University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

IS6052: Descriptive and Predictive Analytics Report

Created by: Jayesh Prakash Rao (120220275)

Reviewer: Prof. Armagan Tarim

Table of Contents

Executive Summary

ABSA bank sales team is looking for insights for making informed decisions for marketing their array of investment products which include bonds, stocks and derivatives. There are time and budget constraints on marketing the products that will allow only potential customers to be contacted for a successful purchase. As a marketing analyst, a dataset of past customer records 'Market.csv' has been provided for analysis of various trends and insights supporting successful purchases. Another file 'Market_pred.csv' has been provided that consists of the customers for which the purchase predictions need to be made. These predictions will help the sales team contact only those who have higher chances of purchasing the investment products.

The initial part of the report consists of exploratory analysis which is eyeballing of the available data to understand the types of variables present. The dataset is also checked for missing and null values such that a clean dataset is available for further analysis. The descriptive analysis involves detecting and studying various patterns and trends by observing the distribution of features of the dataset using descriptive statistics and visualizations using graphs. The dataset is then prepared for predictive analytics for which it is randomly divided into training a testing dataset. Based on the exploratory and descriptive analysis, it is assumed to be a classification problem for which the following machine learning models are considered

1. Classification and Regression Tree (CART)
2. C5.0
3. Random Forest
4. Naïve Bayes
5. Support Vector Machine (SVM)

The above models are initially compared based on their common pros and cons. Each model is then trained using the training dataset and then tested using the testing dataset in R code. The test dataset predictions by the models are then evaluated based on the prediction accuracy, precision, false positive and false negative counts outputted by the confusion matrix. A comparative analysis is then performed on the above parameters generated by each model to decide the final model of selection for the classification problem. K-fold cross-validation is also performed over the top-performing models to eliminate any biases or overfitting that might have caused the models to display high accuracy. The final classification model selected as a result is then used for predicting the 'Market_pred.csv' dataset. A set of recommendations are provided to the ABSA marketing team about the set of customers that have a high chance of purchasing the investment products. The additional parameters that could help make better predictions are also mentioned at the end of the analysis report.

Problem Statement

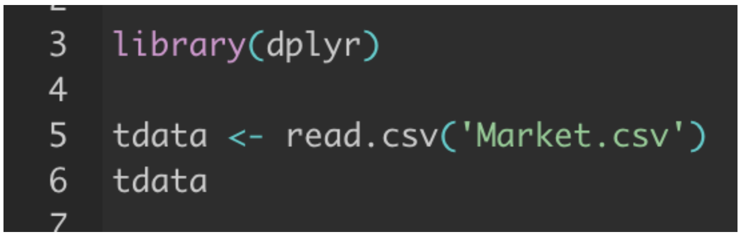
ABSA sales force is looking for potential customers for their range of investment products which includes derivatives, bonds and stocks. Based on past records of customer sales for the aforementioned products, sales prediction is required as an input for targeted marketing for the sales team. This sales prediction will assist the team in targeting the customers with the products who are more likely to buy them, thereby saving time and resources. This analytics report consists of exploring the past customer records, processing the data, performing descriptive analytics, evaluating the predictive algorithms and performing a comparative analysis of the predictions from each of them.

Exploratory and Descriptive Analysis of the dataset

There are two datasets, 'Market.csv' and 'Market_pred.csv' available for our analysis. 'Market.csv' consists of past customer sales records which will be used for the analysis of trends and training the prediction models. 'Market_pred.csv' contains the target customers on which prediction has to be generated by using the model built using the past customer records. The 'Market.csv' dataset is first imported in R as follows.

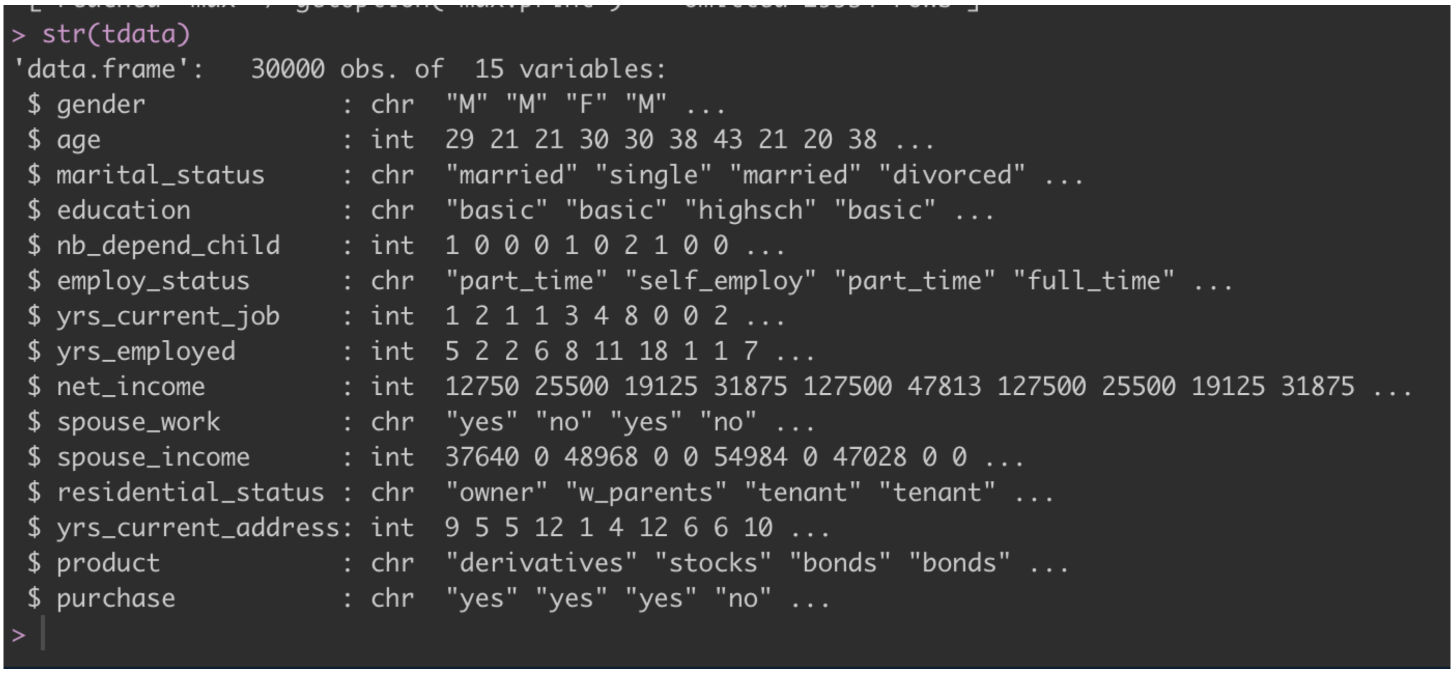
Importing the Data Set

The read.csv() function is used to read the 'Market.csv' file from the system. The data is converted to data frame and is stored in the variable tdata.



```r
library(dplyr)

tdata <- read.csv('Market.csv')
tdata
```

The structure of the data frame imported is observed using the str() function.

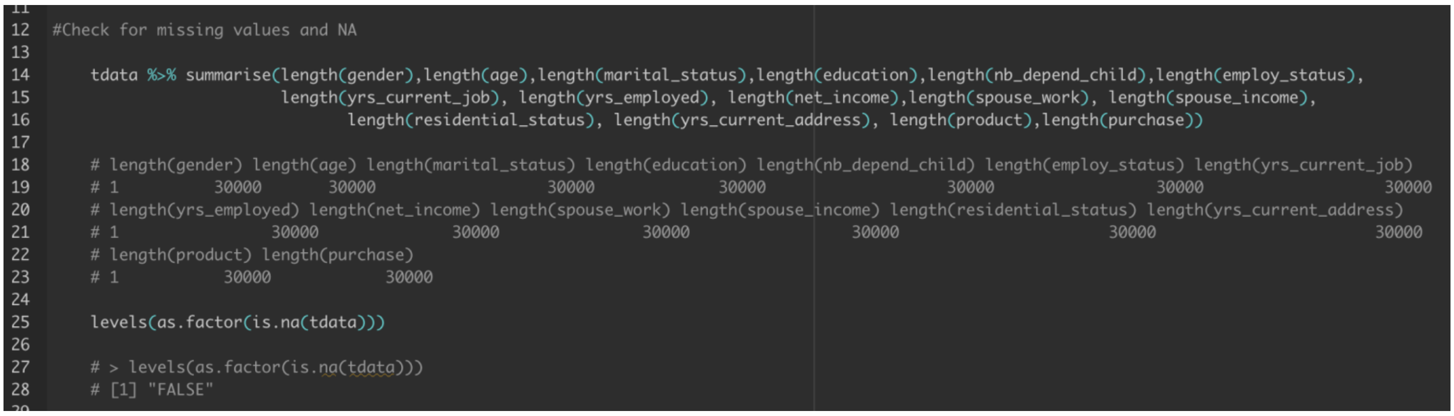


```
> str(tdata)
'data.frame':	30000 obs. of  15 variables:
 $ gender             : chr  "M" "M" "F" "M" ...
 $ age                : int  29 21 21 30 30 38 43 21 20 38 ...
 $ marital_status     : chr  "married" "single" "married" "divorced" ...
 $ education          : chr  "basic" "basic" "highsch" "basic" ...
 $ nb_depend_child    : int  1 0 0 0 1 0 2 1 0 0 ...
 $ employ_status      : chr  "part_time" "self_employ" "part_time" "full_time" ...
 $ yrs_current_job    : int  1 2 1 1 3 4 8 0 0 2 ...
 $ yrs_employed       : int  5 2 2 6 8 11 18 1 1 7 ...
 $ net_income         : int  12750 25500 19125 31875 127500 47813 127500 25500 19125 31875 ...
 $ spouse_work        : chr  "yes" "no" "yes" "no" ...
 $ spouse_income      : int  37640 0 48968 0 0 54984 0 47028 0 0 ...
 $ residential_status : chr  "owner" "w_parents" "tenant" "tenant" ...
 $ yrs_current_address: int  9 5 5 12 1 4 12 6 6 10 ...
 $ product            : chr  "derivatives" "stocks" "bonds" "bonds" ...
 $ purchase           : chr  "yes" "yes" "yes" "no" ...
>
```

The output shows that there are 30000 rows and 15 variables or dimensions present in the data frame tdata. Among the 15 variables, 8 are character type while the remaining 7 are integer type.

Checking for Missing and NA values

The records are now checked for missing values and NA by using the summarise() function to display the length of all the columns. The length of individual columns must be equal to the total records observed for the entire data frame to confirm the absence of missing values.



```
#Check for missing values and NA

    tdata %>% summarise(length(gender),length(age),length(marital_status),length(education),length(nb_depend_child),length(employ_status),
                        length(yrs_current_job), length(yrs_employed), length(net_income),length(spouse_work), length(spouse_income),
                               length(residential_status), length(yrs_current_address), length(product),length(purchase))

    # length(gender) length(age) length(marital_status) length(education) length(nb_depend_child) length(employ_status) length(yrs_current_job)
    # 1          30000       30000                  30000             30000                   30000                 30000                   30000
    # length(yrs_employed) length(net_income) length(spouse_work) length(spouse_income) length(residential_status) length(yrs_current_address)
    # 1                30000              30000               30000                 30000                      30000                        30000
    # length(product) length(purchase)
    # 1           30000            30000

    levels(as.factor(is.na(tdata)))

    # > levels(as.factor(is.na(tdata)))
    # [1] "FALSE"
```

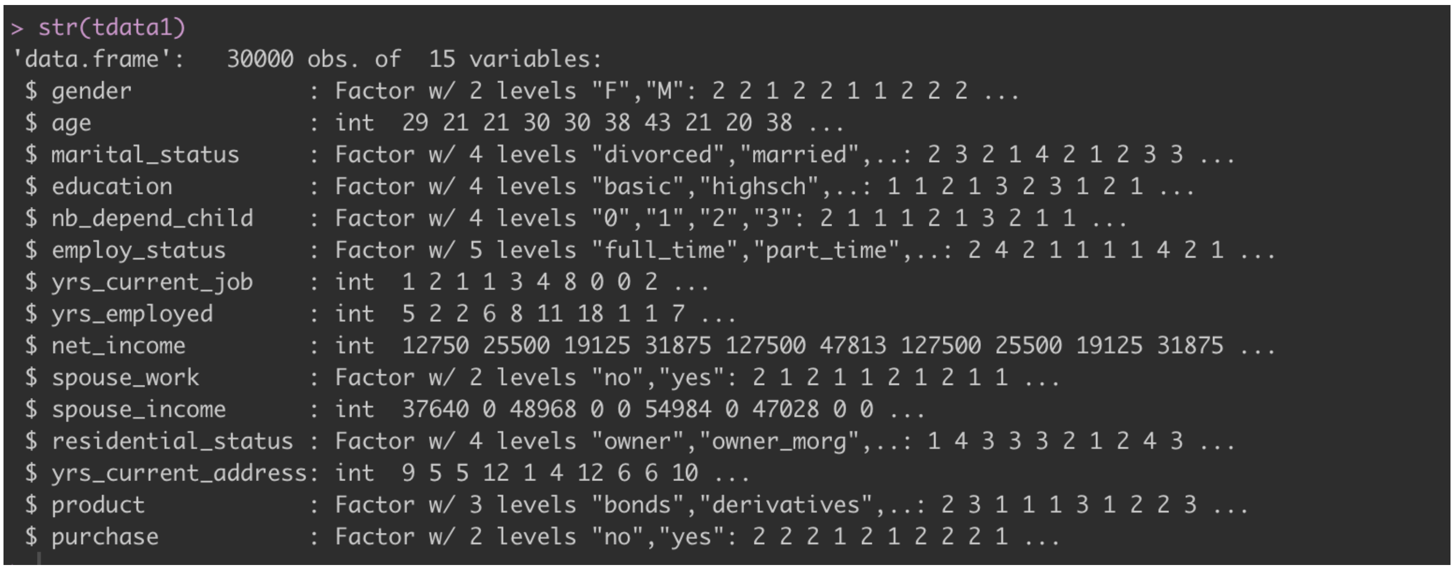
The number of values in each column is 30000 which confirms that there are no missing values. The check for NA in the entire data frame returns 'FALSE' which means that there are no NA values in the entire data frame.

Assumptions

The columns *gender, marital_status, education, nb_depend_child, employ_status, spouse_work, residential_status, product* and *purchase* are assumed to be categories after observing the number of levels/ unique values in the columns by using the levels() function.

Converting Character Columns to Categories

A copy of the tdata data frame is created as tdata1 for modification of the column types. The following R code uses mutate() function to modify the data frame. The factor() will individually convert the columns mentioned above from character type to factor. The str() function is then used to check the updated columns.

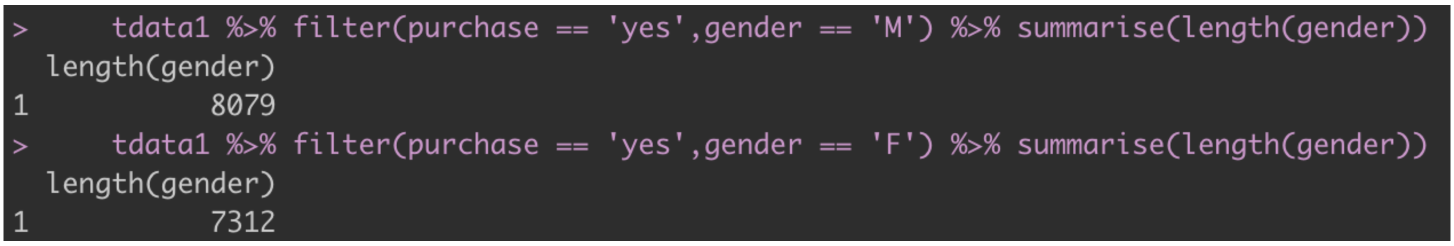


```
> str(tdata1)
'data.frame':	30000 obs. of  15 variables:
 $ gender             : Factor w/ 2 levels "F","M": 2 2 1 2 2 1 1 2 2 2 ...
 $ age                : int  29 21 21 30 30 38 43 21 20 38 ...
 $ marital_status     : Factor w/ 4 levels "divorced","married",..: 2 3 2 1 4 2 1 2 3 3 ...
 $ education          : Factor w/ 4 levels "basic","highsch",..: 1 1 2 1 3 2 3 1 2 1 ...
 $ nb_depend_child    : Factor w/ 4 levels "0","1","2","3": 2 1 1 1 2 1 3 2 1 1 ...
 $ employ_status      : Factor w/ 5 levels "full_time","part_time",..: 2 4 2 1 1 1 1 4 2 1 ...
 $ yrs_current_job    : int  1 2 1 1 3 4 8 0 0 2 ...
 $ yrs_employed       : int  5 2 2 6 8 11 18 1 1 7 ...
 $ net_income         : int  12750 25500 19125 31875 127500 47813 127500 25500 19125 31875 ...
 $ spouse_work        : Factor w/ 2 levels "no","yes": 2 1 2 1 1 2 1 2 1 1 ...
 $ spouse_income      : int  37640 0 48968 0 0 54984 0 47028 0 0 ...
 $ residential_status : Factor w/ 4 levels "owner","owner_morg",..: 1 4 3 3 3 2 1 2 4 3 ...
 $ yrs_current_address: int  9 5 5 12 1 4 12 6 6 10 ...
 $ product            : Factor w/ 3 levels "bonds","derivatives",..: 2 3 1 1 1 3 1 2 2 3 ...
 $ purchase           : Factor w/ 2 levels "no","yes": 2 2 2 1 2 1 2 2 2 1 ...
```

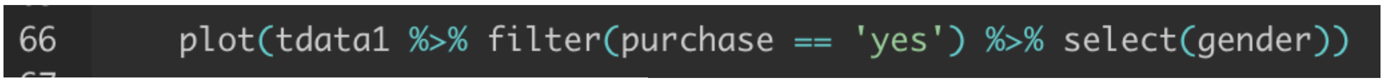
Analysis of the Categorical Columns

This section analyses the categorical columns of the dataset. The records of successful purchases are selected for the analysis to understand the impact each variable has on the purchase decision. In the below R code, the records are filtered using condition such that the purchase is ‘yes’ and the categorical columns are selected individually to observe the bar graphs of the levels or categories. This selection of data is done using the pipe function in R. The following bar charts are then plotted to show the count present in each category.

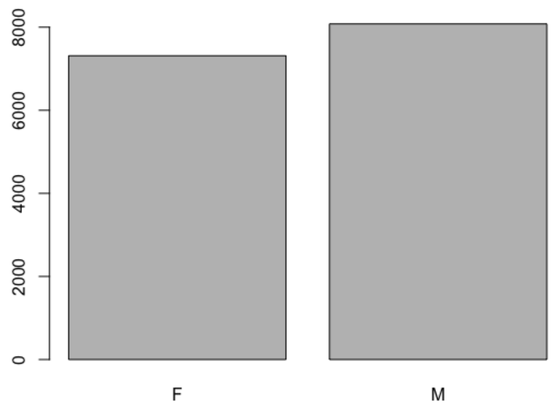
Gender



```
>     tdata1 %>% filter(purchase == 'yes',gender == 'M') %>% summarise(length(gender))
  length(gender)
1           8079
>     tdata1 %>% filter(purchase == 'yes',gender == 'F') %>% summarise(length(gender))
  length(gender)
1           7312
```

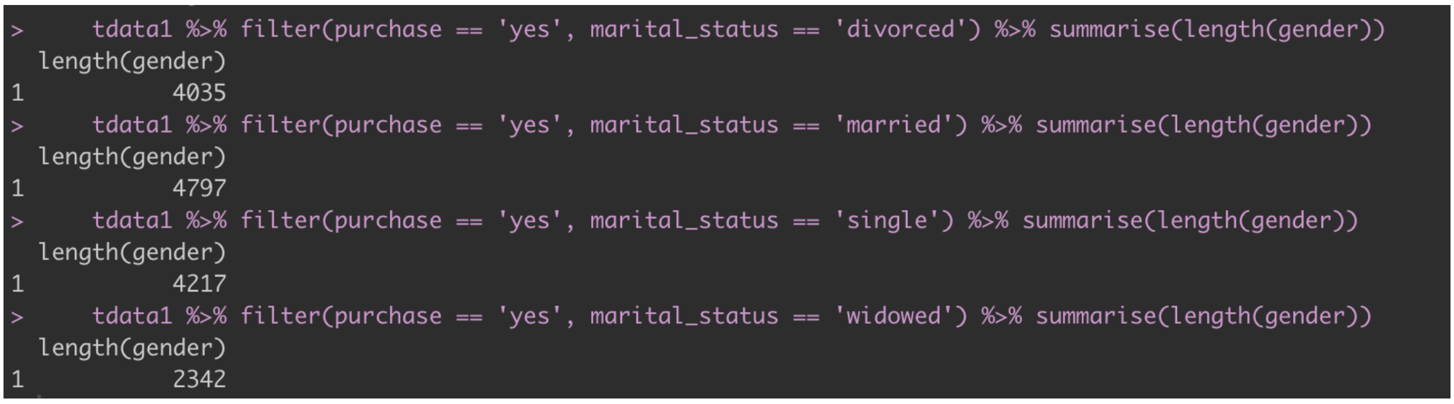


```
      plot(tdata1 %>% filter(purchase == 'yes') %>% select(gender))
```

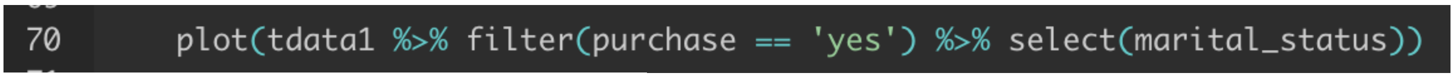


It is observed that from the individuals who have purchased the investment products, 8079 are males and 7312 are females. There is no significant difference between the purchase success based on gender.

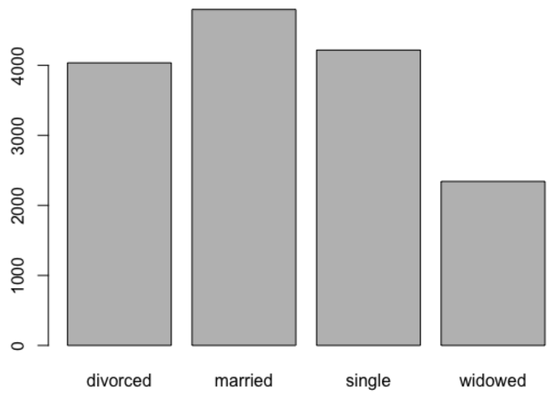
marital_status



```
>     tdata1 %>% filter(purchase == 'yes', marital_status == 'divorced') %>% summarise(length(gender))
  length(gender)
1           4035
>     tdata1 %>% filter(purchase == 'yes', marital_status == 'married') %>% summarise(length(gender))
  length(gender)
1           4797
>     tdata1 %>% filter(purchase == 'yes', marital_status == 'single') %>% summarise(length(gender))
  length(gender)
1           4217
>     tdata1 %>% filter(purchase == 'yes', marital_status == 'widowed') %>% summarise(length(gender))
  length(gender)
1           2342
```

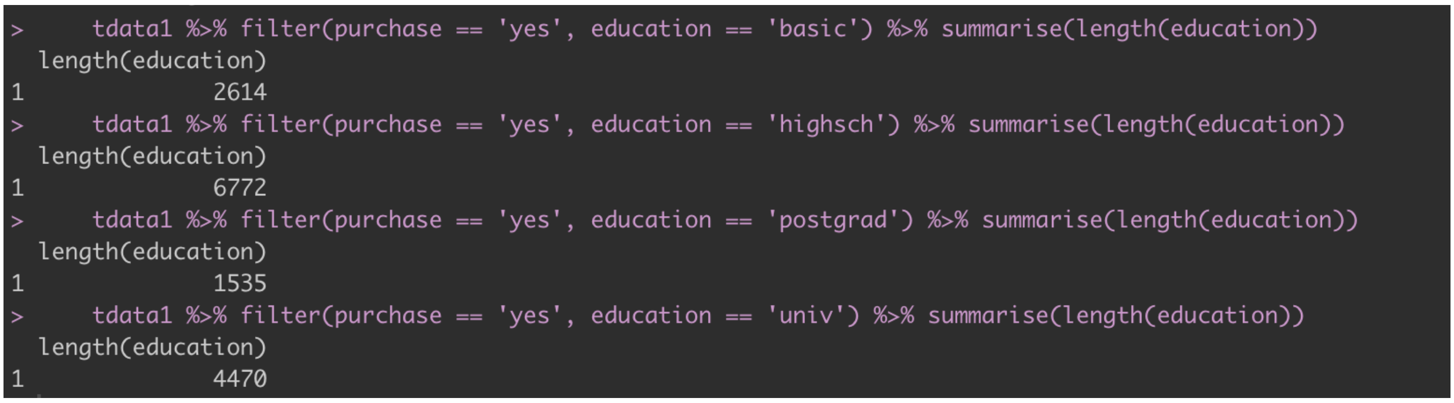


```
plot(tdata1 %>% filter(purchase == 'yes') %>% select(marital_status))
```

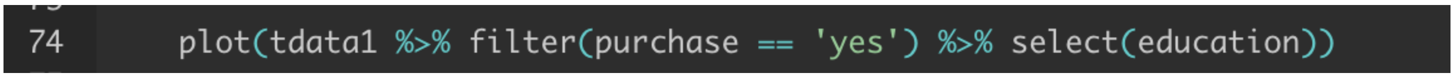


It is observed from the above comparison that the number of married customers who purchased the investment product is the highest at 4797, as marriage translates to significant financial responsibilities of the family. The second highest purchases are done by singles as they mostly belong to the risk-taking age groups as observed previously. The divorced individuals are assumed to have child care or additional financial responsibilities which influence investment purchases. The widowed group have the least purchases as observed as they are assumed to have the least financial responsibilities compared to the other groups.

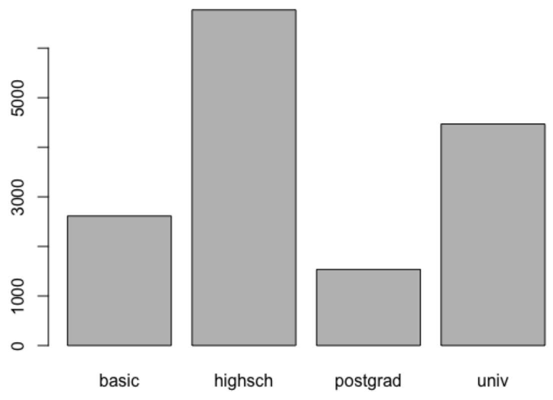
Education



```
>     tdata1 %>% filter(purchase == 'yes', education == 'basic') %>% summarise(length(education))
  length(education)
1              2614
>     tdata1 %>% filter(purchase == 'yes', education == 'highsch') %>% summarise(length(education))
  length(education)
1              6772
>     tdata1 %>% filter(purchase == 'yes', education == 'postgrad') %>% summarise(length(education))
  length(education)
1              1535
>     tdata1 %>% filter(purchase == 'yes', education == 'univ') %>% summarise(length(education))
  length(education)
1              4470
```

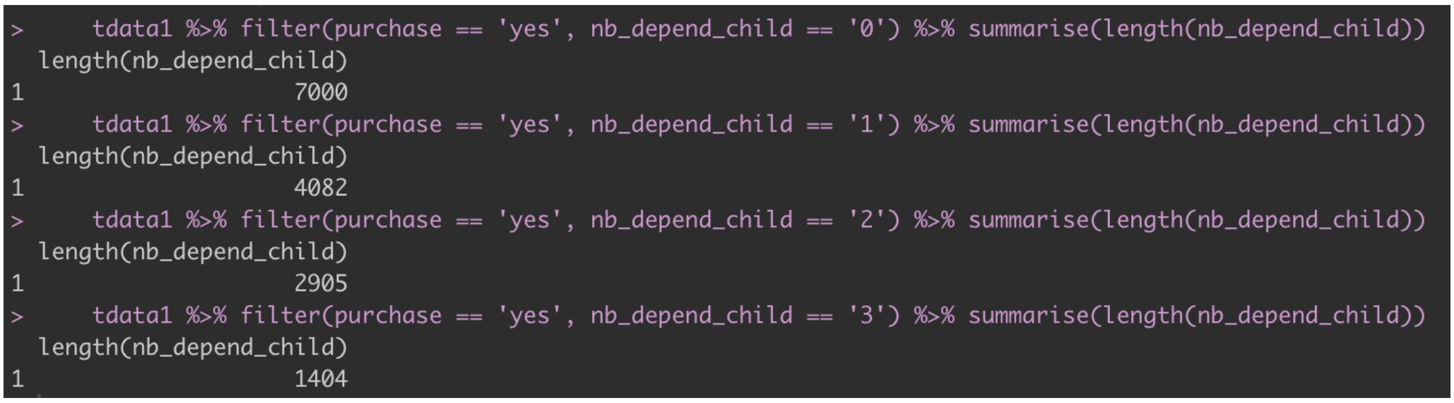


```
plot(tdata1 %>% filter(purchase == 'yes') %>% select(education))
```

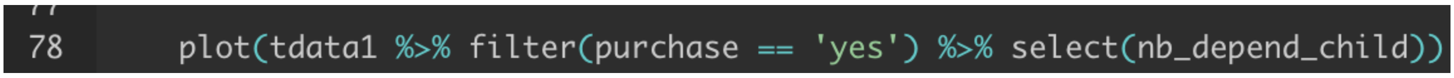


It is observed that the maximum number of customers who have purchased the products have an education level of high school which is at 6772 while customers having the postgrad level is the lowest at a meagre 1535. A distant second highest purchases are made by customers having a university degree and customers having a basic education amount to 2614. It is difficult to establish a reasoning which explains the purchase variations based on these education categories, however the field of education like finance, arts, accounts would be an interesting parameter to observe.

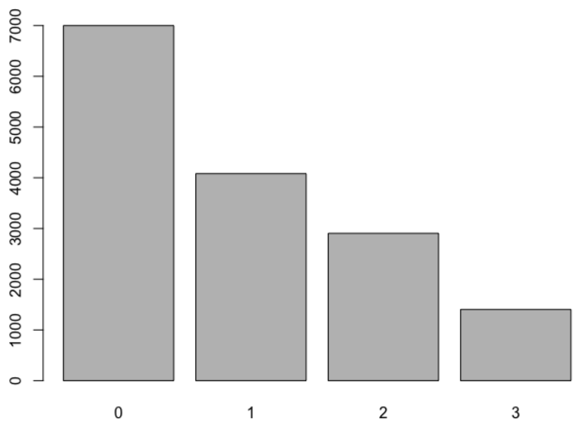
nb_depend_child



```
>     tdata1 %>% filter(purchase == 'yes', nb_depend_child == '0') %>% summarise(length(nb_depend_child))
  length(nb_depend_child)
1                    7000
>     tdata1 %>% filter(purchase == 'yes', nb_depend_child == '1') %>% summarise(length(nb_depend_child))
  length(nb_depend_child)
1                    4082
>     tdata1 %>% filter(purchase == 'yes', nb_depend_child == '2') %>% summarise(length(nb_depend_child))
  length(nb_depend_child)
1                    2905
>     tdata1 %>% filter(purchase == 'yes', nb_depend_child == '3') %>% summarise(length(nb_depend_child))
  length(nb_depend_child)
1                    1404
```

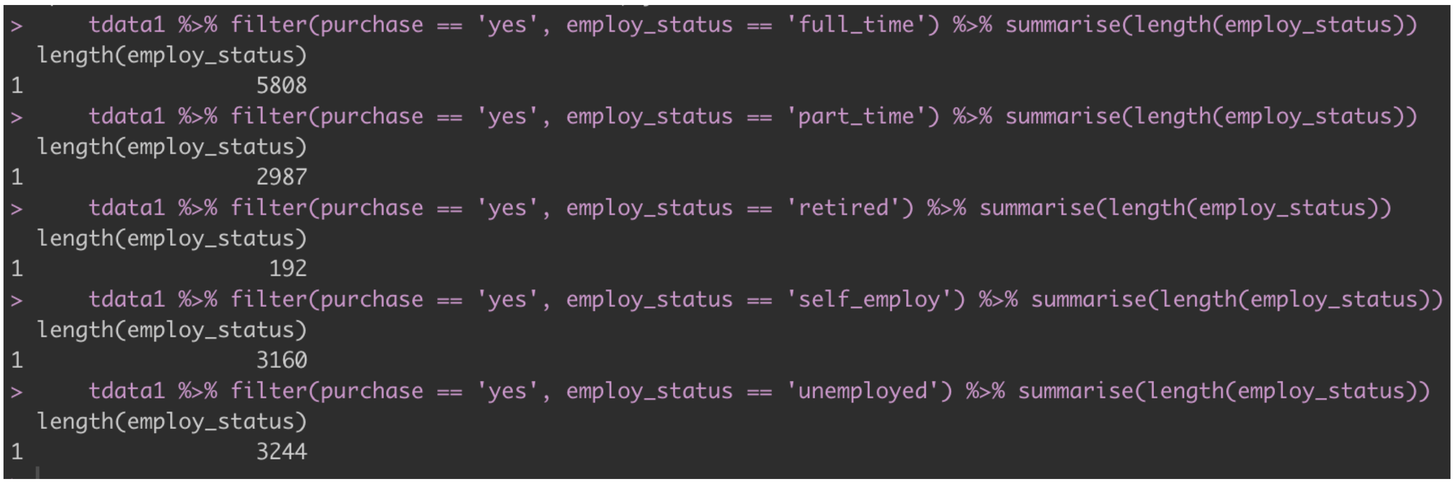


```
    plot(tdata1 %>% filter(purchase == 'yes') %>% select(nb_depend_child))
```

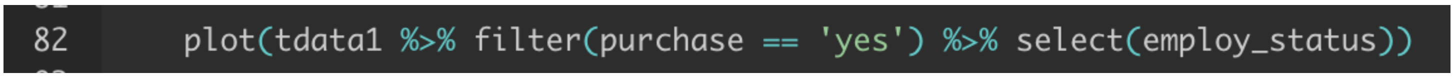


A decreasing trend is observed in the purchases made with increasing depending children. The customers having 0 dependent children can be assumed to be of two groups of single and newly married. There can be a high-risk taking capability during the stages of low financial responsibilities. These group of individuals are likely to invest in volatile investment products compared to individuals having children. As the number of dependent children increases the risk-taking ability decreases and hence the decreasing trend.

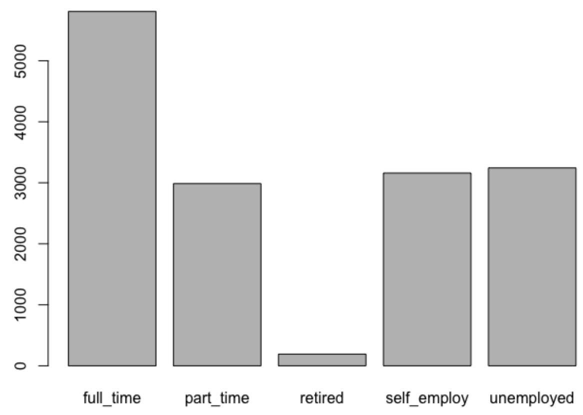
Employ_status



```
>     tdata1 %>% filter(purchase == 'yes', employ_status == 'full_time') %>% summarise(length(employ_status))
  length(employ_status)
1                  5808
>     tdata1 %>% filter(purchase == 'yes', employ_status == 'part_time') %>% summarise(length(employ_status))
  length(employ_status)
1                  2987
>     tdata1 %>% filter(purchase == 'yes', employ_status == 'retired') %>% summarise(length(employ_status))
  length(employ_status)
1                   192
>     tdata1 %>% filter(purchase == 'yes', employ_status == 'self_employ') %>% summarise(length(employ_status))
  length(employ_status)
1                  3160
>     tdata1 %>% filter(purchase == 'yes', employ_status == 'unemployed') %>% summarise(length(employ_status))
  length(employ_status)
1                  3244
```

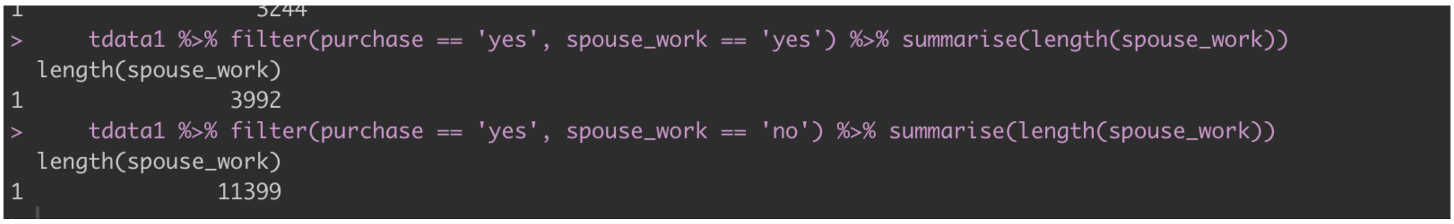


```
      plot(tdata1 %>% filter(purchase == 'yes') %>% select(employ_status))
```

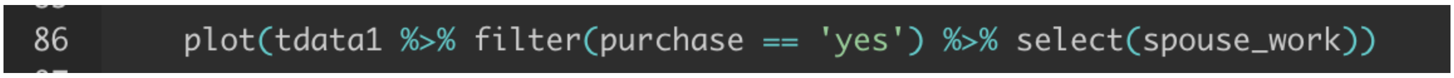


The purchases made by full time employed customers are significantly higher compared to others at 5808. This can be explained by job security. A full-time employee has a steady job and income at all times which guarantees a cover for highs and lows of investment returns. The retired individuals having the lowest investment purchases can be explained by the low cash inflow at that age groups. Self-employed, part time and unemployed individuals do not have a steady or considerable income to assign to investment portfolios and therefore have significantly lower purchases to individuals having a full-time job, however not much variations are observed among themselves.

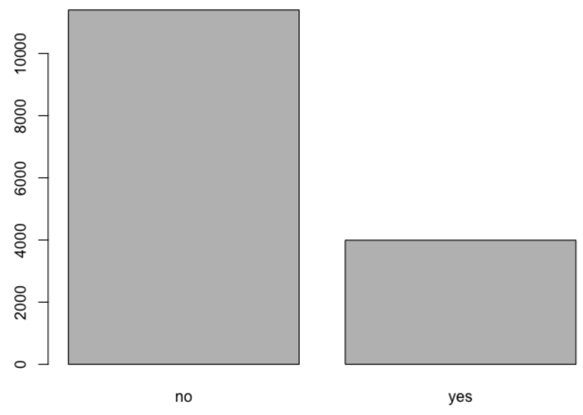
Spouse_work



```
>     tdata1 %>% filter(purchase == 'yes', spouse_work == 'yes') %>% summarise(length(spouse_work))
  length(spouse_work)
1                3992
>     tdata1 %>% filter(purchase == 'yes', spouse_work == 'no') %>% summarise(length(spouse_work))
  length(spouse_work)
1               11399
```

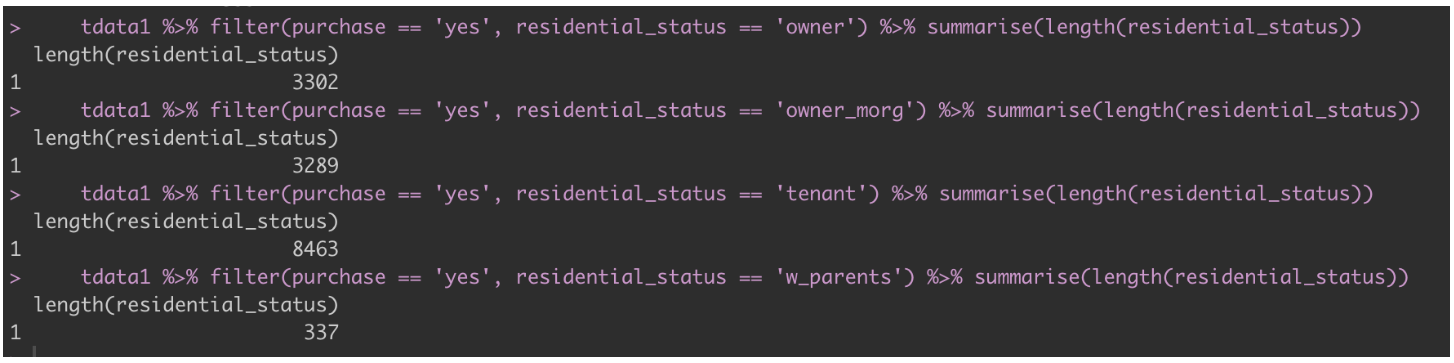


```
      plot(tdata1 %>% filter(purchase == 'yes') %>% select(spouse_work))
```

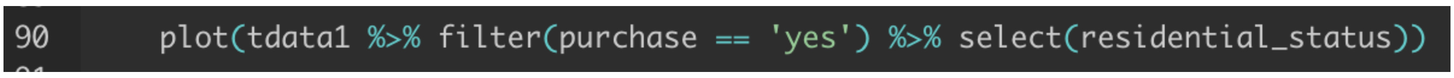


The highest investment purchases are done by individuals whose spouses are not working at 11399 while purchases by individuals whose spouses are working is significantly lower at 3992. This can be explained as the individuals who are single earners of the family observe a higher need to invest money to cover future uncertainties.

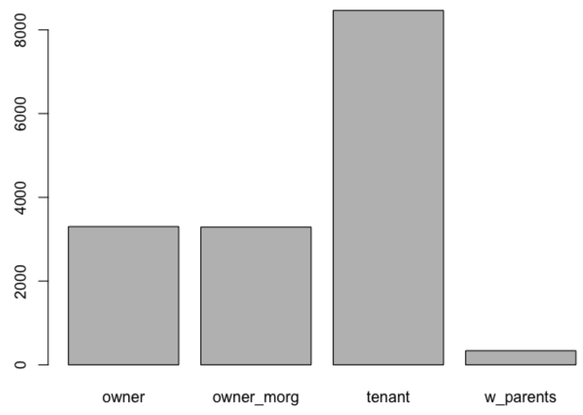
Residential_status



```
>     tdata1 %>% filter(purchase == 'yes', residential_status == 'owner') %>% summarise(length(residential_status))
  length(residential_status)
1                       3302
>     tdata1 %>% filter(purchase == 'yes', residential_status == 'owner_morg') %>% summarise(length(residential_status))
  length(residential_status)
1                       3289
>     tdata1 %>% filter(purchase == 'yes', residential_status == 'tenant') %>% summarise(length(residential_status))
  length(residential_status)
1                       8463
>     tdata1 %>% filter(purchase == 'yes', residential_status == 'w_parents') %>% summarise(length(residential_status))
  length(residential_status)
1                        337
```

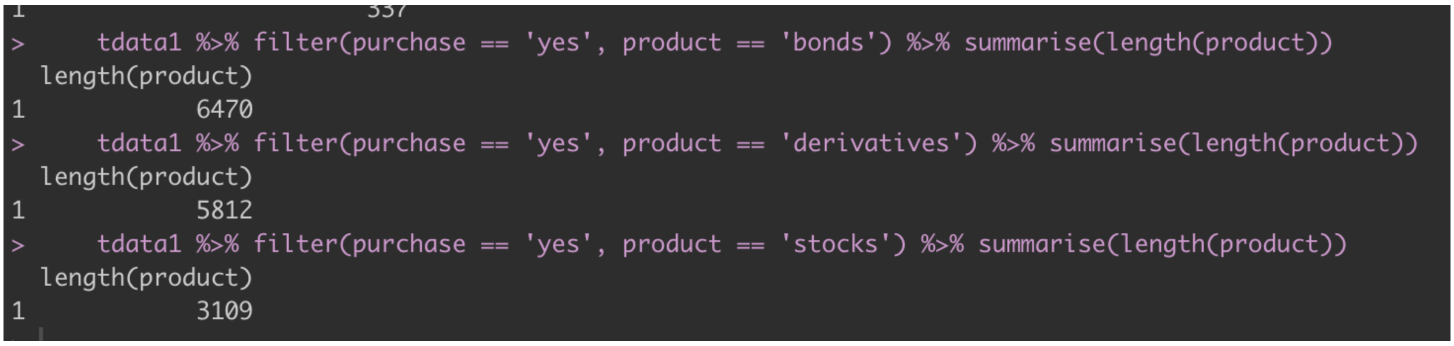


```
      plot(tdata1 %>% filter(purchase == 'yes') %>% select(residential_status))
```

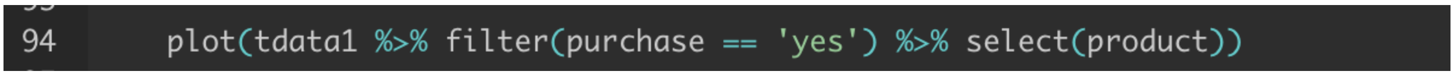


The highest purchases of investment products are made by the tenants at 8463 while the lowest purchases are made by individuals living with parents. Tenants are assumed to have lower financial liability as compared to owners as there are no taxes and expenditure on the maintenance of the property. The risk-taking capability therefore increases. Owners and Owners with mortgage are assumed to be individuals having steady income jobs and under property loan or other financial liabilities which reduces the purchases in investment products.

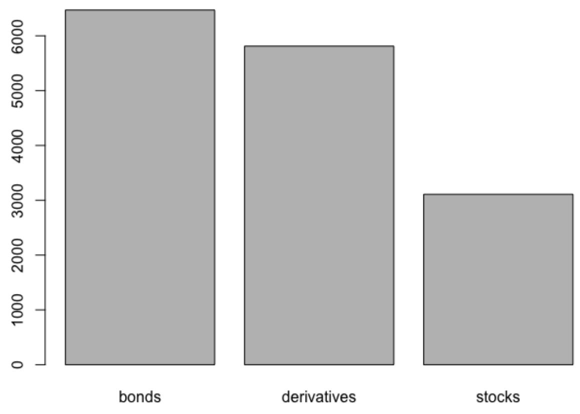
Product



```
1                    337
>     tdata1 %>% filter(purchase == 'yes', product == 'bonds') %>% summarise(length(product))
  length(product)
1            6470
>     tdata1 %>% filter(purchase == 'yes', product == 'derivatives') %>% summarise(length(product))
  length(product)
1            5812
>     tdata1 %>% filter(purchase == 'yes', product == 'stocks') %>% summarise(length(product))
  length(product)
1            3109
```

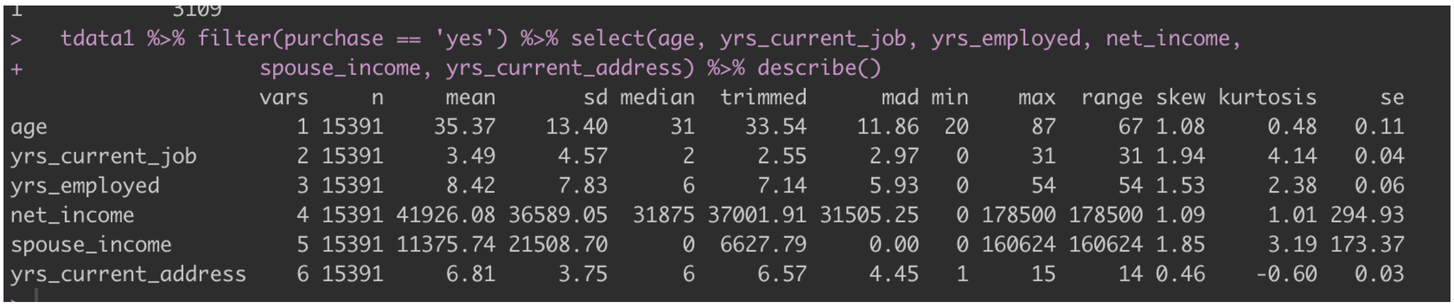


```
    plot(tdata1 %>% filter(purchase == 'yes') %>% select(product))
```



The bonds are observed to have the highest purchases at 6470 as it is the safest investment option which guarantees some interest payment over a period, derivative contracts provide cover against financial investments which is fundamentally outsourcing of risks. This is a product purchased by regular investors and finds itself in the second highest place at 5812. The stocks are the most volatile among the investment options and have noticeably lower purchases at 3109.

Calculating the Descriptive Statistics for the Numerical Columns

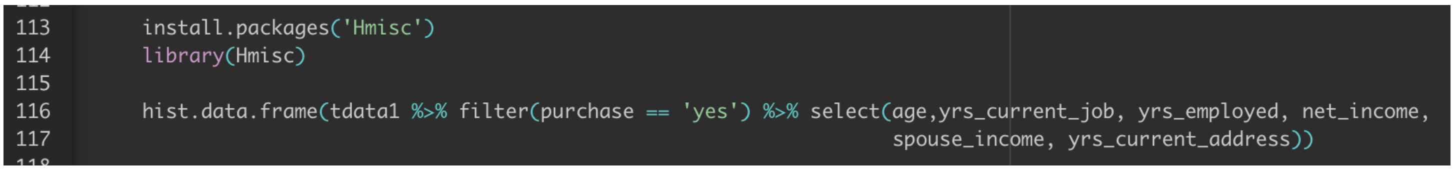


```
1            3109
>   tdata1 %>% filter(purchase == 'yes') %>% select(age, yrs_current_job, yrs_employed, net_income,
+                    spouse_income, yrs_current_address) %>% describe()
                    vars     n     mean       sd median  trimmed      mad min    max  range skew kurtosis     se
age                    1 15391    35.37    13.40     31    33.54    11.86  20     87     67 1.08     0.48   0.11
yrs_current_job        2 15391     3.49     4.57      2     2.55     2.97   0     31     31 1.94     4.14   0.04
yrs_employed           3 15391     8.42     7.83      6     7.14     5.93   0     54     54 1.53     2.38   0.06
net_income             4 15391 41926.08 36589.05  31875 37001.91 31505.25   0 178500 178500 1.09     1.01 294.93
spouse_income          5 15391 11375.74 21508.70      0  6627.79     0.00   0 160624 160624 1.85     3.19 173.37
yrs_current_address    6 15391     6.81     3.75      6     6.57     4.45   1     15     14 0.46    -0.60   0.03
```

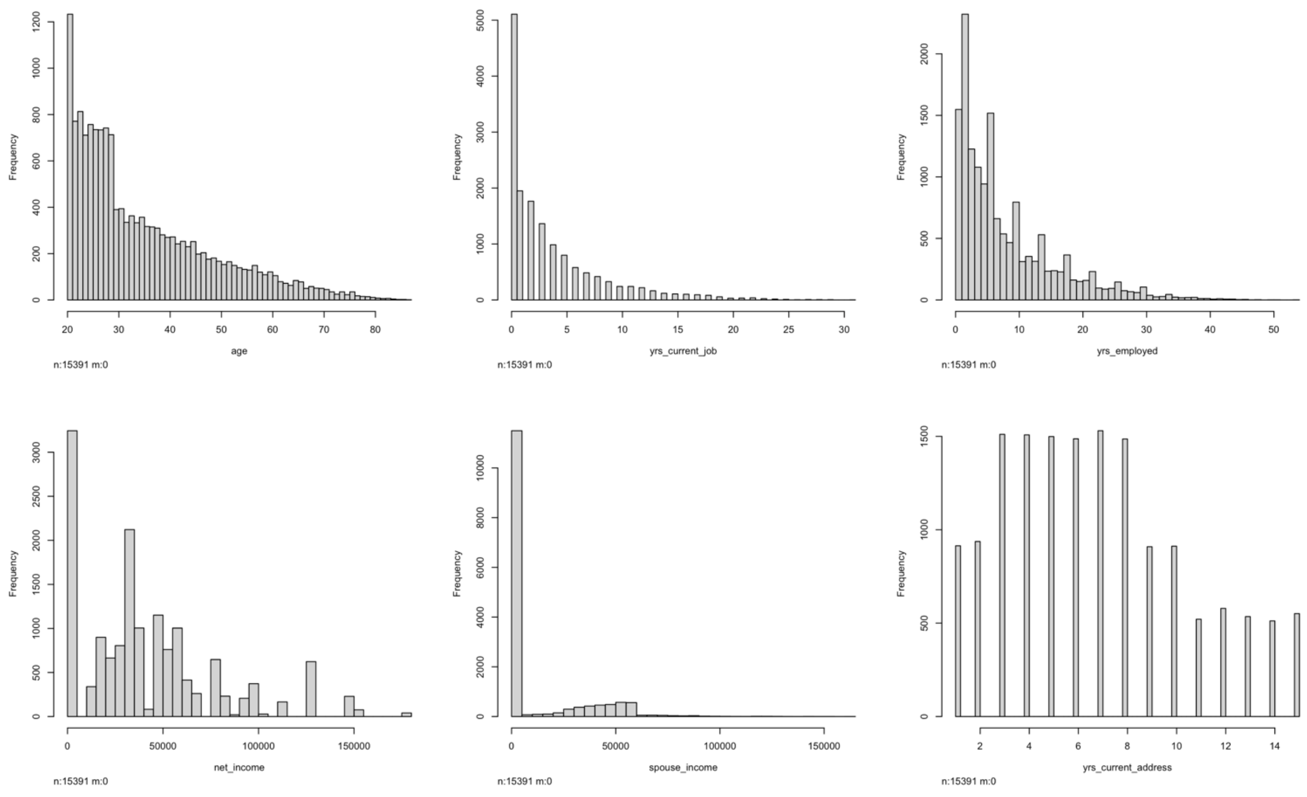
The descriptive statistics are calculated for the numerical columns of the dataset by using the describe() function in R. Being commensurate with the report objective, only the customers who have purchased the products are considered. It is observed that the mean age of customers is 35.37 which can be an ideal starting point for targeted marketing for the investment products. Similarly, means of other variables can be considered as an input to the sales team. The median provides the centre point of the data set as a whole number which can be an additional input. The standard deviation can help understand the dispersion of the data points around the mean. Other parameters include min, max, the difference between them is called the range and skew to understand the symmetry of the distribution of the data set.

Analysis of the Numerical Columns

This section analyses the numerical columns of the dataset. The records of successful purchases are selected for the analysis to understand the impact each variable has on the purchase decision. In the below R code, the records are filtered using condition such that the purchase is 'yes' and the numerical columns are selected. This selection of data is done using the pipe function in R. The following histograms are then plotted.



```
install.packages('Hmisc')
library(Hmisc)

hist.data.frame(tdata1 %>% filter(purchase == 'yes') %>% select(age,yrs_current_job, yrs_employed, net_income,
                                                              spouse_income, yrs_current_address))
```



age

The histogram for age shows the frequency of purchases with respect to age groups of the customers. It is observed that the maximum purchases of the investment products happen between the age groups 20 and 30. This age group consists of high-risk taking individuals who try different volatile and non-volatile investment products to build quick wealth and also for the long term. The risk-taking capacity of an individual is inversely proportional to financial responsibilities. This age group consists of mostly unmarried individuals with no children and therefore have the privilege of taking higher risks.

yrs_current_job

The number of years spent by the individual shows a decreasing trend. The frequency of purchases decreases as the number of years increase at the current job. This can be explained by two possibilities, the individual carries out diversified investment at the start of the new job for the long run and as the years progresses at the current job, there is promotion and increase in the salary which decreases the need to purchase any more investment product as the individual already becomes financially stable.

yrs_employed

The histogram observed for years employed shows an overall decreasing trend. The frequency of the purchase at the 0 years is less at 1600 and spikes to 3000 records at 1 and 2 years. The overall trend shows that the initial years of employment is when most individuals purchase investment products for the long term to fill in gaps of employment or other uncertainties in the future.

net_income

The highest purchases are by individuals having 0 net income. It is assumed that the missing or NA values for the net income was replaced with 0 during cleaning of the dataset. The first trend at 0 is therefore not considered for our analysis. There is a spike observed at 30000 net income which can be the income where the highest number of individuals purchase the investment products.

spouse_income

A similar trend is observed in spouse income as above. The maximum individuals having spouse income 0 have made the most purchases. However, here it can be assumed that this trend is because the individuals feel more need to invest when the spouse does not have a net income.

yrs_current_address

Here the trend observed by the histogram do not show extreme variation in purchase frequencies as per the number of years as compared to previous variables. However, a noticeable increase in purchase of the investment products is observed between individuals who have spent 3 to 8 years at the current address.

Algorithms Considered for Predictive Analytics

Assumptions

1. The variable ‘purchase’ is assumed to be the dependent or target variable of the dataset which decides if the product is purchased by the customer or not. The values of this variable will be the output of the prediction algorithms tested in this report.
2. The remaining 14 variables, gender, age, marital_status, education, nb_depend_child, employ_status, yrs_current_job, yrs_employed, net_income, spouse_work, spouse_income, residential_status, yrs_current_address and product are explanatory variables.
3. A correlation is assumed to be present between the purchase variable and the rest of the column and are included to build the prediction model.
4. As the target variable is categorical, this problem is assumed to be of the classification type. Therefore, algorithms specializing in classifications will be tested on the data set.

The following classification algorithms are considered

Classification and Regression Tree (CART)

Classification and Regression Tree is a commonly used decision tree that has a root node and splits at yes and no decisions. CART can be used for both classification and regression problems.

Pros

- CART is easy to interpret as the algorithm constructs a single tree.
- It is faster to train CART compared to Random Forests on the same dataset.
- No normalization of the data is needed before training the model.

Cons

- There is a risk of overfitting as the classifier is built over a single sample. There can be high errors when unknown test data is presented.
- Imbalanced dataset can cause biases in the prediction.
- The CART algorithm can be limited when the dataset gets complicated.
- A small change in the data can cause large change in the structure of the tree.

C5.0

The C5.0 decision tree is the latest version based on ID3. It has additional feature of boosting over C4.5 which increases accuracy rate of identification on samples.

Pros

- It is more efficient compared to other complex models.
- Adds weightages to the samples generated which presents its importance.
- It automatically excludes the unimportant features.
- C5.0 can be efficient for both small and large datasets.

Cons

- It is easy to overfit or underfit this model. The model while training can fit the anomalies or not fit the important data points and can affect the prediction accuracy.
- Small changes in the training data can cause large deviations in the decision logic.
- Large trees can get complicated and the decisions can seem counter intuitive.

Random Forest

Random Forest is an ensemble learning method that combines n number of individual decision trees by taking row and feature samples from the dataset with replacement. The majority result of the n decision trees is considered as the final output.

Pros

- Combination of multiple individual decision trees reduces the errors caused by overfitting.
- Random Forests are robust to outliers.
- Runs efficiently on large data sets.
- Works well on nonlinear data

Cons

- Random Forests can be biased sometimes while dealing with categorical variables.
- Random Forests can be slower to train compared to other algorithms due to high number of individual decision trees.

Naïve Bayes

Naïve Bayes classifiers are a family of probabilistic classifiers which use Bayes theorem at the core for classification. It assumes that the features contributing to the final classification are independent of each other. The following equation is the Bayes equation where P(c|x), the probability of the target variable is calculated.

$$\underbrace{P(c \mid x)}_{\text{Posterior Probability}} = \frac{\overbrace{P(x \mid c)}^{\text{Likelihood}}\;\overbrace{P(c)}^{\text{Class Prior Probability}}}{\underbrace{P(x)}_{\text{Predictor Prior Probability}}}$$

$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Pros

- The assumption that all the features in the dataset are independent make this model very fast to train.
- Naïve Bayes classification works very well with high dimensional classification problems.
- Naïve Bayes can be suitable for large datasets.
- It can be one of the easiest algorithms to implement.

Cons

- Naïve Bayes is faster comparatively as it assumes feature independence in the dataset, however in real world problems this is not the case. Naïve Bayes therefore has limited application case.
- Naïve Bayes is only limited to classification problems and cannot be used for regression.

Support Vector Machine (SVM)

Support vector machine is a well-known algorithm in the machine learning space known to provide high accuracy while consuming low computational power. The objective of this algorithm is to search for a hyperplane in the N-dimensional space (variables) that classifies the data points.

Pros

- SVM is well suited when there are large number of dimensions.
- It can be slower in training but provide fast predictions.
- It can be used for both classification and prediction of continuous numerical values.

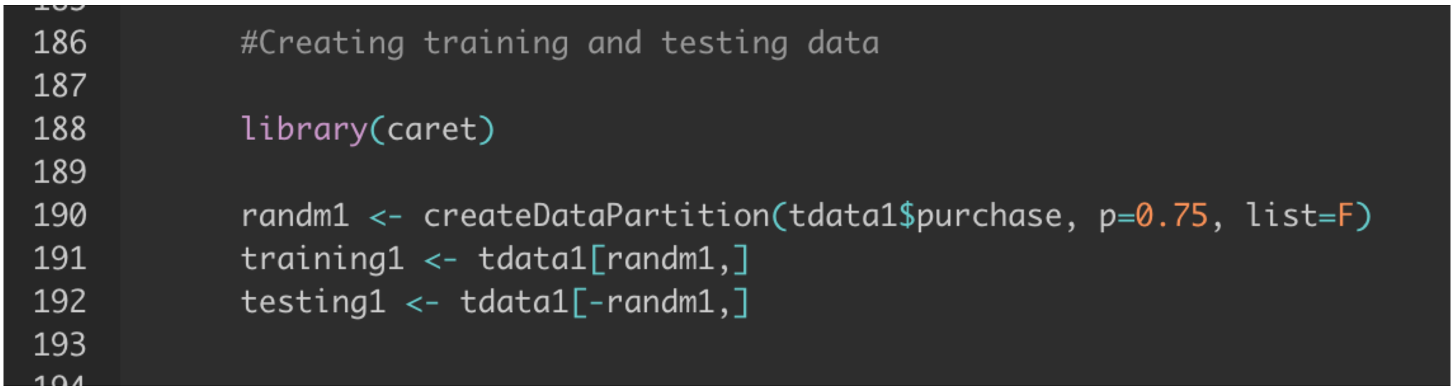
Cons

- SVM is not suitable for larger datasets.
- This model does not perform very well when the dataset has a lot of noise.
- SVM does not provide easily interpretable reports.
- Overfitting can be difficult to detect and fix sometimes.

Preparation of the Data

The customer dataset having 30000 observations is split randomly into training and testing data sets. The training dataset will be used to train the prediction models above and build the classifier and the testing dataset will be used to check the accuracy of the model built by comparing the predictions and the actual values in the column. The predictions will be done thereafter using the second file 'Market_pred.csv' for ABSA sales team.

The following code is used to create these random training and testing samples.

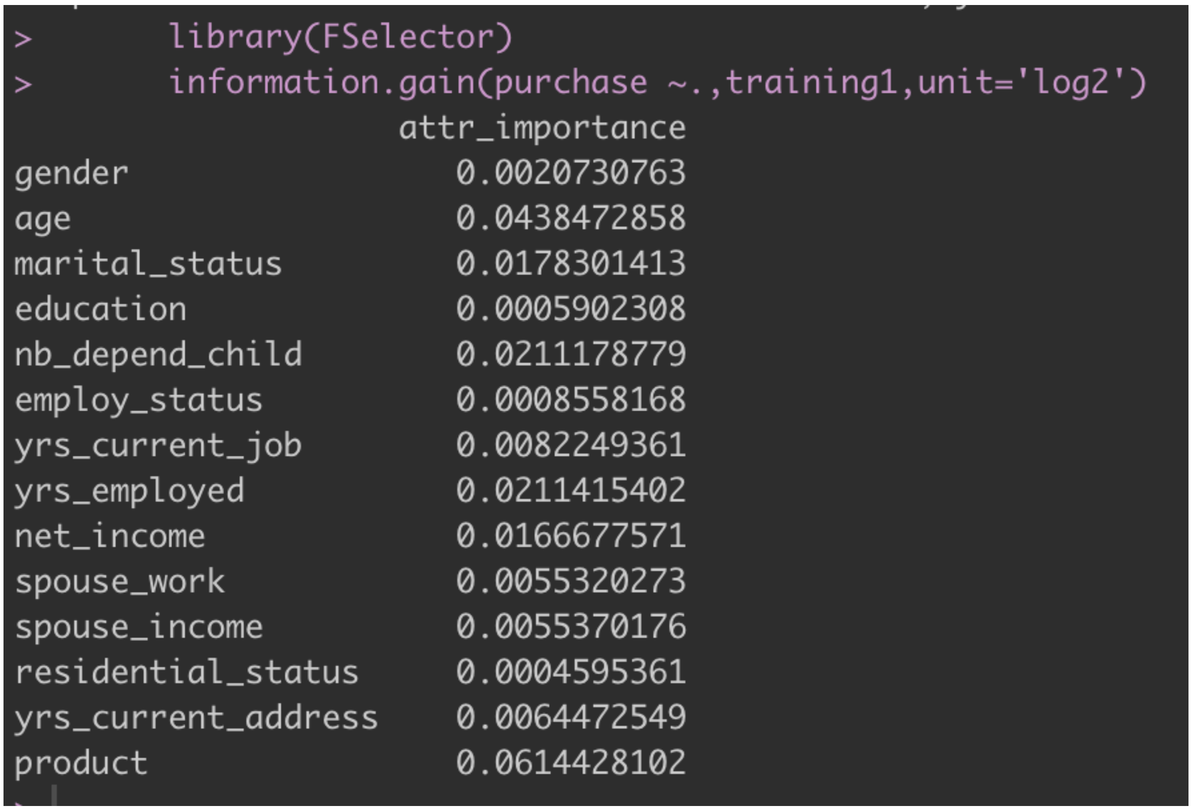


```
#Creating training and testing data

library(caret)

randm1 <- createDataPartition(tdata1$purchase, p=0.75, list=F)
training1 <- tdata1[randm1,]
testing1 <- tdata1[-randm1,]

```

The training dataset will have the sample size of 75% of the original dataset which will be 22501 and the remaining 7499 will be assigned to the testing dataset. This selection is random to avoid any biases while building the models.

Calculating the Information Gain

Information gain shows the impact each dimension has on the decision variable. The information.gain() function is used from the FSelector package in R as shown below. The training1 dataset is used to calculate the information gain as this dataset will be used to train the prediction models. The purchase is the target variable used which is dependent on all the variables of the data set.



```
>       library(FSelector)
>       information.gain(purchase ~.,training1,unit='log2')
                        attr_importance
gender                     0.0020730763
age                        0.0438472858
marital_status             0.0178301413
education                  0.0005902308
nb_depend_child            0.0211178779
employ_status              0.0008558168
yrs_current_job            0.0082249361
yrs_employed               0.0211415402
net_income                 0.0166677571
spouse_work                0.0055320273
spouse_income              0.0055370176
residential_status         0.0004595361
yrs_current_address        0.0064472549
product                    0.0614428102
```

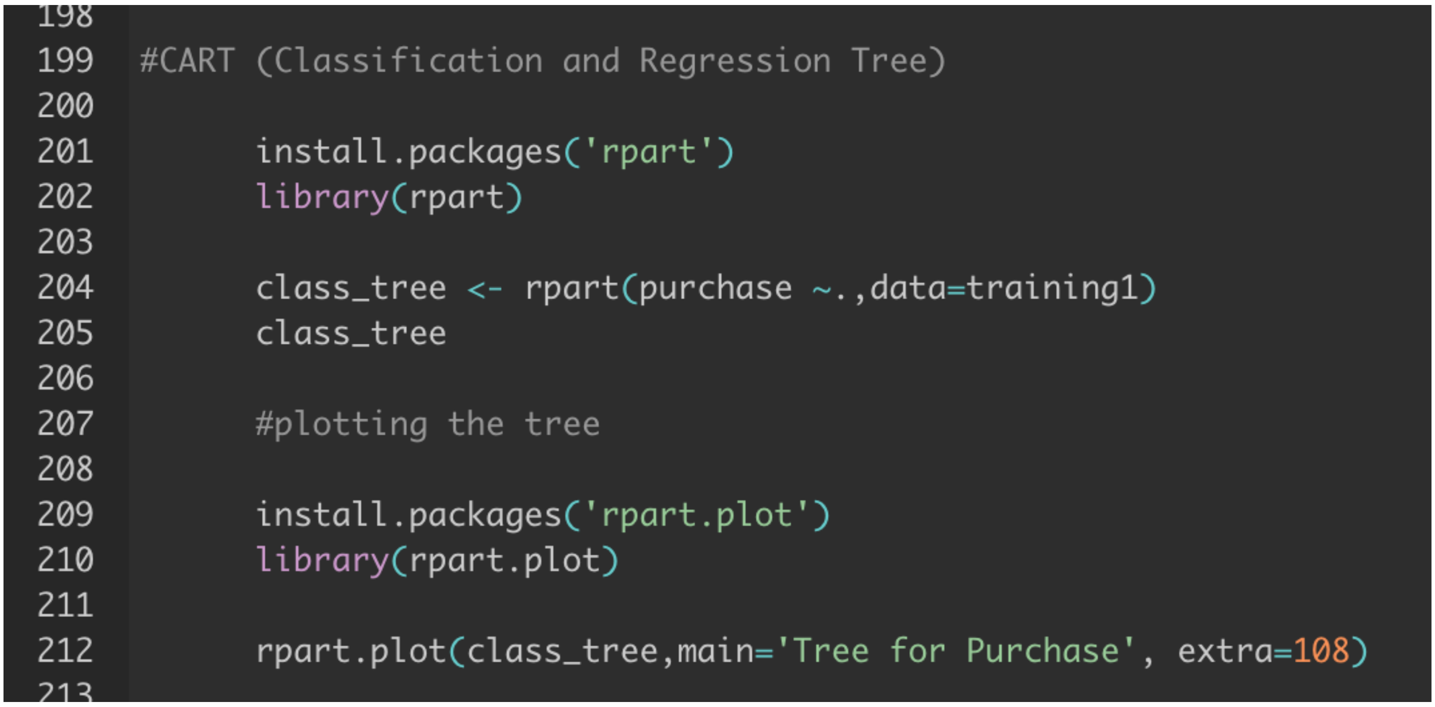
It is observed above that the highest information gain is for the *product* column at 0.0614428102. This means that the product will be the most important deciding factor for the purchase.

Building Classifiers

Classification and Regression Tree (CART)

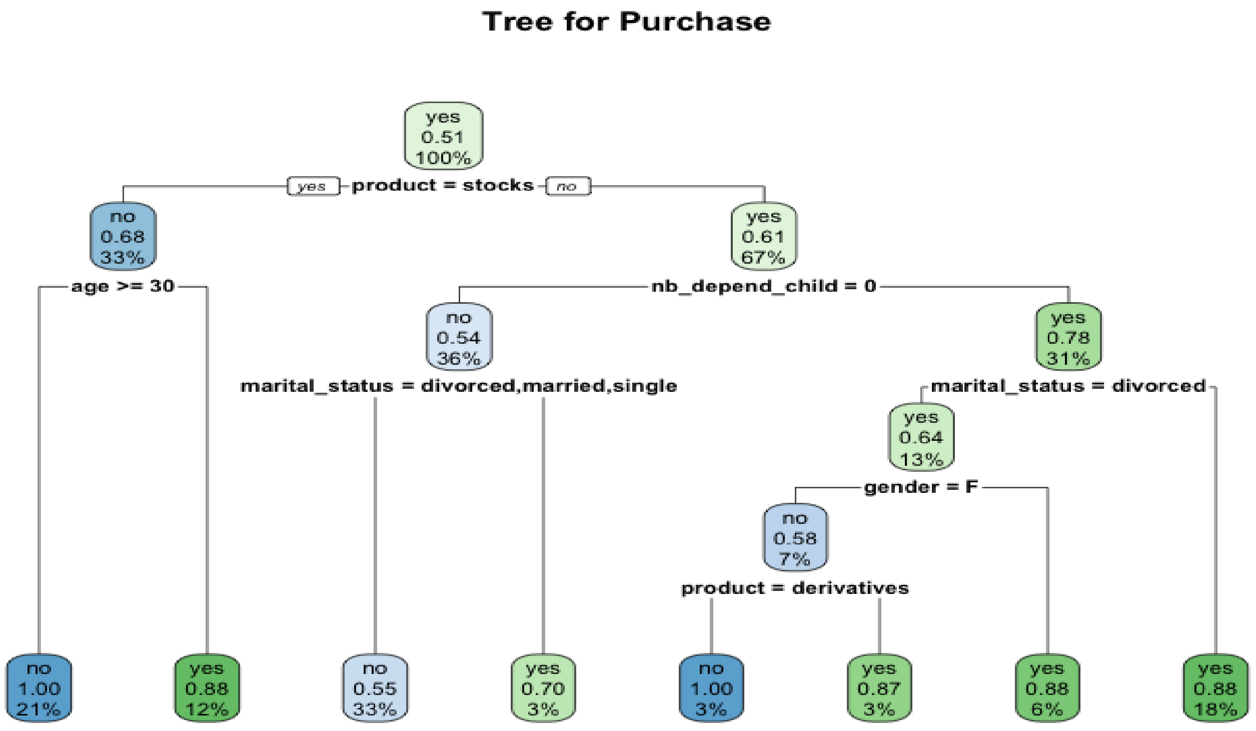
The library rpart is first loaded in R and the function rpart is passed with the parameters 'purchase ~.' and 'data'. Function 'rpart' is used to build the classifier of the decision tree. The 'purchase ~.' means that the dependent or the decision variable that needs to be predicted is 'purchase' while the explanatory variables are all the remaining columns of the data frame. 'data = training1' indicates that the dataset selected is 'training1'. The decision tree takes care of the unimportant variable that do not impact the final purchase decision much as seen in the information gain above. The following is the R code where the classifier is generated and trained using the training data.

Training



```
#CART (Classification and Regression Tree)

        install.packages('rpart')
        library(rpart)

        class_tree <- rpart(purchase ~.,data=training1)
        class_tree

        #plotting the tree

        install.packages('rpart.plot')
        library(rpart.plot)

        rpart.plot(class_tree,main='Tree for Purchase', extra=108)
```

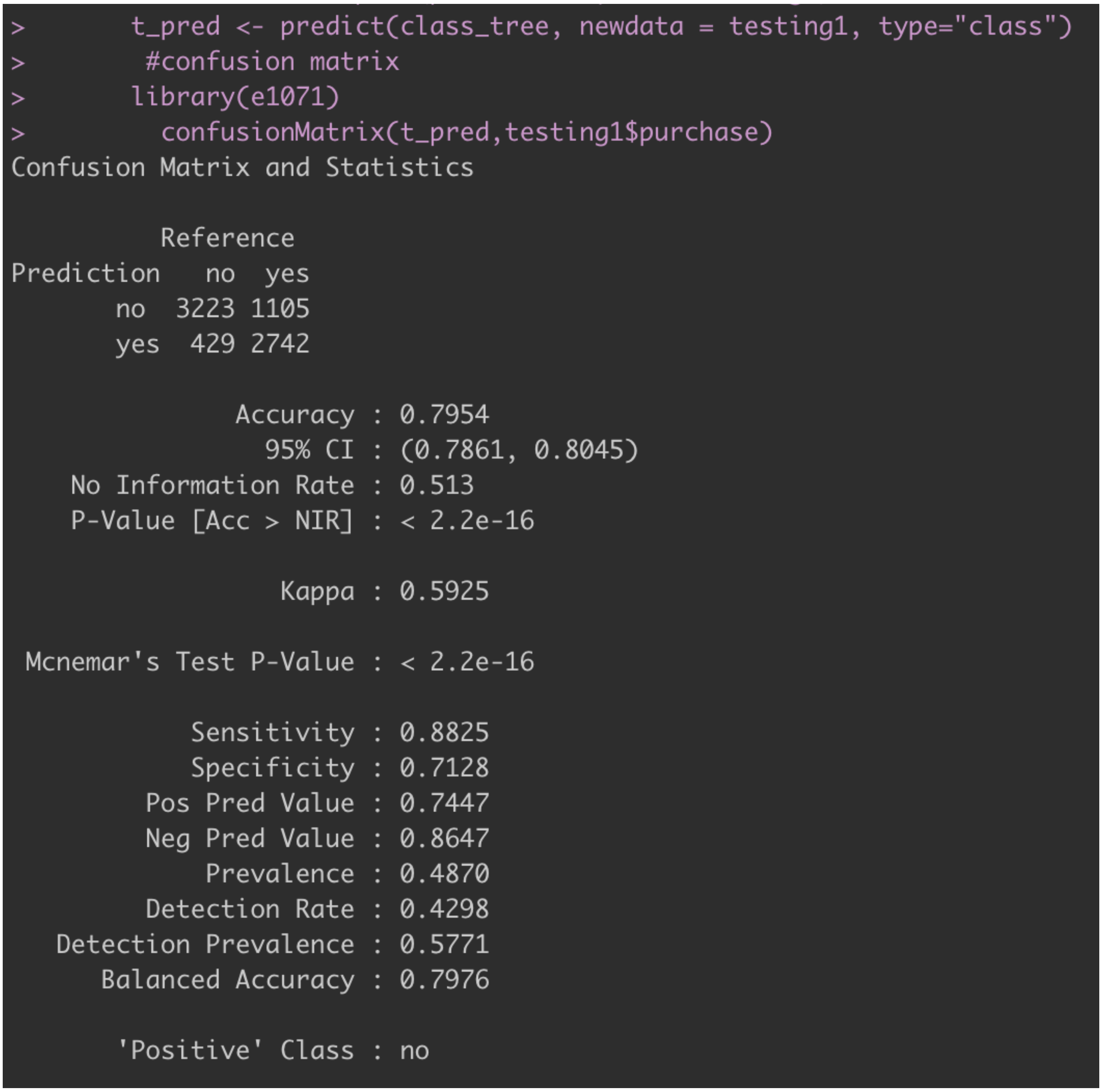
The classifier ‘class_tree’ trained by the training data is plotted using the rpart.plot() function which is part of the library ‘rpart.plot’. The classifier is plotted as follows.



As observed by the information gain, the product is the most important factor that influences the and the tree spreads out further with remaining decision nodes like age, marital status and gender.

Testing

The classifier object is ‘class_tree’ and it will be used to make the predictions. The classifier will now be tested using the ‘testing1’ dataset. Prediction will be performed for the testing data and the accuracy will be calculated.

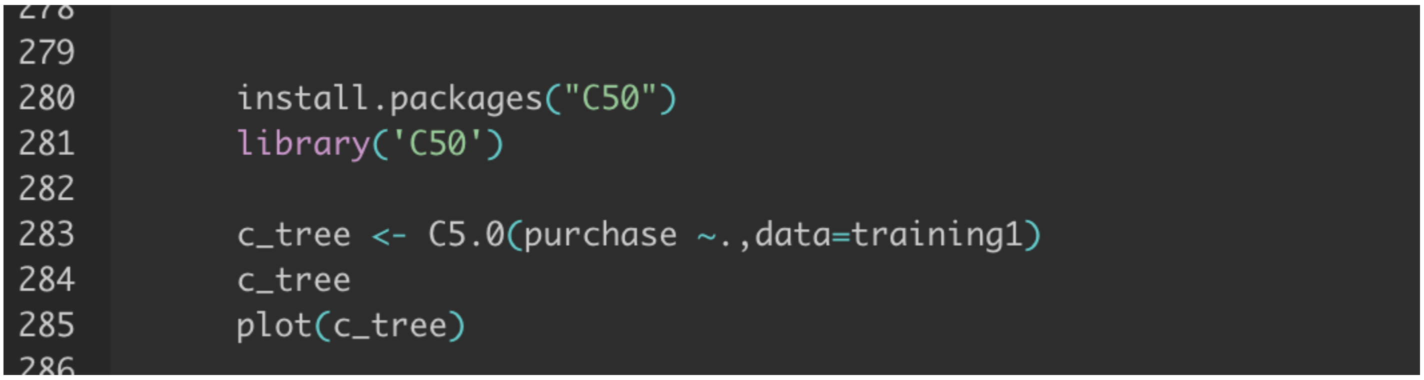


```
>        t_pred <- predict(class_tree, newdata = testing1, type="class")
>         #confusion matrix
>        library(e1071)
>          confusionMatrix(t_pred,testing1$purchase)
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  3223 1105
       yes  429 2742

               Accuracy : 0.7954
                 95% CI : (0.7861, 0.8045)
    No Information Rate : 0.513
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5925

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8825
            Specificity : 0.7128
         Pos Pred Value : 0.7447
         Neg Pred Value : 0.8647
             Prevalence : 0.4870
         Detection Rate : 0.4298
   Detection Prevalence : 0.5771
      Balanced Accuracy : 0.7976

       'Positive' Class : no
```

The purchase decisions are predicted using the predict() function which takes the classifier and the testing1 data set as input and generates the predicted values of the purchase variable. The predicted values are stored in t_pred. Confusion Matrix calculates the final accuracy score for prediction model used on the testing1 dataset by comparing the purchase predictions by the model with the actual purchase values present in the testing1 dataset. The confusionMatrix() function from the e1071 library is used to calculate the scores. The CART algorithm generates 429 false positives which is a Type 1 error and 1105 false negatives which is a type 2 error. This means that CART algorithm wrongly marked 429 decisions as ‘yes’ when they were actually ‘no’ and marked 1105 decisions as no when they were actually a ‘yes’. The overall classification accuracy score calculated is 79.54% which is acceptable. The precision is further calculated to obtain the value of 0.8647 or 86.47%.

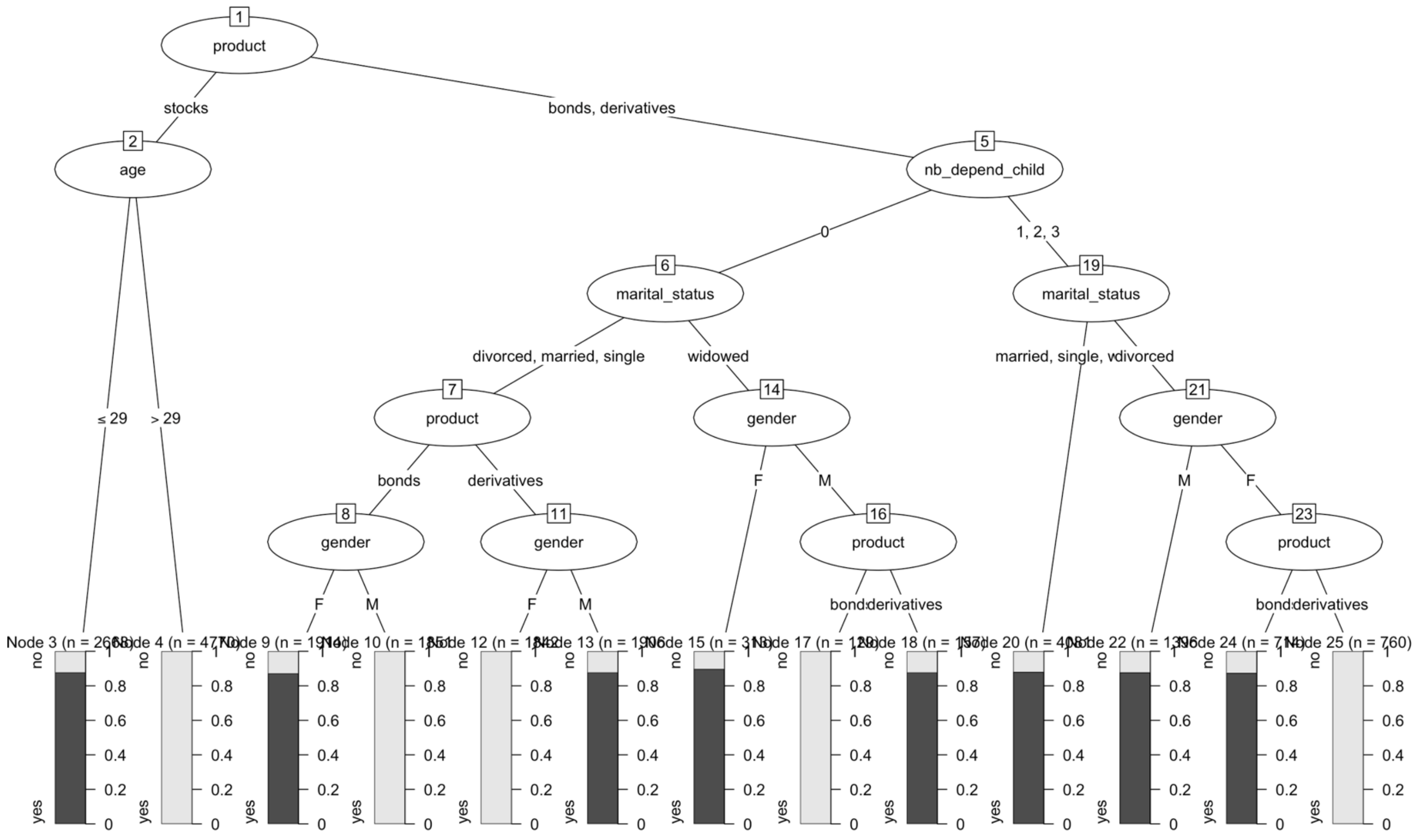
C5.0

The library ‘C50’ is first loaded in R and the function C50() is passed with the parameters ‘purchase ~.’ and ‘data’. Function ‘C50’ is used to build the classifier of the decision tree. The ‘purchase ~.’ means that the dependent or the decision variable that needs to be predicted is ‘purchase’ while the explanatory variables are all the remaining columns of the data frame. ‘data = training1’ indicates that the dataset selected is ‘training1’. The decision tree takes care of the unimportant variable that do not impact the final purchase decision much as seen in the information gain. The following is the R code where the classifier is generated and trained using the training data.



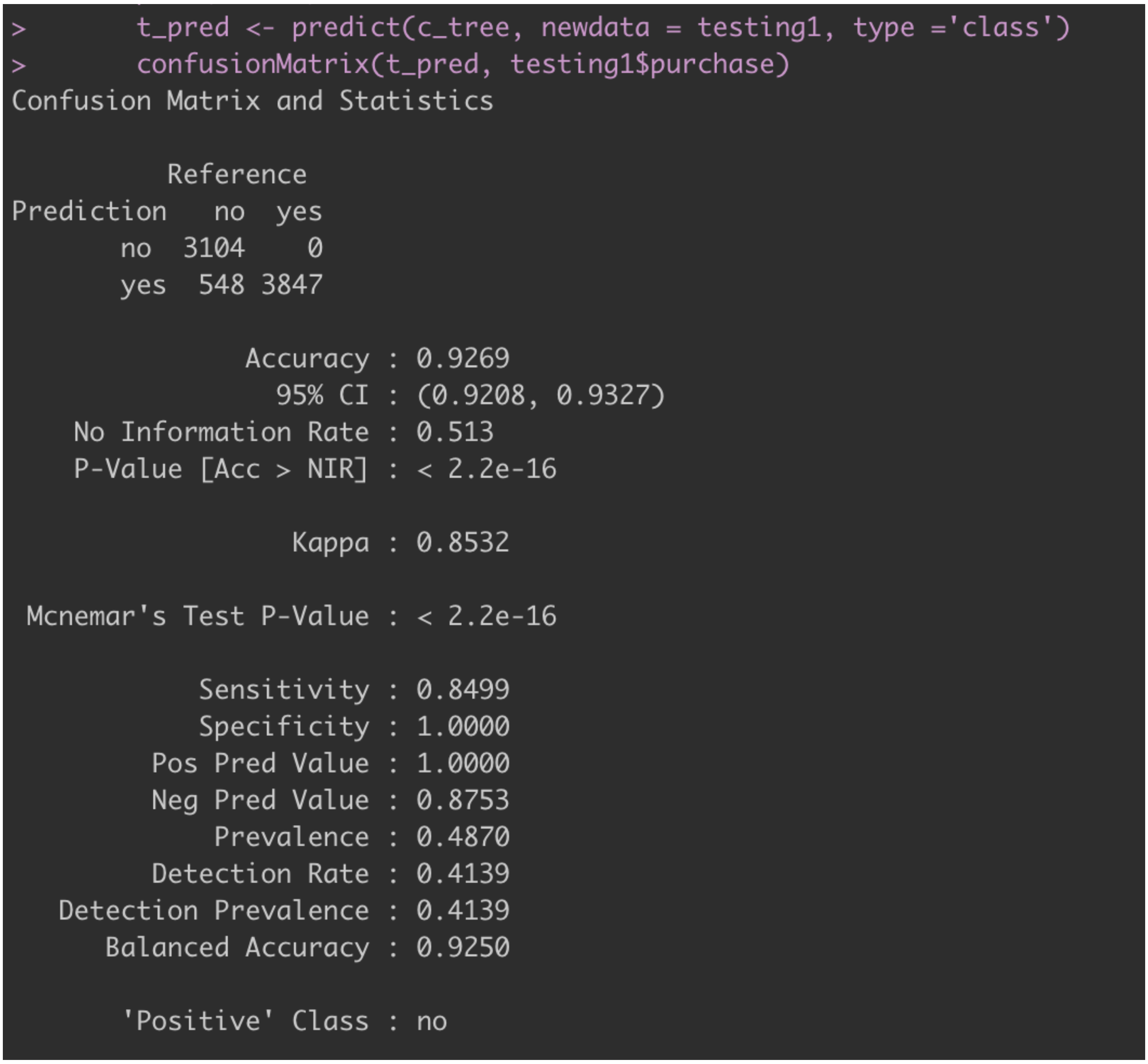
```
install.packages("C50")
library('C50')

c_tree <- C5.0(purchase ~.,data=training1)
c_tree
plot(c_tree)
```

The classifier ‘c_tree’ trained by the training data is plotted using the plot() function. The classifier plotted looks as follows



Testing

The classifier object is ‘c_tree’ and it will be used to make the prediction. The classifier will now be tested using the ‘testing1’ dataset. Prediction will be performed for the testing data and the accuracy will be calculated.



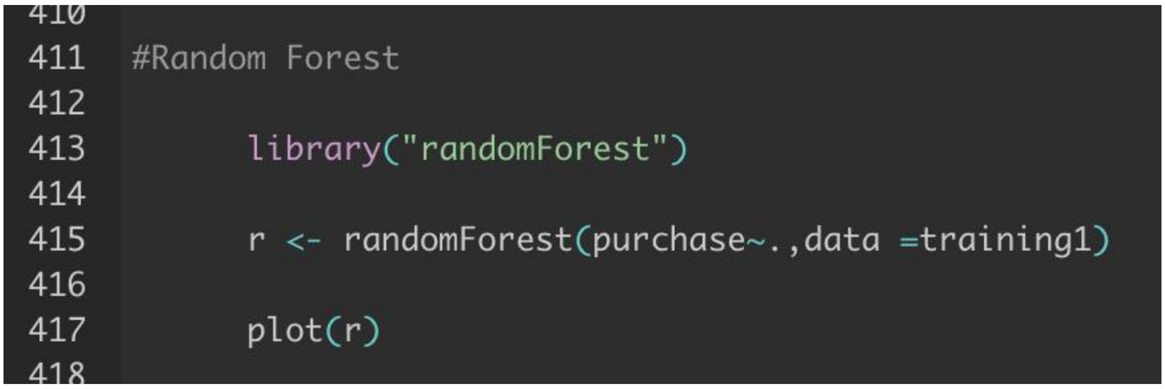
```
>       t_pred <- predict(c_tree, newdata = testing1, type ='class')
>       confusionMatrix(t_pred, testing1$purchase)
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  3104    0
       yes  548 3847

               Accuracy : 0.9269
                 95% CI : (0.9208, 0.9327)
    No Information Rate : 0.513
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8532

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8499
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.8753
             Prevalence : 0.4870
         Detection Rate : 0.4139
   Detection Prevalence : 0.4139
      Balanced Accuracy : 0.9250

       'Positive' Class : no
```

The purchase decisions are predicted using the predict() function which takes the classifier and the testing1 data set as input and generates the predicted values of the purchase variable. The predicted values are stored in t_pred. Confusion Matrix calculates the final accuracy score for prediction model used on the testing1 dataset by comparing the purchase predictions by the model with the actual purchase values present in the testing1 dataset. The confusionMatrix() function from the e1071 library is used to calculate the scores. The C5.0 algorithm generates 548 false positives which is a Type 1 error and 0 false negatives which is a type 2 error. The false positive score of 548 means that C5.0 has a higher possibility of predicting a decision as ‘yes’ when it is actually ‘no’. However, the false negative count is 0 which means that there is no Type 2 error with the C5.0 classification algorithm. The overall accuracy of the algorithm is calculated, which gives the percentage of 92.69%. The precision is further calculated as below to obtain the value of 0.8753 or 87.53%.

$$\textbf{\textit{Precision}} = \frac{TP}{TP + FP}$$

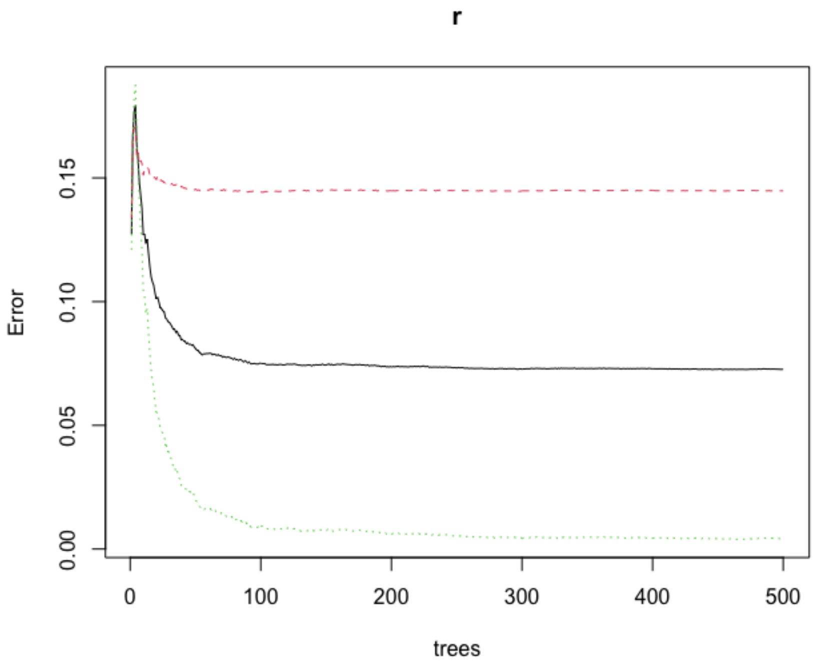
Random Forest

The library 'randomForest' is first loaded in R and the function randomForest() is passed with the parameters 'purchase ~.' and 'data'. Function 'randomForest()' is used to build the classifier of the decision tree. The 'purchase ~.' means that the dependent or the target variable that needs to be predicted is 'purchase' while the explanatory variables are all the remaining columns of the data frame.   'data = training1' indicates that the dataset selected is 'training1'. The random forest will build 500 decision trees by default on bootstrapped training samples. The following is the R code where the classifier is generated and trained using the training data.



```
#Random Forest

        library("randomForest")

        r <- randomForest(purchase~.,data =training1)

        plot(r)
```

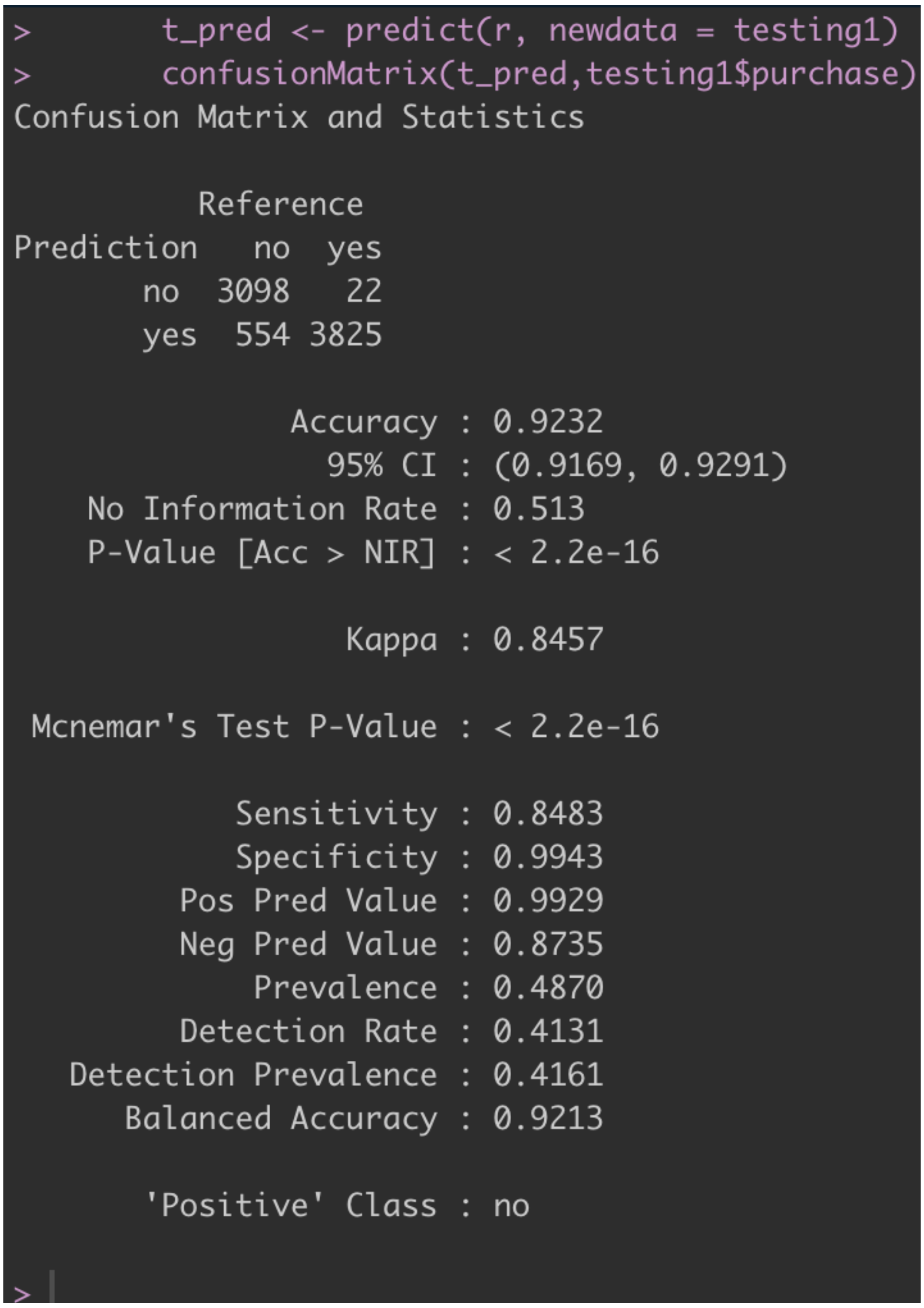
The classifier is plotted using the plot() function as follows.



The graph shows that as the number of decision trees increases in the random forest by row sampling and feature sampling, the high variance or overfitting (error) happening due to individual decision trees gets reduced.

Testing

The classifier object is ‘r’ and it will be used to make the prediction. The classifier will now be tested using the ‘testing1’ dataset. Prediction will be performed for the testing data and the accuracy will be calculated.



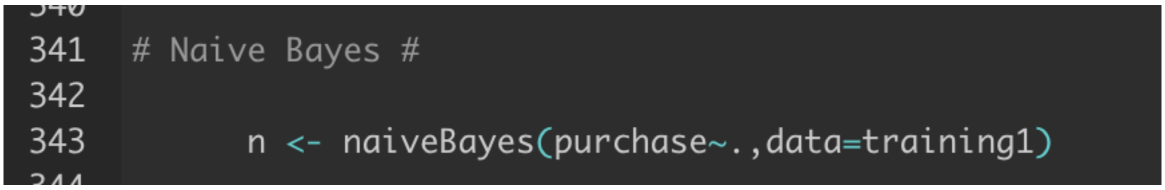
```
>       t_pred <- predict(r, newdata = testing1)
>       confusionMatrix(t_pred,testing1$purchase)
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  3098   22
       yes  554 3825

               Accuracy : 0.9232
                 95% CI : (0.9169, 0.9291)
    No Information Rate : 0.513
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8457

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8483
            Specificity : 0.9943
         Pos Pred Value : 0.9929
         Neg Pred Value : 0.8735
             Prevalence : 0.4870
         Detection Rate : 0.4131
   Detection Prevalence : 0.4161
      Balanced Accuracy : 0.9213

       'Positive' Class : no

>
```

The purchase decisions are predicted using the predict() function which takes the classifier and the testing1 data set as input and generates the predicted values of the purchase variable. The predicted values are stored in t_pred. Confusion Matrix calculates the final accuracy score for prediction model used on the testing1 dataset. It calculates it by comparing the purchase predictions by the model with the actual purchase values present in the testing1 dataset. The confusionMatrix() function from the e1071 library is used to calculate the scores. The Random Forest algorithm generates 554 false positives which is a Type 1 error and 22 false negatives which is a type 2 error. The false positive score of 554 means that Random Forest has a higher possibility of predicting a decision as ‘yes’ when it is actually ‘no’. However, the false negative count is 22 which means that there is still a low Type 2 error with the Random Forest classification algorithm. The overall accuracy of the algorithm is calculated gives 92.32%. The precision of the model is calculated gives 0.8734 or 87.34%

Naïve Bayes

Naïve Bayes is a mathematical classification model and can be used directly using the naiveBayes() function in R code. The naiveBayes() function takes two parameters. The ‘purchase ~.’ means that the dependent or the target variable that needs to be predicted is ‘purchase’ and that the explanatory variables are all the remaining columns of the data frame. ‘data = training1’ indicates that the dataset selected is ‘training1’. The classifier is generated and assigned to a variable ‘n’.

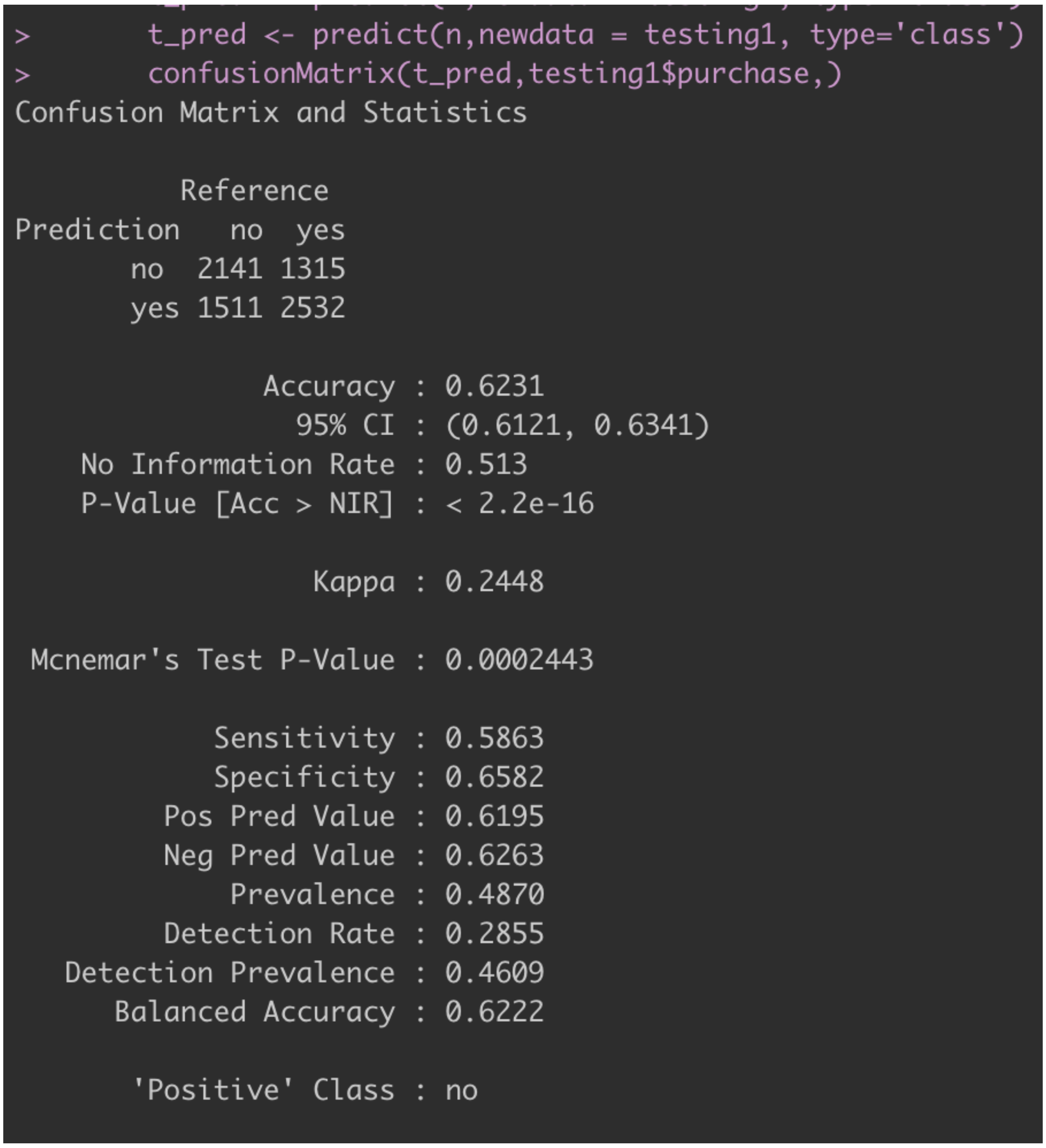
Training



```
# Naive Bayes #

      n <- naiveBayes(purchase~.,data=training1)
```

Testing

The classifier object is ‘n’ and it will be used to make the prediction. The classifier will now be tested using the ‘testing1’ dataset. Prediction will be performed for the testing data and the accuracy will be calculated.



```
>       t_pred <- predict(n,newdata = testing1, type='class')
>       confusionMatrix(t_pred,testing1$purchase,)
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  2141 1315
       yes 1511 2532

               Accuracy : 0.6231
                 95% CI : (0.6121, 0.6341)
    No Information Rate : 0.513
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.2448

 Mcnemar's Test P-Value : 0.0002443

            Sensitivity : 0.5863
            Specificity : 0.6582
         Pos Pred Value : 0.6195
         Neg Pred Value : 0.6263
             Prevalence : 0.4870
         Detection Rate : 0.2855
   Detection Prevalence : 0.4609
      Balanced Accuracy : 0.6222

       'Positive' Class : no
```

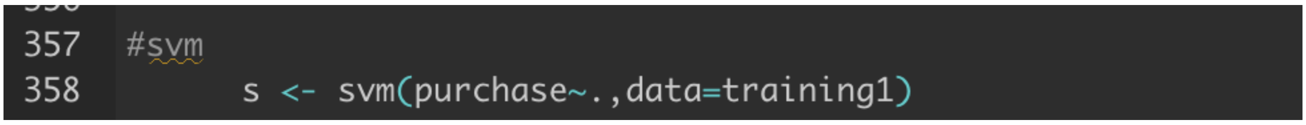
The purchase decisions are predicted using the predict() function which takes the classifier and the testing1 data set as input and generates the predicted values of the purchase variable. The predicted values are stored in t_pred. Confusion Matrix calculates the final accuracy score for the prediction model used on the testing1 dataset. It calculates it by comparing the purchase predictions by the model with the actual purchase values present in the testing1 dataset. The confusionMatrix() function from the e1071 library is used to calculate the scores. The Naïve Bayes algorithm generates 1511 false positives which is a Type 1 error and 1315 false negatives which is a type 2 error. The false positive score of 1511 means that Naïve Bayes has a higher possibility of predicting a decision as 'yes' when it is actually 'no'. However, the false negative count is 1315 which means that there is a high Type 2 error at the same time with the Naïve Bayes classification algorithm. The overall accuracy of the algorithm is calculated and the percentage accuracy is 62.31%. The precision is further calculated as below to obtain the value of 0.6262 or 62.62%.

$$\textbf{\textit{Precision}} = \frac{TP}{TP + FP}$$

Support Vector Machine (SVM)

Support Vector Machine is a predictive classification model that can be used directly using the svm() function in R code. The svm() function takes two parameters. The ‘purchase ~.’ means that the dependent or the target variable that needs to be predicted is ‘purchase’ and that the explanatory variables are all the remaining columns of the data frame. ‘data = training1’ indicates that the dataset selected is ‘training1’. The classifier is generated and assigned to a variable ‘s’.

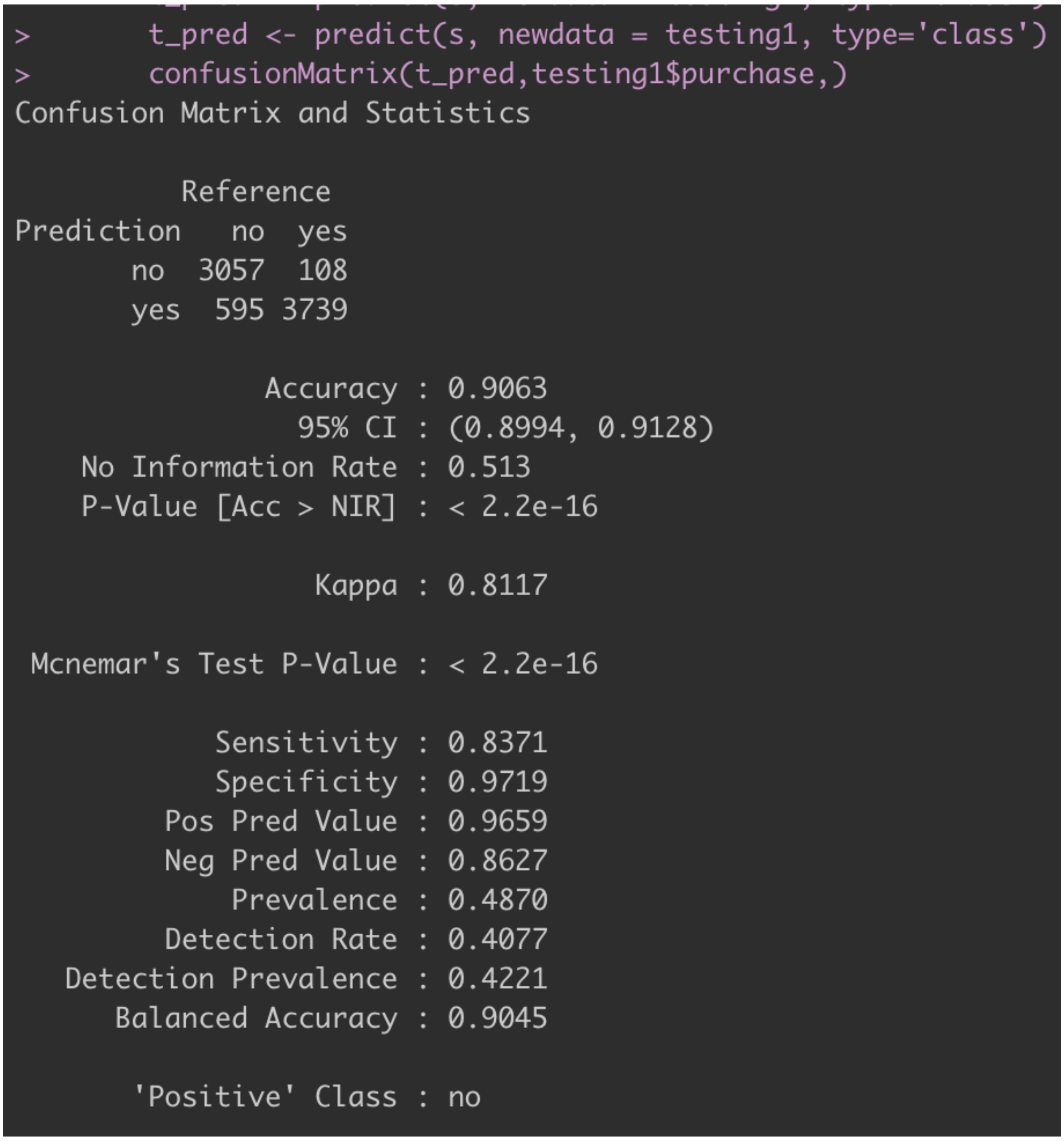
Training



```
#svm
	s <- svm(purchase~.,data=training1)
```

Testing

The classifier object is ‘s’ and it will be used to make the prediction. The classifier will now be tested using the ‘testing1’ dataset. Prediction will be performed for the testing data and the accuracy will be calculated.



```
>       t_pred <- predict(s, newdata = testing1, type='class')
>       confusionMatrix(t_pred,testing1$purchase,)
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  3057  108
       yes  595 3739

               Accuracy : 0.9063
                 95% CI : (0.8994, 0.9128)
    No Information Rate : 0.513
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8117

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8371
            Specificity : 0.9719
         Pos Pred Value : 0.9659
         Neg Pred Value : 0.8627
             Prevalence : 0.4870
         Detection Rate : 0.4077
   Detection Prevalence : 0.4221
      Balanced Accuracy : 0.9045

       'Positive' Class : no
```

The purchase decisions are predicted using the predict() function which takes the classifier and the testing1 data set as input and generates the predicted values of the purchase variable. The predicted values are stored in t_pred. Confusion Matrix calculates the final accuracy score for the prediction model used on the testing1 dataset. It calculates it by comparing the purchase predictions by the model with the actual purchase values present in the testing1 dataset. The confusionMatrix() function from the e1071 library is used to calculate the scores. The Support Vector Machine algorithm generates 595 false positives which is a Type 1 error and 108 false negatives which is a type 2 error. The false positive score of 595 means that Support Vector Machine has a higher possibility of predicting a decision as ‘yes’ when it is actually ‘no’. However, the false negative count is 108 which means that there is still a significant Type 2 error at the same time with the Support Vector Machine classification algorithm. The overall accuracy of the algorithm is 90.63%. The precision is further calculated as below to obtain the value of 0.8627 or 86.27%.

$$\textbf{\textit{Precision}} = \frac{TP}{TP + FP}$$

Comparative Analysis

The following table gives the comparative analysis of the different classification algorithms used for the analysis.

Parameters/Prediction Models	CART	C5.0	Random Forest	Naïve Bayes	SVM
Accuracy	79.54%	92.69%	92.32%	62.31%	90.63%
Precision	86.47%	87.53%	87.34%	62.62%	86.27%
False Positive (Type 1 error)	429	548	554	1511	595
False Negative (Type 2 error)	1105	0	22	1315	108

Accuracy (Higher is Better)

The accuracy of a machine learning model can be calculated by the sum of true positives and true negatives divided by the total dataset. In other words, how many predictions could the model get right relative to the dataset. Based on the accuracy of prediction, C5.0 is marginally higher by 0.37% than Random Forest on this particular problem. Support Vector Machine is a close third at 90.63%. Classification and Regression Tree has an accuracy of 79.54% which is still acceptable for a decision-based problem, however, Naïve Bayes has a lower accuracy comparatively at 62.31%.

Precision (Higher is Better)

The next parameter is the precision of the models, which is the ratio of true positives divided by the sum of true positives and false positives. The precision of a model measures how often the model is correct when it predicts ‘yes’. The precision of C5.0 is the highest at 87.53% followed by the Random Forest at 87.34%. However, things change a bit with precision as the CART algorithm has a higher precision at 86.47% while the Support Vector Machine is a close 86.27%. This means that the CART algorithm will have lower false positives compared to the Support Vector Machine. The Naïve Bayes again has the lowest precision at 62.62% among others.

False Positive (Lower is Better)

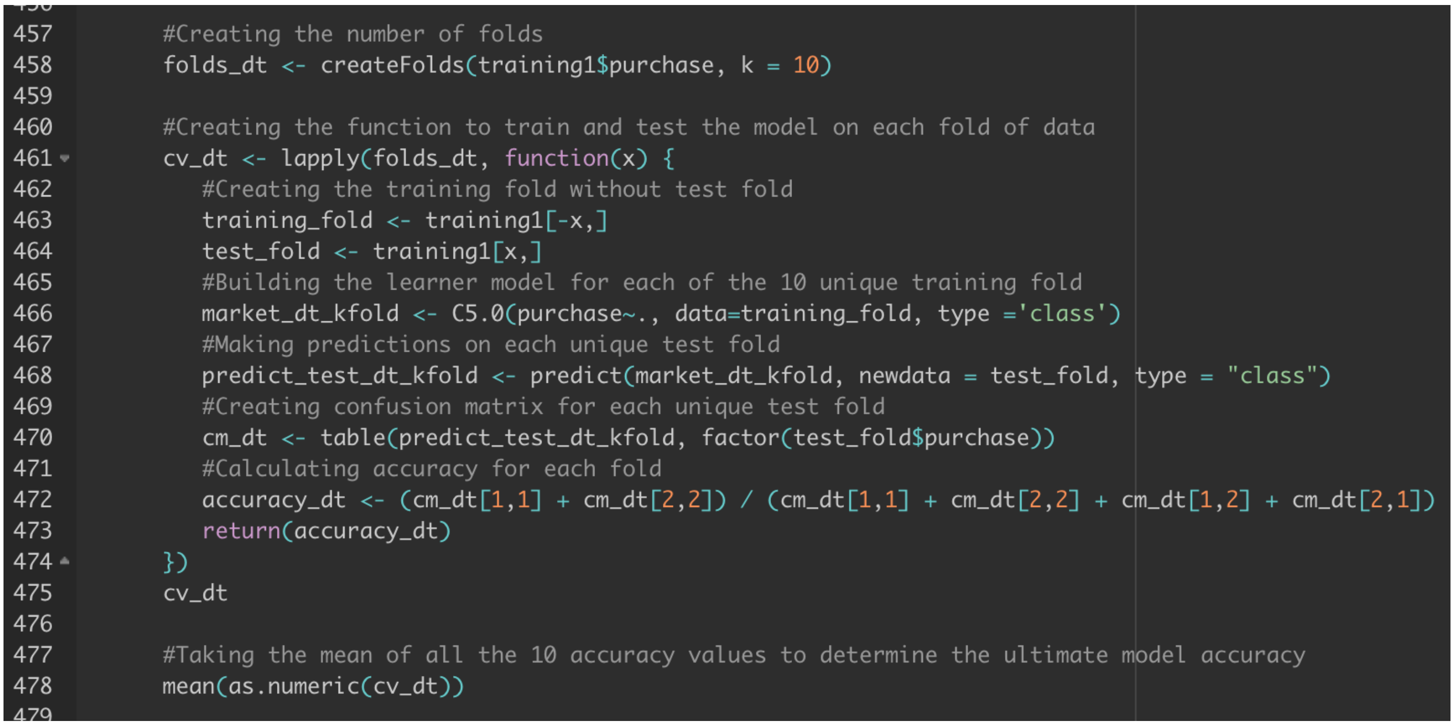
False Positives means how often does the model wrongly predicts a 'yes' for the 'no' decisions. CART algorithm has the lowest of all the false positives among the models, which means that the model will accurately predict a 'yes'. The C5.0 comes a distant second at 548 followed by Random Forest and SVM at 554 and 595 respectively. Naïve Bayes has the highest count of false positives at 1511 which is not ideal for the problem.

False Negative (Lower is Better)

False Negative means how often does the model wrongly predicts a 'no' when the decision is a 'yes'. In this case, C5.0 is at the top with 0 false negatives which means that this algorithm will accurately predict a 'no'. The closest second and third is Random Forest and SVM at 22 and 108 respectively which is also acceptable. However, the algorithms CART and Naïve Bayes have the highest False Negatives at 1105 and 1315 respectively. This algorithm could lose potential customers.
Based on the comparison of the above parameters, it can be observed that C5.0 gives the best prediction results for the test dataset.

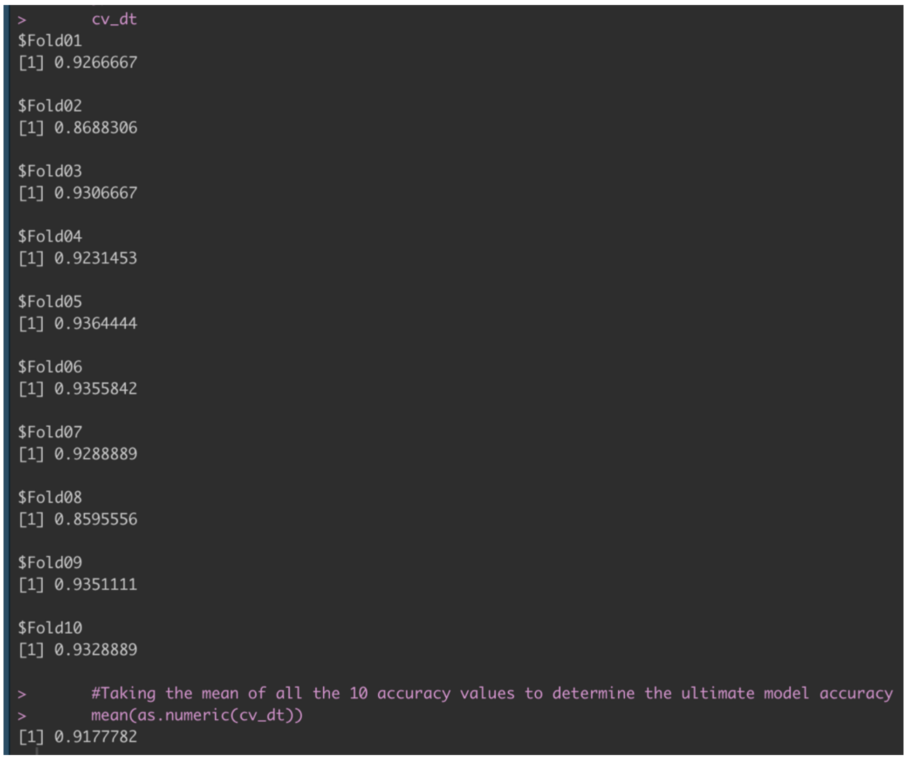
Cross Validation of the Models

Additional validation is performed on the top three algorithms, C5.0, Random Forest and Support Vector Machine to address the biases and overfitting, if present, reflected by testing scores, using a resampling technique called k-fold. Random samples n is selected from the training dataset and split into training fold and test fold which will be the validation dataset. The model built for each fold is then used for prediction using the validation dataset. The accuracies are then calculated for each fold. The mean of the accuracies is then calculated to give the final accuracy of the models.
The createFolds() function takes the target variable and the number of folds which is assigned 10 in this case.
The function lapply() is used to apply the selected algorithm to each sample or fold. The function passed in lapply() consists of creating training and test folds followed by creating the classifier. The C5.0 classifier is selected below. The prediction is then performed on each fold and the prediction accuracy is generated for each of them.



```
#Creating the number of folds
folds_dt <- createFolds(training1$purchase, k = 10)

#Creating the function to train and test the model on each fold of data
cv_dt <- lapply(folds_dt, function(x) {
   #Creating the training fold without test fold
   training_fold <- training1[-x,]
   test_fold <- training1[x,]
   #Building the learner model for each of the 10 unique training fold
   market_dt_kfold <- C5.0(purchase~., data=training_fold, type ='class')
   #Making predictions on each unique test fold
   predict_test_dt_kfold <- predict(market_dt_kfold, newdata = test_fold, type = "class")
   #Creating confusion matrix for each unique test fold
   cm_dt <- table(predict_test_dt_kfold, factor(test_fold$purchase))
   #Calculating accuracy for each fold
   accuracy_dt <- (cm_dt[1,1] + cm_dt[2,2]) / (cm_dt[1,1] + cm_dt[2,2] + cm_dt[1,2] + cm_dt[2,1])
   return(accuracy_dt)
})
cv_dt

#Taking the mean of all the 10 accuracy values to determine the ultimate model accuracy
mean(as.numeric(cv_dt))
```

The function outputs the accuracies for each fold as seen below.



```
>       cv_dt
$Fold01
[1] 0.9266667

$Fold02
[1] 0.8688306

$Fold03
[1] 0.9306667

$Fold04
[1] 0.9231453

$Fold05
[1] 0.9364444

$Fold06
[1] 0.9355842

$Fold07
[1] 0.9288889

$Fold08
[1] 0.8595556

$Fold09
[1] 0.9351111

$Fold10
[1] 0.9328889

>       #Taking the mean of all the 10 accuracy values to determine the ultimate model accuracy
>       mean(as.numeric(cv_dt))
[1] 0.9177782
```

The mean of the accuracies is then calculated. The C5.0 algorithm gets an accuracy score of 91.77% as seen above. Similarly, the function is built for Random Forest and Support Vector Machine. The following table consists of the mean accuracies obtained as a result of the k-fold cross validation.

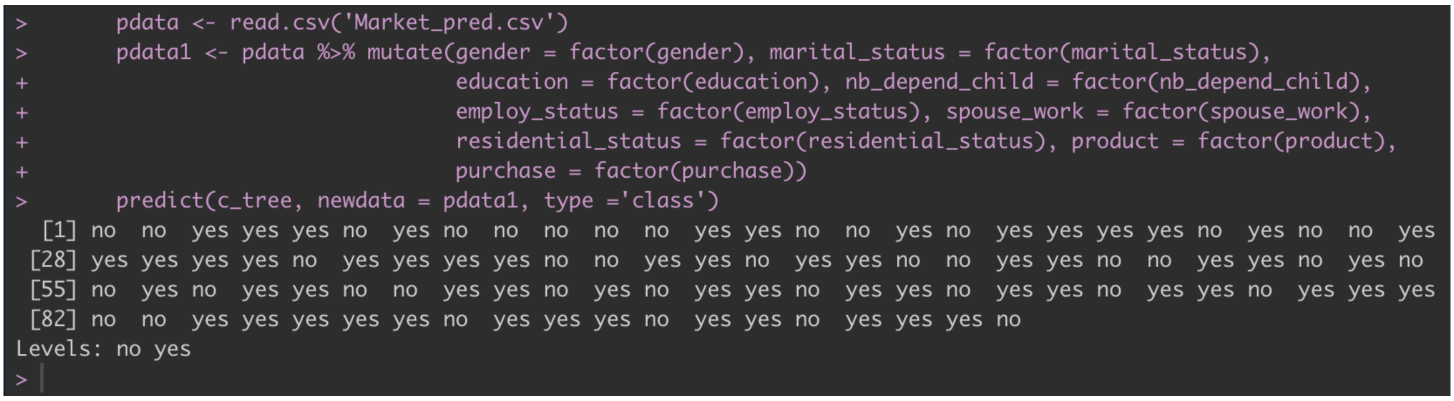
	C5.0	Random Forest	SVM
Confusion Matrix	92.69%	92.32%	90.63%
K-Fold Cross Validation	91.77%	92.83%	90.87%

It is observed that the k-fold cross validation techniques do not provide accuracy results that deviate far apart from those given by the confusion matrix. The Random Forest has the highest accuracy at 92.83% while C5.0 has a minor difference of 1.06% at 91.77% accuracy. Meanwhile, SVM has an accuracy of 90.87% which is almost equal to that given by the Confusion Matrix.

Considering the False Negative count of 0 given by C5.0 in the confusion matrix, and the overall accuracy scores by both the Confusion Matrix and K-Fold Cross Validation, **C5.0 algorithm** will be best suited for the dataset.

Final Prediction

As per the evaluation, C5.0 algorithm is used to predict the purchase decisions in the dataset 'Market_pred.csv'. The dataset is imported using read.csv() function and the columns are converted to factors before performing the prediction to bring the dataset in line with the dataset used to train the classifier 'c_tree'. The predict() function then gives the following values for the 'purchase' column.



```
>       pdata <- read.csv('Market_pred.csv')
>       pdata1 <- pdata %>% mutate(gender = factor(gender), marital_status = factor(marital_status),
+                                  education = factor(education), nb_depend_child = factor(nb_depend_child),
+                                  employ_status = factor(employ_status), spouse_work = factor(spouse_work),
+                                  residential_status = factor(residential_status), product = factor(product),
+                                  purchase = factor(purchase))
>       predict(c_tree, newdata = pdata1, type ='class')
  [1] no  no  yes yes yes no  yes no  no  no  no  no  yes yes no  no  yes no  yes yes yes yes no  yes no  no  yes
 [28] yes yes yes yes no  yes yes yes yes no  no  yes yes no  yes yes no  no  yes yes no  no  yes yes no  yes no
 [55] no  yes no  yes yes no  no  yes yes no  yes no  yes yes no  yes yes no  yes yes no  yes yes no  yes yes yes
 [82] no  no  yes yes yes yes yes no  yes yes yes no  yes yes no  yes yes yes no
Levels: no yes
>
```

Final Recommendation to ABSA

The descriptive analytics section revealed a lot of trends among the customers who have purchased the investment products. As the past customer records are used to train the prediction model, the data trends obtained in the predicted dataset will remain the same. Based on these purchase trends of the investment products, the following group of customers can be targeted.

1. The maximum purchases of the products were made by customers between the age groups 20 and 30. This age group consists of high-risk taking individuals who try different volatile and non-volatile investment products to build quick wealth and also for the long term. The risk-taking capacity of an individual is inversely proportional to financial responsibilities. This age group consists of mostly unmarried individuals with no children and therefore have the privilege of taking higher risks. This age group of customers between 20-30 years can be targeted for marketing the product.
2. Individuals having the least years at the current job can be targeted as there can be seen a higher trend in purchase at the start of the job. This happens mostly because the individual carries out diversified investment at the start of the new job for the long run and as the years progresses at the current job, there is promotion and increase in the salary which decreases the need to purchase any more investment product as the individual already becomes financially stable.
3. Based on the decreasing trend observed with the number of dependent children, customers with 0 dependent children can be targeted as they can be assumed to be of two groups of single and newly married. There can be a high-risk taking capability during the stages of low financial responsibilities. These group of individuals are likely to invest in volatile investment products compared to individuals having children. As the number of dependent children increases the risk-taking ability decreases and hence the decreasing trend.
4. The purchases made by full-time employed individuals are significantly higher and can therefore be targeted for selling the investment products. Job security and steady income with full-time employment plays an important role in investment decisions and can be seen to have a higher investment trend among this group of individuals.
5. The total years employed of the range between 0-5 years have been observed to have the highest number of purchases. This can be explained by the fact that such individuals are at the start of their careers and are planning to build a stable financial future.

In conclusion, individuals between the age of 20-30 having a full-time job, lower number of years at current job, between 0-5 years of total employment and finally having no dependent children can be targeted by the ABSA sales team for marketing bonds, derivatives and stocks.

Additional Data that can be Useful

The dataset containing past customer records have helped explore, analyze and conclude the analysis for the potential customers that ABSA sales team can contact. However, following additional data can also help derive better results from the analysis.

1. There can be other family members apart from children that can be financially dependent on the customer, therefore dependent members count would be a better parameter than dependent children.
2. The number of current ongoing investments and their types can be a highly important parameter to decide what type investment will that customer choose based on his investment portfolio.
3. Educational background like finance, account and arts can be a contributing factor that conveys the customers' knowledge and interest in finance thereby directing the purchase.

R Code Developed

```
library('dplyr')

  tdata <- read.csv('Market.csv')

  tdata
  str(tdata)

  # gender, marital_status, education, nb_depend_child, employ_status, spouse_work,
  # residential_status, product and purchase can be categories.

  #Checking for missing or NA values in the dataset

  tdata %>%
summarise(length(gender),length(age),length(marital_status),
length(education),length(nb_depend_child),
length(employ_status),length(yrs_current_job),
length(yrs_employed),length(net_income),
length(spouse_work),length(spouse_income),length(residential_status),
length(yrs_current_address),  length(product),length(purchase)).


  levels(as.factor(is.na(tdata)))

  #find the levels in the categories identified above.

  # Converting the categories to factors.

  tdata1 <- tdata %>% mutate(gender = factor(gender), marital_status = factor(marital_status),
                   education = factor(education), nb_depend_child = factor(nb_depend_child),
                   employ_status = factor(employ_status), spouse_work = factor(spouse_work),
                   residential_status = factor(residential_status), product = factor(product),
                   purchase = factor(purchase))
```

```
   #Calculating the descriptive statistics for numerical columns

     library(psych)



     tdata1 %>% filter(purchase == 'yes') %>% select(age, yrs_current_job, yrs_employed, net_income,
                spouse_income, yrs_current_address) %>% describe()


# Descriptive Analytics

  #Categorical Columns

    #gender bar chart


    plot(tdata1 %>% filter(purchase == 'yes') %>% select(gender))

    tdata1 %>% filter(purchase == 'yes',gender == 'F') %>% summarise(length(gender))

    #marital_status bar plot


    plot(tdata1 %>% filter(purchase == 'yes') %>% select(marital_status))

    tdata1 %>% filter(purchase == 'yes', marital_status == 'widowed') %>% summarise(length(gender))

    #education bar plot

    plot(tdata1 %>% filter(purchase == 'yes') %>% select(education))

    tdata1 %>% filter(purchase == 'yes', education == 'univ') %>% summarise(length(education))

    #nb_depend_child bar plot

    plot(tdata1 %>% filter(purchase == 'yes') %>% select(nb_depend_child))

    tdata1 %>% filter(purchase == 'yes', nb_depend_child == '3') %>% summarise(length(nb_depend_child))

    #employ_status bar plot

    plot(tdata1 %>% filter(purchase == 'yes') %>% select(employ_status))

    tdata1 %>% filter(purchase == 'yes', employ_status == 'unemployed') %>%
summarise(length(employ_status))

    #spouse_work bar plot

    plot(tdata1 %>% filter(purchase == 'yes') %>% select(spouse_work))

    tdata1 %>% filter(purchase == 'yes', spouse_work == 'no') %>% summarise(length(spouse_work))
    #residential status bar plot

    plot(tdata1 %>% filter(purchase == 'yes') %>% select(residential_status))

    tdata1 %>% filter(purchase == 'yes', residential_status == 'w_parents') %>%
summarise(length(residential_status))


    #product bar plot

    plot(tdata1 %>% filter(purchase == 'yes') %>% select(product))

    tdata1 %>% filter(purchase == 'yes', product == 'stocks') %>% summarise(length(product))

  #Numerical Columns

    install.packages('Hmisc')
    library(Hmisc)

   #plotting histograms for the numerical columns

      hist.data.frame(tdata1 %>% filter(purchase == 'yes') %>%
      select(age,yrs_current_job, yrs_employed, net_income,
      spouse_income, yrs_current_address))

# Information gain

      install.packages('FSelector')
      library(FSelector)

      information.gain(purchase ~.,training1,unit='log2')



#Creating training and testing data

      library(caret)

      randm1 <- createDataPartition(tdata1$purchase, p=0.75, list=F)
      training1 <- tdata1[randm1,]
      testing1 <- tdata1[-randm1,]


      str(training1)
      str(testing1)
```

```
#CART (Classification and Regression Tree)

    install.packages('rpart')
    library(rpart)

    class_tree <- rpart(purchase ~.,data=training1)
    class_tree

    #plotting the tree

    install.packages('rpart.plot')
    library(rpart.plot)

    rpart.plot(class_tree,main='Tree for Purchase', extra=108)

    t_pred <- predict(class_tree, newdata = testing1, type="class")

    # the accuracy of the model


 #confusion matrix

      library(e1071)

      confusionMatrix(t_pred,testing1$purchase)



#C5.0

    install.packages("C50")
    library('C50')

    c_tree <- C5.0(purchase ~.,data=training1)

    plot(c_tree)

    t_pred <- predict(c_tree, newdata = testing1, type ='class')

    confusionMatrix(t_pred, testing1$purchase)

# Naive Bayes #

    n <- naiveBayes(purchase~.,data=training1)


    t_pred <- predict(n,newdata = testing1, type='class')


    confusionMatrix(t_pred,testing1$purchase,)
```

```
#svm
    s <- svm(purchase~.,data=training1)


    t_pred <- predict(s, newdata = testing1, type='class')

    confusionMatrix(t_pred,testing1$purchase,)



#Random Forest

    library("randomForest")

    r <- randomForest(purchase~.,data =training1)

    plot(r)

    t_pred <- predict(r, newdata = testing1)

    confusionMatrix(t_pred,testing1$purchase)


#Applying k-Fold Cross Validation to Decision Tree Model


    #Creating the number of folds

    folds_dt <- createFolds(training1$purchase, k = 10)

    #Creating the function to train and test the model on each fold of data

    #Random Forest

      cv_dt <- lapply(folds_dt, function(x) {

        training_fold <- training1[-x,]
        test_fold <- training1[x,]

        market_dt_kfold <- randomForest(purchase~., data=training_fold)

        predict_test_dt_kfold <- predict(market_dt_kfold, newdata = test_fold, type = "class")

        cm_dt <- table(predict_test_dt_kfold, factor(test_fold$purchase))

        accuracy_dt <- (cm_dt[1,1] + cm_dt[2,2]) / (cm_dt[1,1] + cm_dt[2,2] + cm_dt[1,2] + cm_dt[2,1])

        return(accuracy_dt)
      })
      #Taking the mean of all the 10 accuracy values to determine the ultimate model accuracy

      mean(as.numeric(cv_dt))


    #C5.0

      cv_dt <- lapply(folds_dt, function(x) {

        training_fold <- training1[-x,]
        test_fold <- training1[x,]

        market_dt_kfold <- C5.0(purchase~., data=training_fold)

        predict_test_dt_kfold <- predict(market_dt_kfold, newdata = test_fold, type = "class")

        cm_dt <- table(predict_test_dt_kfold, factor(test_fold$purchase))

        accuracy_dt <- (cm_dt[1,1] + cm_dt[2,2]) / (cm_dt[1,1] + cm_dt[2,2] + cm_dt[1,2] + cm_dt[2,1])

        return(accuracy_dt)
      })


      #Taking the mean of all the 10 accuracy values to determine the ultimate model accuracy

      mean(as.numeric(cv_dt))


#SVM

      cv_dt <- lapply(folds_dt, function(x) {

        training_fold <- training1[-x,]
        test_fold <- training1[x,]

        market_dt_kfold <- svm(purchase~., data=training_fold)

        predict_test_dt_kfold <- predict(market_dt_kfold, newdata = test_fold, type = "class")

        cm_dt <- table(predict_test_dt_kfold, factor(test_fold$purchase))

        accuracy_dt <- (cm_dt[1,1] + cm_dt[2,2]) / (cm_dt[1,1] + cm_dt[2,2] + cm_dt[1,2] + cm_dt[2,1])

        return(accuracy_dt)
      })
```

```
      #Taking the mean of all the 10 accuracy values to determine the ultimate model accuracy

      mean(as.numeric(cv_dt))


#Using the C5.0 classifier c_tree generated previously on the market pred dataset

    pdata <- read.csv('Market_pred.csv')

    # Converting the categories to factors.

    pdata1 <- pdata %>% mutate(gender = factor(gender), marital_status = factor(marital_status),
                     education = factor(education), nb_depend_child = factor(nb_depend_child),
                     employ_status = factor(employ_status), spouse_work = factor(spouse_work),
                     residential_status = factor(residential_status), product = factor(product),
                     purchase = factor(purchase))

    p_pred <- predict(c_tree, newdata = pdata1, type ='class')
    p_pred

    pdata1$purchase <- p_pred

    str(pdata1)
```