**FML FINAL PROJECT**

**GROUP – 19**

**INCOME CLASSIFICATION USING K-NN [ K NEAREST NEIGHBOURS] & NAÏVE BAYES ALGORITHMS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NAME** | **STUDENT ID** | **CONTRIBUTION** |
| 1 | Ajith Raj Periyasamy | 811316336 | * Data Collection * Data Cleaning * Descriptive Stats * K – NN * Naïve Bayes * Word Document |
| 2 | Tejasvin Maddineni | 811314004 | * Data Cleaning * Data Transformation * Result comparison * PPT |

**Project Goal:**

The goal of our project is to use Classification Machine Learning Algorithms [K-NN & Naïve Bayes] to group/classify people into income categories of >50k or <=50k. For this purpose, we have selected a dataset from [***Kaggle***](https://www.kaggle.com/datasets/tawfikelmetwally/census-income-dataset)**,** a public domain which that offers varieties of quality dataset for Analysis.

**Overview:**

The [***dataset***](https://kent.instructure.com/users/200282/files/15639671?wrap=1&verifier=g1iKheURGNKo1RqR1oOxI4PCU6afiXIr6JevErab) we have chosen for our analysis contains **32561**rows and **15**columns. Out of these 15 columns we have a mixture of **8** Categorical and **7** Numerical variables. The target variable in our dataset is Income. We aim to classify Income into 2 categories ‘<=50k’ or ‘>50k’. The procedure used for this purpose will be discussed further in the report.

**Data Exploration & Cleaning:**

1. **Missing Value treatment:**

The first method to perform before proceeding to any analysis is to Explore the data in hand. We started by exploring the data. We found that our dataset had missing values. Those missing values were identified as **‘?’.** We decided to replace these **‘?’** with **‘NA’** values. This replacement will ensure that we are able to deal with the missing values easily in **R.** The percentage of missing values were just **0.87%** of the dataset. Missing values below **5%** will not affect our analysis to a greater extent. Hence, we decided to ignore the missing values. We used the **na.omit()** function for the same.

1. **Relevant conversions:**

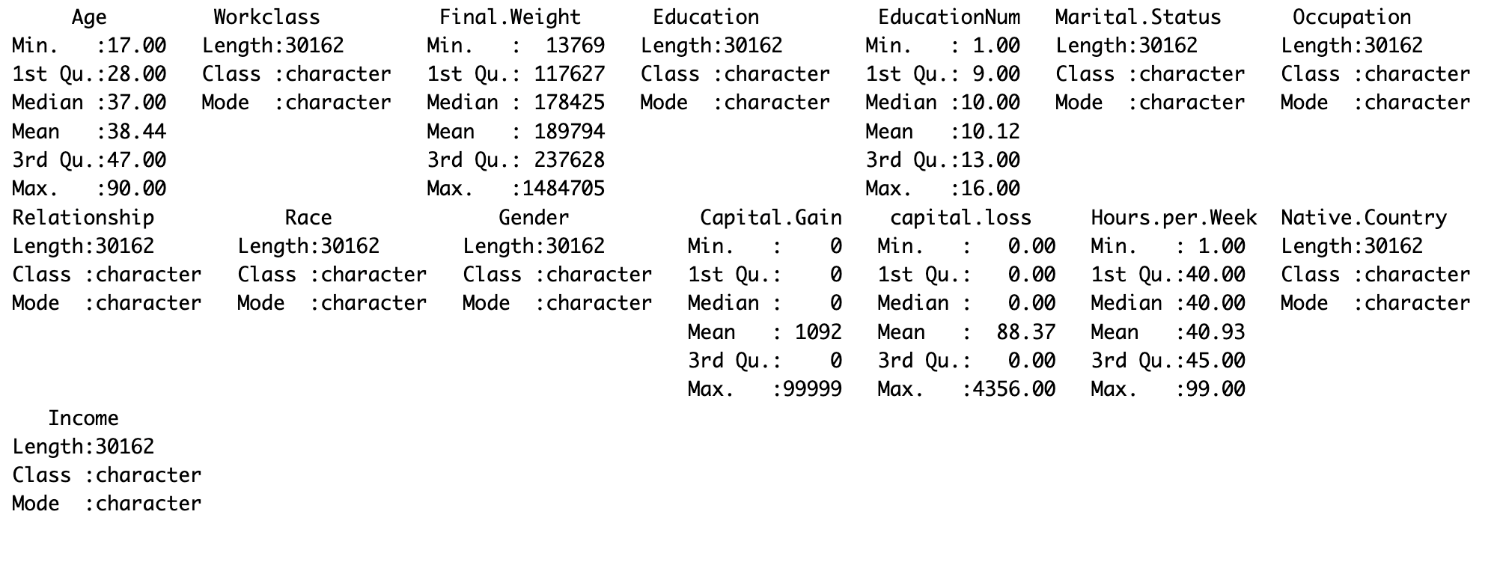
We identified that there were few columns [numerical variables] were of the datatype **‘character’.** We decided to convert these columns into numeric using the **as.numeric()** function for better analysis.

1. **Descriptive Statistics:**

Descriptive statistics provides concise summaries of a dataset's characteristics, helping in understanding the dataset by offering brief overviews and measurements of its features. For our project, we used 3 methods to perform the descriptive statistics.

* **Summary Statistics:**

To begin with, we used basic **summary() statistics** function to understand few basic characteristics such as **Minimum, Maximum, Quartiles, Median, Mean** values for each columns[variables] of the dataset. Here is the result of the summary statistics for our dataset.



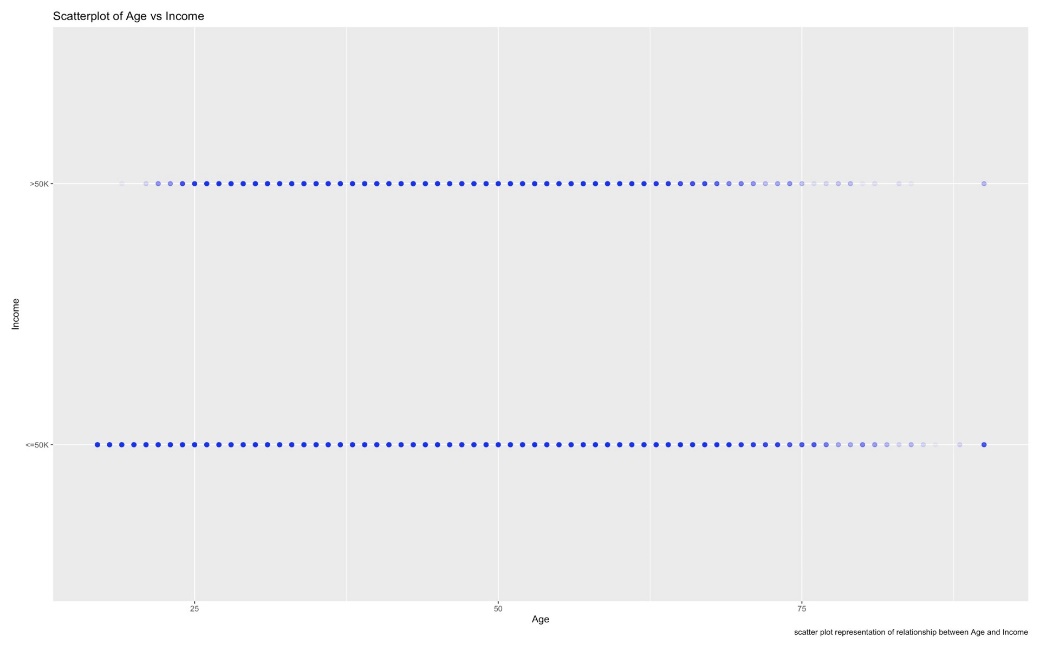
* **Scatter Plot:**

The scatter plot represents pairs of variables, i.e., one variable on each axis, aiming to identify any relationship between them. When variables are correlated, the plotted points tend to align along a line or curve. A stronger correlation results in points clustering more closely around this line. I attempted to plot the scatter plot for two variables:

**Target Variable** – Income<=50k, Income >50k;

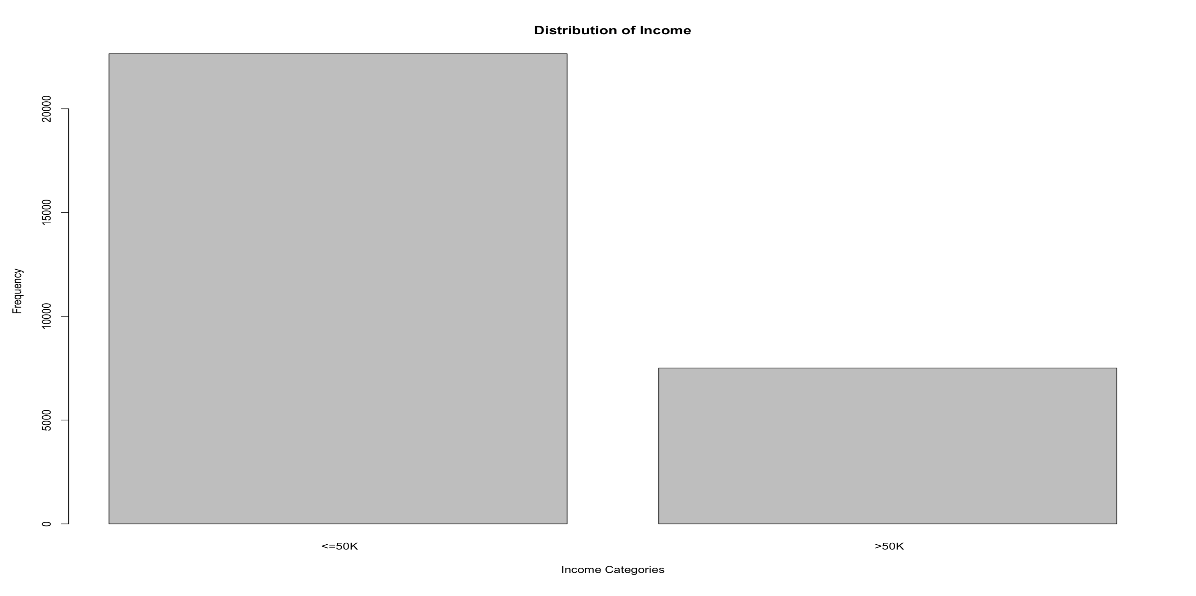
**Input Variable** – Age.

Based on the plot, it can be concluded that **Age** alone as a standalone variable cannot help in accurate predictions of that target variable – Income, rather **Age** along with few other variables might emerge as a good predictor of the target variable. Below is the scatter plot for our dataset.



* **Bar plot:**

Bar plots are an easy and convenient way to depict the distribution of the levels present in a particular variable. Here in our case, we have used the Bar plot to show the distribution between the Target Variable. i.e., to show how many people earn >50k and how many earn <=50k.



**Data Transformation:**

1. **One-Hot-Encoding**

As mentioned in the overview section of this report, our dataset had a mixture of categorical and numerical variables. We decided to convert all the categorical variables in to dummy variables. For this we used ‘**One-Hot-Encoding**’.

1. **Normalization:**

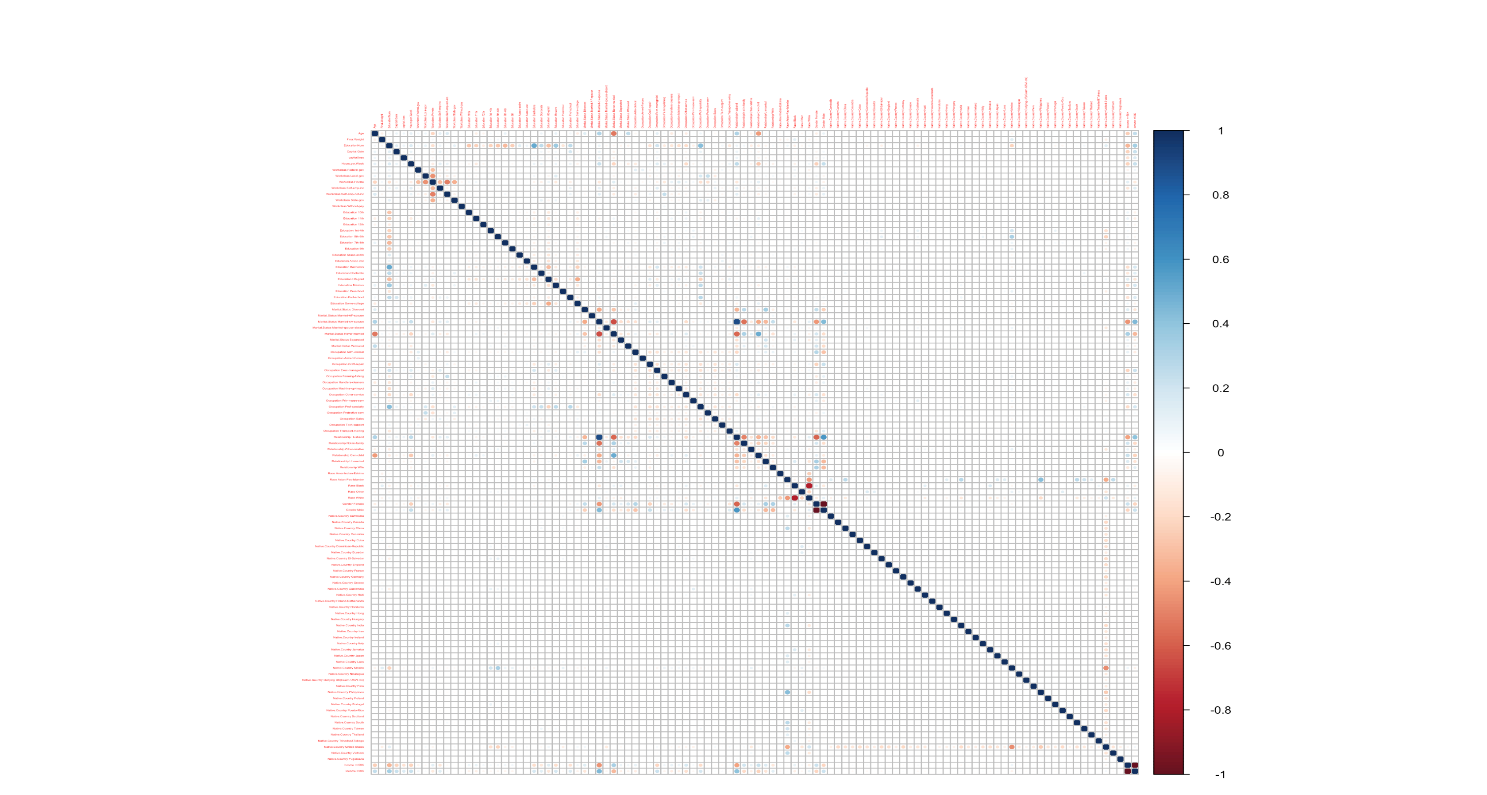
The data in our dataset belonged to different scales. In simple words they were of different units. Hence, we normalized the dataset using the ‘**preProcess()**’ function. We normalized the dataset using the **Min-Max** method. Normalization ensures that analysis can be done without any bias, since after normalization all the values transform to a common scale.

**Feature Selection:**

Feature selection is **the most important** part of our analytics project. During this process, we filter our input variables using certain methods, to predict our target variable. In simple terms, we perform feature selection to shortlist our most important variables for analysis.For feature selection we have used a corrplot, Backward Stepwise Regression, and PCA plot with all the variables and selected the most significant ones which are **Age, Hours. Per. Week, Work class, and Education.** Then after selecting the variables, we have used PCA, and Pair Matrix to depict the selected variables.

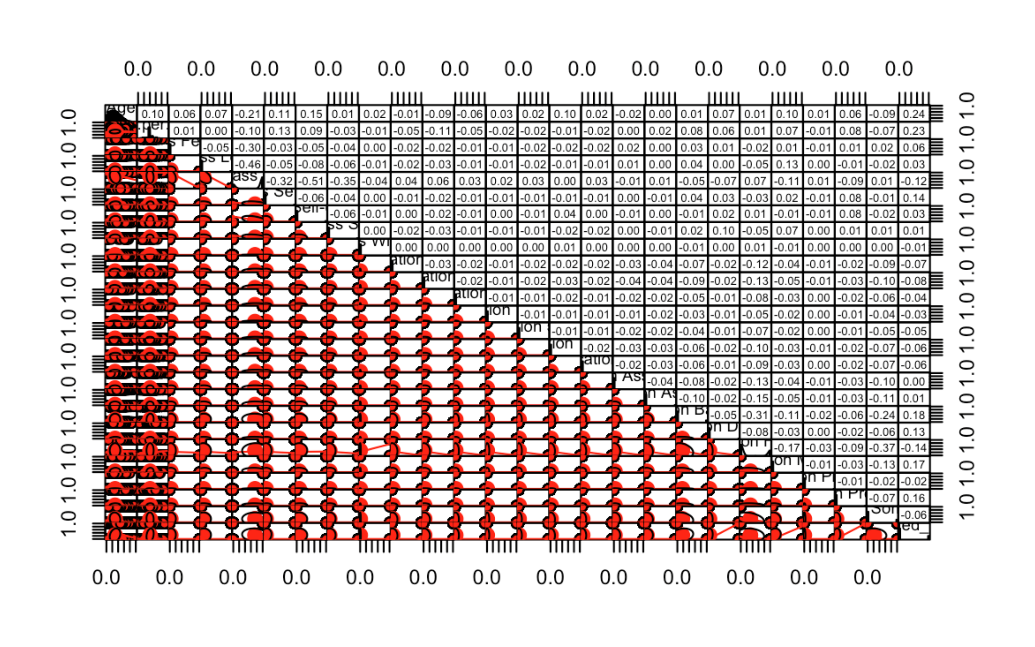
1. **Corrplot:**

Corrplot is a tool that creates pictures of correlation matrices, and it can automatically arrange variables to help spot patterns. Here we have plotted the correlation values of all variables to a corrplot and also towards the end we have plotted a pair matrix that has correlation values and other visual representation that confirms, our input variables - **Age, Hours. Per. Week, Work class, and Education.**



**Corrplot for all Variables**

**Pair Matrix for selected variables**

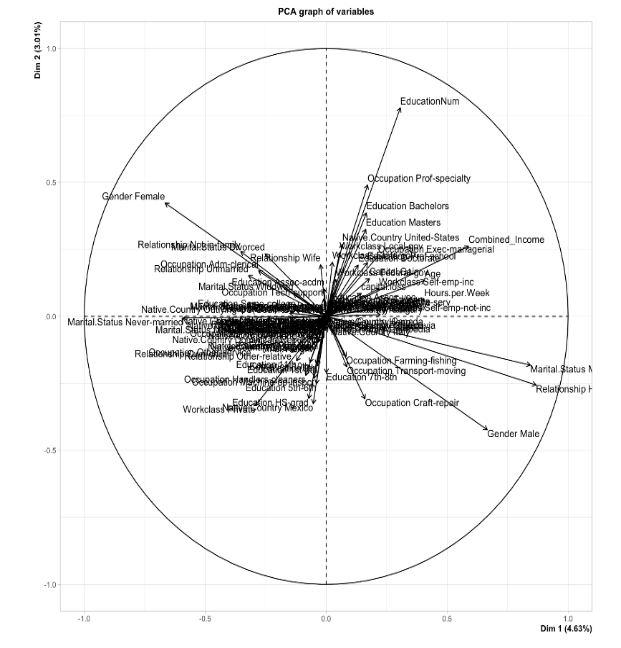
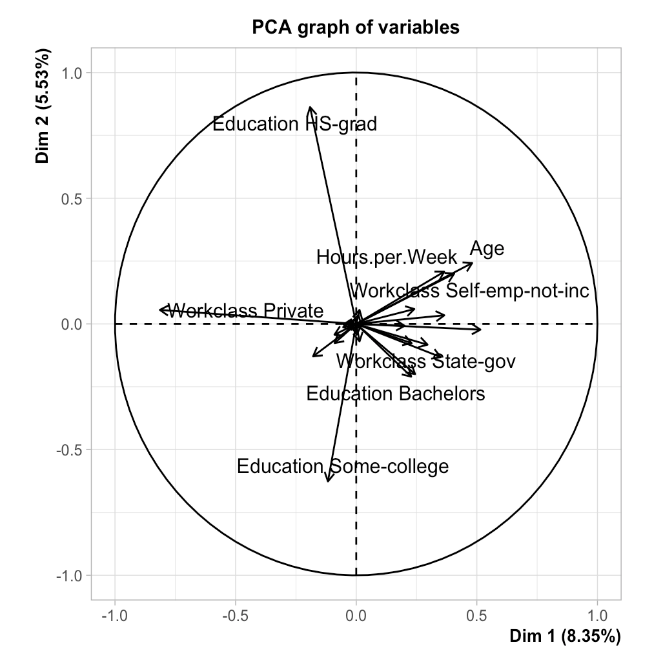


1. **Backward Stepwise Regression:**

Backward stepwise regression is a method that starts with a complete model including all possible predictors and progressively removes them one by one. This approach helps find a simpler model that still effectively explains the data. It's beneficial because it decreases the number of predictors, which helps minimize issues like multicollinearity and overfitting. The result from this method conveyed multiple variables were significant for the analysis of the target variable. We decided to stick to 4 variables – **Age, Hours.Per.Week, Work class, and Eduation** as they were the resultant variables from our correlation analysis as well.

1. **PCA:**

Principal Component Analysis (PCA) is an effective method employed in data analysis, specifically aimed at decreasing the complexity of datasets while retaining essential information. This is achieved by converting the initial variables into a series of fresh, independent variables known as principal components. We have done PCA for all variables, selected the ones needed for our analysis, and also plotted a PCA for the selected variables.

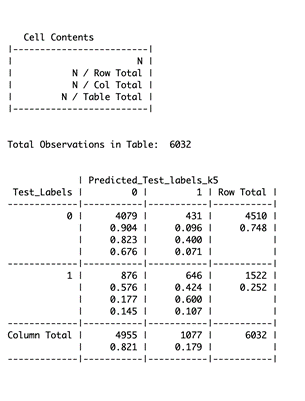
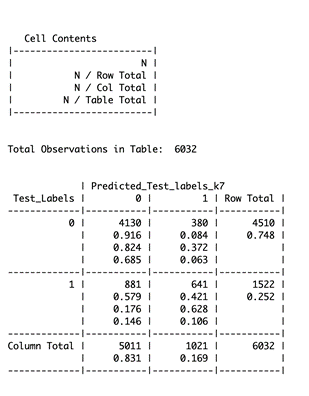
**PCA – SELECTED VARIABLES**

**PCA – ALL VARIABLES**

**Hyper-parameter tuning & K-NN Classification:**

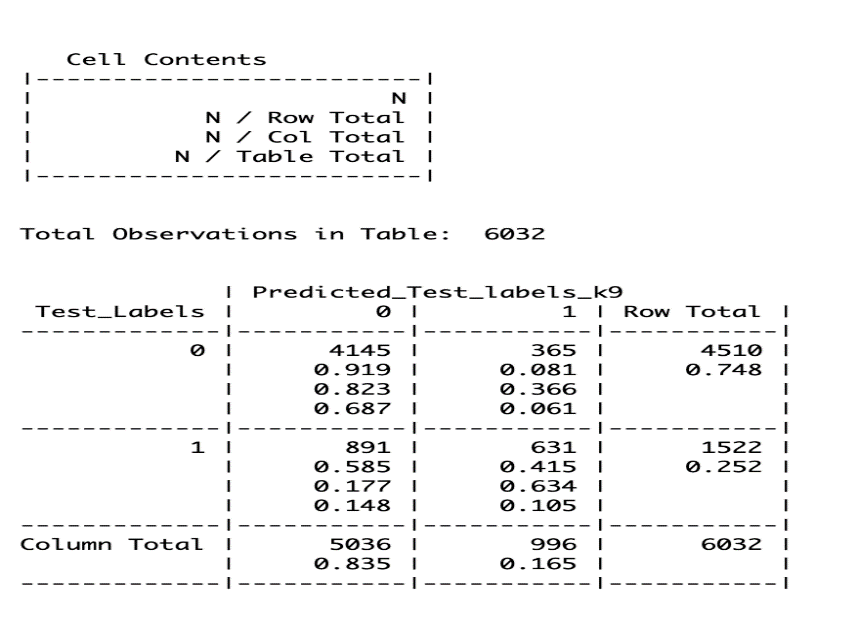
To find out the best **‘k’** value, we used the Grid search method and performed hyper-parameter tuning. We used the **train()** function and set the method to **knn** to find out the best k value through Grid Search. As a result of this, we arrived at three ‘k’ values, i.e., k = 5; k = 7; k = 9.

We then moved on to perform the K-NN classification using the 3 different ‘k’ values we arrived at as a result of our grid search. We tested the model’s performance with all 3 ‘k’ values. After this, we calculated the important metrics – **Accuracy, Recall, Precision, and Specificity** to find out the best ‘k’ value. K = 9 has the highest accuracy, specificity, and precision compared to the other k values, indicating **that k = 9 is the optimum k value** for this K-NN classification analysis.

**K = 5 K = 7**

1. Accuracy – 78.38% 1. Accuracy – 79.07%
2. Recall – 42.42% 2. Recall – 42.11%
3. Precision – 60.02% 3. Precision – 62.77%
4. Specificity – 90.44% 4. Specificity – 91.59%



**K = 9**

1. Accuracy – 79.14%
2. Recall – 41.40%
3. Precision – 63.31%
4. Specificity – 91.91%

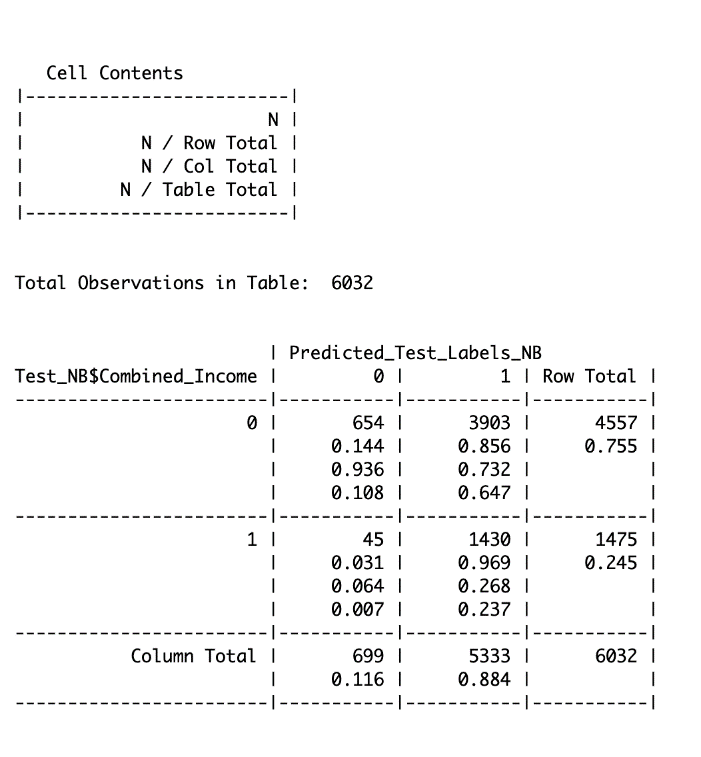
**Naïve Bayes Classification:**

Naive Bayes classifier is a supervised machine learning algorithm used for most of the classification tasks. They make use of principles of probability to perform classification tasks.

**Assumptions of Naïve Bayes Classifier:**

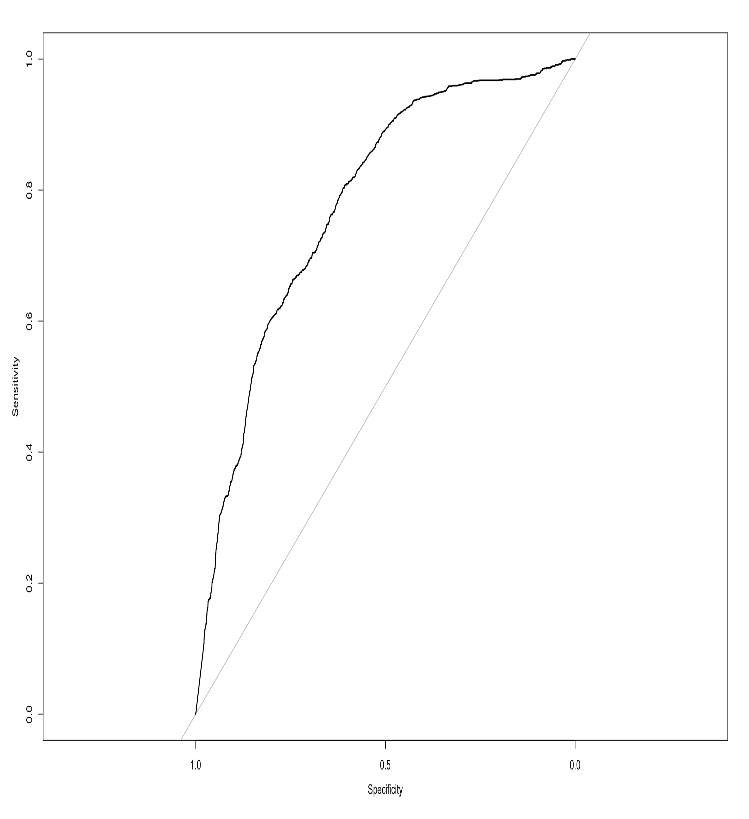
1. **Independence Assumption**: In our analysis, Naive Bayes assumes that the age of a person, the number of hours they work per week, their work class (like private sector, government employee), and their education level (like high school diploma, Bachelor's degree) all independently contribute to predicting their income level. So, the model will treat the probability of someone working a certain number of hours as completely unrelated to their age, work class, or education when it comes to their income. In reality, these variables might be related (for example, those with higher education might work more hours), but Naive Bayes ignores these potential relationships.
2. **Likelihood of Features**: For each feature, Naive Bayes will calculate the likelihood given each income class. For instance, it will estimate the probability that someone works, say, over 40 hours a week given that they earn more than 50K, and compare that to the probability of working the same hours for those earning 50K or less. Similarly, it will look at how likely someone with a Bachelor’s degree is to earn more than 50K versus 50K or less. These probabilities are used to make a prediction about income level for each individual in the dataset.
3. **Feature Relevance**: Naive Bayes in our analysis will assume that all of these factors—age, hours per week, work class, and education—are important in determining whether someone’s income is above or below 50K. For example, it assumes that knowing a person's work class is just as crucial for predicting their income as knowing their education level, even though in reality, some of these features might be more strongly related to income than others. It won't inherently weigh one factor more heavily than another unless the data suggests it should be based on the likelihood calculations.

The following visuals – confusion matrix, and the 4 important metrics [accuracy, precision, recall, and specificity] depict the results of Naïve Bayes Classification.



1. Accuracy – 34.50%
2. Recall – 96.90%
3. Precision – 26.80%
4. Specificity – 14.30 %

**Confusion Matrix - Naïve Bayes Classifer**



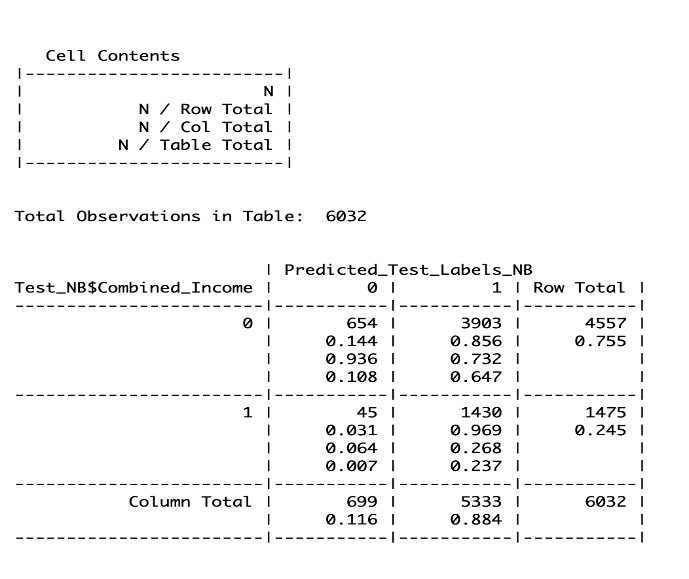
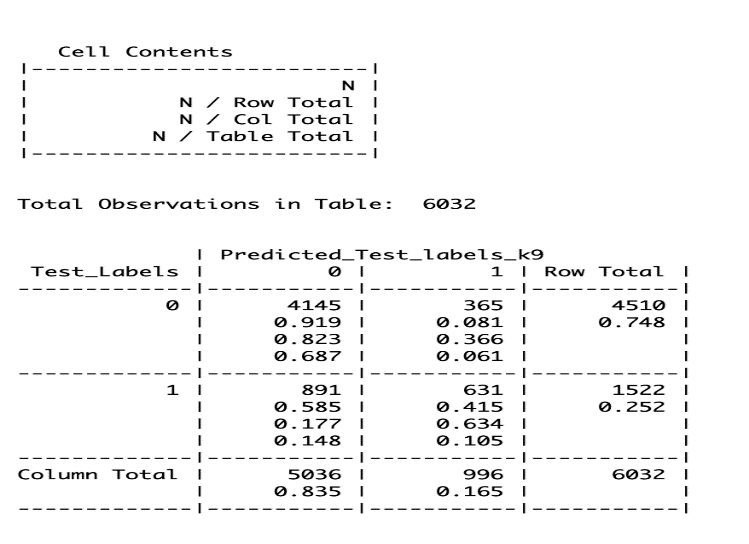
**Interpretation**

The closer the curve to the top-left corner, the higher the AUC, and the better the overall performance of the classifier.

**ROC – Naïve Bayes Classifier**

**K-NN vs Naïve Bayes Classification:**

After successful building up and execution of our model, we now have to compare the results of the two algorithms – K-NN and Naïve Bayes, to find out which algorithm’s prediction is more accurate and suitable for the analysis of our dataset. We aim to establish this comparison through Confusion matrix for best k value **[k = 9]** and the **Naïve Bayes results**.



**Naïve Bayes Classifier**

**K = 9**

* 1. Accuracy – 79.14%

1. Accuracy – 34.50%
2. Recall – 96.90%
3. Precision – 26.80%
4. Specificity – 14.30 %
   1. Recall – 41.40%
   2. Precision – 63.31%
   3. Specificity – 91.91%

**Conclusion:**

From the above comparison it is evident that K=9 has the best accuracy, precision, and specificity when compared to the Naïve Bayes Classification. Hence, we can conclude that for this particular dataset, **K-NN [with k=9] works better** than the Naïve Bayes Algorithm.