introducing bias into our analysis.

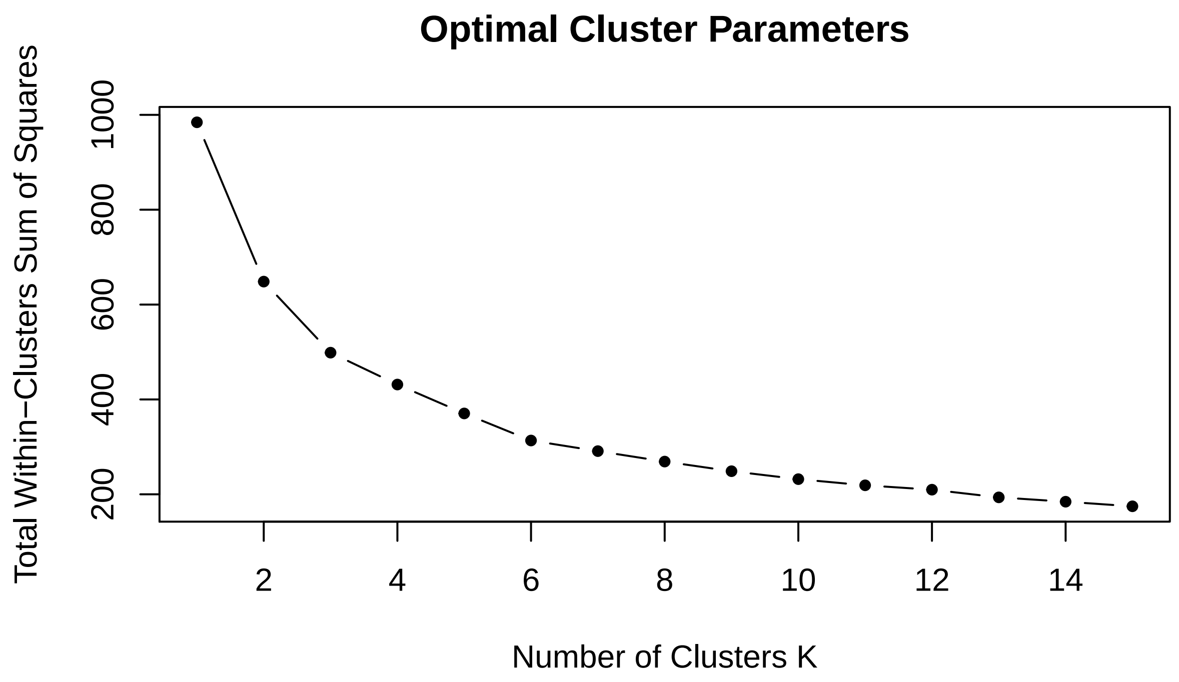
```
## k-Nearest Neighbors
##
## 1338 samples
##    6 predictor
##    3 classes: 'high', 'low', 'medium'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1204, 1204, 1204, 1204, 1204, 1204, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.8378128  0.3523761
##   7  0.8303333  0.2792872
##   9  0.8213612  0.2090717
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

After performing the 10-Fold Cross Validation analysis, we find that the optimal K parameter is 5, however, we find that the accuracy decreases from 90.18% to 83.78%, which we collectively believe that the cross validation was correcting for the level of bias we were introducing by partitioning our data initially.

K-Means

We will now use our data to fit a K-Means Model and observe how the model performs. The concept behind the K-Means model is that we want to partition our dataset into clusters, where each observation belongs to at least one of the number of clusters, K, we assign. The clusters are non-overlapping, which means that no observation belongs to more than one cluster.

```
## K-means clustering with 6 clusters of sizes 159, 278, 269, 115, 299, 218
##
## Cluster means:
##         age sex       bmi  children smoker    region
## 1 0.3679245   1 0.4179932 0.2377358      1 0.5241090
## 2 0.6699640   0 0.4099944 0.2294964      0 0.5119904
## 3 0.1068773   0 0.3732603 0.2052045      0 0.5068154
## 4 0.3826087   0 0.3669753 0.2017391      1 0.4753623
## 5 0.1647157   1 0.3722561 0.2501672      0 0.4024526
## 6 0.6788991   1 0.4337273 0.1752294      0 0.6376147
##
## Clustering vector:
##    [1] 4 5 5 5 5 3 2 3 5 2 5 4 5 2 1 5 2 5 6 1 2 3 5 4 5 2 2 2 5 1 1 3 3 6 1 5 2
##   [38] 5 1 1 3 3 6 3 5 6 3 3 2 1 3 3 1 1 2 1 2 1 4 3 5 5 6 3 4 3 2 5 2 1 4 5 2 6
##   [75] 6 6 3 5 3 2 5 2 1 2 4 1 4 2 2 2 3 2 1 5 4 3 2 6 1 1 2 5 3 4 3 1 3 5 5 1 6
##  [112] 2 6 3 6 6 6 4 2 3 6 5 3 1 2 3 4 2 4 6 2 2 2 5 3 3 5 5 2 3 5 5 1 5 1 3 1 2
##  [149] 2 5 5 6 3 4 2 6 1 1 1 2 4 4 6 3 5 5 3 3 3 5 6 6 5 3 3 4 5 6 2 2 6 3 5 2 6
##  [186] 1 3 3 2 3 6 3 5 2 5 5 3 2 2 2 3 2 2 4 6 3 6 1 2 5 5 5 5 3 2 2 2 5 3 3 3 2
##  [223] 5 1 1 6 5 2 2 5 2 2 3 6 5 4 3 5 1 6 4 3 2 6 4 6 2 5 5 5 1 4 1 5 1 2 1 3 6
##  [260] 1 2 4 1 1 2 1 1 2 6 5 5 1 5 5 5 2 5 3 6 2 4 1 5 2 2 6 2 2 4 6 3 5 1 3 5 5
##  [297] 1 1 1 2 5 4 2 3 2 5 3 3 6 2 6 3 1 6 4 6 6 6 2 5 5 3 1 6 6 5 3 1 4 6 4 1 2
##  [334] 2 2 6 6 6 1 2 3 6 2 6 2 3 5 5 3 5 2 2 3 5 3 6 6 5 5 3 2 6 4 3 3 2 2 2 5 5
##  [371] 2 2 2 1 5 4 4 1 2 6 4 1 6 3 5 5 2 5 3 3 5 3 5 5 2 5 2 5 6 3 2 6 2 5 6 2 3
##  [408] 2 5 5 5 4 4 5 3 2 6 4 6 4 1 1 1 5 6 6 3 3 3 3 5 3 6 2 5 6 5 5 2 5 3 4 5 2
##  [445] 1 2 6 2 2 6 5 5 5 5 5 6 2 2 6 2 2 1 2 6 5 4 2 2 3 3 5 3 3 2 1 1 1 5 5 5 6
##  [482] 6 3 2 6 3 2 5 4 6 3 2 3 6 1 5 3 6 2 2 1 5 1 1 3 5 5 5 3 2 6 5 6 5 1 6 5 5
##  [519] 3 5 2 3 2 3 1 3 3 2 5 5 1 2 6 5 6 5 3 2 2 6 3 3 2 4 6 1 5 2 3 4 6 3 6 2 3
##  [556] 5 5 5 4 5 2 2 5 6 3 3 3 5 2 1 3 3 3 2 2 2 5 4 6 3 6 5 5 3 5 3 5 4 2 3 2 5
##  [593] 5 4 6 2 2 3 6 2 3 6 2 2 4 2 3 4 5 1 2 3 3 3 3 4 2 1 4 2 6 1 2 1 6 3 5 5 6
##  [630] 4 6 5 3 5 6 6 3 3 1 6 5 1 6 3 6 5 5 2 5 2 2 2 2 2 2 4 3 5 2 2 5 2 3 5 4 1
##  [667] 6 4 1 2 5 3 5 2 4 5 2 1 6 2 3 5 1 6 3 6 5 6 2 1 5 6 5 5 3 3 2 1 6 3 3 2 6
##  [704] 3 2 3 4 5 3 3 5 2 2 5 3 6 2 6 2 2 2 6 6 5 2 4 5 4 3 2 1 6 3 2 2 2 4 5 1 1
##  [741] 5 1 1 3 6 2 6 5 2 5 4 5 6 2 5 5 3 4 5 1 3 5 1 5 2 2 6 2 2 3 6 2 2 4 5 6 5
##  [778] 5 5 6 1 5 6 4 3 3 6 5 5 2 3 5 3 1 3 1 5 3 2 1 2 2 5 4 5 2 2 3 5 5 2 3 6 5
##  [815] 5 3 3 5 4 4 6 5 3 2 6 2 1 1 1 5 6 3 3 6 5 6 6 2 3 2 5 6 4 4 6 4 2 5 3 6 4
##  [852] 6 4 2 4 3 4 1 3 6 4 3 2 3 6 6 5 6 6 3 6 3 6 6 5 3 2 5 6 3 5 5 3 4 5 1 1 3
##  [889] 5 6 4 3 6 1 6 2 4 5 3 3 5 1 5 6 2 3 5 2 6 4 5 1 2 2 5 3 4 1 2 3 2 2 6 5 6
##  [926] 5 3 2 2 6 5 3 6 2 5 2 5 3 5 6 5 2 3 5 6 2 6 1 6 1 6 1 3 1 1 5 1 5 1 6 3 3
##  [963] 2 5 6 6 1 5 5 3 2 3 3 3 5 1 6 5 2 3 6 5 1 3 5 2 5 2 5 4 3 3 2 5 4 3 3 2 3
## [1000] 3 1 1 5 6 5 5 5 1 5 6 2 1 2 5 3 6 3 3 2 3 6 4 1 5 2 3 1 5 6 3 4 4 3 1 6 2
## [1037] 1 4 5 5 4 5 1 3 6 4 2 1 3 1 2 6 5 1 3 6 2 2 3 5 5 6 1 5 3 2 6 5 6 2 1 6 5
## [1074] 2 2 3 2 5 1 6 5 5 5 6 2 4 6 6 6 6 1 2 5 4 2 3 4 5 2 3 4 6 5 6 6 2 2 2 5 6
## [1111] 2 1 4 3 5 6 5 1 1 3 4 6 4 3 4 2 6 3 6 3 3 5 6 2 5 2 2 3 5 4 6 2 2 5 6 6 1
## [1148] 3 6 6 3 2 4 3 2 3 1 3 3 3 2 5 5 3 2 3 6 3 6 3 1 4 2 5 5 3 4 2 3 1 2 3 3 2
## [1185] 4 5 1 2 4 3 3 2 2 2 3 3 4 6 5 3 5 5 5 6 4 5 2 1 4 6 5 5 5 2 3 5 6 5 4 3 3
## [1222] 6 6 4 5 3 5 6 6 6 1 4 2 6 2 5 2 2 5 3 1 1 3 3 5 5 2 5 3 1 1 5 1 2 3 2 2 2
## [1259] 6 2 3 5 2 2 2 1 2 1 3 6 5 3 6 5 1 6 3 3 1 3 2 4 4 5 1 2 3 3 1 6 3 1 5 5 6
## [1296] 5 5 3 5 3 1 1 3 1 1 3 4 1 4 6 5 3 6 4 4 5 3 5 5 3 5 1 6 4 5 6 2 6 3 6 2 3
## [1333] 2 5 3 3 3 4
## 
## Within cluster sum of squares by cluster:
## [1] 49.06577 69.91056 63.39345 36.21143 74.32281 53.42560
##  (between_SS / total_SS =  64.8 %)
## 
## Available components:
## 
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```



After fitting our data using the K-Means algorithm, we will now determine what our optimal number of clusters will be for our model. However, we first have to determine our criteria that will be used when determining the optimal number of clusters. For our analysis, we look to minimize the total within-cluster variation or in other words, the total within-cluster sum of square. The total within-cluster sum of square measures the compactness of the clustering, thus, we want the numeric value to be as small as possible. To assist us in minimizing the total within-cluster sum of square, we will use the “Elbow Method” where we will fit a number of different clusters and determine the value where the curve will begin to “bend”. Thus, base on the results derived above, it appears 4 is the optimal number of clusters to use for our data.

```
## K-means clustering with 4 clusters of sizes 676, 176, 198, 288
##
## Cluster means:
##         age sex       bmi  children     smoker    region
## 1 0.3783284   1 0.4028374 0.2230769 0.23520710 0.5069034
## 2 0.1022727   0 0.3431144 0.2068182 0.25568182 0.2140152
## 3 0.6931818   0 0.3956955 0.2121212 0.26767677 0.2575758
## 4 0.3602431   0 0.4092075 0.2215278 0.05902778 0.8495370
##
## Clustering vector:
##    [1] 4 1 1 1 1 4 4 2 1 3 1 3 1 4 1 1 3 1 1 1 3 4 1 2 1 4 3 3 1 1 1 2 4 1 1 1 3
##   [38] 1 1 1 2 4 1 4 1 1 2 2 4 1 2 2 1 1 3 1 3 1 3 2 1 1 1 2 2 4 4 1 4 1 2 1 4 1
##   [75] 1 1 4 1 2 3 1 3 1 3 4 1 3 4 3 3 4 3 1 1 3 4 4 1 1 1 4 1 2 3 4 1 4 1 1 1 1
##  [112] 4 1 2 1 1 1 2 4 2 1 1 2 1 3 2 4 4 2 1 3 3 4 1 2 4 1 1 4 4 1 1 1 1 1 4 1 4
##  [149] 3 1 1 1 2 3 3 1 1 1 1 4 3 2 1 4 1 1 4 2 2 1 1 1 1 4 2 3 1 1 4 3 1 4 1 3 1
##  [186] 1 4 4 4 2 1 4 1 3 1 1 4 4 3 3 2 4 3 2 1 2 1 1 4 1 1 1 1 4 4 4 3 1 4 4 4 3
##  [223] 1 1 1 1 1 4 3 1 3 4 4 1 1 3 4 1 1 1 2 2 4 1 3 1 4 1 1 1 1 3 1 1 1 3 1 4 1
##  [260] 1 4 2 1 1 4 1 1 3 1 1 1 1 1 1 1 3 1 4 1 4 3 1 1 3 4 1 3 3 3 1 4 1 1 4 1 1
##  [297] 1 1 1 3 1 3 4 4 4 1 4 4 1 3 1 4 1 1 4 1 1 1 3 1 1 2 1 1 1 1 4 1 3 1 3 1 3
##  [334] 3 3 1 1 1 1 4 4 1 3 1 4 4 1 1 4 1 3 4 4 1 4 1 1 1 1 4 3 1 4 4 4 3 3 3 1 1
##  [371] 3 3 3 1 1 2 2 1 3 1 2 1 1 4 1 1 4 1 2 2 1 2 1 1 3 1 4 1 1 4 4 1 3 1 1 3 4
##  [408] 4 1 1 1 3 2 1 2 4 1 4 1 3 1 1 1 1 1 1 2 2 2 2 1 2 1 4 1 1 1 1 4 1 2 4 1 4
##  [445] 1 4 1 3 4 1 1 1 1 1 1 1 4 3 1 4 4 1 3 1 1 2 4 3 2 4 1 2 4 3 1 1 1 1 1 1 1
##  [482] 1 4 4 1 2 3 1 3 1 4 3 2 1 1 1 4 1 4 4 1 1 1 1 4 1 1 1 2 4 1 1 1 1 1 1 1 1
##  [519] 4 1 3 4 3 4 1 4 2 4 1 1 1 3 1 1 1 1 4 4 4 1 4 4 3 3 1 1 1 4 2 3 1 4 1 3 2
##  [556] 1 1 1 2 1 3 3 1 1 4 2 2 1 4 1 4 4 4 3 3 3 1 2 1 2 1 1 1 4 1 4 1 2 3 4 4 1
##  [593] 1 2 1 3 4 2 1 3 4 1 4 3 2 4 2 3 1 1 4 4 2 2 4 3 3 1 2 4 1 1 4 1 1 2 1 1 1
##  [630] 3 1 1 4 1 1 1 2 2 1 1 1 1 1 2 1 1 1 3 1 3 4 4 4 4 4 3 4 1 3 3 1 4 2 1 3 1
##  [667] 1 3 1 4 1 2 1 4 3 1 4 1 1 3 4 1 1 1 4 1 1 1 4 1 1 1 1 1 4 2 3 1 1 4 4 3 1
##  [704] 2 3 4 3 1 2 2 1 4 3 1 4 1 3 1 3 3 3 1 1 1 3 2 1 2 2 4 1 1 4 3 4 3 2 1 1 1
##  [741] 1 1 1 4 1 3 1 1 4 1 2 1 1 4 1 1 2 3 1 1 2 1 1 1 3 3 1 4 4 2 1 4 3 2 1 1 1
##  [778] 1 1 1 1 1 1 4 4 4 1 1 1 3 4 1 2 1 2 1 1 2 4 1 4 3 1 2 1 3 3 2 1 1 4 2 1 1
##  [815] 1 4 2 1 3 2 1 1 4 4 1 3 1 1 1 1 1 2 2 1 1 1 1 3 2 3 1 1 2 3 1 3 4 1 4 1 2
##  [852] 1 3 3 3 4 3 1 4 1 4 4 3 2 1 1 1 1 1 4 1 4 1 1 1 2 4 1 1 4 1 1 2 3 1 1 1 2
##  [889] 1 1 3 4 1 1 1 4 3 1 4 2 1 1 1 1 4 2 1 4 1 4 1 1 3 4 1 4 3 1 4 4 4 4 1 1 1
##  [926] 1 4 4 3 1 1 4 1 4 1 4 1 2 1 1 1 4 2 1 1 4 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 4
##  [963] 3 1 1 1 1 1 1 4 4 2 2 4 1 1 1 1 3 4 1 1 1 2 1 4 1 3 1 2 4 2 4 1 2 2 4 3 2
## [1000] 2 1 1 1 1 1 1 1 1 1 1 4 1 3 1 4 1 2 4 3 2 1 2 1 1 4 4 1 1 1 2 3 3 2 1 1 4
## [1037] 1 3 1 1 2 1 1 4 1 3 3 1 2 1 3 1 1 1 2 1 4 4 4 1 1 1 1 1 4 4 1 1 1 4 1 1 1
## [1074] 3 3 4 4 1 1 1 1 1 1 1 3 4 1 1 1 1 1 3 1 2 4 2 3 1 3 4 2 1 1 1 1 4 3 3 1 1
## [1111] 3 1 3 2 1 1 1 1 1 2 4 1 3 2 2 3 1 4 1 4 4 1 1 3 1 3 4 2 1 2 1 4 4 1 1 1 1
## [1148] 2 1 1 2 3 3 2 3 2 1 2 2 4 3 1 1 2 3 2 1 4 1 2 1 4 4 1 1 4 3 4 2 1 3 2 4 3
## [1185] 2 1 1 3 3 4 2 3 3 3 2 2 2 1 1 4 1 1 1 1 2 1 4 1 2 1 1 1 1 4 2 1 1 1 4 2 2
## [1222] 1 1 2 1 4 1 1 1 1 1 4 3 1 4 1 3 3 1 4 1 1 2 4 1 1 4 1 4 1 1 1 1 4 4 4 3 3
## [1259] 1 3 2 1 4 4 3 1 4 1 4 1 1 2 1 1 1 1 2 2 1 2 4 3 2 1 1 3 2 4 1 1 2 1 1 1 1
## [1296] 1 1 4 1 2 1 1 4 1 1 4 2 1 4 1 1 2 1 4 2 1 4 1 1 2 1 1 1 3 1 1 3 1 2 1 4 4
## [1333] 4 1 2 4 4 3
## 
## Within cluster sum of squares by cluster:
## [1] 340.08229  58.64811  74.02341  74.21216
##  (between_SS / total_SS =  44.4 %)
## 
## Available components:
## 
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

As you can see, the "Within cluster sum of squares by cluster" percentage is reported at 44.4% compared to 64.8% when we designated the number of clusters to be at 6 when we initially fit the data on the model. Thus, it seems that by clustering by four, we have greatly minimized the within cluster sum of squares. However, it should be noted that K-Means is an unsupervised model, thus it is possible that there exists another number of clusters that would decrease the within-cluster sum of squares more, but for now, our analysis states that using a clustering of four is preferred.

Since we cannot derive accuracy metrics based on a cross-validation framework, we will bootstrap our model and observe how it performs based on using the number of clusters of four based on the analysis that was completed above. We performed the bootstrap based on 1000 replications.

```
## [1] 0.6684560 0.6138198 0.6005266 0.6299498
```

After we performed the bootstrap on our model, we wanted to observe the Jaccard Coefficients because these coefficients will tell us how stable the clusters are and if the coefficients have a value of 0.5 and greater, then the cluster is deemed important. Thus, based on deriving coefficients that are greater than 0.5, then we can conclude that the number of clusters of 4 is deemed acceptable and every cluster is stable based on the data we are using.

```r
# Accuracy table of results of all 5 models
Accuracy_Table = matrix(c("90.09%","90.73%","87.10%","87.37%","86.35%","85.43%","90.18%","83.78%"),ncol
colnames(Accuracy_Table) = c("Testing Accuracy     ", "Cross Validated Accuracy")
rownames(Accuracy_Table) = c("Mulitnomial Logistic Regression", "Linear Discriminant Analysis (LDA)","Q
Accuracy_Table = as.table(Accuracy_Table)
Accuracy_Table
```

```
##                                         Testing Accuracy
## Mulitnomial Logistic Regression         90.09%
## Linear Discriminant Analysis (LDA)      87.10%
## Quadratic Discriminant Analysis (QDA) 86.35%
## K-Nearest Neighbors (KNN)               90.18%
##                                         Cross Validated Accuracy
## Mulitnomial Logistic Regression         90.73%
## Linear Discriminant Analysis (LDA)      87.37%
## Quadratic Discriminant Analysis (QDA) 85.43%
## K-Nearest Neighbors (KNN)               83.78%
```

After fitting and evaluating 5 different classification models, we have gathered the training/testing and the cross valuated accuracy rates to summarize our results. Based on the results, it is evident that the Multinomial Logistic Regression is the best model to use based on higher accuracy score achieved. The K-Nearest Neighbors (KNN) model performed marginally better in the training/testing analysis, but the cross validated score notably decreased and actually performed the worse out of the classification models. It should be noted that since we cannot derive accuracy metrics for the K-Means, we omitted K-Means from the table.

Thus, we conclude that the Multinomial Logistic Regression Model is the best model to use and also conclude that for our data, a linear model is the best to use. We can conclude that the Multinomial Logistic Regression is the appropriate model to use for our data becuase of the high testing and cross validated accuracy scores that were derived.

III. Regularization (Part 2)

Exploratory Data Analysis

Data Description:

The following data is taken from the DP03 (Selected Economic Characteristics) and DP05 (ACS Demographic and Housing Estimates) tables of the 2017 American Community Survey 5-year estimates. Data is provided for each county or county-equivalent in the US, including DC and Puerto Rico. Counties are political subdivisions, and the boundaries of some have been set for centuries.

While the data is collected from the US census bureau, we sourced it from Kaggle where it was partially cleaned and uploaded. It consists of 3220 county observations and 34 variables. All feature variables are estimated as percentages of the total population in each county, and the target variable represents the per capita Income per county in 2017.

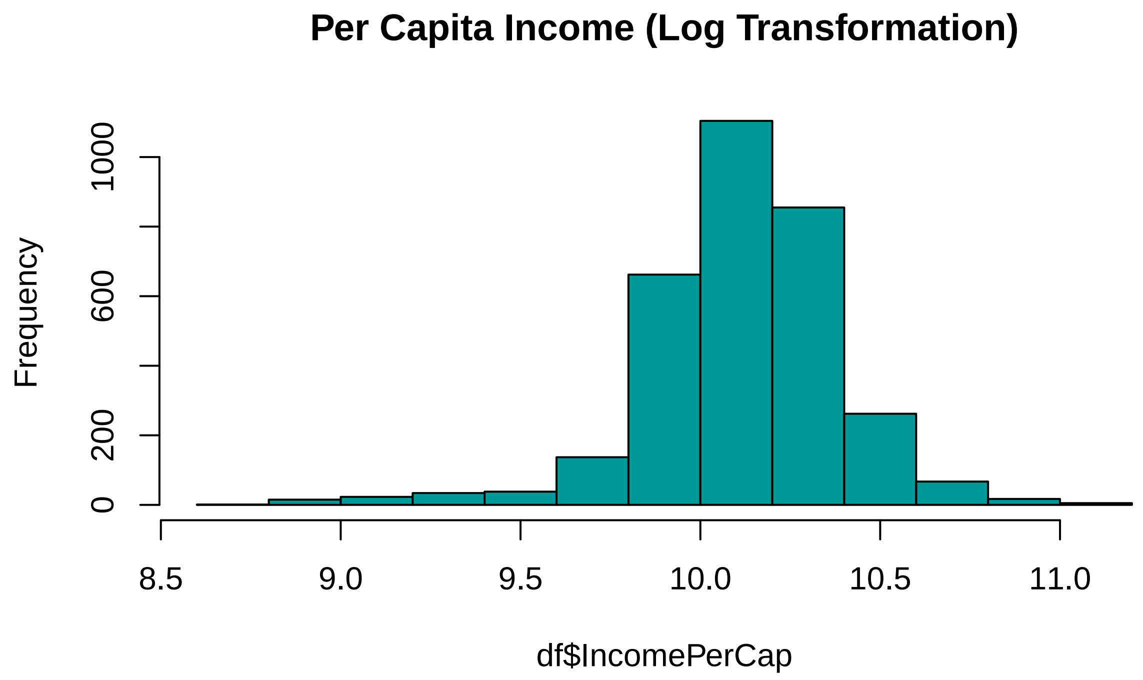
```
## Nans in data: 0
```

```
## 'data.frame':    3219 obs. of  27 variables:
##  $ IncomePerCap    : num  10.23 10.29 9.77 9.95 10 ...
##  $ Men             : num  0.489 0.489 0.533 0.543 0.494 ...
##  $ Women           : num  0.511 0.511 0.467 0.457 0.506 ...
##  $ Hispanic        : num  2.7 4.4 4.2 2.4 9 0.3 0.3 3.6 2.2 1.6 ...
##  $ White           : num  75.4 83.1 45.7 74.6 87.4 21.6 52.2 72.7 56.2 91.8 ...
##  $ Black           : num  18.9 9.5 47.8 22 1.5 75.6 44.7 20.4 39.3 5 ...
##  $ Native          : num  0.3 0.8 0.2 0.4 0.3 1 0.1 0.2 0.3 0.5 ...
##  $ Asian           : num  0.9 0.7 0.6 0 0.1 0.7 1.1 1 1 0.1 ...
##  $ VotingAgeCitizen: num  0.745 0.764 0.774 0.782 0.737 ...
##  $ Poverty         : num  13.7 11.8 27.2 15.2 15.6 28.5 24.4 18.6 18.8 16.1 ...
##  $ ChildPoverty    : num  20.1 16.1 44.9 26.6 25.4 50.4 34.8 26.6 29.1 20 ...
##  $ Professional    : num  35.3 35.7 25 24.4 28.5 19.7 26.9 29 24.3 28.8 ...
##  $ Service         : num  18 18.2 16.8 17.6 12.9 17.1 17.3 17.5 13.5 14.8 ...
##  $ Office          : num  23.2 25.6 22.6 19.7 23.3 18.6 18.5 23.7 23 18.1 ...
##  $ Construction    : num  8.1 9.7 11.5 15.9 15.8 14 11.6 10.4 11.6 11.9 ...
##  $ Production      : num  15.4 10.8 24.1 22.4 19.5 30.6 25.7 19.4 27.6 26.5 ...
##  $ Drive           : num  86 84.7 83.4 86.4 86.8 73.1 83.6 85 87.1 85 ...
##  $ Carpool         : num  9.6 7.6 11.1 9.5 10.2 15.7 12.6 9.2 9.7 12.1 ...
##  $ Transit         : num  0.1 0.1 0.3 0.7 0.1 0.3 0 0.2 0.2 0.4 ...
##  $ Walk            : num  0.6 0.8 2.2 0.3 0.4 6.2 0.9 1.3 0.6 0.3 ...
##  $ OtherTransp     : num  1.3 1.1 1.7 1.7 0.4 1.7 0.9 1.1 0.5 0.3 ...
##  $ WorkAtHome      : num  2.5 5.6 1.3 1.5 2.1 3 2 3.2 2 2 ...
##  $ PrivateWork     : num  74.1 80.7 74.1 76 83.9 81.4 79.1 74.9 84.5 74.8 ...
##  $ PublicWork      : num  20.2 12.9 19.1 17.4 11.9 13.6 15.3 19.9 11.8 17.1 ...
##  $ SelfEmployed    : num  5.6 6.3 6.5 6.3 4 5 5.3 5.1 3.7 8.1 ...
##  $ FamilyWork      : num  0.1 0.1 0.3 0.3 0.1 0 0.3 0.1 0 0 ...
##  $ Unemployment    : num  5.2 5.5 12.4 8.2 4.9 12.1 7.6 10.1 6.4 5.3 ...
##  - attr(*, "na.action")= 'omit' Named int 549
##   ..- attr(*, "names")= chr "Kalawao County, Hawaii"
```

Some variables- Men, Women and Voting Age Citizen- were given as population estimates in the dataframe. We transformed them to represent percent share of total population in each county.We also changed the dataframe's index to the respective county and state names for each row.

Since the dataset was only partially cleaned when downloaded form kaggle, some redundant variables were present in the dataset. We removed these by using the results from Boruta. Variables including Pacific, State, County, MeanCommute, Income and related error variables.

After dropping redundant and irrelevant variables and removing the nans, we ended up with a dataset containing 27 variables including the target variable and 3219 observations, the summary statistics of which can be found below:



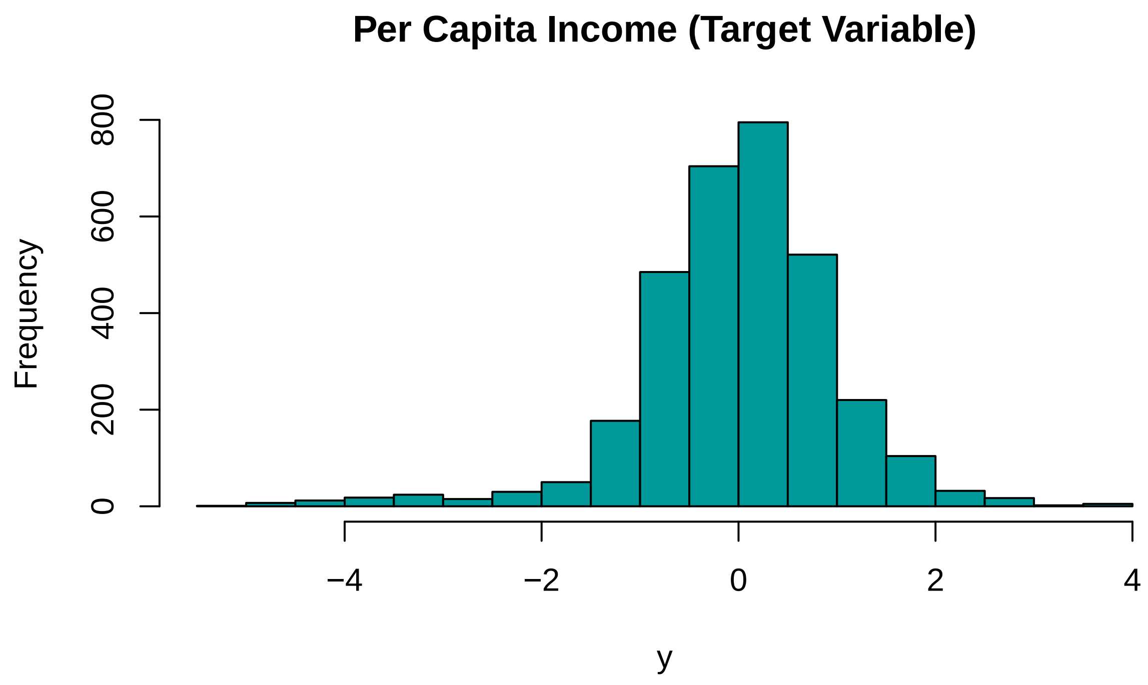
From the statistical summary above, it is evident that the units of all the quantitative variables, are of comparable magnitudes with the log transformed IncomePerCap (target). The histogram of the log transformed dependent variable shows that the dependent variable is negatively skewed and may require further scaling. Since there were 26

** Regularization Methods**

Linear regression models comprising of a large number of features may suffer from overfitting, multicollinearity, or may result in a computationally intensive model.Regularization techniques are thus used to regularize or shrink coefficient estimates toward zero which reduces the risk of overfitting and increases model interpretability (especially for datasets with a large number of features).

Regularization significantly reduces the variance of the model without substantial increase in its bias. Various Regularization models are fitted on the dataset below to determine which model should be preferred for regularization and feature selection.

Splitting and Scaling the data



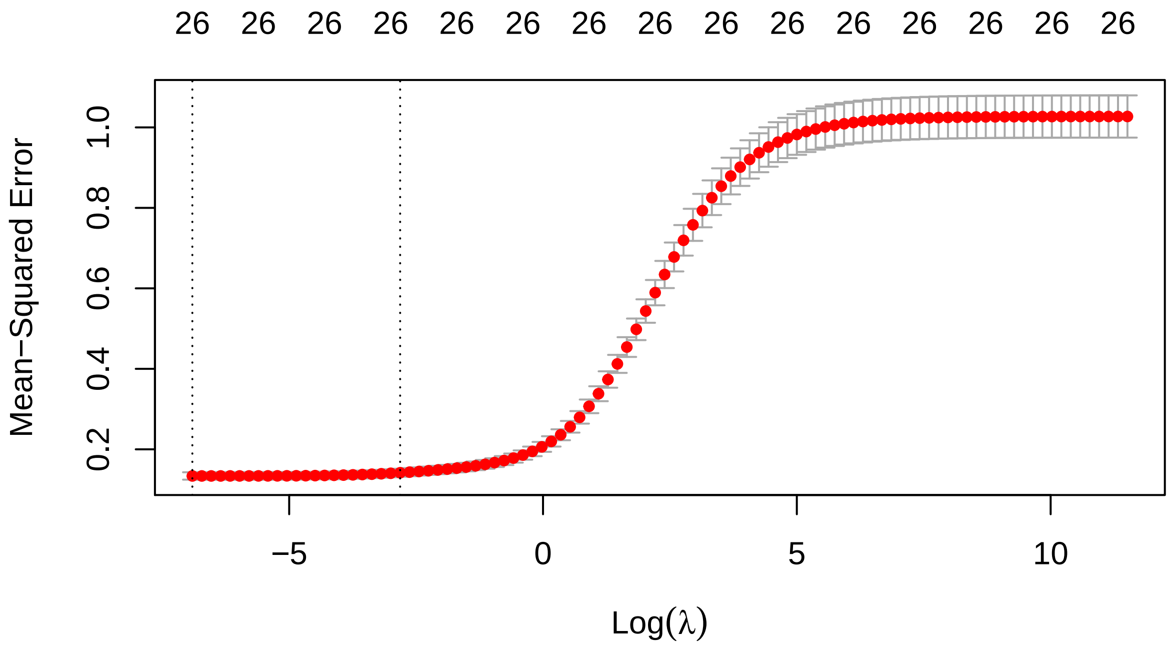
The dataset was randomly split into training (60%) and testing data (40%). Both the features and the target variable were normalized. The train and test features were normalized after splitting them in order to avoid "leaking" any knowledge of testing data while training the model. Feature scaling helped speed up our algorithm's convergence process.

The histogram of the dependent variable shows that after scaling the y variable, Per capita income is normally distributed for the most part, albeit still slightly negatively skewed.

Ridge Regression

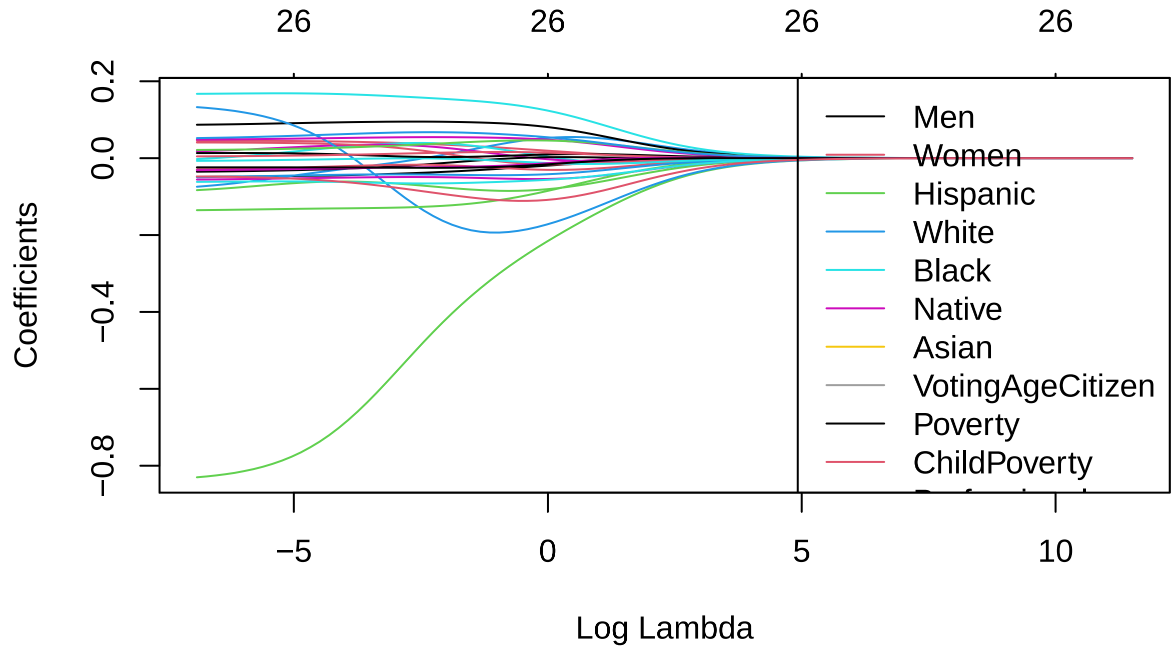
The first model used for regularization is the Ridge regression. It shrinks the regression coefficients, so that variables, with minor contribution to the outcome, have their coefficients close to zero. The shrinkage of the coefficients is achieved by penalizing the regression model with a penalty term called L2-norm, which is the sum of the squared coefficients.

```
## Optimal Lambda: 0.001
```

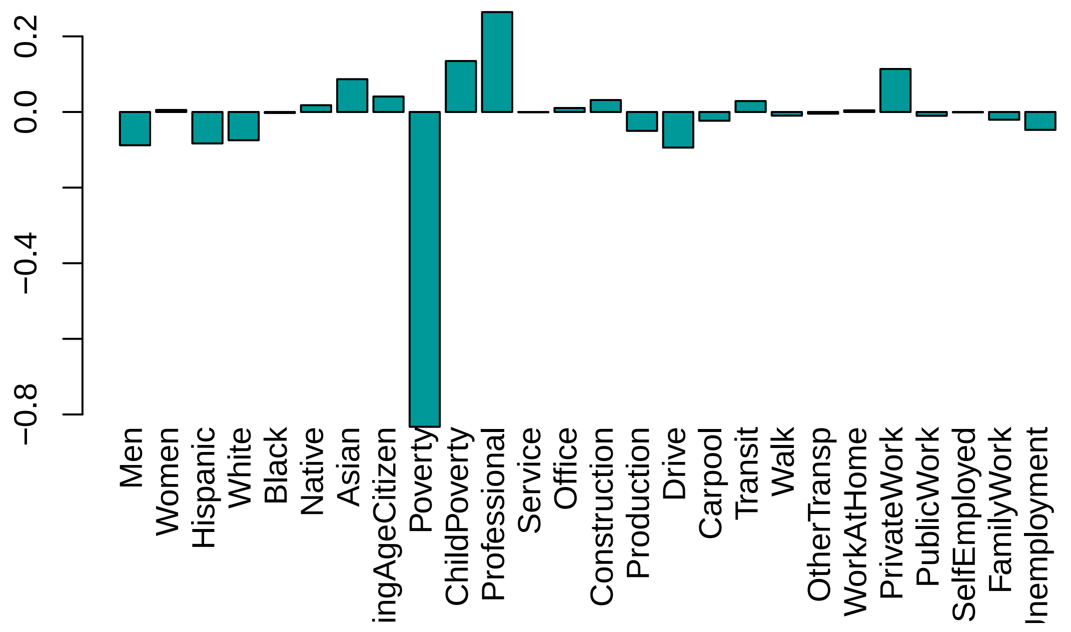


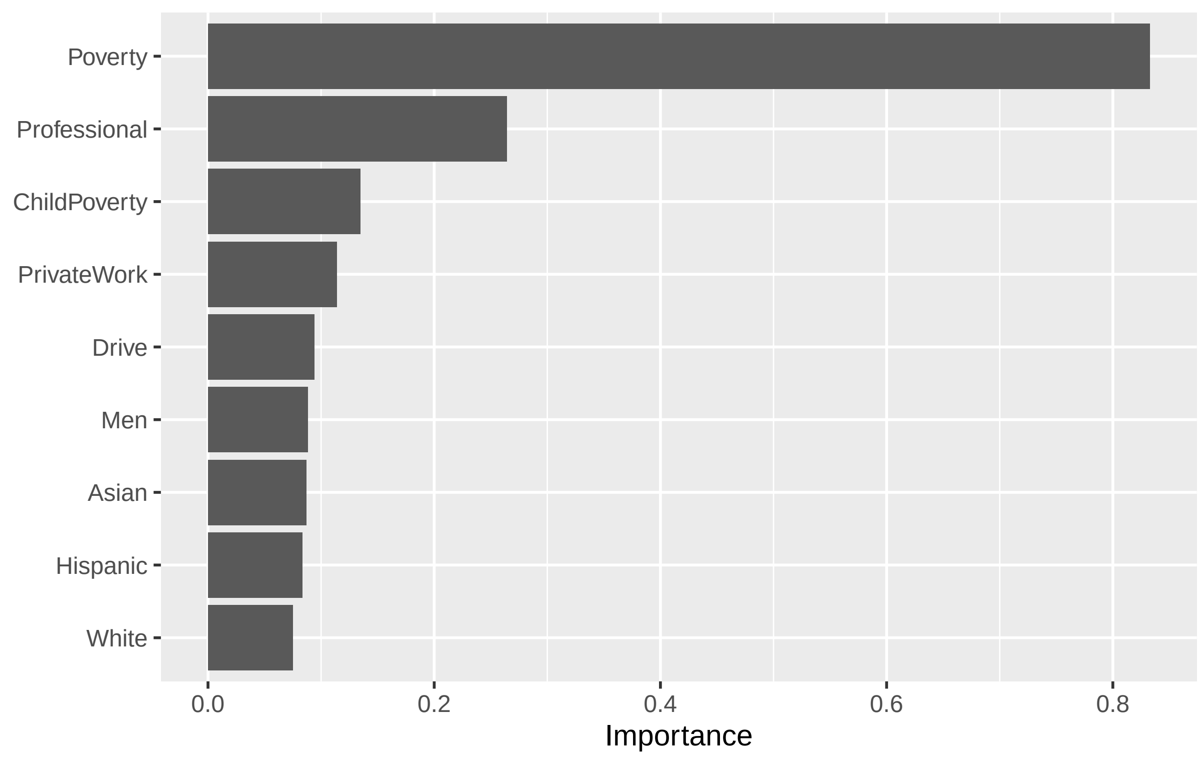
The amount of the penalty can be fine-tuned using a constant called lambda. Selecting a good value for lambda is critical.For the purpose of selecting an optimal lambda value, 10 fold cross-validation is performed on the training data.

The figure above plots the cross validation MSE values as a function of lambda. We see that for the training set, when lambda= 0.01, the MSE is the lowest. The optimal value of the lambda is then used to fit a final ridge model.



Regression Coefficients





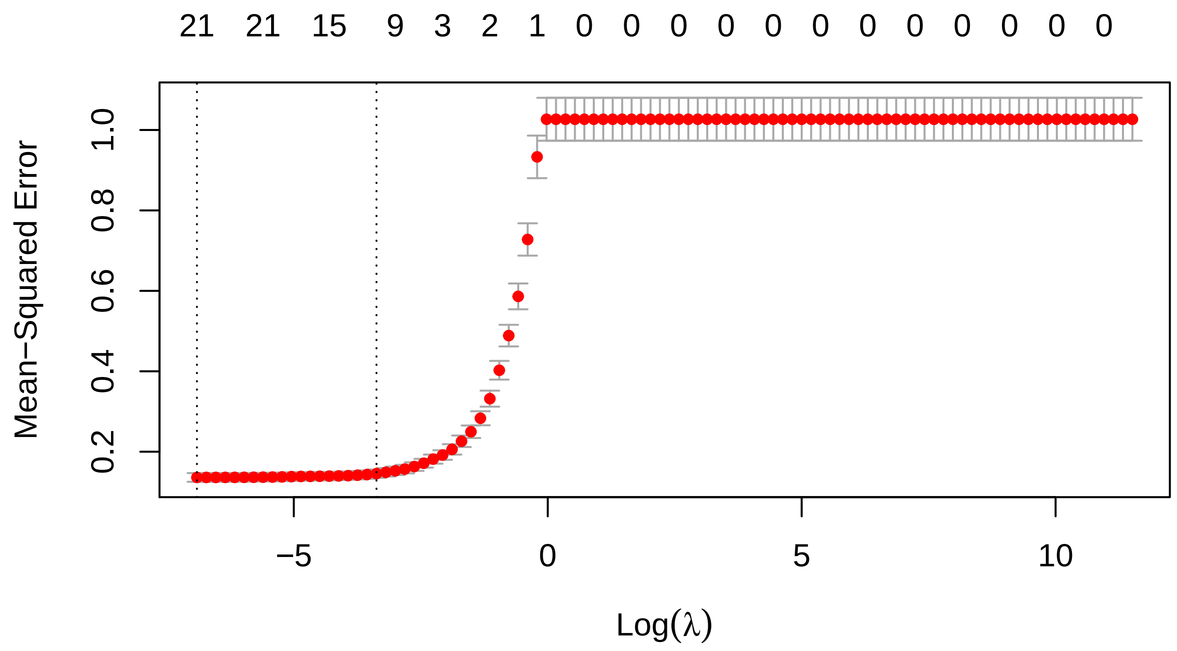
From the first plot above, we see that most coefficient values eventually converge to zero as the value of lambda increases.

In the coefficients plot, when we fit the final model with the optimal lambda, we see that it biases most values toward zero. However, child poverty and professional variables persist and have larger magnitude even after the L2 penalty is imposed. This suggests that those variables have a relatively large influence on per-capita income compared to other variables. The results from the variable importance plot are also in close agreement with the inferences drawn from the regression coefficients plot.

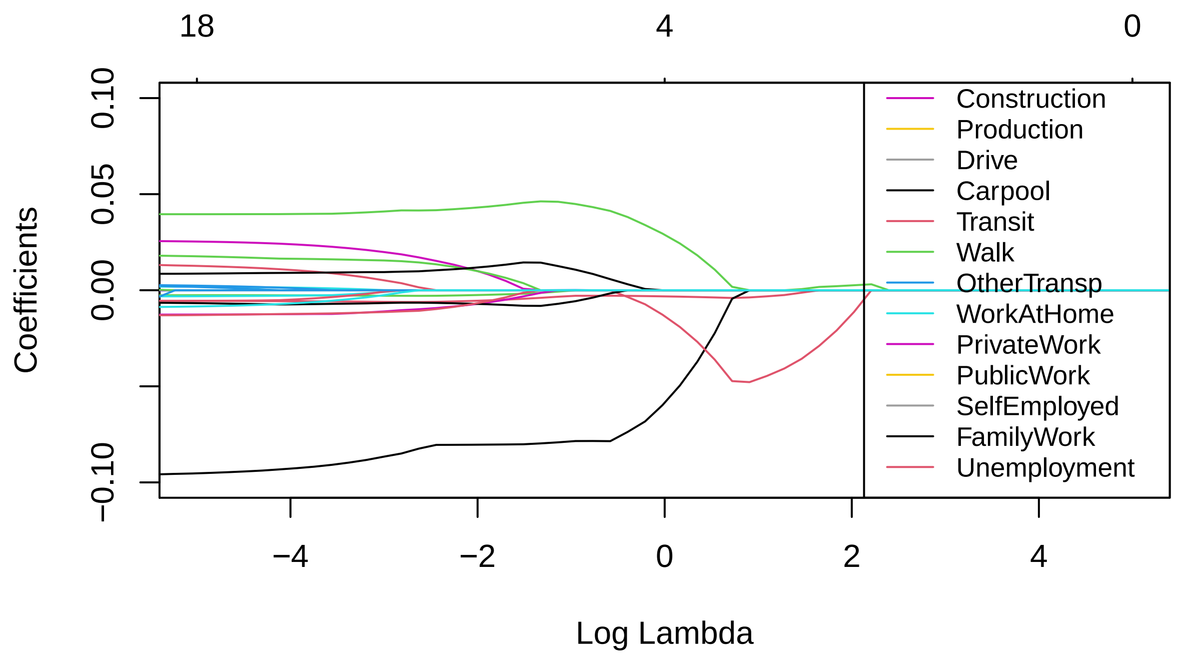
When the trained model is applied to the testing data, the R square drops from 87.6% to 86.08% and the RMSE increases from 0.355 to 0.365. Overall, the testing error is quite low and R square is high, explaining 86% of the variability in present in the data.

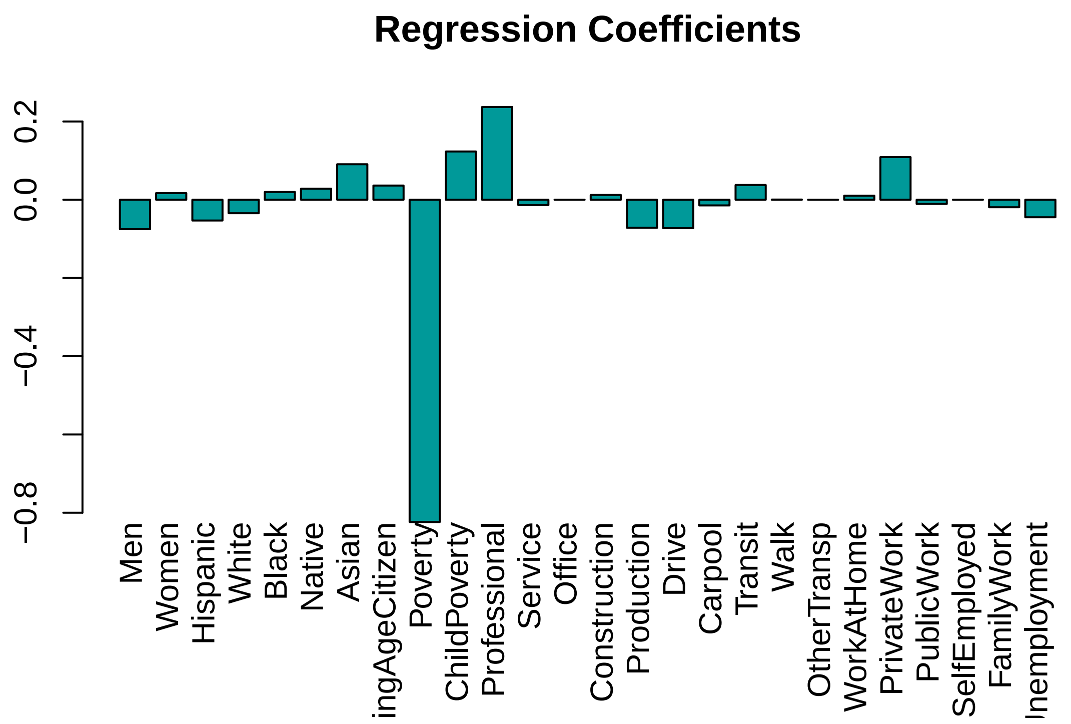
Lasso Regression

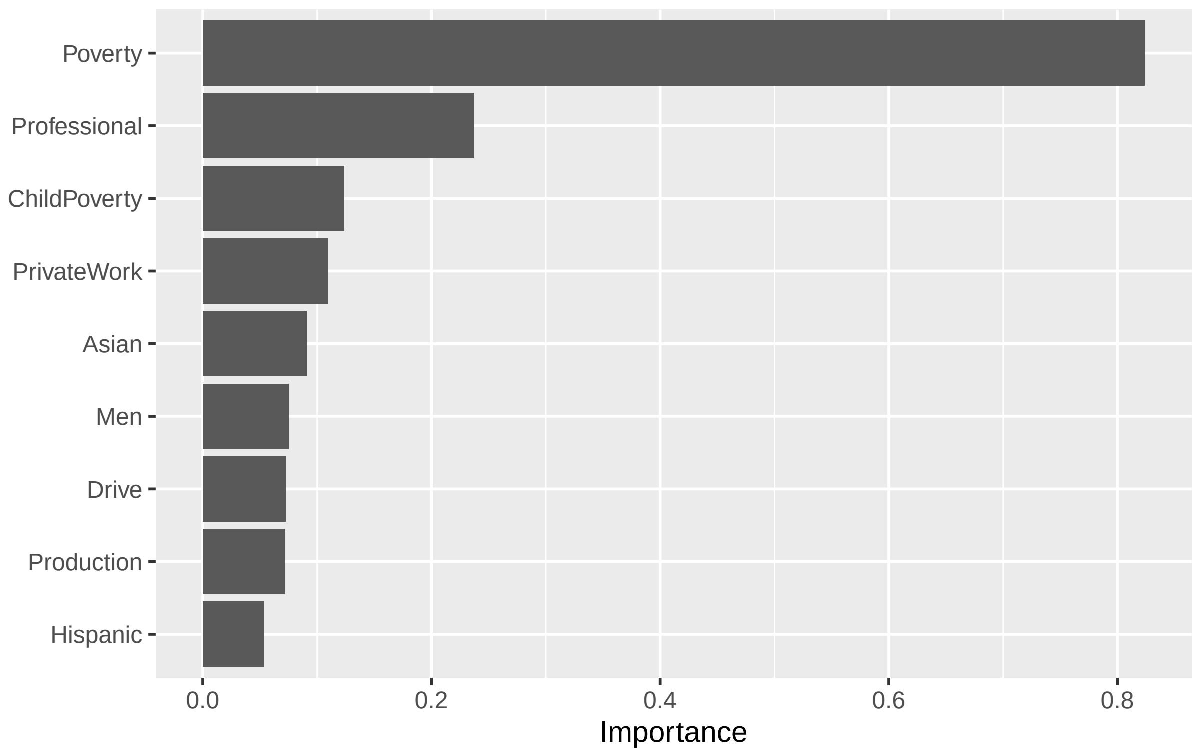
Lasso which stands for Least Absolute Shrinkage and Selection Operator is fit on our data. It shrinks the regression coefficients toward zero by penalizing the regression model with a penalty term called L1-norm, which is the sum of the absolute coefficients.



The plot above shows different values of MSE for increasing values of lambda. After applying 10 fold cross validation on lasso models with all possible lambda values, we find that when lambda is 0.001, the MSE is at it's minimum. This optimal value of lambda is then used to fit a final lasso model on our training data.







In the case of lasso regression, the penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero. After fitting the lasso model on the training data with the optimal lambda value, we find that the following coefficients become exactly zero:

- Office

- Other transport
- Self-employed

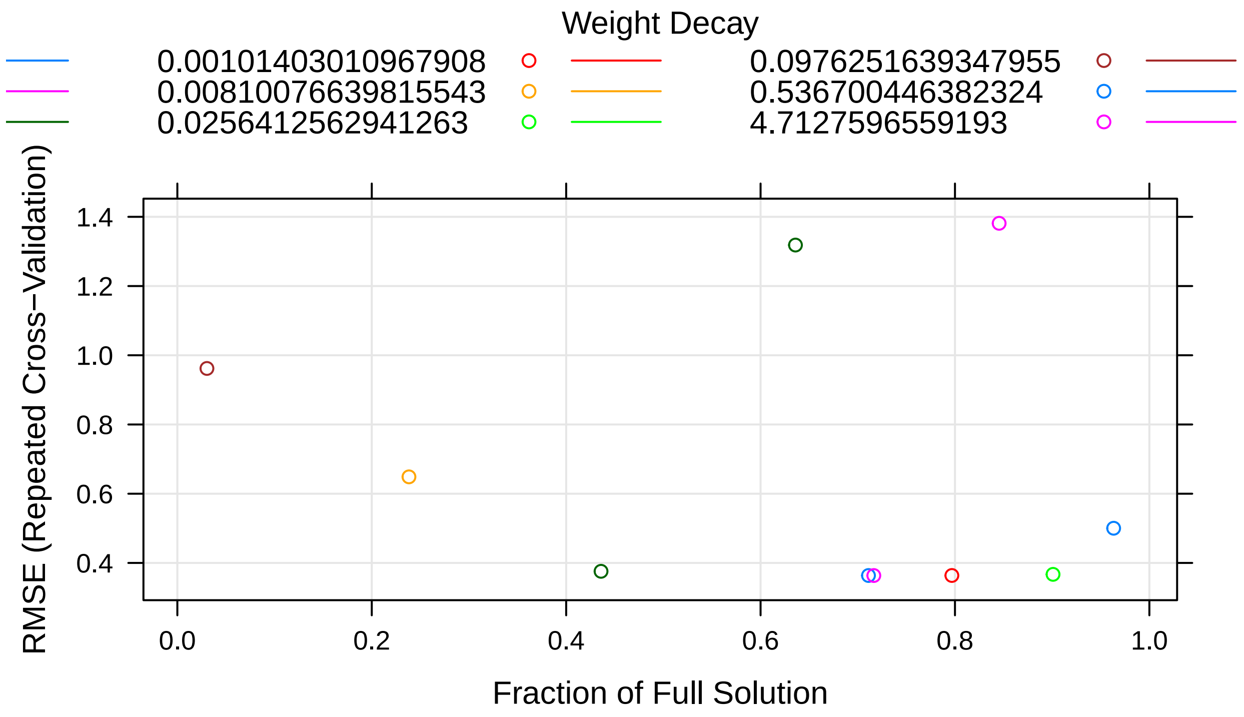
This could indicate that the above mentioned variables may be insignificant in predicting the per-capita income of a county and can be dropped to train a simpler model.

From the coefficient bar chart and the variable importance plot it is seen that variables including Poverty, Professional, Private work, Men and Native continue to have large coefficients despite the imposition of L1 penalty. This implies that these coefficients may be used to construct a simpler model using fewer features to predict per capita income.

When the trained model is used on the testing data, the R square declines from 87.64% to 86.09%, and the RMSE increases from 0.35 to 0.365.

Elastic Net

Elastic net model combines the penalties of ridge regression and lasso to get the best model fit. Here, alpha is the mixing parameter between ridge (alpha=0) and lasso (alpha=1)



The elastic net model was trained using repeated 10-fold cross validation to find the optimum alpha and lambda values. Alpha value is thus 0.7165254 and lambda value is 0.00003196047. RMSE was used to select the optimal model using the smallest value.

When the trained model is used on the testing data, the R square declines from 87.67% to 86.08%, and the RMSE increases from 0.355 to 0.365.

Principle Component Analysis (PCA)

PCA is one of the dimesionality reduction methods. It allows us to keep only the important information for the classification problems with a trade off of simplicity over accuracy. In other words, PCA allows us to reduce the number of variables in a dataset while conserving its information as much as possible.

We now fit the training set to PCA.

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     2.1374 2.0920 1.7070 1.4458 1.33169 1.23763 1.07315
## Proportion of Variance 0.1757 0.1683 0.1121 0.0804 0.06821 0.05891 0.04429
## Cumulative Proportion  0.1757 0.3440 0.4561 0.5365 0.60471 0.66362 0.70792
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     1.03443 0.97877 0.90753 0.84719 0.81983 0.76981 0.74509
## Proportion of Variance 0.04116 0.03685 0.03168 0.02761 0.02585 0.02279 0.02135
## Cumulative Proportion  0.74907 0.78592 0.81760 0.84520 0.87105 0.89384 0.91520
##                           PC15    PC16    PC17    PC18   PC19    PC20    PC21
## Standard deviation     0.70969 0.65405 0.63760 0.57517 0.5073 0.48121 0.21279
## Proportion of Variance 0.01937 0.01645 0.01564 0.01272 0.0099 0.00891 0.00174
## Cumulative Proportion  0.93457 0.95102 0.96666 0.97938 0.9893 0.99819 0.99993
##                           PC22     PC23     PC24     PC25      PC26
## Standard deviation     0.04249 0.006354 0.005879 0.005252 1.635e-15
## Proportion of Variance 0.00007 0.000000 0.000000 0.000000 0.000e+00
## Cumulative Proportion  1.00000 1.000000 1.000000 1.000000 1.000e+00
```

Fitting PCA on our training set tells us that we attein 91.5% of the information by keeping PC1 through PC14. In other words, we can reduce our dimension to 14 from 26, and by selecting only 14 variables of principle components is adequate for constructing a principal component regression.

Since combining PC1 and PC2 contains about 35% of the principle information, we will plot PC1 vs PC2 to see how it captures the information. Since ggplot, which is behind the autoplot function, is not supporting prcomp function, we are inserting the plot image below. (The original code that was used to generate the plot is in the rmd file uploaded separately from the report. The line is hash tagged for the formatting purpose.)

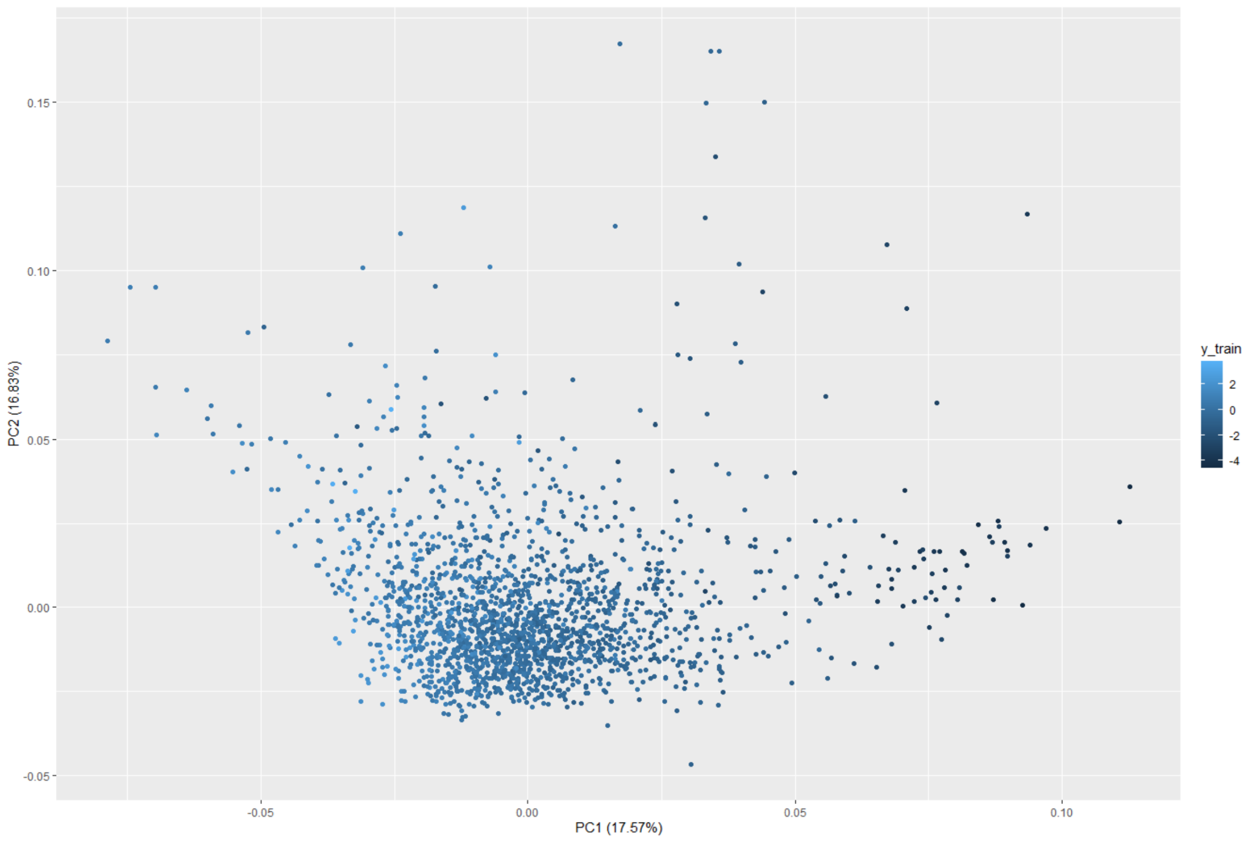


Figure 1: autoplot(pca) result

The plot is not showing us complete information as only PC1 and PC2 captures about 35% of the information. However, it shows a clear separability of the information that is obtained by PC1 and PC2 in light and dark blue. Unfortunately, there is no plotting method that allows us to depict 14 dimensional interaction

Now we will see how much each principle component contains the important information proportionally to the classification model.

```
##  [1] 18 17 11  8  7  6  4  4  4  3  3  3  2  2  2  2  2  1  1  1  0  0  0  0  0
## [26]  0
```

The result above explains that P1 through PC3 contains 46% of the information. However, we need to have 14 principle components in order to have over 90% of the information to our classification model.

Now, we examine the prediction of PCs for validation dataset.

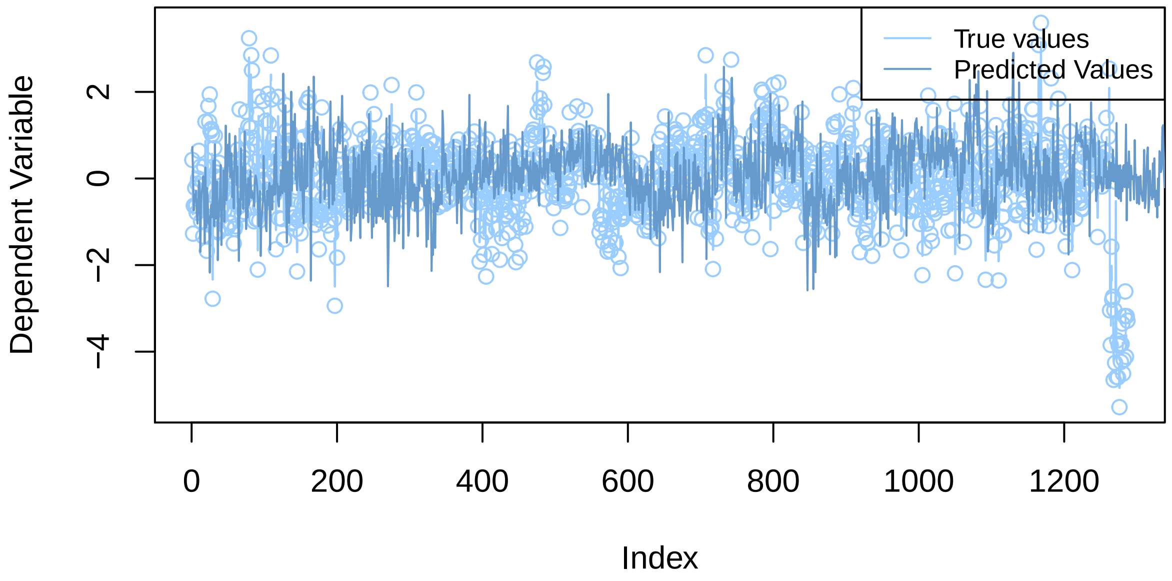
```
##       PC1                PC2                PC3                PC4
##  Min.   :-7.28898   Min.   :-3.20188   Min.   :-5.64200   Min.   :-6.93541
##  1st Qu.:-1.36413   1st Qu.:-1.32356   1st Qu.:-1.06675   1st Qu.:-0.71100
##  Median :-0.31928   Median :-0.49395   Median :-0.33691   Median :-0.09351
##  Mean   :-0.05353   Mean   :-0.02687   Mean   :-0.07137   Mean   :-0.01205  
##  3rd Qu.: 0.82788   3rd Qu.: 0.83931   3rd Qu.: 0.67436   3rd Qu.: 0.56776  
##  Max.   : 9.96231   Max.   :19.87142   Max.   :13.05314   Max.   : 7.09432  
##       PC5                PC6                 PC7                PC8          
##  Min.   :-5.78261   Min.   :-14.98623   Min.   :-6.42696   Min.   :-4.44254  
##  1st Qu.:-0.71908   1st Qu.: -0.63612   1st Qu.:-0.44091   1st Qu.:-0.58573  
##  Median :-0.11441   Median : -0.07730   Median : 0.03417   Median : 0.05541  
##  Mean   : 0.01756   Mean   :  0.01325   Mean   : 0.04625   Mean   :-0.04430  
##  3rd Qu.: 0.62243   3rd Qu.:  0.56680   3rd Qu.: 0.54228   3rd Qu.: 0.54746  
##  Max.   : 6.99627   Max.   :  7.76780   Max.   : 9.05150   Max.   : 7.45768  
##       PC9                 PC10               PC11                PC12           
##  Min.   :-3.745197   Min.   :-7.39017   Min.   :-5.237610   Min.   :-7.465569  
##  1st Qu.:-0.579283   1st Qu.:-0.46854   1st Qu.:-0.418113   1st Qu.:-0.390836  
##  Median : 0.002182   Median : 0.02286   Median : 0.032572   Median :-0.005966  
##  Mean   : 0.011319   Mean   :-0.01894   Mean   :-0.008731   Mean   : 0.007606  
##  3rd Qu.: 0.545835   3rd Qu.: 0.48094   3rd Qu.: 0.386225   3rd Qu.: 0.426116  
##  Max.   : 7.079370   Max.   : 4.84823   Max.   : 9.663029   Max.   : 2.842238  
##       PC13                PC14                PC15               PC16         
##  Min.   :-3.692871   Min.   :-4.743974   Min.   :-6.74317   Min.   :-2.85667  
##  1st Qu.:-0.359435   1st Qu.:-0.374387   1st Qu.:-0.33544   1st Qu.:-0.36337  
##  Median : 0.005585   Median :-0.007806   Median :-0.01230   Median : 0.01254  
##  Mean   :-0.000312   Mean   :-0.008674   Mean   :-0.01757   Mean   : 0.02650  
##  3rd Qu.: 0.355001   3rd Qu.: 0.343697   3rd Qu.: 0.29950   3rd Qu.: 0.38568  
##  Max.   : 3.425873   Max.   : 7.091130   Max.   : 5.08387   Max.   : 2.95621  
##       PC17               PC18                PC19               PC20         
##  Min.   :-2.93307   Min.   :-2.470403   Min.   :-5.09053   Min.   :-2.31233  
##  1st Qu.:-0.31592   1st Qu.:-0.306350   1st Qu.:-0.25358   1st Qu.:-0.25514  
##  Median :-0.05043   Median : 0.002082   Median : 0.01676   Median :-0.00112  
##  Mean   : 0.04119   Mean   : 0.023679   Mean   : 0.02029   Mean   : 0.01414  
##  3rd Qu.: 0.28179   3rd Qu.: 0.343935   3rd Qu.: 0.30636   3rd Qu.: 0.27570  
##  Max.   : 9.25591   Max.   : 3.514670   Max.   : 2.21076   Max.   : 3.17254  
##       PC21                PC22                PC23           
##  Min.   :-1.429672   Min.   :-0.159470   Min.   :-1.988e-02  
##  1st Qu.:-0.120622   1st Qu.:-0.020031   1st Qu.:-1.934e-03  
##  Median :-0.007806   Median :-0.005567   Median :-1.067e-04  
##  Mean   : 0.000970   Mean   : 0.002914   Mean   : 9.959e-05  
##  3rd Qu.: 0.110798   3rd Qu.: 0.012538   3rd Qu.: 2.284e-03  
##  Max.   : 1.743138   Max.   : 0.793376   Max.   : 1.936e-02  
##       PC24                 PC25                 PC26           
##  Min.   :-1.926e-02   Min.   :-1.419e-02   Min.   :-9.416e-15  
##  1st Qu.:-1.404e-03   1st Qu.:-8.858e-04   1st Qu.:-1.069e-15  
##  Median : 2.353e-05   Median :-6.227e-05   Median : 1.435e-16  
##  Mean   :-9.631e-05   Mean   : 1.632e-05   Mean   : 1.492e-17  
##  3rd Qu.: 1.515e-03   3rd Qu.: 7.853e-04   3rd Qu.: 1.332e-15  
##  Max.   : 2.007e-02   Max.   : 1.146e-02   Max.   : 4.291e-15
```

Now, we will fit the training set to the principle regression model, PCR, with PC1 through PC14.

```
## 
## Call:
## lm(formula = y_train ~ pca$x[, 1:14])
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -2.08678 -0.22765  0.00114  0.23040  2.33947
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -0.005468   0.009420  -0.580 0.561679
## pca$x[, 1:14]PC1  -0.366526   0.004408 -83.143  < 2e-16 ***
## pca$x[, 1:14]PC2  -0.052611   0.004504 -11.681  < 2e-16 ***
## pca$x[, 1:14]PC3   0.248748   0.005520  45.063  < 2e-16 ***
## pca$x[, 1:14]PC4   0.055924   0.006517   8.581  < 2e-16 ***
## pca$x[, 1:14]PC5  -0.023813   0.007076  -3.366 0.000779 ***
## pca$x[, 1:14]PC6   0.011689   0.007613   1.535 0.124864
## pca$x[, 1:14]PC7   0.069791   0.008780   7.949 3.19e-15 ***
## pca$x[, 1:14]PC8  -0.079514   0.009109  -8.729  < 2e-16 ***
## pca$x[, 1:14]PC9  -0.153666   0.009627 -15.962  < 2e-16 ***
## pca$x[, 1:14]PC10  0.003448   0.010383   0.332 0.739829
## pca$x[, 1:14]PC11 -0.041004   0.011122  -3.687 0.000233 ***
## pca$x[, 1:14]PC12 -0.093962   0.011493  -8.175 5.31e-16 ***
## pca$x[, 1:14]PC13 -0.022430   0.012240  -1.833 0.067032 .
## pca$x[, 1:14]PC14 -0.015727   0.012646  -1.244 0.213785
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4141 on 1917 degrees of freedom
## Multiple R-squared:  0.8342, Adjusted R-squared:  0.833
## F-statistic: 689.1 on 14 and 1917 DF,  p-value: < 2.2e-16
```

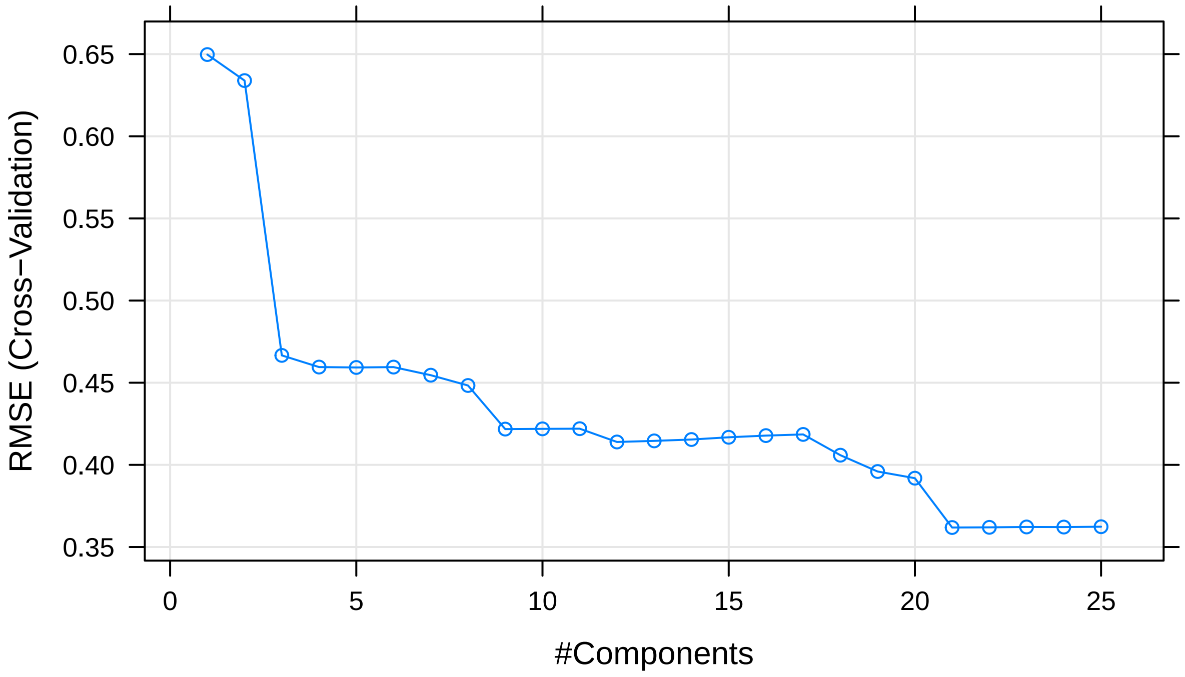
The PCR uses the principle components to be the explanatory variable against the dependent variable. We obtain the R^2 of 0.834. In other words, our model with only 14 dimensions explains about 83% of the data variability of our independent variable, IncomePerCapita. Also, it is noteworthy that PCR typically do not have multicolinearity as PCA avoids the problem of having it.

As our model, pcr, is created based on the training set, we will predict the value of the dependent variable with the validation set to see the model performance. We will then compare our predicted values on testing set to the true values.



The plot above tells that our predicted value is close to the true value. Hence, the model with only 14 key dimensions of information is good enough to construct a model that explains the data variability on our dependent variable.

Next, we will optimize the number of principle components in PCR by minimizing the RMSE.



The above chart shows the the optimum number of the principle components to have in our model is 21 based on the cross-validation on our training set.

We now fit our training set and testing set to the model with 21 principle components.

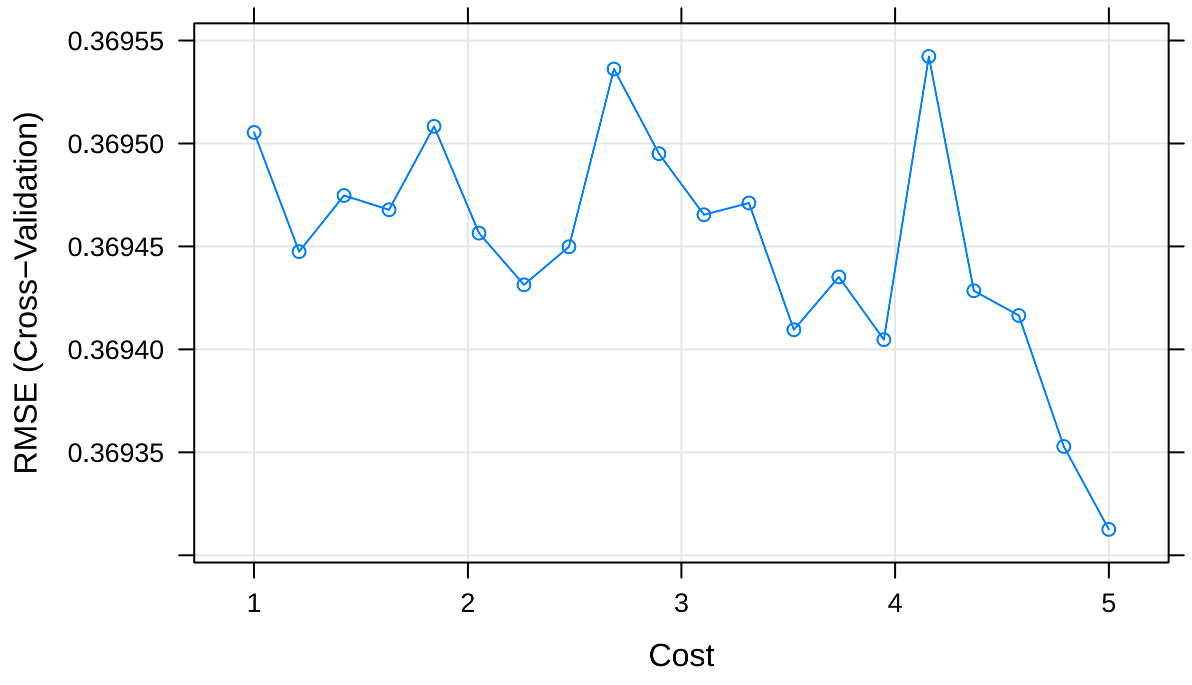
The above three model statistics are coherent with what we expected. Having more principle components in PCR will increase our accuracy, and that will reflect on the R^2. Therefore, the model with 21 principle compoenets yeilds higher R^2. Also, the model with 21 principle componets has lower RMSE than our hand-picked PCR.

Support Vector Regression (SVM)

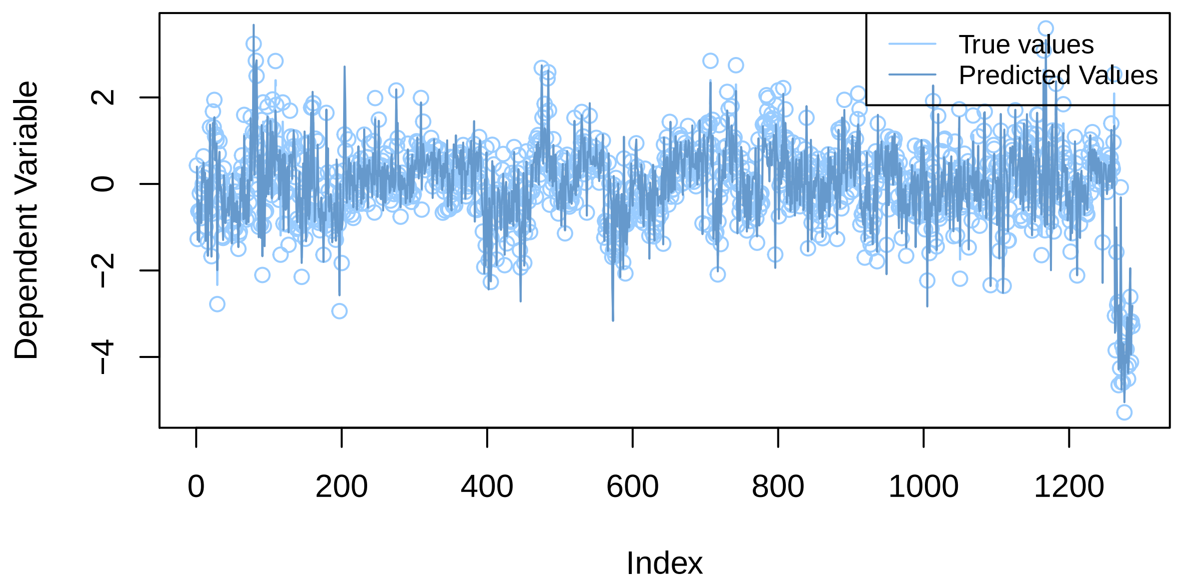
Support vector regression (SVR) is used to find the best fit line, which is a hyperplane that has the maximum number of points. Unlike other regression models that aim to minimize the error between real and predicted value, SVR fits the best fit hyperplane within a threshold value (distance between the hyperplane and boundary line).

For large datasets, like ours, Linear SVR is used since it provides faster implementation than SVR but only considers the linear kernel.

```
## Support Vector Machine object of class "ksvm"
##
## SV type: eps-svr  (regression)
##  parameter : epsilon = 0.1  cost C = 5
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 1391
##
## Objective Function Value : -1628.612
## Training error : 0.125096
```



A linear SVM regression model is fitted above after applying 5-fold cross validation on the training data to identify the optimal parameters. Based on the cross validation results, an epsilon SVR (regression) model is fitted with epsilon of 0.1, cost C=5. The C parameter is the penalty parameter of the error term. For large values of C, the optimization will choose a smaller-margin hyperplane and increase prediction accuracy.



The results from the cross validation are used to train the final model. The final model is then used on the testing data, the R square declines from 87.47% to 86.27%, and the RMSE increases from 0.3584 to 0.3630.

Bias-vairance trade-off

The bias–variance trade-off is the property of a model that the variance of the parameter estimates across samples can be reduced by increasing the bias in the estimated parameters. Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Variance is the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before.

In order to minimize the expected test error, we need to select a statistical learning method that simultaneously achieves low variance and low bias. Regularization tends to decrease the variance at the cost of marginally increases bias in the model. Thus, the regularization models were applied to the ACS dataset. The testing scores and RMSE of the models are reported below:

Model	Testing Score	RMSE
Ridge Regression	0.8609154	0.3653749
Lasso Regression	0.8609468	0.3653336
Elastic Net	0.8608486	0.3654627
Principal Component Analysis	0.8611461	0.3650717
Support Vector Regression	0.8627318	0.3629813
—————-	—————-	—————-

Out of all the models, SVR has the highest testing score and the lowest RMSE making it the best model to regularize our data. This is because the trained model generalizes well to the testing data.

Although Support Vector Regression is used rarely it carries certain advantages. It is robust to outliers, decision model can be easily updated, it has excellent generalization capability, with high prediction accuracy. This is evident from the results above.

IV. Conclusion

In conclusion, After fitting multiple classification models using our insurance premium data, the Multinomial Logistic Regression performed the best overall compared to the other models. We derived this conclusion by training our models by creating a 60-40 training/testing data split to observe the reported accuracies for all the models (except for K-Means, since it is an unsupervised model, thus, an accuracy ratio cannot be derived). However, we recognized that by performing our 60-40 data split, we were introducing a degree of bias into our analysis, thus, to mitigate any bias created, we performed 10-fold cross validations on every model to observe the accuracies for each model. With the Multinomial Logistic Regression deriving the best cross-validated accuracy score, we concluded that the Multinomial Logistic Regression was the best model to use for our data and that a linear model was more appropriate to use for this data specifically. We understand that the data's dependent variable is imbalanced, but regardless, the models were able to derive high accuracies.

V. Future Work

When creating our groupings for insurance premiums, we found that the dependent variable was imbalanced, thus, in terms of future work, we believe that implementing the same analysis on a dataset that contains a dependent multiclass variable that is more balance should be the focus since observing predictors that factor into determining insurance premiums is a topic that is worthy of exploring more. We could have approached to fix the imbalance data by applying SMOTE and other techniques. However, the issues with imbalance data were not covered in this course and other courses yet, so we left it as it is, after consulting with Dr. Rojas, to avoid having further issues by synthetically changing our dataset. However, we acknowledge that fixing the imbalance data would greatly improve our classification models, in particular with LDA and QDA.

VI. Reference

Insurance Premimum Dataset (Classification Data) https://www.kaggle.com/noordeen/insurance-premium-prediction

US Census Demographic Data (Regularization Data) https://www.kaggle.com/muonneutrino/us-census-demographic-data