IS6052 DESCRIPTIVE AND PREDICTIVE ANALYTICS

INDIVIDUAL CA – ABSA BANK PRODUCT MARKETING

Mayur Shailesh Sawant: 120220382

Submission Date: 7th May, 2021

Table of Contents

Executive Summary

The Marketing Analyst of ABSA Bank is responsible for helping ABSA Bank strategic decisions for their strategizing their marketing and promotional activities. Currently, the analyst has been handed a dataset ‘market.csv’ which contained historical records about the bank’s customers who were previously contacted for promoting the 3 investment products – bonds, stocks and derivatives, offered by the bank. These records primarily contained the purchase decisions of the contacted customers along with a few personal details of those customers.

ABSA Bank is now planning to reach out to more customers for promoting their products, however the sales team has been allotted limited hours to contact these customers. Thus, it is essential that they make optimal use of these hours. In order to assist the sales team to make better use of the limited time resource it necessary to recommend them the customers who are most likely to purchase the product upon contacting them.

In this quest, the Marketing Analyst initially analyzed the historical data and attempted to establish relationships between the different features of the customers and their final purchase decision. It was determined that customer features like product being marketed, age, number of dependent children, total employment years, net income and marital status were having an impact on the final purchasing decision of the customers in the written order of significance. The trends indicated that customers in the age group of 21 to 30 years, having zero or one child, fewer years of employment were more likely to purchase the product.

Further, the analyst was also given a dataset ‘market_pred.csv’ of customers who were to be contacted for product marketing in the future. In order to predict which customers are likely to buy the product, predictive analytics was used. The historical data was split into training and test sets and the training set was used to train and build 4 machine learning models which were:

1. Decision Tree Classification
2. Random Forest Classification
3. Naïve Bayes Classification
4. Support Vector Machines (SVM)

After the models were built, they were tested on the test set and their performance was evaluated and compared using confusion matrix. The 2 best performing models were made to undergo another evaluation using k-Fold Cross Validation for ensuring that the accuracies obtained are reliable and there are no instabilities in the model. Finally, using the best performing model, predictions were made on the ‘market_pred.csv’ dataset and the purchase decisions of the customers were predicted. These purchase decisions, along with a few recommendations has been submitted to the bank’s sales team to plan their future marketing activity.

Eyeballing of the Market Data

The data set 'market.csv' contains 30,000 historical records about ABSA Bank's customers which has numerous personal details about all the customers like age, genders, income etc. The main dependent variable in the given dataset is the decision about purchasing one of the 3 investment products offered by ABSA bank – stocks, bonds and derivatives.

The dataset was first read using 'read.csv ()' function in R and then its structure was examined.

Code Snippet

```
#Exploratory Analysis of the Data set

#Importing the dataset
market <- read.csv("Market.csv")
head(market)

summary(market)

str(market)

```

Variables and their Data Types

Amongst all the explanatory and dependent variables in the dataset, some are of numerical type while others are of categorical type.

Integer Type

- age
- nb_depend_child (Number of dependent children)
- yrs_current_job (Numbers of years at the current job)
- yrs_employed (Total number of years the person has been employed for)
- net_income (Annual income of the person)
- spouse_income (Annual income of the spouse given that he/she is employed)
- yrs_current_address (Number of years spent living at the current address)

Character (Categorical) Type

- gender - 2 categories: M (Male), F (Female)
- marital_status - 4 categories: divorced, married, single, widowed)
- education - 4 categories: basic, highsch (high school), postgrad, univ (university)
- employ_status - 4 categories: full_time, part_time, retired, self_employ, unemployed
- spouse_work – 2 categories: yes, no
- residential_status: 4 categories: owner, owner_morg, tenant, w_parents
- product: 3 categories: bonds, derivatives, stocks
- purchase: 2 categories: yes, no

Initially, to proceed with the descriptive and predictive analytics which involved building classification models, it was necessary to convert all the character (categorical) parameters to factors.

Code Snippet

```
#Converting the categorical variables into factors
market$gender <- as.factor(market$gender)
market$marital_status <- as.factor(market$marital_status)
market$education <- as.factor(market$education)
market$employ_status <- as.factor(market$employ_status)
market$spouse_work <- as.factor(market$spouse_work)
market$residential_status <- as.factor(market$residential_status)
market$product <- as.factor(market$product)
market$purchase <- as.factor(market$purchase)

```

Descriptive Analytics of the Market Data

Firstly, all the different parameters were analyzed for their impact on the dependent variable, which in this case is the purchasing decision for the product. For this analysis, information gain of the explanatory attributes was calculated with respect to the purchase attribute. Also, it should be noted that for the purpose of descriptive analytics of the given data, the data related to the customers who had previously purchased ABSA Bank's investment products has been analyzed in order to understand the relation of different parameters with the purchase decision 'yes'.

Code Snippet

```
#Information gain of the parameters
library(FSelector)
market_information_gain <- information.gain(purchase~., market)
market_information_gain %>% arrange(desc(attr_importance))
```

Output

```
                    attr_importance
product                0.0411421923
age                    0.0309859060
nb_depend_child        0.0158975320
yrs_employed           0.0149834405
net_income             0.0128183573
marital_status         0.0126433405
yrs_current_job        0.0054506218
yrs_current_address    0.0042054235
spouse_work            0.0039437468
spouse_income          0.0039371980
gender                 0.0012636246
education              0.0004595292
employ_status          0.0004501333
residential_status     0.0002307776
>
```

The top 6 attributes in the data set which were most important with respect to the purchasing decision were:

1. Product
2. Age
3. Number of dependent children
4. Total employment years

5. Net income
6. Marital status

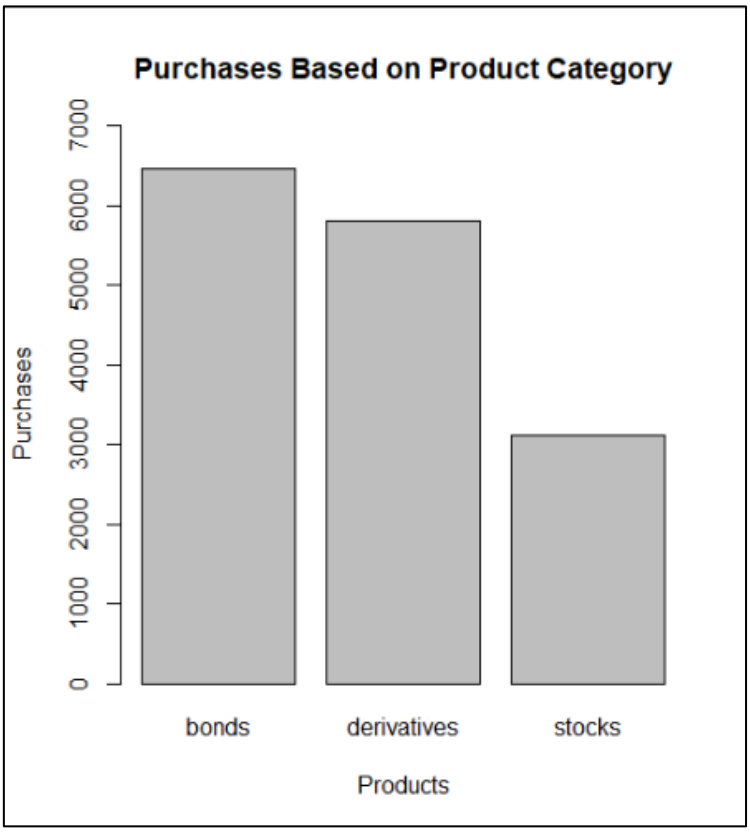
Product

As evident from the market data set, there are 3 investment products offered by ABSA Bank – stocks, bonds and derivatives. Each of the 3 products was almost evenly marketed, i.e., about 10,000 people were contacted for every product. Out of the total 30,000 customers contacted, ABSA Bank was able to sell their products to 15,391 customers.

Code Snippet

```
#DESCRIPTIVE ANALYTICS OF THE DATA
library(dplyr)

##############################  PRODUCT ######################################
#Relation of the product with its purchase
plot(market %>% filter(purchase=="yes") %>% select(product),
     xlab = "Products",
     ylab = "Purchases",
     main = "Purchases Based on Product Category",
     ylim = c(0,7000))

```

Output



Now, upon filtering the data set based on the customers who purchased the products, it was observed that bonds were the most purchased product by the customers, followed by derivatives and stocks. ABSA Bank sold approximately 6500 bonds, 5800 derivatives and 3100 stocks. This provides an insight for the bank as to which product needs more marketing efforts.

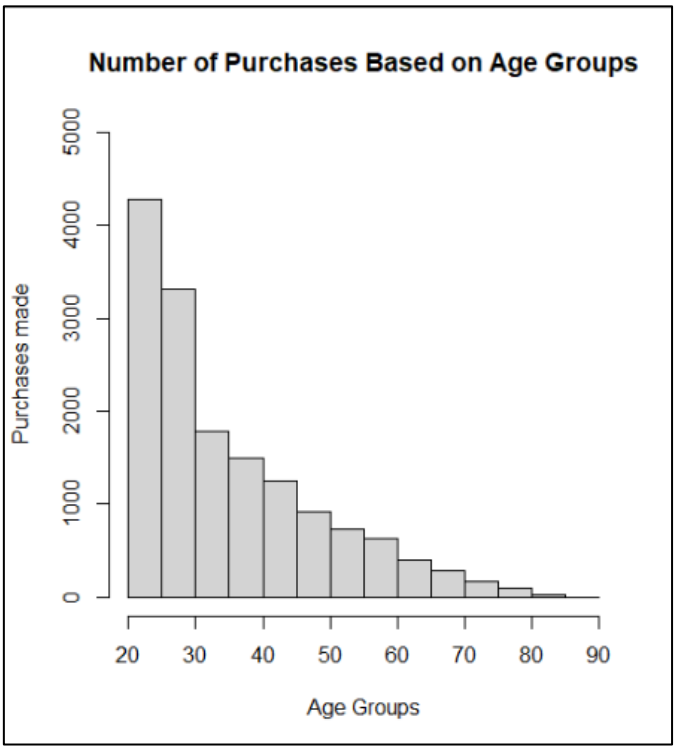
Age

According to the data set, the customers contacted by ABSA Bank for their product marketing were between 20 to 88 years old. The average age of the bank's customers is about 37.5 years. When analyzing the age data of the customers who had previously purchased the bank's products, it was observed that the absolute frequency was the highest for 21 years. This indicates that among all other age groups, most products were purchased by 21-year-old customers. Another interesting observation was that with the increase in age, the number of purchases reduced gradually. Customers belonging to an age group of 20 to 35 years had made the most purchases of the bank's investment products.

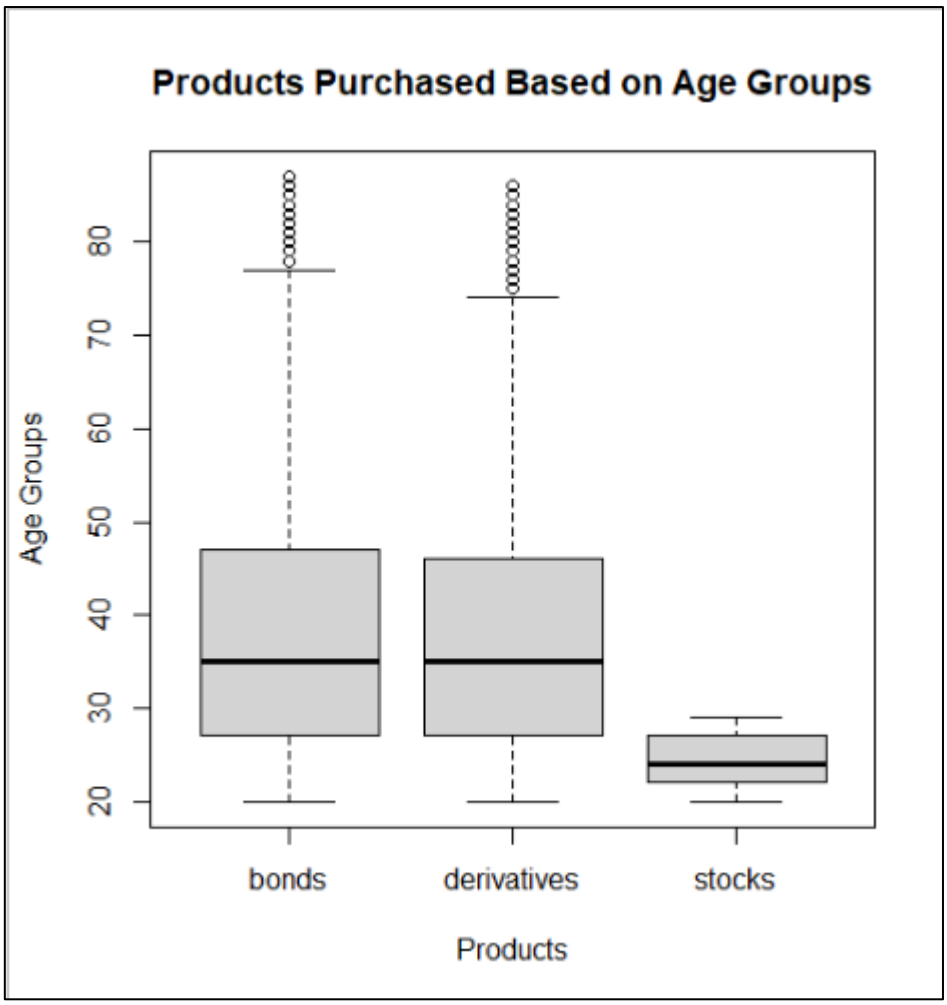
Code Snippet

```
37  ############################  AGE  ##############################
38  #Filtering the data for customers who had purchased the products in the past
39  age_graph <- (market %>% filter(purchase=="yes") %>% select(age))
40  summary(age_graph)
41
42  #Finding absolute frequency
43  sort(table(age_graph))
44
45  #Plotting Age groups vs Purchases
46  hist(age_graph$age, xlab = "Age Groups",
47       ylab = "Purchases made",
48       main = "Number of Purchases Based on Age Groups",
49       ylim = c(0, 5000))
50
51  #Plotting products vs age groups
52  plot(market %>% filter(purchase=="yes") %>% select(product, age), ylab = "Age Groups",
53       xlab = "Products",
54       main = "Products Purchased Based on Age Groups")
55
```

Output



A possible reason for this might be that it is around their 20s and 30s wherein most people get into jobs and are keen upon exploring investment options for securing their futures. People in their early 20s learn about the investment options anew and are curious to explore them. The bank should thus focus more on this younger age group while devising their future marketing campaign.



Also, it was observed that people in the age group 20 to 30, are majorly buying stocks related products. This might be because, stocks are comparatively risky and dynamic than other investment options and younger people are often risk takers as they do not have an immediate responsibility. Thus, while to make sure that the time and resources are utilized efficiently while designing the marketing campaign, ABSA's marketing team should persuade younger customers to buy stocks as the likelihood of that sale would be more.

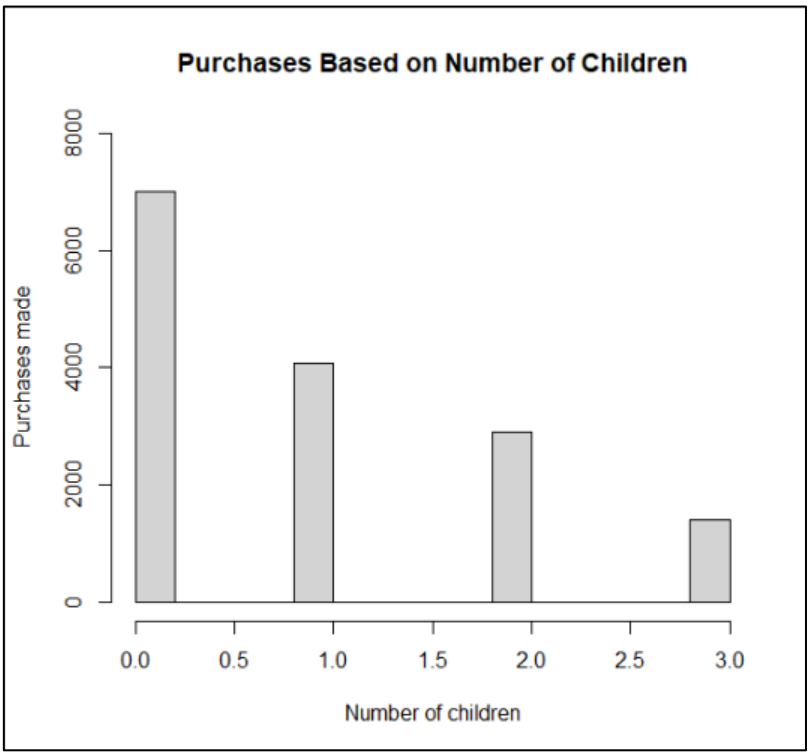
Number of Dependent Children

Customers who were contacted had between 0 to 3 children. Data related to the number of children of those customers who had previously purchased ABSA Bank's products was analyzed. Out of 15,391 customers who had made purchases, 7000 customers had no children while 4082 customers had a single child. People having more than 1 children had made comparatively less purchases.

Code Snippet

```
56  ############################  NUMBER OF CHILDREN  ################################
57  #Filtering the data for customers who had purchased the products in the past
58  children_graph <- (market %>% filter(purchase=="yes") %>% select(nb_depend_child))
59
60  summary(children_graph)
61
62  #Finding absolute frequency
63  sort(table(children_graph))
64
65  #Plotting the relation between number of children and purchases
66  hist(children_graph$nb_depend_child, xlab = "Number of children",
67       ylab = "Purchases made",
68       main = "Purchases Based on Number of Children",
69       ylim = c(0, 8000))
70
```

Output



This observation can be justified by the fact that most of these customers are very young and hence had no or one child and, in such cases, the overall expenditure on their children is saved and this excess money is likely to be invested. Also, as there are no responsibility dependent children, this category of customers tends to invest in comparatively risky investment products like stocks. Therefore, ABSA Bank should primarily target such customers because according to the past data, lesser the number of dependent children, the more likely the customers are to purchase their products.

Total Employment Years

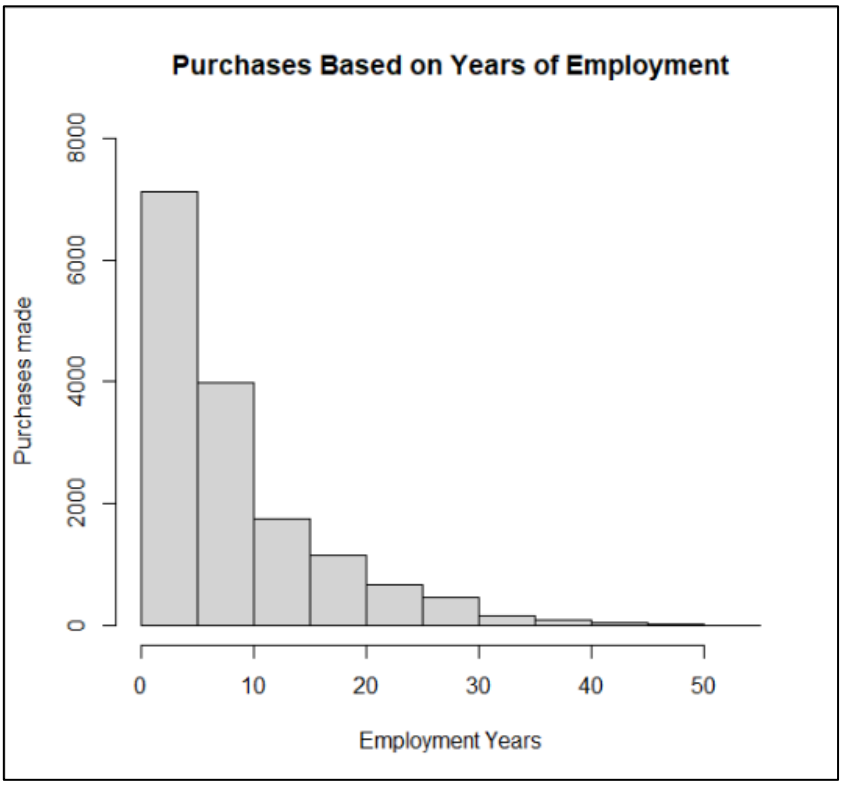
Upon analyzing the employment years data of the customers who had previously purchased ABSA Bank's products, it was observed that this attribute follows a similar trend like all other attributes before. For instance, customers with employment years 0, 1, 2, 3 years had made 604, 944, 2322, 1226 purchases respectively whereas customers with more years of

employment like 43, 44, 45, 46 had made only 7, 7, 3, 4 purchases. Thus, ABSA Bank must target their customers with least number of employment years in order to increase the likelihood of their sales.

Code Snippet

```
71  ######################  YEARS EMPLOYED  ########################################
72  #Filtering the data for customers who had purchased the products in the past
73  employment_graph <- (market %>% filter(purchase=="yes") %>% select(yrs_employed))
74
75  summary(employment_graph)
76
77  #Finding absolute frequency
78  sort(table(employment_graph))
79
80  #Plotting the relation between number of years employed and purchases
81  hist(employment_graph$yrs_employed, xlab = "Employment Years",
82       ylab = "Purchases made",
83       main = "Purchases Based on Years of Employment",
84       ylim = c(0, 8000))
85
```

Output



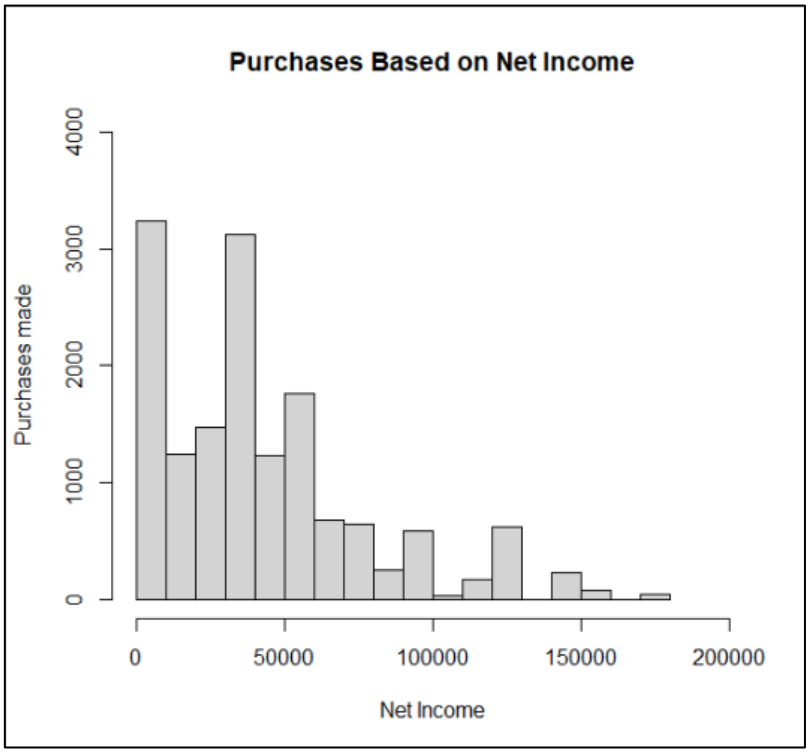
Net Income

While analyzing the net income data about the customers who had previously purchased the products, an anomaly was observed. Customers with no income had the highest previous purchases. Also, it was observed that many of such customers has been employed and still their net income was zero. Thus, it was assumed during the analysis that the net income of such customers was not available and hence was considered as zero while cleaning the data and thus, has not been considered for the sake of this descriptive analysis.

Code Snippet

```
################################  NET INCOME   ##############################
#Filtering the data for customers who had purchased the products in the past
net_income_graph <- (market %>% filter(purchase=="yes") %>% select(net_income))

summary(net_income_graph)

#Finding absolute frequency
sort(table(net_income_graph))

#Plotting the relation between Net Income and purchases
hist(net_income_graph$net_income, xlab = "Net Income",
     ylab = "Purchases made",
     main = "Purchases Based on Net Income",
     xlim = c(0,200000),
     ylim = c(0, 4000))

#Plotting the relation between number of years employed and purchases
plot(market %>% filter(purchase=="yes") %>% select(product, net_income))

```

Output



Upon plotting the relation between the net income and number of purchases, it was seen that people with income in the range of 30,000 to 40,000 had very high number of purchases. Also, when the products purchased were plotted against the net income, it was seen that customers with higher incomes did not invest in stocks.

Marital Status

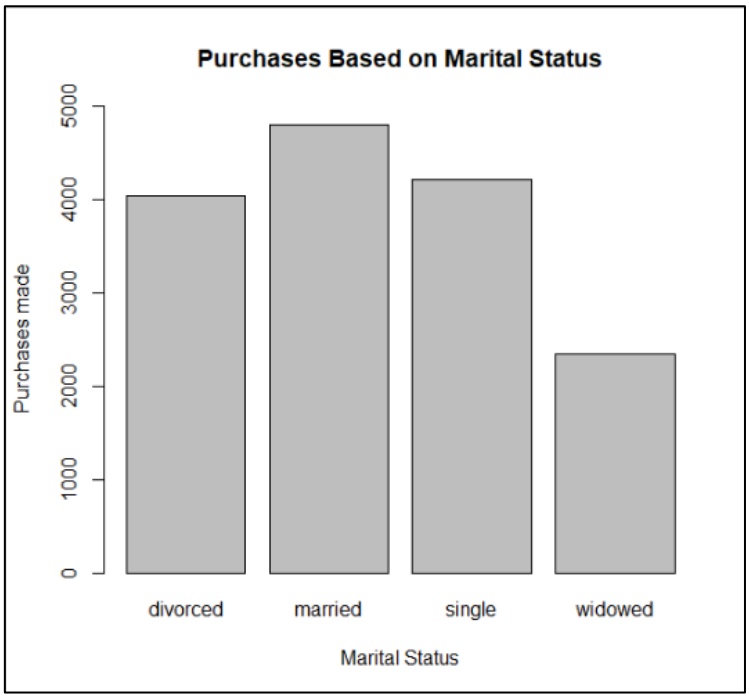
The marital status data of the customers who had previously purchased the products reveal that customers who were married had made the highest number of purchases which were in the range of 4800, followed by customers who were single. There was no distinctive trend observed for this feature as there was not a significant difference in the number of purchases

made by married, single and divorced customers. The customers who are widowed however made the least number of purchases.

Code Snippet

```
105 ############################   MARITAL STATUS  ##########################
106 #Plotting the relation between marital status and purchases
107 plot(market %>% filter(purchase=="yes") %>% select(marital_status),
108      xlab = "Marital Status",
109      ylab = "Purchases made",
110      main = "Purchases Based on Marital Status",
111      ylim = c(0, 5000))
112
```

Output



Overview of the Prediction Methods

For addressing ABSA Bank's marketing problem, classification methods are required for predictions. Classification algorithms are basically used for predicting the class of given observations or data points. They are usually used when the dependent variable is categorical. In the case of given data set "market.csv", the dependent variable 'purchase' is categorical and based on its relationship with the explanatory variables, it can be categorized into 2 classes: 'yes' or 'no'.

3 classification algorithms are used for making predictions in the given problem regarding ABSA Bank. Each of these methods is explained and analyzed below.

Decision Trees Classification

Decision Tree is a machine learning algorithm which is used in both classification and regression models. The model operates in a tree-like structure which answers questions

sequentially while going down a certain route of the tree based on the answer. The model follows a 'if-then' rule set which ultimately provides a specific result.

Advantages

- A straightforward modelling of a single decision tree makes it easy to interpret.
- It is well-suited to handle categorical as well as numerical data.
- Apart from not requiring normalization of the data, decision tree does not require us to manage missing values in the data as it is handled by the decision tree itself.
- Since this method is concerned with only a single tree unlike random forest method, training process of even large datasets is comparatively faster.

Disadvantages

- The risk of overfitting is high with decision trees.
- Even though the decision tree makes the most optimal decision at each step it does not ensure that at by the time it reaches the final node, decision optimality is not guaranteed.
- Decision trees are sensitive to deflections.

Random Forest Classification

It is also a supervised learning algorithm which uses multiple decision trees (forest) unlike decision trees which relies on a single decision tree. Random forest algorithm assembles and works with multiple randomly made decisions and ultimately makes a final decision based on the majority.

Advantages

- It can make better predictions with improved accuracy. This because multiple decision trees enhance its decision-making capability.
- Just like decision tree classification, random forest does not require its data to be normalized or mended for missing data.
- Through random forest classification, optimization parameters can be leveraged to improve the efficiency of the model.

Disadvantages

- Even though it is comparatively less than decision trees, risk of overfitting is still prevalent in in random trees.
- If there are categorical variables in the dataset, each with different level of values, the random forest model can get biased in favor of variables that have more values.
- Use of random forest algorithm for regression, particularly for linearly related data is very limited.

Naïve Bayes Classifier

Naïve Bayes Classifier uses the Bayes' Theorem as its underlying criteria for separating the data into different classes and it assumes that all the predictor variables are independent of each other and that existence of a particular feature in a class is not related to the presence of other features. The Bayes Theorem is given as:

$$P(c \mid x) = (P(x \mid c)\, P(c)) / P(x)$$

Here, ‘c’ represents class while ‘x’ represents the attributes. ‘P (c | x)’ and ‘P (x | c)’ represent the posterior probability of the class with respect to the predictor and that of the predictor with respect to the class, respectively. ‘P(x)’ and ‘P(c)’ stand for prior probability of the predictor and the class, respectively.

Advantages

- It is one of the simplest algorithms to implement and is very fast.
- Naïve Bayes model’s prediction capability is unhampered by noise, irrelevant features or outliers in the dataset, thus eliminating the risk of overfitting.
- Compared to other classification methods, Naïve Bayes model performs better and can be trained easily even with less training data.
- It is less intensive when it comes to computing power and utilizes fewer computing resources compared to other algorithms.

Disadvantages

- Even though the algorithm assumes that all the features are independent, in real world, the features in the datasets are rarely independent of each other. This can lead to more data process steps to suit the model.
- Naïve Bayes algorithm is limited to classification problems and cannot be used for predicting numerical values and thus, it cannot be used in regression.
- Assumption that all features are independent, can cause the algorithm to induce higher bias for certain features.

Support Vector Machines (SVM)

It is also a supervised machine learning algorithm which is mostly used for classification problems. It can however be used for both classification and regression. SVM algorithm plots every single data point in a n-dimensional space, where ‘n’ represents the number of features in the dataset. The value of each feature is the value of the co-ordinate of a particular data point. The model then classifies by identifying the hyperplane that almost most distinctly differentiates the 2 classes.

Advantages

- With higher dimensions of data, i.e., higher number of features, SVM model produces results with very high accuracy.
- Options to choose the right kernel, given the right parameters, provides a high degree of flexibility and diverse toolkit.

Disadvantages

- If the correct parameters are not chosen while building an SVM model, the results can produce substantial inaccuracies and deflections.
- There is no probabilistic explanation for the classification performed by an SVM model as it functions by merely putting data points on either side of the hyperplane.
- SVM models are not inherently scalable and hence they are only suitable for smaller datasets.

Assumptions Made for Prediction Methods

For building any prediction model, a dataset must contain 2 types of variables – explanatory variables and the dependent variable. In the given dataset, it has been assumed that the variable 'purchase' is the dependent or the target variable whose class is to be predicted using the prediction models. Upon looking at the structure of the dataset, there are 2 types of variables – numerical and character. The character variables include 'purchase', 'gender', 'marital_status', 'education' etc. For the sake of building the prediction models, all such variables with character datatype are assumed to be categorical variables, each with distinct classes of values. The dependent or target variable 'purchase' is also assumed to be categorical which forms the basis for the selection of classification models for predictions.

Further, as mentioned previously in this report, it was identified during the descriptive analysis of the data that customers having zero net income despite having 1+ years of employment experience had the highest number of previous purchases compared to the rest of the salary groups. It has thus been assumed that their net income was not available and while cleaning the dataset, N/A or null values must have been replaced by 0.

The decision trees and random forest classification methods usually does not require any assumptions. While implementing the Naïve Bayes method however, it was assumed that the features are independent of each other.

Implementation of Prediction Models

Preparing the Dataset

Before application of any prediction algorithms, it was necessary to make sure that the dataset 'market.csv' is ready to be used by these algorithms. Initially, the dataset was read using the 'read.csv ()' function in R. After this, the structure of the uploaded dataset was examined. It was necessary to convert the variables with character datatype into factors with levels in order to render them efficient while implementing the prediction methods which was done by using 'as.factor ()' function in R.

The provided dataset 'market.csv' did not contain any null or N/A values and thus, it did not require any further cleaning.

Creating Training and Test Set

The next step was splitting the dataset 'market.csv' into training and test set. For the used classification models, the splitting was 75% and 25% for training and testing, respectively. 'createDataPartition ()' function was used from the 'caret' library for splitting the dataset and the output is in the form of indexes of the randomly selected rows from the dataset which is the stored in 'indxTrain' variable. Further, the data related to these indexes is stored in 'train_data' variable and the remaining 25% of data is stored in 'test_data' variable.

Code Snippet

```
################  SPLITTING THE DATA INTO TRAINING AND TEST SETS  ##################
#Loading the required libraries
library(caret) #Classification and Regression Training
library(rpart) #Recursive partitioning and regression trees
library(rpart.plot) #Plotting the rpart model
library(e1071) #Naive Bayes
library(randomForest) #Random Forest Classification

#Splitting the data set into Training and Test Set
indxTrain <- createDataPartition(market$purchase, p=0.75, list=F)

train_data <- market[indxTrain,]
test_data <- market[-indxTrain,]
```

Building the Classification Models

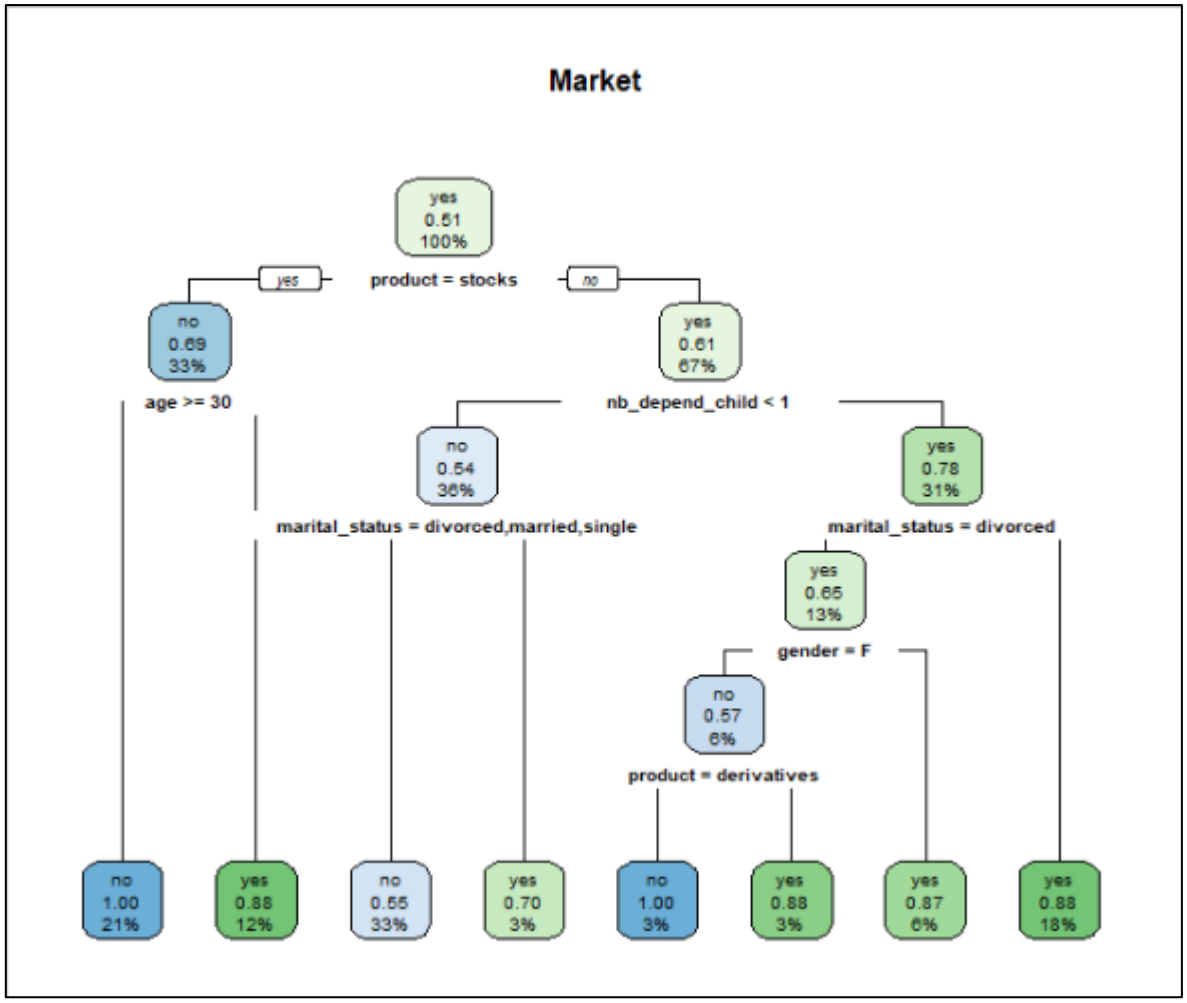
Decision Trees Classification

The 3 libraries used for building a decision tree classification model are ‘caret’, ‘rpart’ and ‘rpart.plot’. The ‘caret’ library uses the CART algorithm of building a decision tree. Initially a control frame is built in which the values for minimum splits and buckets are passed as arguments using the ‘rpart.control ()’ function or the ‘rpart’ library. Next step is building the actual learner model using ‘rpart ()’ function. The first argument represents the 2 entities between whom the relationship is meant to be analyzed. Thus, in this case, to the left of the tilde ‘~’ is target variable ‘purchase’ and to the right side is a dot ‘.’which is a wildcard character representing all the explanatory variables. The second argument is the training data followed by the control frame as the last argument. Finally, the learner model is stored in ‘market_dt’.

Code Snippet

```
############################  DECISION TREES  ################################

#Configuring the control settings
ctrl_dt <- rpart.control(minsplit = 1, minbucket = 1, maxdepth = 10, cp = 0.01)

#Building the learner model
market_dt <- rpart(purchase~., data = train_data, control = ctrl_dt)
market_dt

#Plotting the decision tree
rpart.plot(market_dt, main = "Market", extra = 108)

#Making predictions on the test data
predict_test_dt <- predict(market_dt, newdata = test_data, type = "class")
predict_test_dt
```

Using ‘rpart.plot ()’ function, the decision tree can be visualized. Further, in order to test the learner model, the test data is used for making the predictions about the target variable ‘purchase’ using ‘predict ()’ function and the prediction results are stored in ‘predict_test_dt’ as type class because the target variable ‘purchase’ has 2 categorical classes ‘yes’ or ‘no’.



Random Forest Classification

The library used for random forest classification method is 'randomForest'. The process of building the learner model and making predictions on the test data is similar to that of the decision trees. The only difference is that for random forest, the whole 'market.csv' dataset can be used instead of using the training data as the measures are implicitly taken by the algorithm. However, since the models would be evaluated and compared against one another, it is necessary that they are built on the same training dataset. There, the same training data set has been used for random forest classification as well.

Thus, for building the learner model, 'randomForest ()' function is used and the model is stored in 'market_rf'. The number of trees to be built are set to '500'. The predictions on the test data are made by using the 'predict ()' function and passing the trained model as an argument. The predicted values of the target variables are then stored in 'predict_test_rf' as type class because the target variable 'purchase' has 2 categorical classes 'yes' or 'no'.

Code Snippet

```
143 ###################################  RANDOM FOREST   ########################
144
145 #Building the learner model
146 market_rf <- randomForest(purchase~., data=market, ntree = 500)
147 market_rf
148
149 #Making predictions on the test data
150 predict_test_rf <- predict(market_rf, newdata = test_data, type = "class")
151 predict_test_rf
```

Naïve Bayes Classification

The library used for Naïve Bayes classification is ‘e1071’. The process of building the learner model and making predictions on the test data is similar to that of the decision trees. Thus, for building the learner model, ‘naiveBayes ()’ function is used and the model is stored in ‘market_nb’. The predictions on the test data are made by using the ‘predict ()’ function and passing the trained model as an argument. The predicted values of the target variables are then stored in ‘predict_test_nb’ as type class because the target variable ‘purchase’ has 2 categorical classes ‘yes’ or ‘no’.

Code Snippet

```
157 ###################################  NAIVE BAYES  ##########################
158 #Building the learner model
159 market_nb <- naiveBayes(purchase~., data=train_data)
160 market_nb
161
162 #Making predictions on the test data
163 predict_test_nb <- predict(market_nb, newdata = test_data, type = "class")
164 predict_test_nb
```

Support Vector Machine (SVM)

The library used for SVM is ‘e1071’, the same as Naïve Bayes. Again, the process of building the learner model and making predictions on the test data is similar to that of the decision trees. Thus, for building the learner model, ‘svm ()’ function is used and the model is stored in ‘market_svm’. The predictions on the test data are made by using the ‘predict ()’ function and passing the trained model as an argument. The predicted values of the target variables are then stored in ‘predict_test_svm’ as type class because the target variable ‘purchase’ has 2 categorical classes ‘yes’ or ‘no’.

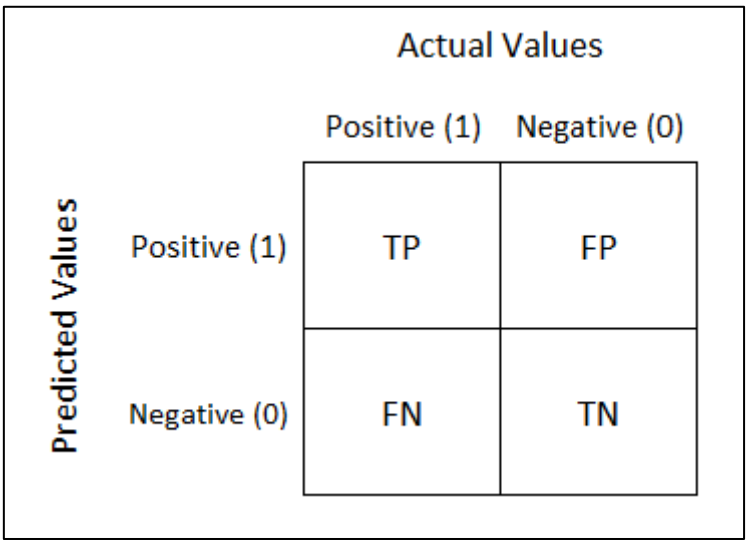
Code Snippet

```
168 ###################################  SVM  ##################################
169 #Building the learner model
170 market_svm <- svm(purchase~., data=train_data)
171 market_svm
172
173 #Making predictions on the test data
174 predict_test_svm <- predict(market_svm, newdata = test_data, type = "class")
175 predict_test_svm
```

Evaluation of Model Performance

Confusion Matrix

Confusion Matrix is a common way to evaluate the performance of a classification model by comparing the predicted values of the target variable in the test data against the actual values of the target variable in the test data.



The above figure represents the structure of the confusion matrix where the predicted values are compared against actual values of the test data.

TP: True Positive

TN: True Negative

FP: False Positive

FN: False Negative

The accuracy of the model is determined by the formula:

$$(TP + TN / TP + TN + FP + FN) * 100$$

The precision of the model which is the ratio of correct positive predictions by the model is given by the formula:

$$(TP / TP + FP) * 100$$

For the given ABSA Bank's problem, the confusion matrix has been built for all the 4 prediction models which are mentioned above. The 'caret' library contains a function 'confusionMatrix ()' using which the matrix can be built easily. The two arguments which are passed in the function are the predicted set of values and the actual target variable values from the test data.

Confusion Matrix for Decision Tree Model

Code Snippet

```
#Building the confusion matrix for Decision Tree Classification
confusionMatrix(predict_test_dt, factor(test_data$purchase))
```

Output

```
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  3240 1113
       yes  412 2734

               Accuracy : 0.7966
                 95% CI : (0.7873, 0.8057)
    No Information Rate : 0.513
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.595

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

            Sensitivity : 0.8872
            Specificity : 0.7107
         Pos Pred Value : 0.7443
         Neg Pred Value : 0.8690
             Prevalence : 0.4870
         Detection Rate : 0.4321
   Detection Prevalence : 0.5805
      Balanced Accuracy : 0.7989

       'Positive' Class : no
```

As seen in the output, the decision tree model had an accuracy of about 79.66%. The number of true positives and true negatives were 2734 and 3240, respectively while that of false positives and false negatives were 412 and 1113, respectively. The precision of the Decision Tree model comes out to be 86.9%.

Confusion Matrix for Random Forest Classification Model

Code Snippet

```
153 #Building the confusion matrix
154 confusionMatrix(predict_test_rf, factor(test_data$purchase))
```

Output

```
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  3129   24
       yes  523 3823

               Accuracy : 0.9271
                 95% CI : (0.9209, 0.9328)
    No Information Rate : 0.513
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8535

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

            Sensitivity : 0.8568
            Specificity : 0.9938
         Pos Pred Value : 0.9924
         Neg Pred Value : 0.8797
             Prevalence : 0.4870
         Detection Rate : 0.4173
   Detection Prevalence : 0.4205
      Balanced Accuracy : 0.9253

       'Positive' Class : no
```

As seen in the output, the random forest model had an accuracy of about 92.71%. The number of true positives and true negatives were 3823 and 3129, respectively while that of false positives and false negatives were 523 and 24, respectively. The precision of the Random Forest model comes out to be 87.9%.

Confusion Matrix for Naïve Bayes Classification Model

Code Snippet

```
##Building the confusion matrix
confusionMatrix(predict_test_nb, factor(test_data$purchase))
```

Output

```
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  2088 1349
       yes 1564 2498

               Accuracy : 0.6115
                 95% CI : (0.6004, 0.6226)
    No Information Rate : 0.513
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.2214

 Mcnemar's Test P-Value : 7.339e-05

            Sensitivity : 0.5717
            Specificity : 0.6493
         Pos Pred Value : 0.6075
         Neg Pred Value : 0.6150
             Prevalence : 0.4870
         Detection Rate : 0.2784
   Detection Prevalence : 0.4583
      Balanced Accuracy : 0.6105

       'Positive' Class : no
```

As seen in the output, the Naïve Bayes model had an accuracy of about 61.15%. The number of true positives and true negatives were 2498 and 2088, respectively while that of false positives and false negatives were 1564 and 1349, respectively. The precision of the Naïve Bayes model comes out to be 61.49%.

Confusion Matrix for SVM Classification Model

Code Snippet

```
177  ##Building the confusion matrix
178  confusionMatrix(predict_test_svm, factor(test_data$purchase))
```

Output

```
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  3098   81
       yes  554 3766

               Accuracy : 0.9153
                 95% CI : (0.9088, 0.9215)
    No Information Rate : 0.513
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.83

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

            Sensitivity : 0.8483
            Specificity : 0.9789
         Pos Pred Value : 0.9745
         Neg Pred Value : 0.8718
             Prevalence : 0.4870
         Detection Rate : 0.4131
   Detection Prevalence : 0.4239
      Balanced Accuracy : 0.9136

       'Positive' Class : no
```

As seen in the output, the SVM model had an accuracy of about 91.53%. The number of true positives and true negatives were 3766 and 3098, respectively while that of false positives and false negatives were 81 and 554, respectively. The precision of the SVM model comes out to be 87.37%.

Comparative Analysis of Models' Performance

The results of how each of the 4 models performed have been evaluated in the previous section of this report. Below table summarizes the performance of these models in comparison with each other.

Evaluation Parameters	Decision Tree	Random Forest	Naïve Bayes	SVM
True Positives	2734	3823	2498	3766
True Negatives	3240	3129	2088	3098
False Positives	412	523	1564	81
False Negatives	1113	24	1349	554

Precision	86.9%	87.9%	61.49%	87.37%
Accuracy	79.66%	92.71%	61.15%	91.53%

From the above table, it is evident that Random Forest and SVM are the top 2 contenders based on their accuracies as well as precision values. False positives (Type 1) in this case would mean that model is predicting that the customers would purchase the product while in reality, they did not. This means that ABSA Bank's sales team would waste their resources in contacting and persuading such customers to by their investment products which is an unwanted phenomenon. The false negatives (Type 1) on the other hand mean that the model predicted that the customers would not purchase the product while in reality, they did. This would mean that ABSA Bank might lose on potential customers for their products which would affect their sales numbers. This is also an unwanted phenomenon. From the above table, type 1 error was higher for Random Forest model than the SVM model and with Type 2 error, the case was exactly the opposite. Thus, neither of the 2 parameters provide a conclusive comparison of the 2 models' performance. Therefore, accuracy of the model would be a decisive criterion in this case.

However, it should be noted that the above values of accuracy are solely based on values obtained from the confusion matrix. As accuracy is being considered as the conclusive parameter for model selection in this case, it was necessary to verify the accuracy levels presented by the confusion matrix to mitigate the possibilities of the accuracy paradox and overfitting of the model. For this purpose, k-fold cross validation has been performed on the Random Forest and SVM models.

k-Fold Cross Validation

In k-Fold Cross Validation technique, the data is divided into 'k' unique subsets and then the model is trained on 'k-1' subsets while using the remaining subset as the test data. This process is then repeated 'k' times, which ensures that each time a different subset is used for training and testing. This also implies that every single data point would be used for training 'k-1' times and would be tested exactly once. The bias and variance of the resulting estimate then tends to reduce and it will give us a more thorough and precise accuracy of the model's performance.

In order to implement this technique in R, 'caret' library is used and using the 'createFolds ()' function the data is partitioned into 'k = 10' folds. Further, for the main algorithm the 'lapply ()' function which applies a particular function to different elements of a list is used. The particular function created in this case would compute the accuracy of each of these 10 folds. The 'training_fold' contains the whole training set, holding back the testing fold which is stored in 'test_fold'. The model is then trained on the training fold and tested against the training fold. The accuracy of every single subset is calculated using the confusion matrix and finally, the mean of those 10 accuracies and their standard deviation would provide the ultimate performance values of the 2 models. The value of standard deviation indicates the stability of the model and thus, should ideally be less.

k-Fold Cross Validation for Random Forest Model

Code Snippet

```
#Creating the function to train and test the model on each fold of data
cv_rf <- lapply(folds_rf, function(x) {
        #Creating the training fold without test fold
        training_fold <- train_data[-x,]
        test_fold <- train_data[x,]
        #Building the learner model for each of the 10 unique training fold
        market_rf_kfold <- randomForest(purchase~., data=training_fold, ntree = 500)
        #Making predictions on each unique test fold
        predict_test_rf_kfold <- predict(market_rf_kfold, newdata = test_fold, type = "class")
        #Creating confusion matrix for each unique test fold
        cm_rf <- table(predict_test_rf_kfold, factor(test_fold$purchase))
        #Calculating accuracy for each fold
        accuracy_rf <- (cm_rf[1,1] + cm_rf[2,2]) / (cm_rf[1,1] + cm_rf[2,2] + cm_rf[1,2] + cm_rf[2,1])
        return(accuracy_rf)
})
cv_rf
```

Output

```
$Fold01
[1] 0.919147

$Fold02
[1] 0.9288573

$Fold03
[1] 0.9212984

$Fold04
[1] 0.9337483

$Fold05
[1] 0.9218125

$Fold06
[1] 0.9177778

$Fold07
[1] 0.9306975

$Fold08
[1] 0.9275877

$Fold09
[1] 0.9355556

$Fold10
[1] 0.9231111
```

Further, the mean and the standard deviation of the 10 accuracies is calculated.

Code Snippet

```
#Calculating the mean of all the 10 accuracy values to determine the ultimate model accuracy
mean(as.numeric(cv_rf))
#Calculating the standard deviation of the 10 accuracy values to determine model stability
sd(as.numeric(cv_rf))
```

Output

```
> #Taking the mean of all the 10 accuracy values to determine the ultimate model accuracy
> mean(as.numeric(cv_rf))
[1] 0.9259593
> sd(as.numeric(cv_rf))
[1] 0.006206035
```

k-Fold Cross Validation for SVM Model

Code Snippet

```
#Creating the function to train and test the model on each fold of data
cv_svm <- lapply(folds_svm, function(x) {
        #Creating the training fold without test fold
        training_fold <- train_data[-x,]
        test_fold <- train_data[x,]
        #Building the learner model for each of the 10 unique training fold
        market_svm_kfold <- svm(purchase~., data=training_fold)
        #Making predictions on each unique test fold
        predict_test_svm_kfold <- predict(market_svm_kfold, newdata = test_fold, type = "class")
        #Creating confusion matrix for each unique test fold
        cm_svm <- table(predict_test_svm_kfold, factor(test_fold$purchase))
        #Calculating accuracy for each fold
        accuracy_svm <- (cm_svm[1,1] + cm_svm[2,2]) / (cm_svm[1,1] + cm_svm[2,2] + cm_svm[1,2] + cm_svm[2,1])
        return(accuracy_svm)
})
cv_svm
```

Output

```
> cv_svm
$Fold01
[1] 0.9168889

$Fold02
[1] 0.8987117

$Fold03
[1] 0.904402

$Fold04
[1] 0.92

$Fold05
[1] 0.9040426

$Fold06
[1] 0.9266341

$Fold07
[1] 0.9071111

$Fold08
[1] 0.9190751

$Fold09
[1] 0.9111506

$Fold10
[1] 0.9089294
```

Further, the mean and the standard deviation of the 10 accuracies is calculated.

Code Snippet

```
#Calculating the mean of all the 10 accuracy values to determine the ultimate model accuracy
mean(as.numeric(cv_svm))
#Calculating the standard deviation of the 10 accuracy values to determine model stability
sd(as.numeric(cv_svm))
```

Output

```
> #Calculating the mean of all the 10 accuracy values to determine the ultimate model accuracy
> mean(as.numeric(cv_svm))
[1] 0.9116945
> #Calculating the standard deviation of the 10 accuracy values to determine model stability
> sd(as.numeric(cv_svm))
[1] 0.008710544
```

Summary of the results:

	Random Forest	SVM
Mean Accuracy	92.6%	91.16%
Standard Deviation	0.0062	0.0087

As it is evident from the results of k-Fold Cross Validation, the accuracy of Random Forest model is still higher than SVM Model. Also, the standard deviation is lesser for the Random Forest Model which indicates less model instabilities. Therefore, after a 2-step evaluation of the models, it can be concluded that the Random Forest Classification Model is the most suitable model for ABSA Bank's marketing problem.

Final Predictions

Code Snippet

```
############################  FINAL PREDICTIONS  ####################################

#Reading the dataset for predictions
market_pred <- read.csv("market_pred.csv")

#Converting the categorical variables into factors
market_pred$gender <- as.factor(market_pred$gender)
market_pred$marital_status <- as.factor(market_pred$marital_status)
market_pred$education <- as.factor(market_pred$education)
market_pred$employ_status <- as.factor(market_pred$employ_status)
market_pred$spouse_work <- as.factor(market_pred$spouse_work)
market_pred$residential_status <- as.factor(market_pred$residential_status)
market_pred$product <- as.factor(market_pred$product)
market_pred$purchase <- as.factor(market_pred$purchase)

str(market_pred)

#Making the predictions using the model built using Random Forest Classification
market_pred_rf <- predict(market_rf, newdata = market_pred, type = "class")
market_pred_rf

#Replacing the target variable column in the market_pred dataset with the predictions
market_pred$purchase <- market_pred_rf
market_pred$purchase
```

Output

```
> market_pred$purchase
  [1] no  no  yes yes yes no  yes 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  no  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 no  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
```

Recommendations to ABSA Bank

The final recommendations to ABSA Bank reflect the trends which were analyzed during the descriptive analytics of the historical data about the purchase decisions of the customers. Upon analyzing the dataset, it was observed that around 50% of the contacted customers purchased the investment product which was marketed to them. If the trends and relationships of the customer details with the purchase decision are analyzed, ABSA Bank's sales team can utilize their time and resources in an appropriate manner while maximizing the number of sales.

The historical data indicates that younger customers are making the most purchases. Thus, firstly, the sales team must try to concentrate more on younger customers aging between 21 to 30 years. Customers lying in this age group start making their first earnings and are curious to explore options to invest them. If the sales team can persuade such customers about the benefits of the bank's investment products and overall importance of making investments in early years, it is very likely that these customers would purchase the marketed product. For customers with relatively higher age, above 40 years, already own investments and thus, are less likely to purchase any new investment products.

Stock investments are dynamic in nature and possess a certain degree of risk. Given the dynamic nature of young-aged people and relatively lesser responsibilities of families etc., are willing to invest in risk-prone options with quick benefits. Thus, they are likely to invest in stocks rather than bonds and derivatives, which is evident from the trends of the historical data. Thus, the bank's sales team should avoid marketing derivatives and bonds to young customers and instead suggest stocks.

More the number of children, more is the overall expenditure and responsibilities. It is again evident from the data trends that people with more than 1 children are less likely to purchase any new investment products. The bank's sales team thus should try to market their product with customers with one or zero children, which is again analogous with young age.

Another feature which impacted the purchase decision was the total number of years for which the customer has been employed. Customers with relatively more years of employment made fewer purchases than those who were recently employed. This trend also follows the age criteria as it is obvious that younger customers would be employed for fewer years than the older ones.

In conclusion, if the final recommendation for the bank's sales team is to be summed up in a single line, it would be utilizing the available time and resources to market the investment products preferably to a younger lot of customers belonging to the age range of 21 to 30 years. Apart from this, the predictions made by the machine learning model provides a highly accurate estimate regarding the exact customers who are most likely to purchase the

marketed product which can be referred by the sales team in order to utilize their time and resources optimally while maximizing the sales.

Possibly Useful Additional Data

From the overall analysis of the dataset and studying about the relationship of the explanatory variables with the dependent variable 'purchase', this report has identified 2 new features data which can help to make better predictions in the future. These 2 features are described below.

Field of Education

From the information gain calculated for all the explanatory variables, it was observed that the feature 'education' had a very low information gain. It was observed that customers whose education was 'high school' had the highest number of purchases according to the historical data. A logical reasoning could not be established as to why the customers who have completed only high schooling had such a significant number of purchases. Thus, this feature was as good as redundant. However, if there was another feature which included data about the field of a customer's education, it might have provided helpful insights. It might be possible that customers who have an educational background in Finance, Economics and Commerce could more likely be interested in different investment products, given their pre-existing knowledge about this domain. Also, the sales team while marketing their products can use less time and resources while explaining about what the actual product is to such customers as they would already be aware about it.

Existing Investment Portfolio

There are 3 helpful insights which could potentially be obtained from the data regarding a customer's existing investment portfolio:

1. It can provide information about what kind of investment product the customer has invested and usually prefers. The bank's sales team can thus market similar products to these customers.
2. If the customer has made a lot of investments or has made financially hefty investments, it is not very likely that he/she will purchase another investment product.
3. Considering the net income and the existing investments, a customer's available budget for purchasing a new investment product could be calculated which can be useful for the sales team while marketing the products to the customer.

R-Code Developed

```
#Exploratory Analysis of the Data set

#Importing the dataset
market <- read.csv("Market.csv")
head(market)

summary(market)

str(market)

#Converting the categorical variables into factors
market$gender <- as.factor(market$gender)
market$marital_status <- as.factor(market$marital_status)
market$education <- as.factor(market$education)
market$employ_status <- as.factor(market$employ_status)
market$spouse_work <- as.factor(market$spouse_work)
market$residential_status <- as.factor(market$residential_status)
market$product <- as.factor(market$product)
market$purchase <- as.factor(market$purchase)

#Information gain of the parameters
library(FSelector)
market_information_gain <- information.gain(purchase~., market)
market_information_gain %>% arrange(desc(attr_importance))

#DESCRIPTIVE ANALYTICS OF THE DATA
library(dplyr)

############################## PRODUCT ########################

#Relation of the product with its purchase
plot(market %>% filter(purchase=="yes") %>% select(product),
    xlab = "Products",
    ylab = "Purchases",
    main = "Purchases Based on Product Category",
    ylim = c(0,7000))

############################## AGE ##############################

#Filtering the data for customers who had purchased the products in the past
age_graph <- (market %>% filter(purchase=="yes") %>% select(age))
summary(age_graph)

#Finding absolute frequency
sort(table(age_graph))

#Plotting Age groups vs Purchases
```

```
hist(age_graph$age, xlab = "Age Groups",
    ylab = "Purchases made",
    main = "Number of Purchases Based on Age Groups",
    ylim = c(0, 5000))

#Plotting products vs age groups
plot(market %>% filter(purchase=="yes") %>% select(product, age), ylab = "Age
Groups",
    xlab = "Products",
    main = "Products Purchased Based on Age Groups")

######################## NUMBER OF CHILDREN ###################

#Filtering the data for customers who had purchased the products in the past
children_graph <- (market %>% filter(purchase=="yes") %>%
select(nb_depend_child))

summary(children_graph)

#Finding absolute frequency
sort(table(children_graph))

#Plotting the relation between number of children and purchases
hist(children_graph$nb_depend_child, xlab = "Number of children",
    ylab = "Purchases made",
    main = "Purchases Based on Number of Children",
    ylim = c(0, 8000))

######################## YEARS EMPLOYED ##########################

#Filtering the data for customers who had purchased the products in the past
employment_graph <- (market %>% filter(purchase=="yes") %>%
select(yrs_employed))

summary(employment_graph)

#Finding absolute frequency
sort(table(employment_graph))

#Plotting the relation between number of years employed and purchases
hist(employment_graph$yrs_employed, xlab = "Employment Years",
    ylab = "Purchases made",
    main = "Purchases Based on Years of Employment",
    ylim = c(0, 8000))

######################## NET INCOME  ############################

#Filtering the data for customers who had purchased the products in the past
net_income_graph <- (market %>% filter(purchase=="yes") %>% select(net_income))
```

```
summary(net_income_graph)

#Finding absolute frequency
sort(table(net_income_graph))

#Plotting the relation between Net Income and purchases
hist(net_income_graph$net_income, xlab = "Net Income",
    ylab = "Purchases made",
    main = "Purchases Based on Net Income",
    xlim = c(0,200000),
    ylim = c(0, 4000))

#Plotting the relation between number of years employed and purchases
plot(market %>% filter(purchase=="yes") %>% select(product, net_income))

########################## MARITAL STATUS ######################

#Plotting the relation between marital status and purchases
plot(market %>% filter(purchase=="yes") %>% select(marital_status),
    xlab = "Marital Status",
    ylab = "Purchases made",
    main = "Purchases Based on Marital Status",
    ylim = c(0, 5000))

######### SPLITTING THE DATA INTO TRAINING AND TEST SETS ############

#Loading the required libraries
library(caret) #Classification and Regression Training
library(rpart) #Recursive partitioning and regression trees
library(rpart.plot) #Plotting the rpart model
library(e1071) #Naive Bayes
library(randomForest) #Random Forest Classification

#Splitting the data set into Training and Test Set
indxTrain <- createDataPartition(market$purchase, p=0.75, list=F)

train_data <- market[indxTrain,]
test_data <- market[-indxTrain,]

######################### DECISION TREES ########################

#Configuring the control settings
ctrl_dt <- rpart.control(minsplit = 1, minbucket = 1, maxdepth = 10, cp = 0.01)

#Building the learner model
market_dt <- rpart(purchase~., data = train_data, control = ctrl_dt)
market_dt
```

```
#Plotting the decision tree
rpart.plot(market_dt, main = "Market", extra = 108)

#Making predictions on the test data
predict_test_dt <- predict(market_dt, newdata = test_data, type = "class")
predict_test_dt

#Building the confusion matrix for Decision Tree Classification
confusionMatrix(predict_test_dt, factor(test_data$purchase))

######################## RANDOM FOREST  ##########################

#Building the learner model
market_rf <- randomForest(purchase~., data=train_data, ntree = 500)
market_rf

#Making predictions on the test data
predict_test_rf <- predict(market_rf, newdata = test_data, type = "class")
predict_test_rf

#Building the confusion matrix
confusionMatrix(predict_test_rf, factor(test_data$purchase))

######################### NAIVE BAYES ############################

#Building the learner model
market_nb <- naiveBayes(purchase~., data=train_data)
market_nb

#Making predictions on the test data
predict_test_nb <- predict(market_nb, newdata = test_data, type = "class")
predict_test_nb

##Building the confusion matrix
confusionMatrix(predict_test_nb, factor(test_data$purchase))

############################## SVM ############################

#Building the learner model
market_svm <- svm(purchase~., data=train_data)
market_svm

#Making predictions on the test data
predict_test_svm <- predict(market_svm, newdata = test_data, type = "class")
predict_test_svm

##Building the confusion matrix
confusionMatrix(predict_test_svm, factor(test_data$purchase))
```

```
################ K-FOLD CROSS VALIDATION ######################

#Applying k-Fold Cross Validation to Random Forest Model
#Creating the number of folds
folds_rf <- createFolds(train_data$purchase, k = 10)

#Creating the function to train and test the model on each fold of data
cv_rf <- lapply(folds_rf, function(x) {
      #Creating the training fold without test fold
      training_fold <- train_data[-x,]
      test_fold <- train_data[x,]
      #Building the learner model for each of the 10 unique training fold
      market_rf_kfold <- randomForest(purchase~., data=training_fold, ntree = 500)
      #Making predictions on each unique test fold
      predict_test_rf_kfold <- predict(market_rf_kfold, newdata = test_fold, type = "class")
      #Creating confusion matrix for each unique test fold
      cm_rf <- table(predict_test_rf_kfold, factor(test_fold$purchase))
      #Calculating accuracy for each fold
      accuracy_rf <- (cm_rf[1,1] + cm_rf[2,2]) / (cm_rf[1,1] + cm_rf[2,2] + cm_rf[1,2] +
cm_rf[2,1])
      return(accuracy_rf)
})
cv_rf

#Calculating the mean of all the 10 accuracy values to determine the ultimate model
accuracy
mean(as.numeric(cv_rf))
#Calculating the standard deviation of the 10 accuracy values to determine model
stability
sd(as.numeric(cv_rf))

#*****************************************************************************#

#Applying k-Fold Cross Validation to SVM Model
#Creating the number of folds
folds_svm <- createFolds(train_data$purchase, k = 10)

#Creating the function to train and test the model on each fold of data
cv_svm <- lapply(folds_svm, function(x) {
      #Creating the training fold without test fold
      training_fold <- train_data[-x,]
      test_fold <- train_data[x,]
      #Building the learner model for each of the 10 unique training fold
      market_svm_kfold <- svm(purchase~., data=training_fold)
      #Making predictions on each unique test fold
      predict_test_svm_kfold <- predict(market_svm_kfold, newdata = test_fold, type =
"class")
      #Creating confusion matrix for each unique test fold
      cm_svm <- table(predict_test_svm_kfold, factor(test_fold$purchase))
```

```
    #Calculating accuracy for each fold
    accuracy_svm <- (cm_svm[1,1] + cm_svm[2,2]) / (cm_svm[1,1] + cm_svm[2,2] +
cm_svm[1,2] + cm_svm[2,1])
    return(accuracy_svm)
})
cv_svm

#Calculating the mean of all the 10 accuracy values to determine the ultimate model
accuracy
mean(as.numeric(cv_svm))
#Calculating the standard deviation of the 10 accuracy values to determine model
stability
sd(as.numeric(cv_svm))

###################### FINAL PREDICTIONS ##########################

#Reading the dataset for predictions
market_pred <- read.csv("market_pred.csv")

#Converting the categorical variables into factors
market_pred$gender <- as.factor(market_pred$gender)
market_pred$marital_status <- as.factor(market_pred$marital_status)
market_pred$education <- as.factor(market_pred$education)
market_pred$employ_status <- as.factor(market_pred$employ_status)
market_pred$spouse_work <- as.factor(market_pred$spouse_work)
market_pred$residential_status <- as.factor(market_pred$residential_status)
market_pred$product <- as.factor(market_pred$product)
market_pred$purchase <- as.factor(market_pred$purchase)

str(market_pred)

#Making the predictions using the model built using Random Forest Classification
market_pred_rf <- predict(market_rf, newdata = market_pred, type = "class")
market_pred_rf

#Replacing the target variable column in the market_pred dataset with the predictions
market_pred$purchase <- market_pred_rf
market_pred$purchase
```