SAN DIEGO STATE UNIVERSITY

MIS-620 FINAL PROJECT

US Census Data Income Analysis

By,

Shruti Venkatesh Kulkarni
824677549

1. Executive summary

This assessment aims to develop analytical models in order to solve the business problem. The problem is detailed as trying to predict if a person earns an income that is greater than 50K dollars when certain demographic, socio-economic data is known.

The document is segmented into 3 major sections. The initial section details the exploratory data analysis performed on the dataset that consisted of 48840 observations over 15 variables. This section also includes visualizations that show predictors as a function of another. Visualizations detailed in this section provide insight into variables, demonstrate variation in values for some and contrast the overall rate against one another.

During exploration, predictors like occupation, workclass and native_country had an unknow level '?', this had to be replaced by NA and later imputed using knn impute. Predictors capital_gain, capital_loss and native_country were found to have high frequency of zero values or near zero variance. As a result, these predictors and the other non-representative levels from native_country were dropped to avoid undue influence on the models. Additionally, predictors fnlwgt, which is a method of census data collection and education which deduces information from educational_num predictor were dropped from the dataset. This section also explains the creation of training and test dataset to validate model's results and measure its performance. Methods like SMOTE were applied to correct the data imbalance that was observed.

The second section outlines the process of building predictive models. Decision Trees, Random Forest, Logistic Regression, Support vector machines, Naïve Bayes and Neural Network methods were selected as good candidates to compare and contrast for this classification problem. A 10-fold cross validation method was incorporated for all models to get a better estimate of test error, avoid over-fitting and tune parameters using customized grid search. Model performance is outlined and key metrics that are relevant to our business case such as sensitivity (TPR), ROC/AUC and accuracy are compared across models to determine best performing model.

The next section deals with predicting the dependent variable on the test dataset. All of the models that were developed are used to predict the income on the same test dataset. Specificity and Accuracy loss due to having the training dataset balanced is described and found to be an acceptable tradeoff that helped in increasing the sensitivity of the predictions.

Model	**Sensitivity**	**Accuracy**	**AUC**
Decision Tree	0.485	0.830	0.834
Logistic Regression	0.534	0.836	0.885
Random Forest	0.583	0.842	0.877
Naive Bayes	0.527	0.828	0.874
SVM	0.471	0.832	0.862
Neural Network	0.548	0.836	0.889

Table 1 – Test performance metrics for original dataset

Model	**Sensitivity**	**Accuracy**	**AUC**
Decision Tree	0.640	0.810	0.813
Logistic Regression	0.805	0.801	0.885
Random Forest	0.698	0.804	0.865
Naive Bayes	0.928	0.697	0.874
SVM	0.811	0.799	0.869
Neural Network	0.825	0.806	0.893

Table 2 – Test performance metrics for SMOTE dataset

Table 1 and Table 2 shows the test performance for the original dataset and the SMOTE dataset respectively. Random Forest performed best on the original dataset with high sensitivity, AUC and Accuracy, followed by Neural Network. These models indicate that they predict the minority class better and the true values more accurately.

On the SMOTE dataset, we chose Neural Network as the best model since it helped balance the trade-off by having a high Sensitivity, Accuracy and AUC, followed by Logistic Regression. Although Naïve Bayes had highest sensitivity value, the model had significant loss in accuracy indicating we can accurately predict a person's true income only 70% of the time, and thus cannot be accepted as a best performing model. It was also observed that years of formal education, marital_status (married with a spouse) and age were the most informative predictors. Overall, the analytical models were able to satisfactorily answer the business question within the limits on computational resources and available predictors.

2. Discovery and Data Preparation

2.1 Business case for selecting data

Federal, state and business use case:

The income analysis helps in identifying the areas and regions which need housing assistance and rehabilitation loans, housing subsidies, allocation of funds for Federal educational programs such as vocational and education, identifying accurate assessment of economic well-being for different regional populations, community development, Medicare, women and child welfare and generating employment to list a few. On the business use case side, the income analysis helps in product development, forecasting the supply and chain, identifying newer market locations by studying the communities, targeted marketing and recommendations for different groups and ethnicities, setup factories and manufacturing units that provides employment to nearby communities.

This dataset is the US Census 1994 data which contains information related to socio-economic and demographic data such as ethnicity, age educational level, marital status and others that help in predicting whether a person earns >50K or <=50K. The income currency is in US Dollars.

Data link: http://archive.ics.uci.edu/ml/datasets/Census+Income

We form our hypothesis and success criteria as follows:

Null Hypothesis: Income of an individual is not related or influenced by the demographic or socio-economic information.

Alternative Hypothesis: There is a relationship between the predictors and the income earned and influence of the predictors in earning an income >50K

Success criteria: Using this dataset, we aim to predict if a person earns >50K or <=50K and also determine which factors which lead to an earning of >50K.

2.2 Count and column explanations

This dataset contains information related to the US census data; there are a total of 48840 rows and 15 columns. The target variable is income, and the other 14 columns are predictors.

Table 3 outlines the structure of the dataset:

VARIABLE NAME	DATATYPE	DESCRIPTION
AGE	numeric	Age of the adult
WORKCLASS	categorical with 9 levels	Type of employment such as government, private
FNLWGT	numeric	Information about the data collection method
EDUCATION	categorical with 16 levels	Highest education level achieved such as master's, bachelor's
EDUCATIONAL-NUM	numeric (number of years of education)	Number of years of education (related to education)
MARITAL-STATUS	categorical with 7 levels	Current marital status of the adult
OCCUPATION	categorical with 7 levels	Occupation of the adult, such as sales, admin-clerical, manager
RELATIONSHIP	categorical with 6 levels	Relationship of the person (adult) living in the house to the "householder".

		Householder is the person who rents, owns or is going to buy the place.
RACE	categorical with 5 levels	Ethnicity of the adult
SEX	categorical with 2 levels	Gender of the adult
CAPITAL.GAIN	numeric	The gains or proceeding derived from the sale of property, a derived value
CAPITAL.LOSS	numeric	The loss derived from the sale of property, a derived value
HOURS.PER.WEEK	numeric	The total number of working hours
NATIVE.COUNTRY	categorical with 42 levels	Native country of the adult
INCOME	Categorical with 2 levels	<=50K and >50K

Table 3 – Variable definitions in the dataset

Prior to visualizing, the dataset had some potential issues:

1) The file had spaces across all cells which had to be removed to be read as a dataframe
2) The income variable had four levels <=50K., <=50K, >50K and >50K, two levels had an extra '.' which had to be replaced so that income has just two levels <=50K & >50K

2.3 Data visualization and inferences

Fig 1 shows the distribution of workclass in the dataset. About 70% of the people work in the Private sector. There are some levels in this category which has fewer representation. '?' exists as a category, this could be due to missing data.

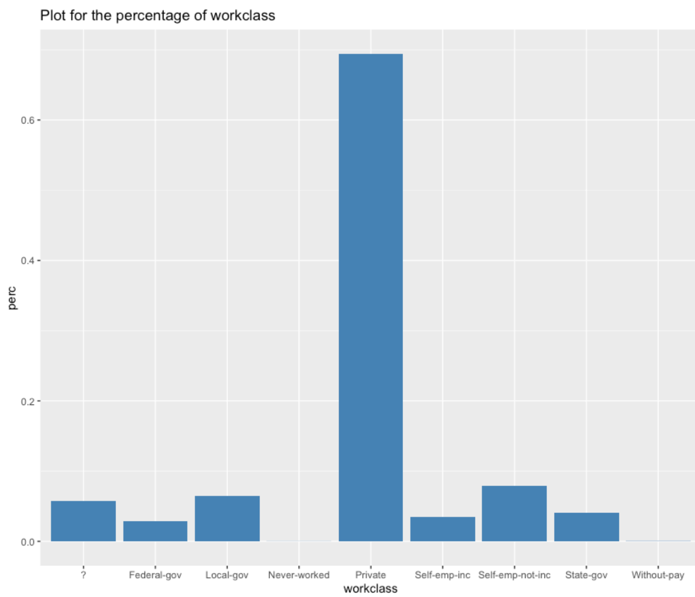
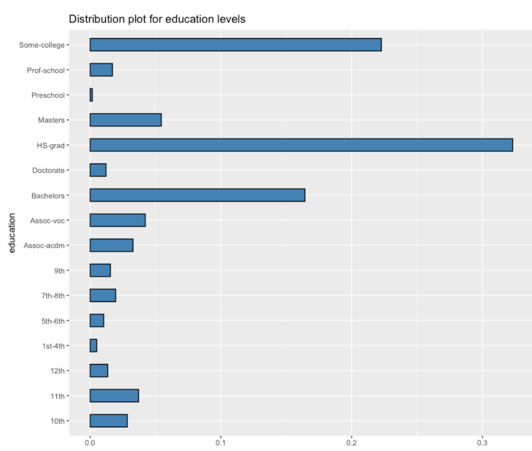


Fig 1 – Percentage of workclass distribution

Fig 2 shows the distribution of education levels observed in the dataset.



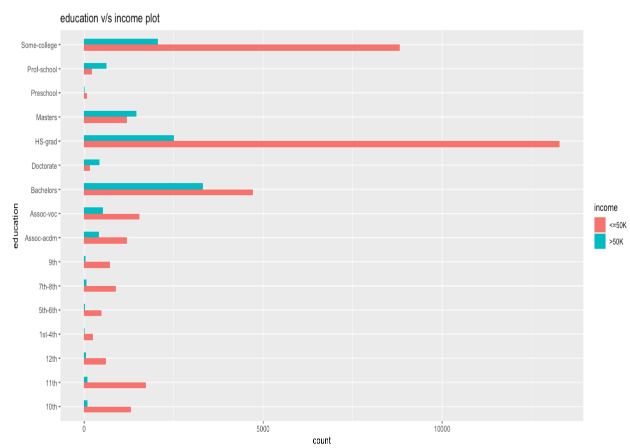
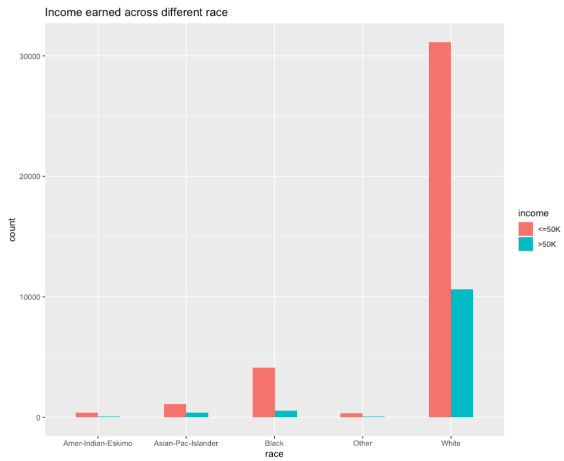


Fig 2 – Distribution of education levels

Fig 3 – Income distribution vs education levels

Fig 3 shows the distribution of income among the various education levels. From fig 3, we observe that the majority population with a master's degree, prof-school or a doctorate earned >50K, and also that those having lower education earned <=50K. Around 33% of the observations were HS-grad, this was followed by Some-college category. From Fig 4, we see the distribution of income vs race and it can be observed that dataset is skewed towards the White population.



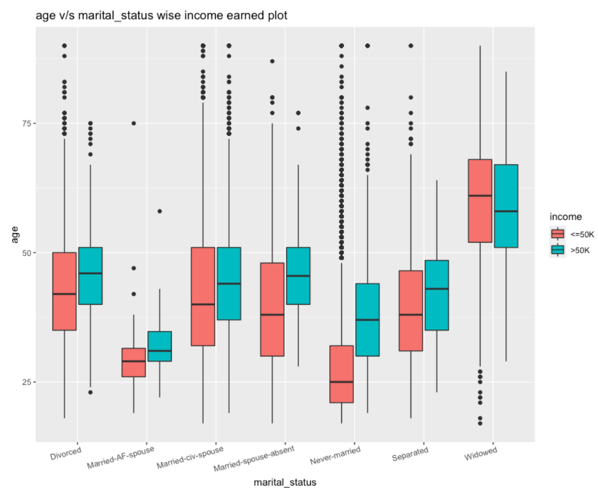


Fig 4 – Income distribution vs race

Fig 5 – Income vs marital status and age

Fig 5 shows the income data based on the marital status and age. Number of people earning >50K is observed to be higher among never-married and in their late 30's. Those who were widowed typically earned <=50K and were of age-group 60. People who earned >50K and widowed were around 52-57 of age.

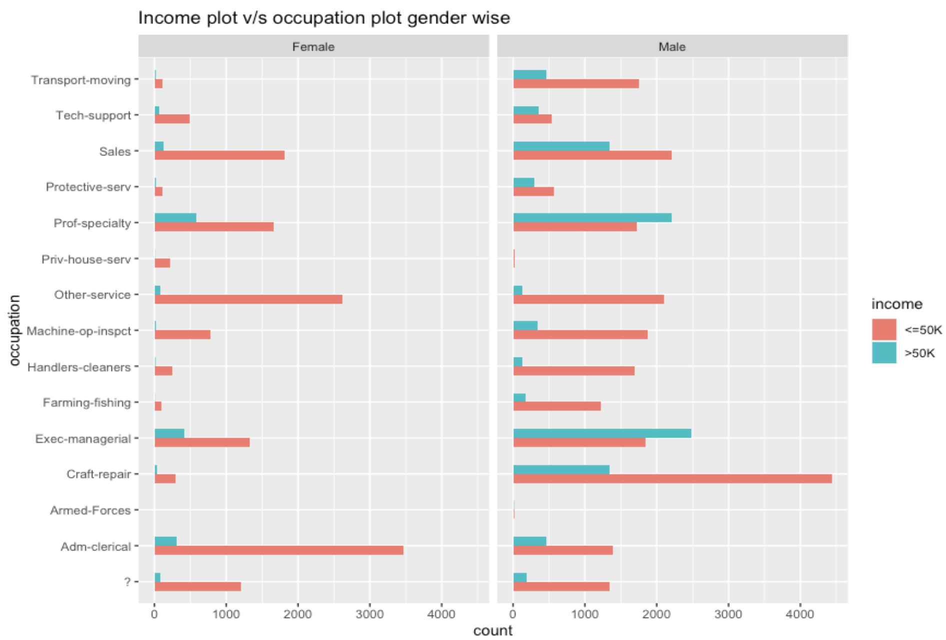
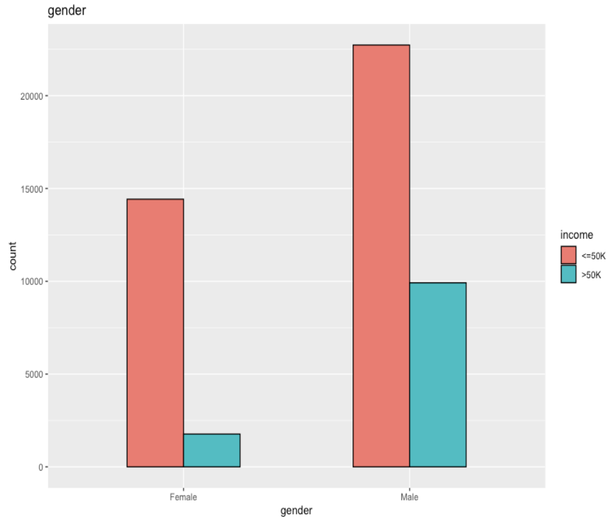


Fig 6 – Income vs occupation and gender

Fig 6 shows the income by gender distribution and occupation. Males were highest in craft-repair and exec-managerial categories, and females were part of adm-clerical followed by other services. There are more males who held exec-managerial position and earned >50K. It is also important to note that '?' exists as a category of occupation. Similarly, Fig 7 shows the overall gender distribution of the dataset with males having a higher representation and higher percentage of them earning >50K.



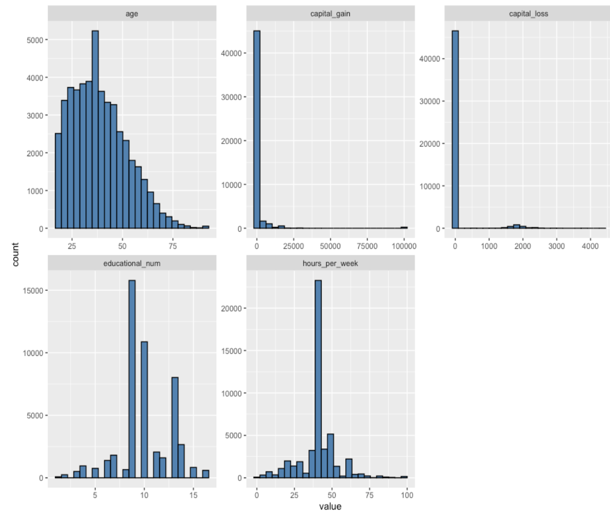
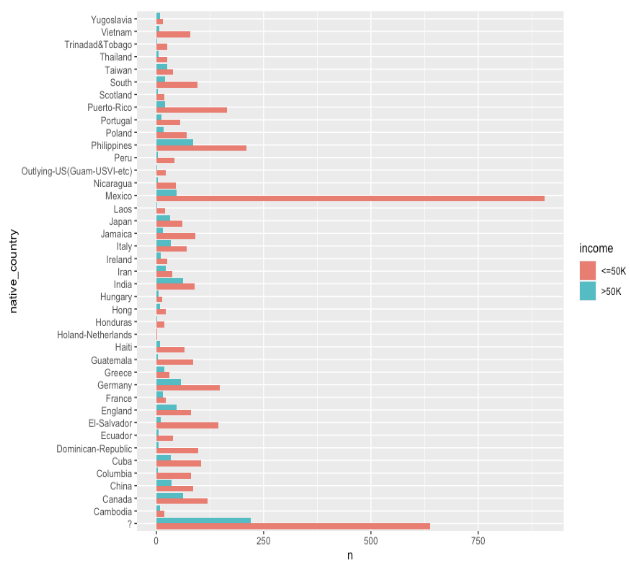


Fig 7 – Income vs gender

Fig 8 –numerical predictors distribution

Fig 8 shows the distribution of all the numerical predictors observed in the dataset.



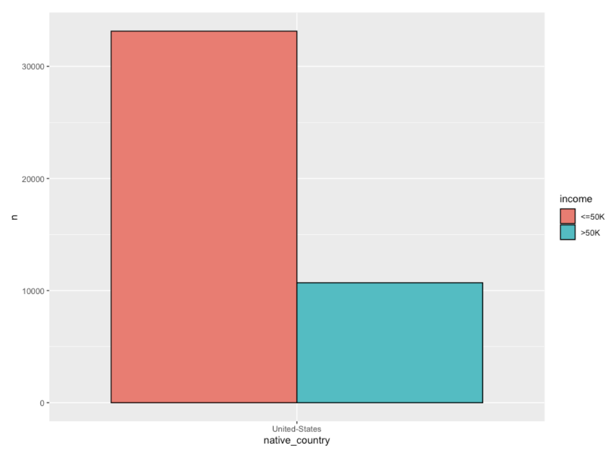


Fig – 9 Income distribution by native country

Fig 10 – Native country - US income distribution

Fig 9 and Fig 10 visualizes the income distribution by native country (non-US) and US respectively. Mexico had the highest number of people earning <=50K. Taiwanese, French and Indian natives typically had a higher percentage of earners earning >50K.

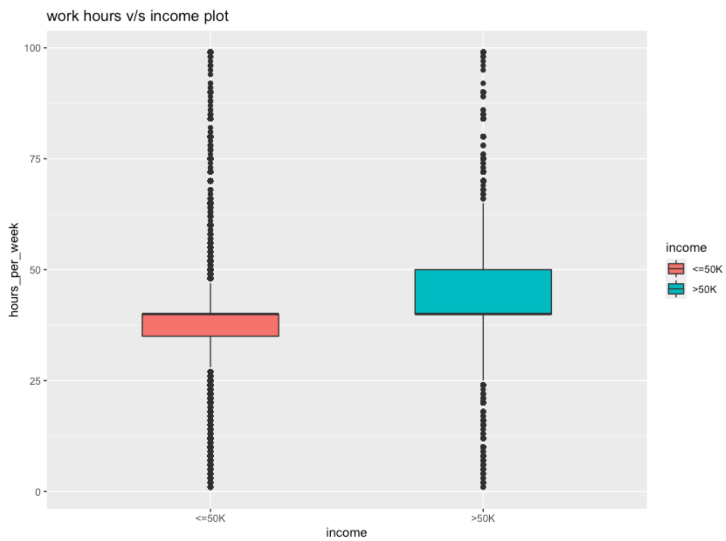


Fig 11 – Work hours vs income

Fig 11 shows the income distribution by number of hours worked. People who earned >50K, it appears that they typically worked more than 40 hours per week.

2.4 Data preparation

2.4.1 Understanding the variables

Fig 12 demonstrates predictors capital_gain, capital_loss have many 0 values (extremely low variances) and thus does not provide much information.

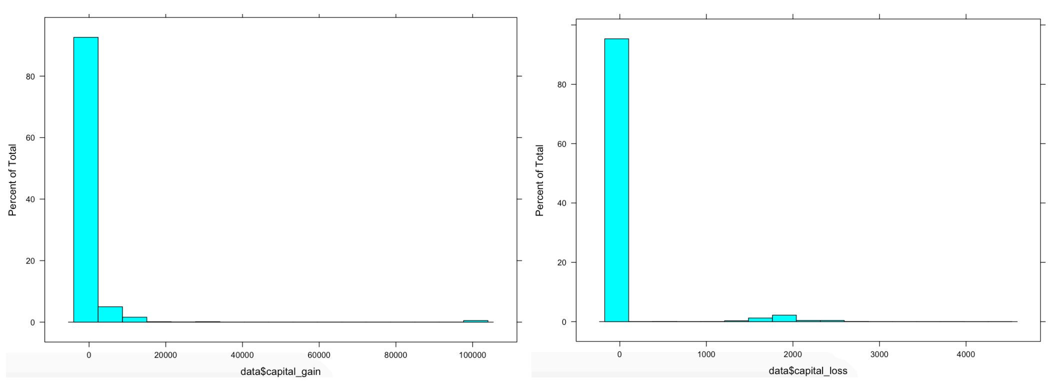


Fig 12 – capital_gain and capital_loss value distribution

Similarly, native_country has levels whose count vary significantly. The predictor with United States as a level alone has around 43830 observations and other countries like Honduras have far fewer observations. These three variables could become near zero variables after split in cross-validation and cause issues during modeling and analysis.

From the visualizations, it was observed that few predictors like work occupation, workclass and native_country had an unknow level '?', this had to be replaced by NA. Additionally, this level '?' had to be dropped after replacement since they now had no values, and caused the dataset to have missing values. Variables occupation and workclass have about 6% missing data and native_country has about 2% missing data, although the overall percentage of missing data is about 7.5% and we decided to impute the values, as seen in the Fig 13 the nature of missing values.

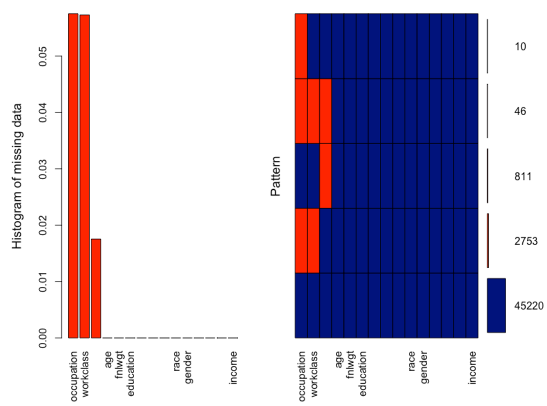


Fig 13 – Missing values observed in the dataset

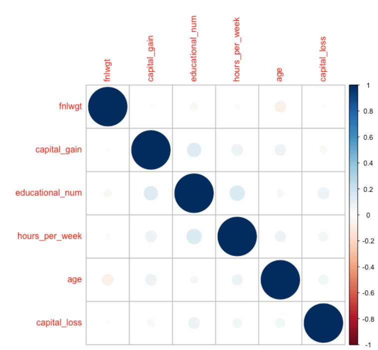


Fig 14 – Corrplot for numerical variables

The dataset didn't have any correlated features as suggested by Fig 14. It was observed that variables educational_num and education deduce similar information about the education levels achieved by people and having such similar predictors can cause issues. As a result, education was removed prior to modelling.

Finally, the predictor fnlwgt is a way of census data collection and it is not useful for our analysis and hence it was removed from the dataset. Since we are interested in understanding who earns >50K, we set the minority class as positive for our analysis after relabeling >50K and <=50K to gt50K and lte50K respectively.

2.4.2 Data Preprocessing

After splitting the dependent variable from the predictors, we dummy coded our predictors prior to preprocessing. Preprocessing was performed to scale and center the data in order to represent all the data in similar units. Imputation was performed on missing values using knnImpute and variables capital_gain and capital_loss were removed by setting method as nzv. Similarly, levels in native_country which had near zero variance was also removed.

2.4.3 Test/Train split and Cross Validation:

The original dataset was split into a training and a testing set using a 70 – 30 ratio. Training data consisted of a 70% of the original observations on which we used cross validation to tune the algorithm and the test set consisted of the other 30%. We use 10-fold cross validation for resampling in order to optimize the parameters and gain robust performance from the folds without overfitting.

2.4.4 Data imbalance:

Our data set is imbalanced, the target variable income consists of 37,153 observations for <=50K, that is, around 76% and 11,687 observations for >50K, around 24%. This imbalance will impact our model’s performance since it won’t be able to predict well on the >50K class and could be biased towards <=50K.

To handle the class imbalance, we use the hybrid SMOTE (Synthetic Minority Oversampling Technique) on the training data to increase the minority class. After SMOTE, we now have 16,362 as >50K records and 20,452 as <=50K records. The hybrid SMOTE technique was applied only to the training dataset and the test dataset retained the original imbalance.

3. Model planning and building

3.1 Models used

This is a binary classification problem; we are using models such as Logistic Regression, Decision tree, Naïve bayes, Random Forest, Neural Network and SVM with Radial Basis kernel function. All the models were trained using caret’s train, we also use the custom trainControl to help provide a finer control over how caret searches for models. We set the parameters method as cv and the number to 10 to indicate that we are using 10 fold cross validation, we also set the summaryFunction parameter to indicate compute performance metrics other than default ones and we set the classProbs to compute the class probabilities for AUC calculation. In this section we have summarized the tuning parameters of the models on the original as well as imbalanced dataset.

<u>Logistic Regression</u>

Since this is a binary classification problem and we are trying to predict whether a person makes >50K or <=50K, we use logistic regression since it is a low variance and high bias model, 10-fold cross validation was used.

Decision Tree

In our hypothesis, we are also interested in identifying the predictors that are important in predicting the >50K using information gain concept, decision tree helps in visualizing these predictors and typically uses only a single predictor in constructing the tree. The cp tuning parameter was set on both training datasets using 10-fold CV. The final Cp on Imbalanced data was 0.00459 and on SMOTE data final cp was 0.01287.

Random Forest

Using bagged trees, multiple decision trees are fit and these decision trees in the forest are implemented using different predictors so that no subset of tree includes the same predictors to achieve more accurate model on this imbalanced dataset. In order to identify the tuneGrid mtry hyperparameter range, an initial search was performed with tuneLength of 7 on both datasets using 10-fold CV, final mtry applied on imbalanced data was 5 & on SMOTE data mtry was 12.

Naïve Bayes

Naïve Bayes is a probabilistic classifier that makes a bold and naïve assumption that all predictors are independent of each other. 10-fold cross-validation was used for tuning hyperparameters fL, adjust on the training dataset. Final parameters for imbalanced data were fLwas 0 & adjust was 1. On SMOTE final fL was 0 & adjust was 1.

Support Vector machines

Support vector machine is a flexible classification model for two class prediction which allows flexibility by softening the margin and thus generalize the model, SVM with radial basis function kernel method was used for implementation. In order to identify the tuneGrid parameters sigma and cost parameter (C) an initial search was performed with tuneLength 7 using 10-fold cross-validation on both the datasets and the final model was set using the range for sigma and cost parameter (C) using tuneGrid. On the original dataset the final sigma was 0.04 & final C was 0.5. On SMOTE data final sigma was 0.04 and final C was 0.5.

Neural Networks

For our classification, neural network was implemented using ’nnet’. In order to identify the tuneGrid parameters, size (number of neurons) in the single hidden layer and decay(weight), an initial search was performed with tuneLength of 7 using 10-fold CV for both the datasets. The data was preprocessed to allow all the values to be in the range of 0 and 1. The final parameters applied on the imbalanced data were size was 4 and decay was 1. On SMOTE data, the final size was 5 and decay was 0.1.

3.2 Metrics used

Our business case is to predict whether a person earns >50K or <=50 K income on an imbalanced dataset, the following metrics were used across all the models and compared:

Accuracy: The percentage of correctly classified instances of majority class <=50K and minority class >50K out of all instances.

Sensitivity: True positive rate to understand how well the positive class >50K was predicted

ROC: plots the Sensitivity (true positive rate) and the Sensitivity (1-FPR)

AUC: This single score can be calculated using the area under the ROC curve.

4. Model performance

4.1 Training data performance

The following section describes the model performance using the 4 models. Fig 15 and Fig 16 below compare the models using the ROC metric. The models trained with the balanced dataset performed better compared to the ones that were trained on the original dataset.

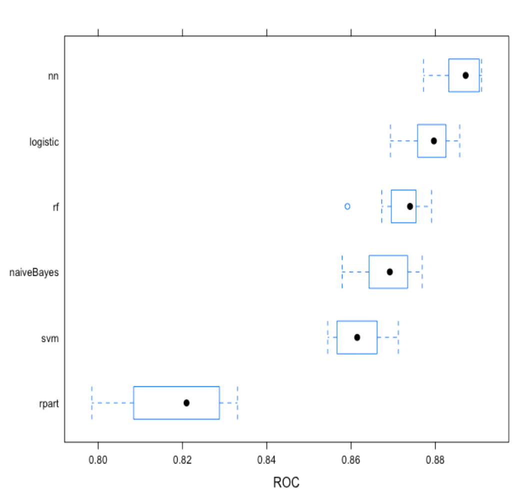


Fig 15 - ROC plot imbalanced data

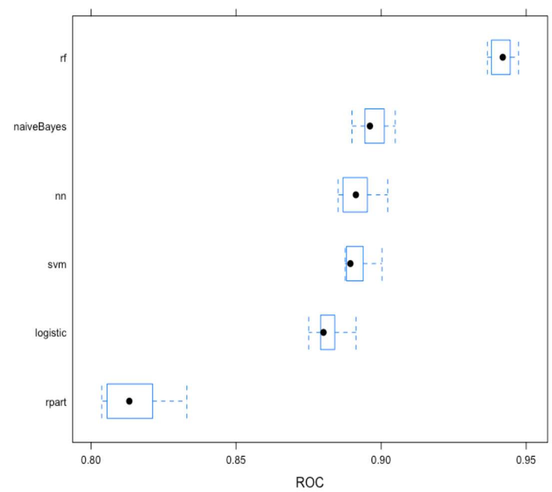


Fig 16 - ROC plot SMOTE data

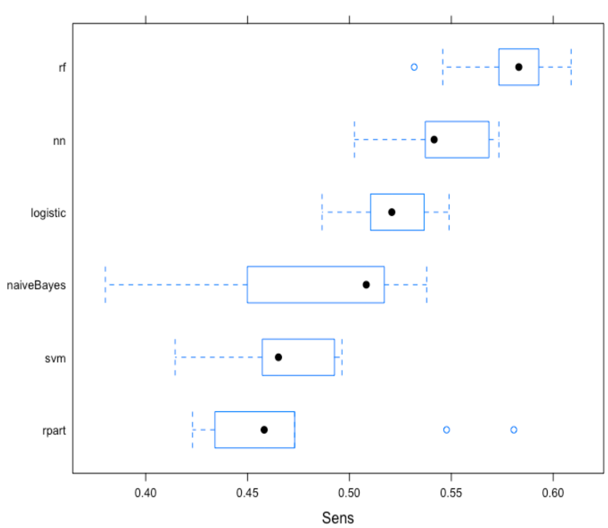


Fig 17 – Sensitivity plot on imbalanced data

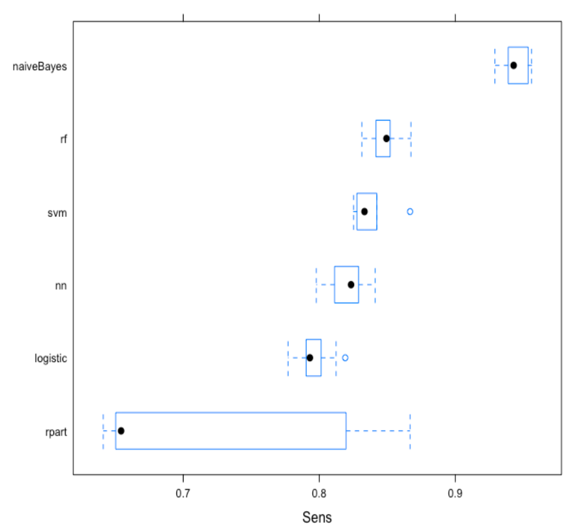


Fig 18 – Sensitivity plot on SMOTE data

Fig 17 and Fig 18 show the Sensitivity values for all the 6 models used on original and SMOTE dataset. It is evident that the balanced training set was much more effective in training the models to increase sensitivity.

4.2 Test data performance

Logistic Regression:

	Actual		
Prediction		gt50K	lte50K
	gt50K	1871	766
	lte50K	1635	10379

Table 4 - Test data (original) Confusion Matrix

Accuracy = 0.8361 Sensitivity = 0.5337

	Actual		
Prediction		gt50K	lte50K
	gt50K	2823	2230
	lte50K	683	8915

Table 5 – Test data confusion matrix (SMOTE)

Accuracy = 0.8012 Sensitivity = 0.8052

Table 4 and Table 5 describe the confusion matrices for Logistic regression. Sensitivity greatly increased on the models trained with SMOTE data whereas accuracy saw a decline. The performance shows this simple model is a good fit for this dataset.

Decision Tree:

	Actual		
Prediction		gt50K	lte50K
	gt50K	1700	689
	lte50K	1806	10456

Table 6 - Test dataset (original) Confusion Matrix

Accuracy = 0.8297 Sensitivity = 0.4849

	Actual		
Prediction		gt50K	lte50K
	gt50K	2244	1515
	lte50K	1262	9630

Table 7 – Test dataset SMOTE confusion matrix

Accuracy = 0.8105 Sensitivity = 0.6400

Table 6 and Table 7 describes the confusion matrices for the Decision Tree model. Sensitivity did have a considerable increase on model trained with SMOTE data, but the accuracy saw a negligible decrease. Fig. 19 displays the decision tree plot used to determine income using the information gain concept.

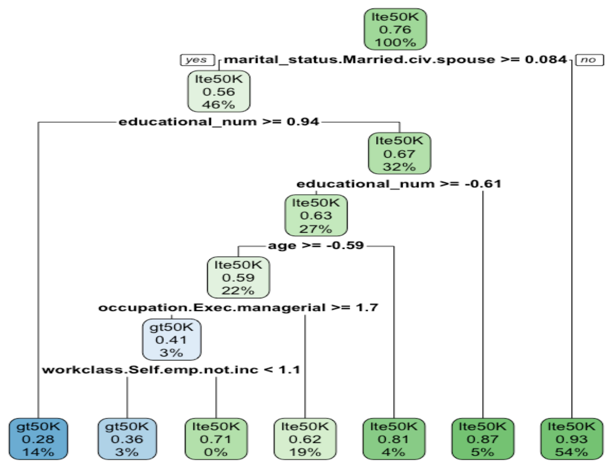


Fig 19 – Decision Tree plot

Naïve Bayes:

		Actual	
Prediction		gt50K	lte50K
	gt50K	1847	868
	lte50K	1659	10277

Table 8 - Test dataset (original) Confusion Matrix

Accuracy = 0.8275 Sensitivity = 0.5268

		Actual	
Prediction		gt50K	lte50K
	gt50K	3252	4189
	lte50K	254	6956

Table 9 – Test dataset SMOTE confusion matrix

Accuracy = 0.6967 Sensitivity = 0.9276

Table 8 and Table 9 describe the confusion matrices for Naïve Bayes model. Sensitivity had the highest gain among any of the other models but caused the accuracy to significantly drop when using the SMOTE dataset. The SMOTE data confusion matrix for Naïve Bayes had the highest number of false positives compared to other models indicating bad test performance.

SVM:

		Actual	
Prediction		gt50K	lte50K
	gt50K	1653	602
	lte50K	1853	10543

Table 10 - Test dataset (original) Confusion Matrix

Accuracy = 0.8324 Sensitivity = 0.4715

		Actual	
Prediction		gt50K	lte50K
	gt50K	2845	2285
	lte50K	661	8860

Table 11 – Test dataset SMOTE confusion matrix

Accuracy = 0.7989 Sensitivity = 0.811

Table 10 and Table 11 describe the confusion matrices for SVM. SVM also had Sensitivity greatly increase with a minor reduction in accuracy when using the SMOTE dataset.

Random Forest:

		Actual	
Prediction		gt50K	lte50K
	gt50K	2045	851
	lte50K	1461	10294

Table 12 - Test dataset (original) Confusion Matrix

Accuracy = 0.8422 Sensitivity = 0.5833

		Actual	
Prediction		gt50K	lte50K
	gt50K	2446	1808
	lte50K	1060	9337

Table 13 – Test dataset SMOTE confusion matrix

Accuracy = 0.8042 Sensitivity = 0.6977

Table 12 and Table 13 describe the confusion matrices for the Random Forest model. Sensitivity had a modest gain but also resulted the accuracy to noticeably drop when using the SMOTE dataset. Figure 20 displays educational_num, marital_status, age and hours_per_week as four topmost predictors.

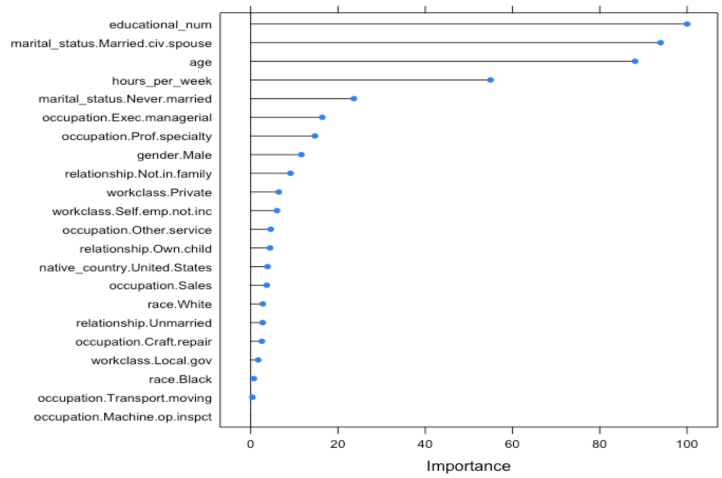


Fig 20: Random Forest Attribute importance

Neural Network:

	Actual	
Prediction	gt50K	lte50K
gt50K	1920	811
lte50K	1586	10334

Table 14 - Test dataset (original) Confusion Matrix

Accuracy = 0.8364 Sensitivity = 0.5476

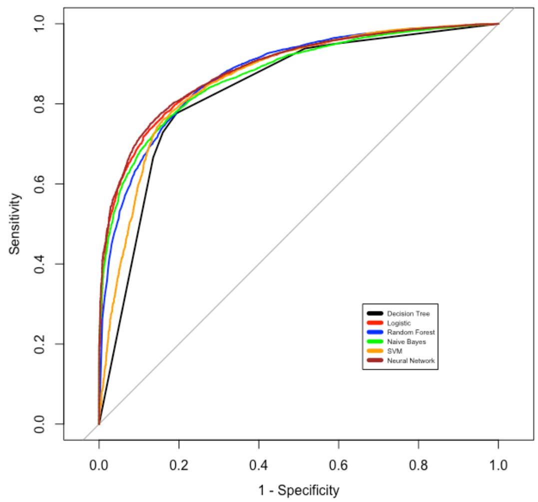
	Actual	
Prediction	gt50K	lte50K
gt50K	2892	2230
lte50K	614	8915

Table 15 – Test dataset SMOTE confusion matrix

Accuracy = 0.8059 Sensitivity = 0.8249

Table 14 and Table 15 describe the confusion matrices for Neural Network model. Neural Network also had Sensitivity greatly increase with a minor decrease in accuracy when using the SMOTE dataset.

Fig 21 and Fig 22 show the ROC curves for all the models using the original dataset vs using the SMOTE dataset respectively.



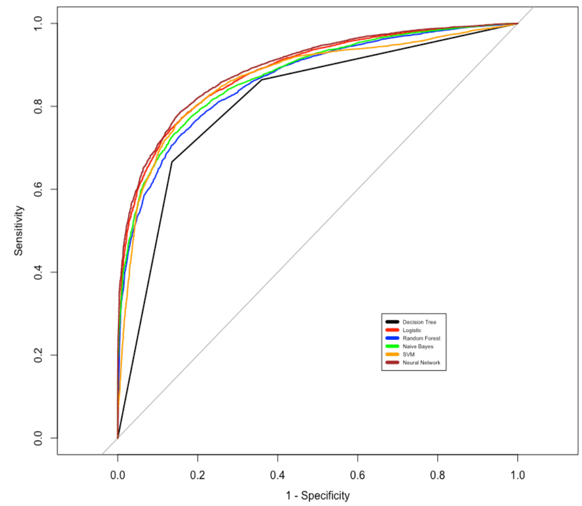


Fig 21 – ROC curve for predictions (original dataset)   Fig 22 – ROC curve for predictions (SMOTE)

Table 16 and Table 17 summarizes the key test performance metrics on original and SMOTE dataset respectively.

Model	**Sensitivity**	**Specificity**	**Accuracy**	**AUC**
Decision Tree	0.485	0.938	0.830	0.834
Logistic Regression	0.534	0.931	0.836	0.885
Random Forest	0.583	0.924	0.842	0.877
Naive Bayes	0.527	0.922	0.828	0.874
SVM	0.471	0.946	0.832	0.862
Neural Network	0.548	0.927	0.836	0.889

Table 16 – Test performance metrics for original dataset

Model	**Sensitivity**	**Specificity**	**Accuracy**	**AUC**
Decision Tree	0.640	0.864	0.810	0.813
Logistic Regression	0.805	0.800	0.801	0.885
Random Forest	0.698	0.838	0.804	0.865
Naive Bayes	0.928	0.624	0.697	0.874
SVM	0.811	0.795	0.799	0.869
Neural Network	0.825	0.800	0.806	0.893

Table 17 – Test performance metrics for SMOTE dataset

5. Conclusion

5.1 Discussion and recommendations

Based on the testing performance, random forest, neural network models and logistic regression models fared better on the original dataset showcasing high sensitivity, high AUC and high accuracy indicating they predict the true classes better than the other models. On the models tested for SMOTE dataset fit, it was observed that Neural Network and Logistic Regression models were the best performing since they had high sensitivity, accuracy and AUC values. Also, although Naïve Bayes model had the highest sensitivity metric, the model's accuracy was the lowest, indicating that it could predict true income values accurately only 69% of the times. It was found that the precision of this model was also extremely low due to the increased false positive (Type I) errors, thus this model was not considered for best performance.

Recommendations are made towards utilizing a better tune length for few models given we had better computational capabilities. The dataset had fewer observations of people from non-US native countries and people who earned >50K, collecting information related to these predictor levels might help in higher performance. The analytical models satisfy our success criteria and it was found that predictors Education, Marital status married with a spouse and age were the most influential. Typically, people who had higher education and were married with a spouse earned >50K, these findings supported the alternate hypothesis.

`

Appendix

1. R code

```
library(caret)
library(stringr)
install.packages("dplyr")
library(dplyr)
library(mice)
library(VIM)
library(tidyverse)
library(doParallel)
library(scales)
library(corrplot)
library(car)
library(purrr)
library(tidyr)
library(ggplot2)
install.packages("klaR")
install.packages("promises")
install.packages("fastmap")
library(promises)
library(klaR)
library(fastmap)
install.packages('rpart.plot')
library(rpart.plot)
library(plyr)
install.packages("pROC")
library(pROC)

unloadNamespace("VIM")

setwd("~/ISLR_RCode/620_Project/NZV_Try")
column_names = c('age', 'workclass', 'fnlwgt', 'education',
            'educational_num','marital_status', 'occupation', 'relationship',
            'race', 'gender','capital_gain', 'capital_loss', 'hours_per_week',
            'native_country','income')

datafile1 <- read.csv("adult_train.csv", header=T, sep=",", col.names = column_names,
strip.white = TRUE)
datafile2 <- read.csv("adult_test.csv", header=T, sep=",", col.names = column_names,
strip.white = TRUE)

#?read.csv()
dim(datafile1)
dim(datafile2)
```

```
datafile <- rbind(datafile1,datafile2)
dim(datafile)

write.csv(datafile,"adult_income_data.csv")

data = as.data.frame(read.csv('adult_income_data.csv'), na.strings = "?")

data$X <- NULL

summary(data)
str(data)


#one of the dataset has <=50K. and >50K. as levels
table(data$income)
# <=50K <=50K.   >50K  >50K.
# 24719  12434   7841   3846

#replacing the . in the income column with nothing and changing it back to a factor
data$income = str_replace_all(data$income, "[[.]]", "")
data$income = as.factor(data$income)
table(data$income)
#Aafter
# <=50K  >50K
# 37153 11687

summary(data)
# > summary(data)
# age                  workclass         fnlwgt              education
# Min.   :17.00   Private         :33905   Min.   :  12285   HS-grad     :15784
# 1st Qu.:28.00   Self-emp-not-inc: 3862   1st Qu.: 117554   Some-college:10878
# Median :37.00   Local-gov       : 3136   Median : 178144   Bachelors   : 8024
# Mean   :38.64   ?               : 2799   Mean   : 189666   Masters     : 2657
# 3rd Qu.:48.00   State-gov       : 1980   3rd Qu.: 237647   Assoc-voc   : 2061
# Max.   :90.00   Self-emp-inc    : 1695   Max.   :1490400   11th        : 1811
#                 (Other)         : 1463                     (Other)     : 7625
# educational_num              marital_status           occupation
# Min.   : 1.00   Divorced             : 6633   Prof-specialty : 6172
# 1st Qu.: 9.00   Married-AF-spouse    :   37   Craft-repair   : 6112
# Median :10.00   Married-civ-spouse   :22379   Exec-managerial: 6086
# Mean   :10.08   Married-spouse-absent:  628   Adm-clerical   : 5610
# 3rd Qu.:12.00   Never-married        :16115   Sales          : 5504
# Max.   :16.00   Separated            : 1530   Other-service  : 4923
# Widowed              : 1518   (Other)        :14433
# relationship                  race               gender       capital_gain
```

```
# Husband       :19716   Amer-Indian-Eskimo:  470   Female:16192   Min.   :    0
# Not-in-family :12582   Asian-Pac-Islander: 1519   Male  :32648   1st Qu.:    0
# Other-relative: 1506   Black             : 4684                  Median :    0
# Own-child     : 7580   Other             :  406                  Mean   : 1079
# Unmarried     : 5125   White             :41761                  3rd Qu.:    0
# Wife          : 2331                                             Max.   :99999
#
# capital_loss     hours_per_week       native_country    income
# Min.   :   0.00   Min.   : 1.00   United-States:43830   <=50K:37153
# 1st Qu.:   0.00   1st Qu.:40.00   Mexico       :  951   >50K :11687
# Median :   0.00   Median :40.00   ?            :  857
# Mean   :  87.51   Mean   :40.42   Philippines  :  295
# 3rd Qu.:   0.00   3rd Qu.:45.00   Germany      :  206
# Max.   :4356.00   Max.   :99.00   Puerto-Rico  :  184
#                                   (Other)      : 2517


######Plots:
#barchart of workclass:
ggplot(data, aes(x=workclass))+
  geom_bar(width = 0.5) +
  ggtitle('Plot for the percentage of workclass')
#? exists as a category, missing data


#percentage of people avcross different workclass
data %>%
  dplyr::count(workclass) %>%
  dplyr::mutate(perc = n / nrow(data)) -> workclass_perc
ggplot(workclass_perc, aes(x = workclass, y = perc)) + geom_bar(stat = "identity",
fill="steelblue")+
  ggtitle('Plot for the percentage of workclass')
#about 70% of the people were from "Private" sector and ? is an unknow level to be handled

#barchart of race:
data %>%
  dplyr::count(race) %>%
  dplyr::mutate(perc = n / nrow(data)) -> race_perc
ggplot(aes(race_perc, n, fill = income)) + geom_bar(position = "dodge")+
  ggtitle('Plot for the percentage of race')

#race income earned plot
ggplot(data, aes(x=race, fill=income))+
  geom_bar(width = 0.5, position = "dodge") +
  ggtitle('Income earned across different race')
```

```
#education plot
data %>%
  dplyr::count(education) %>%
  dplyr::mutate(perc = n / nrow(data)) -> education_perc
ggplot(education_perc, aes(x=education, y =perc))+
  geom_bar(width = 0.5, stat="identity", fill="steelblue", colour="black") +
  ggtitle('Distribution plot for education levels') +coord_flip()
#most people were from HS-grad (around 33%) category followed by Some-college category
#education_num plot

#marital_status v/s age over income plot
ggplot(data = data) + geom_boxplot(aes(y = age, x = marital_status, fill = income))+
  xlab("marital_status") + ylab("age") + ggtitle('age v/s marital_status wise income earned plot')+
  theme(legend.position = "right", axis.text.x.bottom = element_text(angle=15,hjust=0.9))
#most people who earned >50K were never-married and were in their late 30's
#also, most people who were widowed and earned <=50K were of age-group 60,
#the poeple who earned >50K and widwed were around 52-57


#occupation v/s income plot for each gender
ggplot(data, aes(x=occupation, fill=income))+
  geom_bar(width = 0.5, position = "dodge") +
  ggtitle('Income plot v/s occupation plot')+
  coord_flip() + facet_grid(~gender)
#? exists as a category,most males were into craft-repair category and exec-mangerioal
#most females belonged to adm-clerical category, followed by other services
#There are more males who held exec-managerial position and earned >50K


#relationship plot genderwise
ggplot(data, aes(x=relationship))+
  geom_bar(width = 0.5, fill="steelblue", colour="black") +
  ggtitle('relationship')

#race plot
ggplot(data, aes(x=race, fill=income))+
  geom_bar(width = 0.5, colour="black", position="dodge") +
  ggtitle('income earned w.r.t each race')


#gender plot
ggplot(data, aes(x=gender, fill=income))+
  geom_bar(width = 0.5, position = "dodge") +
  ggtitle('gender')
#more males than females
```

```
#native_country overall plot
ggplot(data, aes(x=native_country, fill=income))+
  geom_bar(width = 0.5, position = "dodge")+
  ggtitle('native_country v/s income plot')+
  coord_flip()

#plotting income data for all countries except US_and_Territories to understand earning of other nationalities
data %>% filter(!native_country == "United-States") %>%dplyr::group_by(income, native_country) %>%
  dplyr::summarise(n = n()) %>% dplyr::arrange(desc(n)) %>% ggplot(aes(native_country, n, fill = income)) +
  geom_bar(stat = "identity", position = "dodge") +
  coord_flip()


data %>% filter(native_country == "United-States") %>%dplyr::group_by(income, native_country) %>%
  dplyr::summarise(n = n()) %>% dplyr::arrange(desc(n)) %>% ggplot(aes(native_country, n, fill = income)) +
  geom_bar(stat = "identity", position = "dodge")


histogram(data$capital_gain)
histogram(data$capital_loss)

#numeric data arepresentation
data %>%
  keep(is.numeric) %>%
  gather() %>%
  ggplot(aes(value)) +
  facet_wrap(~ key, scales = "free") +
  geom_histogram(bins = 25, fill="steelblue", colour="black")



#replacing ? with NA in the dataset
data = na_if(data, "?")

#dropping all  ? categories after conevrting them to NA
summary(data$workclass)
data$workclass <- droplevels(data$workclass)

summary(data$occupation)
data$occupation <- droplevels(data$occupation)
```

```
summary(data$native_country)
data$native_country <- droplevels(data$native_country)

summary(data)
# age              workclass          fnlwgt             education       educational_num
# Min.   :17.00   Private          :33905   Min.   :  12285   HS-grad     :15784   Min.   : 1.00
# 1st Qu.:28.00   Self-emp-not-inc: 3862   1st Qu.: 117554   Some-college:10878   1st Qu.: 9.00
# Median :37.00   Local-gov       : 3136   Median : 178144   Bachelors   : 8024   Median :10.00
# Mean   :38.64   State-gov       : 1980   Mean   : 189666   Masters     : 2657   Mean   :10.08
# 3rd Qu.:48.00   Self-emp-inc    : 1695   3rd Qu.: 237647   Assoc-voc   : 2061   3rd Qu.:12.00
# Max.   :90.00   (Other)         : 1463   Max.   :1490400   11th        : 1811   Max.   :16.00
#                 NA's            : 2799                     (Other)     : 7625
#              marital_status            occupation            relationship
#  Divorced             : 6633   Prof-specialty : 6172   Husband       :19716
#  Married-AF-spouse    :   37   Craft-repair   : 6112   Not-in-family :12582
#  Married-civ-spouse   :22379   Exec-managerial: 6086   Other-relative: 1506
#  Married-spouse-absent:  628   Adm-clerical   : 5610   Own-child     : 7580
#  Never-married        :16115   Sales          : 5504   Unmarried     : 5125
#  Separated            : 1530   (Other)        :16547   Wife          : 2331
#  Widowed              : 1518   NA's           : 2809
# race                      gender        capital_gain    capital_loss     hours_per_week
# Amer-Indian-Eskimo:  470   Female:16192   Min.   :    0   Min.   :   0.00   Min.   : 1.00
# Asian-Pac-Islander: 1519   Male  :32648   1st Qu.:    0   1st Qu.:   0.00   1st Qu.:40.00
# Black             : 4684                  Median :    0   Median :   0.00   Median :40.00
# Other             :  406                  Mean   : 1079   Mean   :  87.51   Mean   :40.42
# White             :41761                  3rd Qu.:    0   3rd Qu.:   0.00   3rd Qu.:45.00
#                                           Max.   :99999   Max.   :4356.00   Max.   :99.00
#
# native_country        income
# United-States:43830   <=50K:37153
# Mexico       :  951   >50K :11687
# Philippines  :  295
# Germany      :  206
# Puerto-Rico  :  184
# (Other)      : 2517
# NA's         :  857


#Checking for missing values
data[complete.cases(data),]
data[!complete.cases(data),]
sapply(data, function(x) sum(is.na(x)))
# age      workclass       education educational_num  marital_status      occupation
# 0           2799              0               0               0           2809
# relationship  race          gender    capital_gain    capital_loss  hours_per_week
```

```
# 0              0              0              0              0              0
# native_country    income
# 857              0

#missing data analysis using VIM
library(VIM)
md.pattern(data)
#OCCUPATION, WORKCLASS AND NATIVE_COUNTRY HAVE  missing values
# Variables sorted by number of missings:
# Variable      Count
# occupation 0.05751433
# workclass 0.05730958
# native_country 0.01754709
# age            0.00000000
# fnlwgt 0.00000000
# education 0.00000000
# educational_num 0.00000000
# marital_status 0.00000000
# relationship 0.00000000
# race 0.00000000
# gender 0.00000000
# capital_gain 0.00000000
# capital_loss 0.00000000
# hours_per_week 0.00000000
# income 0.00000000
aggr_plot <- aggr(data, col=c('navyblue','red'), numbers=TRUE, prop=c(TRUE, FALSE),
                  sortVars=TRUE,
                  ylab=c("Histogram of missing data","Pattern"))
#From the plot, we see that 5% data is missing due to occupation and workclass variables &
#around 1.8% data is missin for column native_country
#On the left plot, we see that
#a) around 46 rows where all the three variables have no data & there are 45220 complete cases
#b) 2753 rows were just workclass and occupation have missing values and
# c) 10 missing rows were just related to occupation and
# d) 811 rows related to native_coutnry missing data,



#removing fnlwgt since it is a method of data collection only
data <- data[,!(colnames(data) == "fnlwgt")]
str(data)
# 'data.frame': 48840 obs. of  14 variables:
# $ age            : int  50 38 53 28 37 49 52 31 42 37 ...
# $ workclass      : Factor w/ 9 levels "?","Federal-gov",..: 7 5 5 5 5 7 5 5 5 ...
# $ education      : Factor w/ 16 levels "10th","11th",..: 10 12 2 10 13 7 12 13 10 16 ...
# $ educational_num: int  13 9 7 13 14 5 9 14 13 10 ...
```

```
# $ marital_status : Factor w/ 7 levels "Divorced","Married-AF-spouse",..: 3 1 3 3 3 4 3 5 3 3 ...
# $ occupation     : Factor w/ 15 levels "?","Adm-clerical",..: 5 7 7 11 5 9 5 11 5 5 ...
# $ relationship   : Factor w/ 6 levels "Husband","Not-in-family",..: 1 2 1 6 6 2 1 2 1 1 ...
# $ race           : Factor w/ 5 levels "Amer-Indian-Eskimo",..: 5 5 3 3 5 3 5 5 5 3 ...
# $ gender         : Factor w/ 2 levels "Female","Male": 2 2 2 1 1 1 2 1 2 2 ...
# $ capital_gain   : int  0 0 0 0 0 0 0 14084 5178 0 ...
# $ capital_loss   : int  0 0 0 0 0 0 0 0 0 0 ...
# $ hours_per_week : int  13 40 40 40 40 16 45 50 40 80 ...
# $ native_country : Factor w/ 42 levels "?","Cambodia",..: 40 40 40 6 40 24 40 40 40 40 ...
# $ income         : Factor w/ 2 levels "<=50K",">50K": 1 1 1 1 1 1 2 2 2 2 ...


#setting <=50k as lte50K & >50k as gt50k
data$income <- revalue(data$income, c("<=50K" = "lte50K", ">50K" = "gt50K"))
table(data$income)
# lte50K  gt50K
# 37153  11687
levels(data$income)
#[1] "lte50K" "gt50K"

#final check before dummy coding and preprocessing
for (i in colnames(data)){
  print(class(data[[i]]))
}
# [1] "integer"
# [1] "factor"
# [1] "integer"
# [1] "factor"
# [1] "integer"
# [1] "factor"
# [1] "factor"
# [1] "factor"
# [1] "factor"
# [1] "factor"
# [1] "integer"
# [1] "integer"
# [1] "integer"
# [1] "factor"
# [1] "factor"

#checking correlations and near Zero variance amongst the predictors
correlations <- cor(select_if(data, is.numeric))
correlations
findCorrelation(correlations, cutoff = 0.75)
#integer(0) - No correlation found between predictors
```

```
#Displaying  correlation using corrplot()
df_cor <- select_if(data, is.numeric) %>% cor()
corrplot(df_cor, method = "circle", order = "hclust")
# df_cor
#                 age        fnlwgt     educational_num  capital_gain capital_loss hours_per_week
# age             1.00000000 -0.076622108      0.03091655   0.07722681  0.056940150
0.07155839
# fnlwgt         -0.07662211  1.000000000     -0.03872893  -0.00370220 -0.004369368    -
0.01351938
# educational_num  0.03091655 -0.038728928      1.00000000   0.12514304  0.080974005
0.14369288
# capital_gain     0.07722681 -0.003702200      0.12514304   1.00000000 -0.031440805
0.08215732
# capital_loss     0.05694015 -0.004369368      0.08097400  -0.03144081  1.000000000
0.05446697
# hours_per_week   0.07155839 -0.013519384      0.14369288   0.08215732  0.054466967
1.00000000
# #there is no correlation, since none of the variables show correlation coeeficient magnitude of
more than 0.6

#identifying the nearZeroVariance in the dataset
nzv <- nearZeroVar(data)
print(names(data[nzv]))
#[1] "capital_gain"   "capital_loss"   "native_country"
# Capital_gain and capital_loss variables have a large number of 0 values which will cause
#issues during model fitting impacting results, thus will remove these variables in preprocess

#since educational_num provides similar info removing the variable
table(data$educational_num, data$education)

data$education <- NULL

glimpse(data)

#Target variable income has two levels: <=50K: lte50K and >50K: gt50K
#setting gt50K as the positive class
data$income <- relevel(data$income, ref = "gt50K")
levels(data$income)
summary(data$income)
# gt50K lte50K
# 11687  37153

#seperating target and independent variables before dummy coding:
y <- data$income
x <- data[,1:12] #only first 12 columns
```

```
#dummycoding
x_dummy.model <- dummyVars("~ .", data=x, fullRank=TRUE)

#apply model to data with predict function
x_dummy <- data.frame(predict(x_dummy.model, x))

str(x)
str(x_dummy) #had 12 variables before,now it has 81 variables


#removing nzv variables, imputing the missing data using knnImpute and preProcessing using
caret
set.seed(192)
x.prepmodel <- preProcess(x_dummy, method=c("knnImpute","center", "scale", "nzv"))
x.prepmodel
# Created from 45220 samples and 81 variables
#
# Pre-processing:
#   - centered (22)
# - ignored (0)
# - 5 nearest neighbor imputation (22)
# - removed (59)
# - scaled (22)

x.prep <- predict(x.prepmodel, x_dummy)

str(x.prep) #22 variables

#adding back the y column before splitting to test and train
data_census <- cbind(x.prep, y)
summary(data_census)
glimpse(data_census)
# Rows: 48,840
# Columns: 23
# $ age                          <dbl> 0.82827154, -0.04696039, 1.04707952, -0.77632033, -0.119…
# $ workclass.Local.gov              <dbl> -0.2703519, -0.2703519, -0.2703519, -0.2703519, -
0.27035…
# $ workclass.Private                <dbl> -1.6714345, 0.5982755, 0.5982755, 0.5982755,
0.5982755, …
# $ workclass.Self.emp.not.inc        <dbl> 3.3047399, -0.3025891, -0.3025891, -0.3025891, -
0.302589…
# $ educational_num                  <dbl> 1.13650756, -0.41933534, -1.19725679, 1.13650756,
1.5254…
# $ marital_status.Married.civ.spouse <dbl> 1.0873725, -0.9196292, 1.0873725, 1.0873725,
1.0873725, …
```

```
# $ marital_status.Never.married      <dbl> -0.7017314, -0.7017314, -0.7017314, -0.7017314, -
0.70173…
# $ occupation.Craft.repair          <dbl> -0.3912885, -0.3912885, -0.3912885, -0.3912885, -
0.39128…
# $ occupation.Exec.managerial        <dbl> 2.5618903, -0.3903283, -0.3903283, -0.3903283,
2.5618903…
# $ occupation.Machine.op.inspct      <dbl> -0.2650244, -0.2650244, -0.2650244, -0.2650244, -
0.26502…
# $ occupation.Other.service         <dbl> -0.3460565, -0.3460565, -0.3460565, -0.3460565, -
0.34605…
# $ occupation.Prof.specialty        <dbl> -0.3935003, -0.3935003, -0.3935003, 2.5412391, -
0.393500…
# $ occupation.Sales                 <dbl> -0.3685210, -0.3685210, -0.3685210, -0.3685210, -
0.36852…
# $ occupation.Transport.moving       <dbl> -0.2322038, -0.2322038, -0.2322038, -0.2322038, -
0.23220…
# $ relationship.Not.in.family       <dbl> -0.5890721, 1.6975502, -0.5890721, -0.5890721, -
0.589072…
# $ relationship.Own.child           <dbl> -0.4286132, -0.4286132, -0.4286132, -0.4286132, -
0.42861…
# $ relationship.Unmarried           <dbl> -0.3423949, -0.3423949, -0.3423949, -0.3423949, -
0.34239…
# $ race.Black                       <dbl> -0.3256935, -0.3256935, 3.0703082, 3.0703082, -
0.3256935…
# $ race.White                       <dbl> 0.4117144, 0.4117144, -2.4288184, -2.4288184,
0.4117144,…
# $ gender.Male                      <dbl> 0.7042348, 0.7042348, 0.7042348, -1.4199518, -
1.4199518,…
# $ hours_per_week                   <dbl> -2.21296556, -0.03408731, -0.03408731, -0.03408731, -
0.0…
# $ native_country.United.States      <dbl> 0.3078157, 0.3078157, 0.3078157, -3.2486300,
0.3078157, …
# $ y                                <fct> lte50K, lte50K, lte50K, lte50K, lte50K, lte50K, gt50K, g…

data_census$y


#test train split
set.seed(192)
#using 70% of the data as train and 30 as test
split_index<-createDataPartition(data_census$y, p=.70, list=FALSE)
train_data <- data_census[split_index,]  #34,189 rows
test_data <- data_census[-split_index,]  #14,651 rows

#creating trCtrl for fine-grained control over the tuning parameters that can be explored
ctrl <- trainControl(method="cv", number=10,
```

```
                 classProbs=TRUE,
                 summaryFunction = twoClassSummary,
                 verboseIter = TRUE)


##Logistic Regression
modelLookup("glm")
set.seed(192)
glm_fit<- train(y ~ ., data= train_data,
             trControl = ctrl,
             metric = "ROC",
             method = "glm",
             family=binomial)
glm_fit
# Generalized Linear Model
#
# 34189 samples
# 22 predictor
# 2 classes: 'gt50K', 'lte50K'
#
# No pre-processing
# Resampling: Cross-Validated (10 fold)
# Summary of sample sizes: 30770, 30770, 30770, 30770, 30770, 30770, ...
# Resampling results:
#
#   ROC       Sens       Spec
# 0.878871  0.5215757  0.9256003

summary(glm_fit) #try fitting with most significant variables tomorrow
# Call:
#   NULL
#
# Deviance Residuals:
#   Min       1Q   Median       3Q      Max
# -3.5524   0.0506   0.2355   0.5898   2.6160
#
# Coefficients:
#   Estimate Std. Error z value Pr(>|z|)
# (Intercept)                      2.038690   0.027059  75.341  < 2e-16 ***
#   age                           -0.332093   0.019518 -17.015  < 2e-16 ***
#   workclass.Local.gov              0.053233   0.018457   2.884 0.003925 **
#   workclass.Private                0.034798   0.021620   1.610 0.107499
# workclass.Self.emp.not.inc         0.203513   0.018816  10.816  < 2e-16 ***
#   educational_num                -0.776486   0.021447 -36.204  < 2e-16 ***
#   marital_status.Married.civ.spouse -1.127571   0.083606 -13.487  < 2e-16 ***
#   marital_status.Never.married       0.222101   0.033397   6.650 2.92e-11 ***
```

```
#   occupation.Craft.repair          -0.054451   0.018887  -2.883 0.003939 **
#   occupation.Exec.managerial        -0.292979   0.017907 -16.361  < 2e-16 ***
#   occupation.Machine.op.inspct       0.066598   0.020237   3.291 0.000998 ***
#   occupation.Other.service          0.269251   0.030668   8.780  < 2e-16 ***
#   occupation.Prof.specialty         -0.221992   0.019429 -11.426  < 2e-16 ***
#   occupation.Sales                 -0.103362   0.018938  -5.458 4.82e-08 ***
#   occupation.Transport.moving        0.005417   0.017111   0.317 0.751580
# relationship.Not.in.family        -0.182091   0.072318  -2.518 0.011805 *
#   relationship.Own.child            0.257022   0.065463   3.926 8.63e-05 ***
#   relationship.Unmarried           -0.001002   0.055208  -0.018 0.985527
# race.Black                        0.018207   0.030892   0.589 0.555616
# race.White                        -0.006279   0.029810  -0.211 0.833179
# gender.Male                       -0.070170   0.021824  -3.215 0.001304 **
#   hours_per_week                  -0.360732   0.017518 -20.592  < 2e-16 ***
#   native_country.United.States      -0.049706   0.019164  -2.594 0.009496 **
#   ---
#   Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
#
# (Dispersion parameter for binomial family taken to be 1)
#
# Null deviance: 37625  on 34188  degrees of freedom
# Residual deviance: 24551  on 34166  degrees of freedom
# AIC: 24597
#
# Number of Fisher Scoring iterations: 7

#predicting on test data
p_glm_fit <- predict(glm_fit,test_data, type="prob")
class_glm_fit <- predict(glm_fit,test_data)
confusionMatrix(class_glm_fit,test_data$y)
# Confusion Matrix and Statistics
#
# Reference
# Prediction gt50K lte50K
# gt50K   1871    766
# lte50K  1635  10379
#
# Accuracy : 0.8361
# 95% CI : (0.83, 0.8421)
# No Information Rate : 0.7607
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.5081
#
# Mcnemar's Test P-Value : < 2.2e-16
#
```

```
#            Sensitivity : 0.5337
#            Specificity : 0.9313
#         Pos Pred Value : 0.7095
#         Neg Pred Value : 0.8639
#             Prevalence : 0.2393
#         Detection Rate : 0.1277
#   Detection Prevalence : 0.1800
#      Balanced Accuracy : 0.7325
#
#       'Positive' Class : gt50K


#applying Decision tree
modelLookup("rpart")
set.seed(192)
rpart_fit<- train(y ~ ., data= train_data,
                 trControl = ctrl,
                 metric = "ROC", #using AUC to find best performing parameters
                 method = "rpart")
rpart_fit
# CART
#
# 34189 samples
# 22 predictor
# 2 classes: 'gt50K', 'lte50K'
#
# No pre-processing
# Resampling: Cross-Validated (10 fold)
# Summary of sample sizes: 30770, 30770, 30770, 30770, 30770, 30770, ...
# Resampling results across tuning parameters:
#
#   cp           ROC        Sens       Spec
# 0.004583792  0.8188427  0.4699929  0.9336354
# 0.007334067  0.8096903  0.4246384  0.9451714
# 0.123395673  0.6234069  0.1587536  0.9787389
#
# ROC was used to select the optimal model using the largest value.
# The final value used for the model was cp = 0.004583792.

getTrainPerf(rpart_fit)
#    TrainROC TrainSens TrainSpec method
# 1 0.8188427 0.4699929 0.9336354  rpart
rpart.plot(rpart_fit$finalModel)

#predict using test_data
p_rpart_fit <- predict(rpart_fit,test_data, type="prob")
```

```
class_rpart_fit <- predict(rpart_fit,test_data)
confusionMatrix(class_rpart_fit,test_data$y)
# Confusion Matrix and Statistics
#
# Reference
# Prediction gt50K lte50K
# gt50K   1700    689
# lte50K  1806  10456
#
# Accuracy : 0.8297
# 95% CI : (0.8235, 0.8358)
# No Information Rate : 0.7607
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.4749
#
# Mcnemar's Test P-Value : < 2.2e-16
#
#            Sensitivity : 0.4849
#            Specificity : 0.9382
#         Pos Pred Value : 0.7116
#         Neg Pred Value : 0.8527
#             Prevalence : 0.2393
#         Detection Rate : 0.1160
#   Detection Prevalence : 0.1631
#      Balanced Accuracy : 0.7115
#
#       'Positive' Class : gt50K



##applying Naive Bayes
modelLookup("nb")
install.packages("klaR")
install.packages("promises")
install.packages("fastmap")
library(promises)
library(klaR)
library(fastmap)
modelLookup("nb")
set.seed(192)
nb_fit<- train(y ~ .,data= train_data,
               trControl = ctrl,
               metric = "ROC",
               method = "nb"
               #tuneLength=8
```

```
)
nb_fit
# Naive Bayes
#
# 34189 samples
# 22 predictor
# 2 classes: 'gt50K', 'lte50K'
#
# No pre-processing
# Resampling: Cross-Validated (10 fold)
# Summary of sample sizes: 30770, 30770, 30770, 30770, 30770, 30770, ...
# Resampling results across tuning parameters:
#
#   usekernel  ROC        Sens       Spec
# FALSE      0.8478530  0.8168927  0.7168180
# TRUE      0.8685166  0.4885689  0.9236777
#
# Tuning parameter 'fL' was held constant at a value of 0
# Tuning parameter 'adjust' was held constant at a value of 1
# ROC was used to select the optimal model using the largest value.
# The final values used for the model were fL = 0, usekernel = TRUE and adjust = 1.

getTrainPerf(nb_fit)
#    TrainROC TrainSens TrainSpec method
# 1 0.8685166 0.4885689 0.9236777     nb


varImp(nb_fit) #marital_status.Married.civ.spouse & educational_num are the most important
(latest)
# ROC curve variable importance
#
# only 20 most important variables shown (out of 22)
#
# Importance
# marital_status.Married.civ.spouse   100.0000
# educational_num                      83.9641
# age                                  69.1091
# marital_status.Never.married         67.3959
# hours_per_week                       66.0347
# gender.Male                          45.2659
# relationship.Not.in.family           36.3633
# relationship.Own.child               35.8707
# occupation.Exec.managerial           29.3395
# occupation.Prof.specialty            28.3265
# occupation.Other.service             26.5754
# workclass.Private                    19.8465
```

```
# relationship.Unmarried              18.5084
# race.White                         10.9141
# race.Black                         10.0077
# occupation.Machine.op.inspct         8.7857
# native_country.United.States         3.0485
# occupation.Craft.repair             2.0928
# occupation.Transport.moving          1.3724
# workclass.Local.gov                 0.5657
plot(nb_fit)

#predict using test_data
p_nb_fit <- predict(nb_fit,test_data, type="prob")
class_nb_fit <- predict(nb_fit,test_data)
confusionMatrix(class_nb_fit,test_data$y)
# Confusion Matrix and Statistics
#
# Reference
# Prediction gt50K lte50K
# gt50K   1847    868
# lte50K  1659  10277
#
# Accuracy : 0.8275
# 95% CI : (0.8213, 0.8336)
# No Information Rate : 0.7607
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.4865
#
# Mcnemar's Test P-Value : < 2.2e-16
#
#            Sensitivity : 0.5268
#            Specificity : 0.9221
#         Pos Pred Value : 0.6803
#         Neg Pred Value : 0.8610
#             Prevalence : 0.2393
#         Detection Rate : 0.1261
#   Detection Prevalence : 0.1853
#      Balanced Accuracy : 0.7245
#
#       'Positive' Class : gt50K


##svm with radial kernel
set.seed(192)
modelLookup("svmRadial")
svm.grid<- expand.grid(sigma=c(.03, .04), C=c(0.5, 1))
```

```
#sigma=0.031 and C=0.5 were identified as optimal parameters using tunelength = 8
svm_fit <- train(y~ .,
              trControl = ctrl,
              metric = "ROC",
              data = train_data,
              #tuneLength=8,
              tuneGrid=svm.grid,
              method = "svmRadial")
svm_fit
# Support Vector Machines with Radial Basis Function Kernel
#
# 34189 samples
# 22 predictor
# 2 classes: 'gt50K', 'lte50K'
#
# No pre-processing
# Resampling: Cross-Validated (10 fold)
# Summary of sample sizes: 30770, 30770, 30770, 30770, 30770, 30770, ...
# Resampling results across tuning parameters:
#
#   sigma  C    ROC        Sens       Spec
# 0.03   0.5  0.8607485  0.4593607  0.9429639
# 0.03   1.0  0.8563782  0.4477384  0.9425170
# 0.04   0.5  0.8619039  0.4651589  0.9416250
# 0.04   1.0  0.8546501  0.4760181  0.9406959
#
# ROC was used to select the optimal model using the largest value.
# The final values used for the model were sigma = 0.04 and C = 0.5.
plot(svm_fit)
getTrainPerf(svm_fit)
#   TrainROC TrainSens TrainSpec    method
# 1 0.8619039 0.4651589  0.941625 svmRadial
p_svm_fit <- predict(svm_fit,test_data, type="prob")
class_svm_fit <- predict(svm_fit,test_data)
confusionMatrix(class_svm_fit,test_data$y)
# Confusion Matrix and Statistics
#
# Reference
# Prediction gt50K lte50K
# gt50K   1653    602
# lte50K  1853  10543
#
# Accuracy : 0.8324
# 95% CI : (0.8263, 0.8384)
# No Information Rate : 0.7607
# P-Value [Acc > NIR] : < 2.2e-16
```

```
#
# Kappa : 0.4756
#
# Mcnemar's Test P-Value : < 2.2e-16
#
#            Sensitivity : 0.4715
#            Specificity : 0.9460
#         Pos Pred Value : 0.7330
#         Neg Pred Value : 0.8505
#             Prevalence : 0.2393
#         Detection Rate : 0.1128
#   Detection Prevalence : 0.1539
#      Balanced Accuracy : 0.7087
#
#       'Positive' Class : gt50K


##applying random forest
modelLookup("rf")
library(randomForest)
set.seed(192)
mtryGrid <- expand.grid(mtry = c(4,5,6))   #grid search mtry=5 was the optimal mtry
rf_fit<- train(y ~ .,
              data= train_data,
              trControl = ctrl,
              #tuneLength = 7,
              tuneGrid = mtryGrid,
              metric = "ROC",
              method = "rf")
rf_fit
# Random Forest
#
# 34189 samples
# 22 predictor
# 2 classes: 'gt50K', 'lte50K'
#
# No pre-processing
# Resampling: Cross-Validated (10 fold)
# Summary of sample sizes: 30770, 30770, 30770, 30770, 30770, 30770, ...
# Resampling results across tuning parameters:
#
#   mtry  ROC        Sens       Spec
# 4     0.8712519  0.5600815  0.9246004
# 5     0.8721923  0.5779284  0.9187176
# 6     0.8719236  0.5796405  0.9139113
#
```

```
# ROC was used to select the optimal model using the largest value.
# The final value used for the model was mtry = 5.
getTrainPerf(rf_fit)
#   TrainROC TrainSens TrainSpec method
# 1 0.8721923 0.5779284 0.9187176     rf
plot(rf_fit)  #5 was the best
varImp(rf_fit)
# rf variable importance
#
# only 20 most important variables shown (out of 22)
#
# Overall
# educational_num                 100.0000
# marital_status.Married.civ.spouse  93.9234
# age                              88.0876
# hours_per_week                   54.9374
# marital_status.Never.married      23.6406
# occupation.Exec.managerial        16.3908
# occupation.Prof.specialty         14.7052
# gender.Male                      11.5972
# relationship.Not.in.family        9.1274
# workclass.Private                 6.4268
# workclass.Self.emp.not.inc        5.9949
# occupation.Other.service          4.5819
# relationship.Own.child            4.4211
# native_country.United.States      3.8454
# occupation.Sales                  3.6626
# race.White                        2.7788
# relationship.Unmarried            2.7336
# occupation.Craft.repair           2.5386
# workclass.Local.gov               1.7095
# race.Black                        0.7182
#plotting important variables
plot(varImp(rf_fit))

p_rf_fit <- predict(rf_fit , test_data, type="prob")
class_rf_fit <- predict(rf_fit, test_data)
confusionMatrix(class_rf_fit,test_data$y)
# Confusion Matrix and Statistics
#
# Reference
# Prediction gt50K lte50K
# gt50K   2045    851
# lte50K  1461  10294
#
# Accuracy : 0.8422
```

```
# 95% CI : (0.8362, 0.8481)
# No Information Rate : 0.7607
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.5391
#
# Mcnemar's Test P-Value : < 2.2e-16
#
#            Sensitivity : 0.5833
#            Specificity : 0.9236
#         Pos Pred Value : 0.7061
#         Neg Pred Value : 0.8757
#             Prevalence : 0.2393
#         Detection Rate : 0.1396
#   Detection Prevalence : 0.1977
#      Balanced Accuracy : 0.7535
#
#       'Positive' Class : gt50K


##neural network
modelLookup("nnet")
set.seed(192)
#classification method type to evaluate different nodes and weights for a single hidden layer
nnet_grid <- expand.grid(size = c(2,3,4), decay = c(0.1,1,2))
nn_fit <- train(y ~ .,
             trControl = ctrl,
             data = train_data,
             metric = "ROC",
             preProcess="range",
             tuneGrid = nnet_grid,
             method = "nnet")
nn_fit
# Neural Network
#
# 34189 samples
# 22 predictor
# 2 classes: 'gt50K', 'lte50K'
#
# Pre-processing: re-scaling to [0, 1] (22)
# Resampling: Cross-Validated (10 fold)
# Summary of sample sizes: 30770, 30770, 30770, 30770, 30770, 30770, ...
# Resampling results across tuning parameters:
#
#   size  decay  ROC        Sens       Spec
# 2     0.1    0.8822078  0.5411352  0.9215246
```

```
# 2     1.0    0.8818600  0.5408894  0.9216013
# 2     2.0    0.8818413  0.5391783  0.9229469
# 3     0.1    0.8845766  0.5394215  0.9259077
# 3     1.0    0.8846331  0.5375879  0.9254849
# 3     2.0    0.8838687  0.5400350  0.9244083
# 4     0.1    0.8856304  0.5505447  0.9246389
# 4     1.0    0.8860935  0.5459001  0.9248310
# 4     2.0    0.8851024  0.5390541  0.9257926
#
# ROC was used to select the optimal model using the largest value.
# The final values used for the model were size = 4 and decay = 1.
plot(nn_fit)  #4 neurons in a single layer gave better performance
getTrainPerf(nn_fit)
#   TrainROC TrainSens TrainSpec method
# 1 0.8860935 0.5459001  0.924831   nnet

#predicting on the test set:
p_nn_fit <- predict(nn_fit , test_data, type="prob")
class_nn_fit <- predict(nn_fit, test_data)
confusionMatrix(class_nn_fit,test_data$y)
# Confusion Matrix and Statistics
#
# Reference
# Prediction gt50K lte50K
# gt50K   1920    811
# lte50K  1586  10334
#
# Accuracy : 0.8364
# 95% CI : (0.8303, 0.8424)
# No Information Rate : 0.7607
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.5138
#
# Mcnemar's Test P-Value : < 2.2e-16
#
#            Sensitivity : 0.5476
#            Specificity : 0.9272
#         Pos Pred Value : 0.7030
#         Neg Pred Value : 0.8669
#             Prevalence : 0.2393
#         Detection Rate : 0.1310
#   Detection Prevalence : 0.1864
#      Balanced Accuracy : 0.7374
#
#       'Positive' Class : gt50K
```

```
#################PLOTTING AND COMPARING THE PERFORMANCE OF THE MODELS
#Train performance
rValues <- resamples(list(rpart=rpart_fit, logistic=glm_fit, naiveBayes= nb_fit, rf=rf_fit,
svm=svm_fit, nn=nn_fit))
summary(rValues)
# Call:
#   summary.resamples(object = rValues)
#
# Models: rpart, logistic, naiveBayes, rf, svm, nn
# Number of resamples: 10
#
# ROC
# Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
# rpart      0.7985021 0.8101445 0.8209701 0.8188427 0.8279165 0.8330536    0
# logistic   0.8693322 0.8765204 0.8796554 0.8788710 0.8820774 0.8857768    0
# naiveBayes 0.8578879 0.8649675 0.8692016 0.8685166 0.8728086 0.8768764    0
# rf         0.8591377 0.8695493 0.8739882 0.8721923 0.8753762 0.8790881    0
# svm        0.8544593 0.8571216 0.8614543 0.8619039 0.8657314 0.8712208    4
# nn         0.8771967 0.8839409 0.8871866 0.8860935 0.8901305 0.8909460    0
#
# Sens
#                 Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
# rpart      0.4229829 0.4345966 0.4581553 0.4699929 0.4712714 0.5806846    0
# logistic   0.4865526 0.5105345 0.5207824 0.5215757 0.5351467 0.5488998    0
# naiveBayes 0.3801956 0.4627139 0.5082477 0.4885689 0.5165037 0.5378973    0
# rf         0.5317848 0.5736553 0.5831296 0.5779284 0.5919927 0.6088020    0
# svm        0.4144254 0.4587408 0.4651589 0.4651589 0.4862469 0.4963325    4
# nn         0.5024450 0.5383216 0.5415648 0.5459001 0.5644866 0.5733496    0
#
# Spec
# Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
# rpart      0.8985006 0.9300269 0.9375240 0.9336354 0.9485775 0.9507692    0
# logistic   0.9177240 0.9221453 0.9236832 0.9256003 0.9300202 0.9346154    0
# naiveBayes 0.9088812 0.9139754 0.9234762 0.9236777 0.9314496 0.9427143    0
# rf         0.9100346 0.9140715 0.9192618 0.9187176 0.9228105 0.9277201    0
# svm        0.9365629 0.9388697 0.9430988 0.9416250 0.9438677 0.9454056    4
# nn         0.9196463 0.9215686 0.9240532 0.9248310 0.9275279 0.9315648    0


#plots comparing them
bwplot(rValues, metric="ROC")
bwplot(rValues, metric="Sens")
bwplot(rValues, metric="Spec")
```

```
#RF has the highest sensitivity value and it doesn't vary much compared to other models
#it predicts the gt50K class more precisely than other models, NB's sensitivity performance varies the most
#Decision tree and SVM provde lowest sensitivity for this imbalanced dataset
#although RF has slightly lower accuracy than logistic regression, it predicts the positive class more accurately

#TEST METRICS PLOTTING
rpart_roc <- roc(test_data$y, p_rpart_fit$gt50K)
glm_roc <- roc(test_data$y, p_glm_fit$gt50K)
rf_roc<- roc(test_data$y, p_rf_fit$gt50K)
nb_roc<- roc(test_data$y, p_nb_fit$gt50K)
svm_roc<- roc(test_data$y, p_svm_fit$gt50K)
nn_roc <- roc(test_data$y, p_nn_fit$gt50K)

#auc
auc(rpart_roc) #Area under the curve: 0.8336
auc(glm_roc)   #Area under the curve: 0.8847
auc(rf_roc)    #Area under the curve: 0.8766
auc(nb_roc)    #Area under the curve: 0.8743
auc(svm_roc)   #Area under the curve: 0.862
auc(nn_roc)    #Area under the curve: 0.8888

#combined ROC plot
plot(rpart_roc, col="black", legacy.axes=TRUE)
plot(glm_roc, add=T, col="red", legacy.axes=TRUE)
plot(rf_roc, add=T, col="blue", legacy.axes=TRUE)
plot(nb_roc, add=T, col="green", legacy.axes=TRUE)
plot(svm_roc, add=T, col="orange", legacy.axes=TRUE)
plot(nn_roc, add=T, col="brown", legacy.axes=TRUE)
legend(x=.34, y=.3, cex=.5, legend=c("Decision Tree","Logistic", "Random Forest", "Naive Bayes", "SVM", "Neural Network"),
       col=c("black", "red", "blue", "green", "orange", "brown"), lty= 1 , lwd=5)
#Decision tree has the lowest ROC
#RF, Neural network and logistic regression models appear to provide better performance for our dataset

result_auc_all <- data_frame("Models"=c("Decision Tree","Logistic", "Random Forest", "Naive Bayes", "SVM", "Neural Network"),

"AUC"=c(auc(rpart_roc),auc(glm_roc),auc(rf_roc),auc(nb_roc),auc(svm_roc),auc(nn_roc)))
print(result_auc_all)
# # A tibble: 6 x 2
# Models          AUC
# <chr>         <dbl>
# 1 Decision Tree  0.834
```

```
# 2 Logistic       0.885
# 3 Random Forest  0.877
# 4 Naive Bayes    0.874
# 5 SVM            0.862
# 6 Neural Network 0.889

#table format for all test metrics
rpart_cm <- confusionMatrix(class_rpart_fit,test_data$y)
glm_cm <- confusionMatrix(class_glm_fit, test_data$y)
rf_cm <- confusionMatrix(class_rf_fit, test_data$y)
nb_cm <- confusionMatrix(class_nb_fit, test_data$y)
svm_cm <- confusionMatrix(class_svm_fit, test_data$y)
nn_cm <- confusionMatrix(class_nn_fit,test_data$y)
model_cm_metrics_all <- data_frame("Models"=c("Decision Tree","Logistic", "Random
Forest", "Naive Bayes", "SVM", "Neural Network"),

"Sensitivity"=c(rpart_cm$byClass["Sensitivity"],glm_cm$byClass["Sensitivity"],
rf_cm$byClass["Sensitivity"],
                                          nb_cm$byClass["Sensitivity"],svm_cm$byClass["Sensitivity"],
nn_cm$byClass["Sensitivity"]),

"Specificity"=c(rpart_cm$byClass["Specificity"],glm_cm$byClass["Specificity"],
rf_cm$byClass["Specificity"],
                                          nb_cm$byClass["Specificity"],svm_cm$byClass["Specificity"],
nn_cm$byClass["Specificity"]),
                            "Balanced_Accuracy"=c(rpart_cm$byClass["Balanced
Accuracy"],glm_cm$byClass["Balanced Accuracy"], rf_cm$byClass["Balanced Accuracy"],
                                               nb_cm$byClass["Balanced
Accuracy"],svm_cm$byClass["Balanced Accuracy"], nn_cm$byClass["Balanced Accuracy"]),

"Accuracy"=c(rpart_cm$overall["Accuracy"],glm_cm$overall["Accuracy"],
rf_cm$overall["Accuracy"],
                                        nb_cm$overall["Accuracy"],svm_cm$overall["Accuracy"],
nn_cm$overall["Accuracy"]))
print(model_cm_metrics_all)
# # A tibble: 6 x 5
# Models         Sensitivity Specificity    Balanced_Accuracy Accuracy
# <chr>                <dbl>       <dbl>             <dbl>    <dbl>
# 1 Decision Tree        0.485       0.938             0.712    0.830
# 2 Logistic             0.534       0.931             0.732    0.836
# 3 Random Forest        0.583       0.924             0.753    0.842
# 4 Naive Bayes          0.527       0.922             0.724    0.828
# 5 SVM                  0.471       0.946             0.709    0.832
# 6 Neural Network       0.548       0.927             0.737    0.836
```

```
######SMOTE CODE
## Data imbalance - Using SMOTE on train data to balance the income variable
summary(train_data$y)
# gt50K lte50K
# 8181  26008
summary(test_data$y)
# gt50K lte50K
# 3506  11145

library(DMwR)
set.seed(123)
train_data_smote <- SMOTE(y ~ ., data = train_data, perc.over=100, perc.under=250,k=5)
histogram(train_data_smote$y)
summary(train_data_smote$y)
# gt50K lte50K
# 16362  20452


#applying Decision tree
modelLookup("rpart")
set.seed(192)
rpart_fit_smote <- train(y ~ .,
                    trControl = ctrl,
                    data = train_data_smote,
                    metric = "ROC",
                    method = "rpart")
rpart_fit_smote
# CART
#
# 36814 samples
# 22 predictor
# 2 classes: 'gt50K', 'lte50K'
#
# No pre-processing
# Resampling: Cross-Validated (10 fold)
# Summary of sample sizes: 33133, 33132, 33133, 33133, 33133, 33133, ...
# Resampling results across tuning parameters:
#
#   cp          ROC        Sens       Spec
# 0.01286518  0.8146151  0.7247861  0.8140969
# 0.04559345  0.7878291  0.7279776  0.7824098
# 0.43356558  0.6009603  0.3378457  0.8640748
#
# ROC was used to select the optimal model using the largest value.
# The final value used for the model was cp = 0.01286518.
```

```
getTrainPerf(rpart_fit_smote)
#    TrainROC TrainSens TrainSpec method
# 1 0.8146151 0.7247861 0.8140969  rpart

varImp(rpart_fit_smote)        # marital_status is most important , followed by educational_num
# rpart variable importance
#
# only 20 most important variables shown (out of 22)
#
# Overall
# marital_status.Married.civ.spouse 100.000
# educational_num                   71.008
# marital_status.Never.married      58.611
# age                               58.495
# hours_per_week                    33.213
# occupation.Prof.specialty          7.601
# occupation.Exec.managerial         6.720
# occupation.Other.service           6.599
# occupation.Craft.repair            0.000
# gender.Male                        0.000
# race.Black                         0.000
# occupation.Transport.moving        0.000
# relationship.Own.child             0.000
# workclass.Self.emp.not.inc         0.000
# relationship.Unmarried             0.000
# race.White                         0.000
# relationship.Not.in.family         0.000
# occupation.Sales                   0.000
# workclass.Private                  0.000
# native_country.United.States       0.000
rpart.plot(rpart_fit_smote$finalModel)

p_rpart_fit_smote <- predict(rpart_fit_smote,test_data, type="prob")
class_rpart_fit_smote <- predict(rpart_fit_smote,test_data)
confusionMatrix(class_rpart_fit_smote,test_data$y)
# Confusion Matrix and Statistics
#
# Reference
# Prediction gt50K lte50K
# gt50K   2244   1515
# lte50K  1262   9630
#
# Accuracy : 0.8105
# 95% CI : (0.804, 0.8168)
# No Information Rate : 0.7607
# P-Value [Acc > NIR] : < 2.2e-16
```

```
#
# Kappa : 0.4919
#
# Mcnemar's Test P-Value : 1.735e-06
#
#            Sensitivity : 0.6400
#            Specificity : 0.8641
#         Pos Pred Value : 0.5970
#         Neg Pred Value : 0.8841
#             Prevalence : 0.2393
#         Detection Rate : 0.1532
#   Detection Prevalence : 0.2566
#      Balanced Accuracy : 0.7521
#
#       'Positive' Class : gt50K

#applying glm
modelLookup("glm")
set.seed(192)
glm_fit_smote <-train(y ~ .,
                 trControl = ctrl,
                 data = train_data_smote,
                 metric = "ROC", #using AUC to find best performing parameters
                 method = "glm",
                 family = "binomial")

glm_fit_smote
# Generalized Linear Model
#
# 36814 samples
# 22 predictor
# 2 classes: 'gt50K', 'lte50K'
#
# No pre-processing
# Resampling: Cross-Validated (10 fold)
# Summary of sample sizes: 33133, 33132, 33133, 33133, 33133, 33133, ...
# Resampling results:
#
#   ROC        Sens       Spec
# 0.8813868  0.7958699  0.7986994

summary(glm_fit_smote)
# Call:
#   NULL
#
# Deviance Residuals:
```

```
#   Min       1Q   Median       3Q      Max
# -3.3588  -0.6902   0.1503   0.6119   2.9219
#
# Coefficients:
#   Estimate Std. Error z value Pr(>|z|)
# (Intercept)                         1.156308   0.021156  54.656  < 2e-16 ***
#   age                             -0.367486   0.018019 -20.395  < 2e-16 ***
#   workclass.Local.gov               0.053196   0.016461   3.232  0.00123 **
#   workclass.Private                 0.025580   0.019493   1.312  0.18944
# workclass.Self.emp.not.inc         0.177856   0.016943  10.497  < 2e-16 ***
#   educational_num                 -0.816186   0.019165 -42.586  < 2e-16 ***
#   marital_status.Married.civ.spouse -1.174243   0.068043 -17.257  < 2e-16 ***
#   marital_status.Never.married      0.193501   0.026847   7.208 5.69e-13 ***
#   occupation.Craft.repair          -0.041436   0.016224  -2.554  0.01065 *
#   occupation.Exec.managerial       -0.290089   0.016137 -17.976  < 2e-16 ***
#   occupation.Machine.op.inspct      0.068466   0.016806   4.074 4.62e-05 ***
#   occupation.Other.service          0.285964   0.024415  11.713  < 2e-16 ***
#   occupation.Prof.specialty        -0.221358   0.017216 -12.857  < 2e-16 ***
#   occupation.Sales                 -0.075988   0.016555  -4.590 4.43e-06 ***
#   occupation.Transport.moving       0.009141   0.014655   0.624  0.53277
# relationship.Not.in.family       -0.182945   0.058921  -3.105  0.00190 **
#   relationship.Own.child            0.257089   0.052763   4.873 1.10e-06 ***
#   relationship.Unmarried           -0.010884   0.044669  -0.244  0.80750
# race.Black                          0.018167   0.026442   0.687  0.49207
# race.White                         -0.043500   0.026036  -1.671  0.09476 .
# gender.Male                        -0.089377   0.018198  -4.911 9.05e-07 ***
#   hours_per_week                  -0.417324   0.016885 -24.716  < 2e-16 ***
#   native_country.United.States     -0.079230   0.016728  -4.736 2.18e-06 ***
#   ---
#   Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
#
# (Dispersion parameter for binomial family taken to be 1)
#
# Null deviance: 50580  on 36813  degrees of freedom
# Residual deviance: 31314  on 36791  degrees of freedom
# AIC: 31360
#
# Number of Fisher Scoring iterations: 6

#predicting
p_glm_fit_smote <- predict(glm_fit_smote,test_data, type="prob")
class_glm_fit_smote <- predict(glm_fit_smote,test_data)
confusionMatrix(class_glm_fit_smote,test_data$y)
# Confusion Matrix and Statistics
#
# Reference
```

```
# Prediction gt50K lte50K
# gt50K   2823   2230
# lte50K   683   8915
#
# Accuracy : 0.8012
# 95% CI : (0.7946, 0.8076)
# No Information Rate : 0.7607
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.5256
#
# Mcnemar's Test P-Value : < 2.2e-16
#
#            Sensitivity : 0.8052
#            Specificity : 0.7999
#         Pos Pred Value : 0.5587
#         Neg Pred Value : 0.9288
#             Prevalence : 0.2393
#         Detection Rate : 0.1927
#    Detection Prevalence : 0.3449
#       Balanced Accuracy : 0.8026
#
#        'Positive' Class : gt50K

#applying Naive-Bayes
modelLookup("nb")
set.seed(192)
nb_fit_smote<- train(y ~ .,
                  data= train_data_smote,
                  trControl = ctrl,
                  metric = "ROC", #using AUC to find best performing parameters
                  method = "nb")
nb_fit_smote
# Naive Bayes
#
# 36814 samples
# 22 predictor
# 2 classes: 'gt50K', 'lte50K'
#
# No pre-processing
# Resampling: Cross-Validated (10 fold)
# Summary of sample sizes: 33133, 33132, 33133, 33133, 33133, 33133, ...
# Resampling results across tuning parameters:
#
# usekernel  ROC       Sens       Spec
# FALSE      0.851580  0.8285054  0.7045773
```

```
# TRUE       0.897385  0.9442004  0.6293279
#
# Tuning parameter 'fL' was held constant at a value of 0
# Tuning parameter 'adjust' was held constant at a value of 1
# ROC was used to select the optimal model using the largest value.
# The final values used for the model were fL = 0, usekernel = TRUE and adjust = 1.
getTrainPerf(nb_fit_smote)
#   TrainROC TrainSens TrainSpec method
# 1 0.897385 0.9442004 0.6293279     nb

varImp(nb_fit_smote) #marital_status.Married.civ.spouse & educational_num are the most
important (latest)
# ROC curve variable importance
#
# only 20 most important variables shown (out of 22)
#
# Importance
# marital_status.Married.civ.spouse   100.0000
# educational_num                      84.9900
# age                                  69.0885
# hours_per_week                       67.7366
# marital_status.Never.married         66.4884
# gender.Male                          45.0492
# relationship.Not.in.family           36.8984
# relationship.Own.child               35.2916
# occupation.Exec.managerial           30.2269
# occupation.Prof.specialty            28.6929
# occupation.Other.service             26.5384
# workclass.Private                    19.6519
# relationship.Unmarried               18.2396
# race.White                           12.5688
# race.Black                           10.6113
# occupation.Machine.op.inspct          8.6756
# native_country.United.States          3.3532
# occupation.Craft.repair               1.7839
# occupation.Transport.moving           1.0020
# workclass.Self.emp.not.inc            0.7803


#predict using test_data
p_nb_fit_smote <- predict(nb_fit_smote,test_data, type="prob")
class_nb_fit_smote <- predict(nb_fit_smote,test_data)
confusionMatrix(class_nb_fit_smote,test_data$y)
# Confusion Matrix and Statistics
#
#           Reference
```

```
# Prediction gt50K lte50K
# gt50K     3252   4189
# lte50K    254   6956
#
# Accuracy : 0.6967
# 95% CI : (0.6892, 0.7042)
# No Information Rate : 0.7607
# P-Value [Acc > NIR] : 1
#
# Kappa : 0.3984
#
# Mcnemar's Test P-Value : <2e-16
#
#            Sensitivity : 0.9276
#            Specificity : 0.6241
#         Pos Pred Value : 0.4370
#         Neg Pred Value : 0.9648
#             Prevalence : 0.2393
#         Detection Rate : 0.2220
#   Detection Prevalence : 0.5079
#      Balanced Accuracy : 0.7758
#
#       'Positive' Class : gt50K

#applying Random forest
set.seed(192)
mtryGrid_smote <- expand.grid(mtry = c(5,8,12))  #using tunelegth=7 gave us mtry=12 as the
optimal mtry
rf_fit_smote<- train(y ~ .,
                 data= train_data_smote,
                 trControl = ctrl,
                 #tuneLength = 7,
                 tuneGrid = mtryGrid_smote,
                 metric = "ROC",
                 method = "rf")

rf_fit_smote
# Random Forest
#
# 36814 samples
# 22 predictor
# 2 classes: 'gt50K', 'lte50K'
#
# No pre-processing
# Resampling: Cross-Validated (10 fold)
# Summary of sample sizes: 33133, 33132, 33133, 33133, 33133, 33133, ...
```

```
# Resampling results across tuning parameters:
#
#   mtry  ROC        Sens       Spec
# 5    0.9292088  0.8682311  0.8421669
# 8    0.9397689  0.8668256  0.8740958
# 12    0.9415564  0.8479401  0.8916983
#
# ROC was used to select the optimal model using the largest value.
# The final value used for the model was mtry = 12.

plot(rf_fit_smote)
varImp(rf_fit_smote)    #marital_status, age and educational_num are the most important parameters
# rf variable importance
# only 20 most important variables shown (out of 22)
#
# Overall
# marital_status.Married.civ.spouse 100.000
# age                             95.852
# educational_num                 81.323
# hours_per_week                  55.131
# marital_status.Never.married      21.114
# workclass.Private                8.211
# gender.Male                      6.912
# occupation.Exec.managerial          6.865
# occupation.Prof.specialty          6.370
# occupation.Sales                 4.396
# occupation.Craft.repair            4.307
# workclass.Self.emp.not.inc          4.008
# native_country.United.States        3.919
# race.White                        3.333
# occupation.Other.service           3.316
# workclass.Local.gov               2.693
# relationship.Not.in.family          2.120
# occupation.Transport.moving         1.980
# occupation.Machine.op.inspct        1.391
# race.Black                        1.276
getTrainPerf(rf_fit_smote)
#    TrainROC TrainSens TrainSpec method
# 1 0.9415564 0.8479401 0.8916983     rf

#predicting on the test data
p_rf_fit_smote <- predict(rf_fit_smote,test_data, type="prob")
class_rf_fit_smote <- predict(rf_fit_smote,test_data)
confusionMatrix(class_rf_fit_smote,test_data$y)  #we predict the positive class better now compared to imbalanced dataset
```

```
# Confusion Matrix and Statistics
#
# Reference
# Prediction gt50K lte50K
# gt50K   2446   1808
# lte50K  1060   9337
#
# Accuracy : 0.8042
# 95% CI : (0.7977, 0.8106)
# No Information Rate : 0.7607
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.499
#
# Mcnemar's Test P-Value : < 2.2e-16
#
#            Sensitivity : 0.6977
#            Specificity : 0.8378
#         Pos Pred Value : 0.5750
#         Neg Pred Value : 0.8980
#             Prevalence : 0.2393
#         Detection Rate : 0.1670
#   Detection Prevalence : 0.2904
#      Balanced Accuracy : 0.7677
#
#       'Positive' Class : gt50K


##neural network
modelLookup("nnet")
set.seed(192)
nnet_grid <- expand.grid(size = c(3,4,5), decay = c(0.1,1))
nn_fit_smote <- train(y ~ .,
                  trControl = ctrl,
                  data = train_data_smote,
                  metric = "ROC",
                  preProcess="range",
                  #tunelength = 7,
                  tuneGrid = nnet_grid,
                  method = "nnet")
nn_fit_smote
# Neural Network
#
# 36814 samples
# 22 predictor
# 2 classes: 'gt50K', 'lte50K'
```

```
#
# Pre-processing: re-scaling to [0, 1] (22)
# Resampling: Cross-Validated (10 fold)
# Summary of sample sizes: 33133, 33132, 33133, 33133, 33133, 33133, ...
# Resampling results across tuning parameters:
#
#   size  decay  ROC        Sens       Spec
# 3     0.1    0.8877366  0.8208053  0.7883824
# 3     1.0    0.8884742  0.8173218  0.7943483
# 4     0.1    0.8901919  0.8173217  0.7939569
# 4     1.0    0.8894922  0.8126162  0.7953745
# 5     0.1    0.8921760  0.8206840  0.7978679
# 5     1.0    0.8908129  0.8154901  0.7978197
#
# ROC was used to select the optimal model using the largest value.
# The final values used for the model were size = 5 and decay = 0.1.
getTrainPerf(nn_fit_smote)
#   TrainROC TrainSens TrainSpec method
# 1 0.892176  0.820684 0.7978679   nnet
plot(nn_fit_smote)
#predicting on the test set
p_nn_fit_smote <- predict(nn_fit_smote,test_data, type="prob")
class_nn_fit_smote <- predict(nn_fit_smote,test_data)
confusionMatrix(class_nn_fit_smote,test_data$y)
# Confusion Matrix and Statistics
#
# Reference
# Prediction gt50K lte50K
# gt50K   2892   2230
# lte50K   614   8915
#
# Accuracy : 0.8059
# 95% CI : (0.7994, 0.8123)
# No Information Rate : 0.7607
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.5396
#
# Mcnemar's Test P-Value : < 2.2e-16
#
#            Sensitivity : 0.8249
#            Specificity : 0.7999
#         Pos Pred Value : 0.5646
#         Neg Pred Value : 0.9356
#             Prevalence : 0.2393
#         Detection Rate : 0.1974
```

```
#    Detection Prevalence : 0.3496
#       Balanced Accuracy : 0.8124
#
#        'Positive' Class : gt50K


##svm with radial kernel
set.seed(192)
modelLookup("svmRadial")
smote_svm.grid<- expand.grid(sigma=c(.01, .03, .04), C=c(0.05, 1, 14))
#sigma=0.031 and C=0.5 were identified as optimal parameters using tunelength = 8
svm_fit_smote <- train(y~ .,
                       trControl = ctrl,
                       metric = "ROC",
                       data = train_data_smote,
                       #tuneLength=7,
                       tuneGrid=smote_svm.grid,
                       method = "svmRadial")
svm_fit_smote
# Support Vector Machines with Radial Basis Function Kernel
#
# 36814 samples
# 22 predictor
# 2 classes: 'gt50K', 'lte50K'
#
# No pre-processing
# Resampling: Cross-Validated (10 fold)
# Summary of sample sizes: 33133, 33132, 33133, 33133, 33133, 33133, ...
# Resampling results across tuning parameters:
#
#   sigma  C      ROC        Sens       Spec
# 0.01    0.05  0.8789919  0.7892692  0.7815379
# 0.01    1.00  0.8877591  0.8133864  0.7928445
# 0.01   14.00  0.8908919  0.8232257  0.8008557
# 0.03    0.05  0.8830890  0.7793682  0.8007535
# 0.03    1.00  0.8884951  0.8129216  0.8052026
# 0.03   14.00  0.8908786  0.8340730  0.8009039
# 0.04    0.05  0.8834774  0.7777182  0.8057407
# 0.04    1.00  0.8873805  0.8144500  0.8074028
# 0.04   14.00  0.8918338  0.8390179  0.8019942
#
# ROC was used to select the optimal model using the largest value.
# The final values used for the model were sigma = 0.04 and C = 14.
plot(svm_fit_smote)

#predicting on the test set
```

```
p_svm_fit_smote <- predict(svm_fit_smote,test_data, type="prob")
class_svm_fit_smote <- predict(svm_fit_smote,test_data)
confusionMatrix(class_svm_fit_smote,test_data$y)
# Confusion Matrix and Statistics
#
# Reference
# Prediction gt50K lte50K
# gt50K   2845   2285
# lte50K   661   8860
#
# Accuracy : 0.7989
# 95% CI : (0.7923, 0.8054)
# No Information Rate : 0.7607
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.5234
#
# Mcnemar's Test P-Value : < 2.2e-16
#
#            Sensitivity : 0.8115
#            Specificity : 0.7950
#         Pos Pred Value : 0.5546
#         Neg Pred Value : 0.9306
#             Prevalence : 0.2393
#         Detection Rate : 0.1942
#    Detection Prevalence : 0.3501
#       Balanced Accuracy : 0.8032
#
#        'Positive' Class : gt50K
#

######################PLOTTING AND COMPARING THE PERFORMANCE OF THE MODELS
#Training performance
rValues_smote <- resamples(list(rpart=rpart_fit_smote, logistic=glm_fit_smote, naiveBayes=
nb_fit_smote, rf=rf_fit_smote, svm=svm_fit_smote, nn=nn_fit_smote))
summary(rValues_smote)
# Call:
#   summary.resamples(object = rValues_smote)
#
# Models: rpart, logistic, naiveBayes, rf, svm, nn
# Number of resamples: 10
#
# ROC
#               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
# rpart      0.8037098 0.8058688 0.8132049 0.8146151 0.8211387 0.8330332    0
```

```
# logistic   0.8750215 0.8792974 0.8801389 0.8813868 0.8833411 0.8913730    0
# naiveBayes 0.8899398 0.8945275 0.8961419 0.8973850 0.9010786 0.9048692    0
# rf         0.9366333 0.9387677 0.9419292 0.9415564 0.9439345 0.9473477    0
# svm        0.8875876 0.8880610 0.8893963 0.8918338 0.8938229 0.9003013    5
# nn         0.8851560 0.8873387 0.8912948 0.8921760 0.8951746 0.9022619    0
#
# Sens
#                 Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
# rpart      0.6411980 0.6505196 0.6543399 0.7247861 0.8173900 0.8667482    0
# logistic   0.7770312 0.7906798 0.7930929 0.7958699 0.8010391 0.8190709    0
# naiveBayes 0.9290954 0.9390281 0.9428652 0.9442004 0.9520171 0.9559902    0
# rf         0.8312958 0.8422983 0.8493741 0.8479401 0.8519254 0.8673594    0
# svm        0.8251834 0.8276284 0.8332315 0.8390179 0.8422983 0.8667482    5
# nn         0.7978009 0.8127393 0.8233496 0.8206840 0.8285452 0.8410758    0
#
# Spec
#                 Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
# rpart      0.7222494 0.7498778 0.8572476 0.8140969 0.8611754 0.8753056    0
# logistic   0.7882641 0.7929348 0.7953545 0.7986994 0.8036916 0.8180929    0
# naiveBayes 0.6044010 0.6236401 0.6261614 0.6293279 0.6397311 0.6484108    0
# rf         0.8797066 0.8887939 0.8907090 0.8916983 0.8979218 0.9026895    0
# svm        0.7911980 0.8015640 0.8034230 0.8019942 0.8063570 0.8074291    5
# nn         0.7833741 0.7914425 0.7998046 0.7978679 0.8018580 0.8156479    0


bwplot(rValues_smote, metric="ROC")
bwplot(rValues_smote, metric="Sens")
bwplot(rValues_smote, metric="Spec")
#RF has the highest ROC value follwed by NaiveBayes and NN
#NaiveBayes has highest sensitivity training performance, followed by RF and SVM


#Test data performance metrics
#Plots and AUC-ROC for test data
rpart_smote_roc <- roc(test_data$y, p_rpart_fit_smote$gt50K)
glm_smote_roc <- roc(test_data$y, p_glm_fit_smote$gt50K)
rf_smote_roc<- roc(test_data$y, p_rf_fit_smote$gt50K)
nb_smote_roc<- roc(test_data$y, p_nb_fit_smote$gt50K)
svm_smote_roc<- roc(test_data$y, p_svm_fit_smote$gt50K)
nn_smote_roc <- roc(test_data$y, p_nn_fit_smote$gt50K)

# auc
auc(rpart_smote_roc) #Area under the curve: 0.8134
auc(glm_smote_roc)   #Area under the curve: 0.8848
auc(rf_smote_roc)    #Area under the curve: 0.865
auc(nb_smote_roc)    #Area under the curve: 0.8741
```

```
auc(svm_smote_roc)   #Area under the curve: 0.8686
auc(nn_smote_roc)    #Area under the curve: 0.8929

#ROC plot
plot(rpart_smote_roc, col="black", legacy.axes=TRUE)
plot(glm_smote_roc, add=T, col="red", legacy.axes=TRUE)
plot(rf_smote_roc, add=T, col="blue", legacy.axes=TRUE)
plot(nb_smote_roc, add=T, col="green", legacy.axes=TRUE)
plot(svm_smote_roc, add=T, col="orange", legacy.axes=TRUE)
plot(nn_smote_roc, add=T, col="brown", legacy.axes=TRUE)
legend(x=.34, y=.3, cex=.5, legend=c("Decision Tree","Logistic", "Random Forest", "Naive
Bayes", "SVM", "Neural Network"),
       col=c("black", "red", "blue", "green", "orange", "brown"), lty= 1 , lwd=5)
#Decision tree has the lowest ROC
#Neural network, Logistic regression are best performing


result_smote_auc_all <- data_frame("Models"=c("Decision Tree","Logistic", "Random Forest",
"Naive Bayes", "SVM", "Neural Network"),

"AUC"=c(auc(rpart_smote_roc),auc(glm_smote_roc),auc(rf_smote_roc),auc(nb_smote_roc),auc(
svm_smote_roc),auc(nn_smote_roc)))
print(result_smote_auc_all)
#AUC confirms Neural network and Logistic regression are best performing
# # A tibble: 6 x 2
# Models           AUC
# <chr>          <dbl>
# 1 Decision Tree  0.813
# 2 Logistic       0.885
# 3 Random Forest  0.865
# 4 Naive Bayes    0.874
# 5 SVM            0.869
# 6 Neural Network 0.893


#table format for the accuracy
rpart_smote_cm <- confusionMatrix(class_rpart_fit_smote,test_data$y)
glm_smote_cm <- confusionMatrix(class_glm_fit_smote, test_data$y)
rf_smote_cm <- confusionMatrix(class_rf_fit_smote, test_data$y)
nb_smote_cm <- confusionMatrix(class_nb_fit_smote, test_data$y)
svm_smote_cm <- confusionMatrix(class_svm_fit_smote, test_data$y)
nn_smote_cm <- confusionMatrix(class_nn_fit_smote,test_data$y)
model_cm_metrics_all <- data_frame("Models"=c("Decision Tree","Logistic", "Random
Forest", "Naive Bayes", "SVM", "Neural Network"),
```

```
"Sensitivity"=c(rpart_smote_cm$byClass["Sensitivity"],glm_smote_cm$byClass["Sensitivity"],
rf_smote_cm$byClass["Sensitivity"],

nb_smote_cm$byClass["Sensitivity"],svm_smote_cm$byClass["Sensitivity"],
nn_smote_cm$byClass["Sensitivity"]),

"Specificity"=c(rpart_smote_cm$byClass["Specificity"],glm_smote_cm$byClass["Specificity"],
rf_smote_cm$byClass["Specificity"],

nb_smote_cm$byClass["Specificity"],svm_smote_cm$byClass["Specificity"],
nn_smote_cm$byClass["Specificity"]),
                          "Balanced_Accuracy"= c(rpart_smote_cm$byClass["Balanced
Accuracy"],glm_smote_cm$byClass["Balanced Accuracy"], rf_smote_cm$byClass["Balanced
Accuracy"],
                                         nb_smote_cm$byClass["Balanced
Accuracy"],svm_smote_cm$byClass["Balanced Accuracy"], nn_smote_cm$byClass["Balanced
Accuracy"]),

"Accuracy"=c(rpart_smote_cm$overall["Accuracy"],glm_smote_cm$overall["Accuracy"],
rf_smote_cm$overall["Accuracy"],

nb_smote_cm$overall["Accuracy"],svm_smote_cm$overall["Accuracy"],
nn_smote_cm$overall["Accuracy"]))
print(model_cm_metrics_all)
# Models         Sensitivity Specificity  Balanced_Accuracy    Accuracy
# <chr>               <dbl>       <dbl>             <dbl>    <dbl>
# 1 Decision Tree       0.640       0.864             0.752    0.810
# 2 Logistic            0.805       0.800             0.803    0.801
# 3 Random Forest       0.698       0.838             0.768    0.804
# 4 Naive Bayes         0.928       0.624             0.776    0.697
# 5 SVM                 0.811       0.795             0.803    0.799
# 6 Neural Network      0.825       0.800             0.812    0.806
```