Illinois Institute of Technology

CS - 571 Data Preparation and Analysis

CREDIT CARD CUSTOMER CHURN (IN R)

Instructed by: **Prof. Jawahar Panchal**

Submitted by:

Raj Shah - A20524266

Akshay Singh – A20498211

Ravi Teja Batchala - A20512513

TABLE OF CONTENTS

Sr. No.	Topic	Page No.
1	Introduction	1
2	Data Description	1-3
3	Data Cleaning	3
4	Exploratory Data Analysis	4-6
5	Principal Component Analysis	7-8
6	Dimension Reduction	9
7	Data Pre-processing	9-10
8	Models	11-14
9	Model Evaluation and Interpretation	15-20
10	Conclusion	21
11	Challenges faced	21-22
12	References	22
13	Contributions	23

TABLE OF FIGURES

Fig No	Fig Name	Page No.
1	Attrition number by Gender	4
2	Attrition number by Age	4
3	Attrition number by Education	4
4	Attrition number by Income	4
5	Attrition number by Card type	5
6	Attrition Status by Gender against existing customer	5
7	Attrition by Card Category	5
8	Attrition by Income Category	5
9	Attrition by Age and Education	6
10	Scree Plot of PCA	8

I. Introduction

The main objective is to select the model that predicts credit card customer churning best by comparing them with the performance of different Machine Learning models using R. Customer churn is a common measure of lost customers. By minimizing customer churn, a company can maximize its profits. Companies recognize that existing customers are one of the most valuable assets a company could have and customer retention is critical for a good marketing strategy. The prevention of customer churn through customer retention is the core problem of Customer Relationship Management. Here, an analysis is done on purchasing behavior of bank customers from a certain dataset. A detailed analysis is worked out to convert raw customer data into meaningful and useful data that suits the buying behavior and in turn, converts this meaningful data into knowledge for which predictive data mining techniques are adopted.

II. Data Description

This dataset includes the information of banks along with its existing and attrited customers. We will use the attrition flag that is within the dataset as our target variable, we will train and test with this variable. To define the success of the solution that we will deliver let's define the metrics as: Accuracy and Recall score. This metrics were chosen since normally churn problems are imbalanced, but all depends on the definition of churn and the cost driven by each scenario. The Dataset consists of 10127 Rows and 21 Columns.

Link to Dataset - https://www.kaggle.com/varunbarath/credit-card-customers-bank-churners

Predicted attribute: Attrition Flag

Feature Information:
1. CLIENTNUM: Client number. Unique identifier for the customer holding the account.
2. Customer Age: Customer's Age in Years.
3. Gender: M=Male, F=Female.
4. Dependent count: Number of dependents.
5. Education Level: Educational Qualification of the account holder.
6. Marital Status: Married, Single, Divorced, Unknown.

7. Income Category: Annual Income Category of the account holder.
8. Card Category: Type of Card (Blue, Silver, Gold, Platinum).
9. Months on book: Period of relationship with bank.
10. Total Relationship Count: Total number of products held by the customer.
11. Months Inactive 12 mon: No. of months inactive in the last 12 months.
12. Contacts Count 12 mon: No. of Contacts in the last 12 months.
13. Credit Limit: Credit limit on the Credit Card.
14. Total Revolving Bal: Total Revolving Balance on the Credit Card.
15. Avg Open To Buy: Open to Buy Credit Line (Average of last 12 months).
16. Total Amt Chng Q4 Q1: Change in Transaction Amount (Q4 over Q1).
17. Total Trans Amt: Total Transaction Amount (Last 12 months).
18. Total Trans Ct: Total Transaction Count (Last 12 months).
19. Total Ct Chng Q4 Q1: Change in Transaction Count (Q4 over Q1).
20. Avg Utilization Ration: Average Card Utilization Ratio.
21. Attrition Flag: If the account is closed then 1 (Attrited Customer) else 0 (Existing Customer).

```
## 'data.frame':    10127 obs. of  23 variables:
##  $ CLIENTNUM
##  $ Attrition_Flag
##  $ Customer_Age
##  $ Gender
##  $ Dependent_count
##  $ Education_Level
##  $ Marital_Status
##  $ Income_Category
##  $ Card_Category
##  $ Months_on_book
##  $ Total_Relationship_Count
##  $ Months_Inactive_12_mon
##  $ Contacts_Count_12_mon
##  $ Credit_Limit
##  $ Total_Revolving_Bal
##  $ Avg_Open_To_Buy
##  $ Total_Amt_Chng_Q4_Q1
##  $ Total_Trans_Amt
##  $ Total_Trans_Ct
##  $ Total_Ct_Chng_Q4_Q1
##  $ Avg_Utilization_Ratio
##  $ Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Educati
##  $ Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Educati
```

```
sapply(churn, function(x) sum(is.na(x)))
```

III. Data Cleaning

In this step, we will check for missing and duplicate values. We will also drop the last 2 columns (Naive_Bayes_Classifier….) because it contains garbage values and will not be useful for our model.

```r
sapply(churn, function(x) sum(is.na(x)))

churn$Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_1  <- NULL
churn$Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_2 <- NULL
churn$CLIENTNUM <- NULL
```

Missing Attribute Values: None

We also converted columns to numeric data for ease of training and testing our model.

```r
#Converting all features to categorical data
churn[sapply(churn, is.character)]<- lapply(churn[sapply(churn, is.character)], as.factor)
```

IV. Exploratory Data Analysis

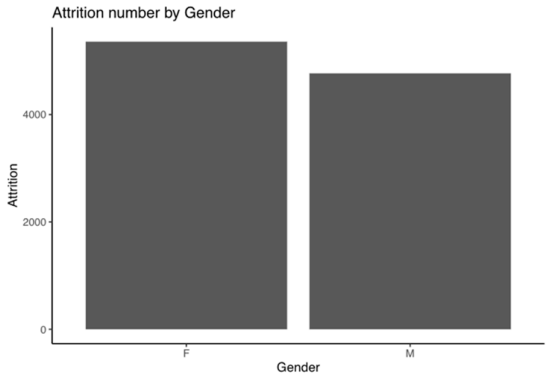


Fig 1: Attrition number by Gender

You can see through this graph that Females are more prone to attrition compared to males.

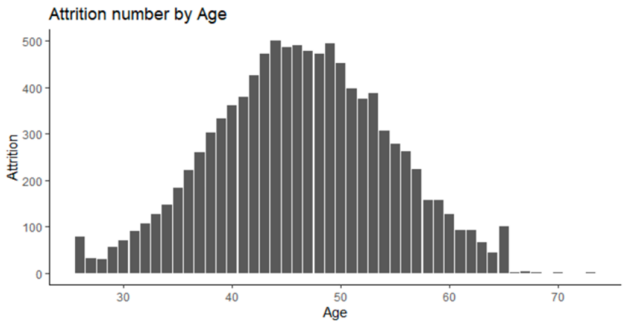


Fig 2: Attrition number by Age

You can see through this graph that Age 44 are more prone to attrition compared to people of other ages.

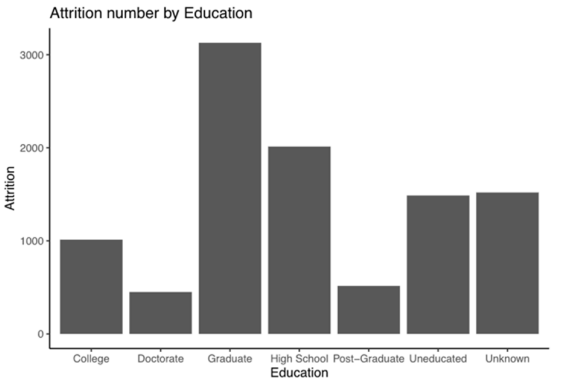


Fig 3: Attrition number by Education

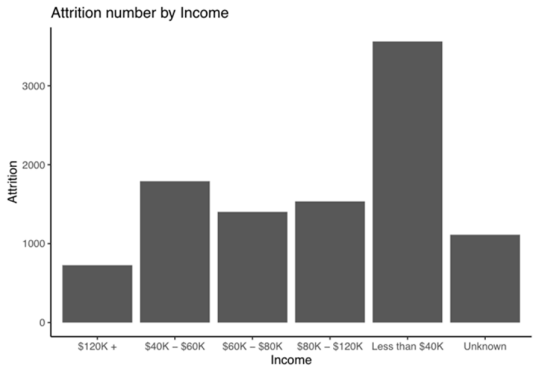
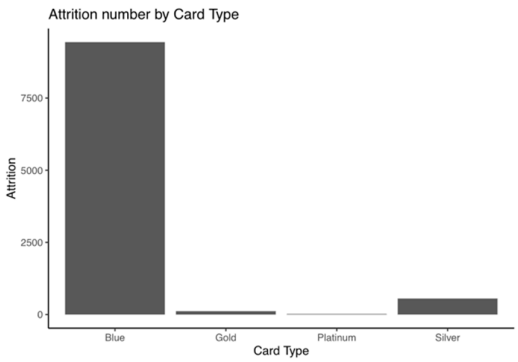


Fig 4: Attrition number by Income

You can see through this graph that Graduates are more prone to attrition compared to people of other education streams.

You can see through this graph that people earning less than 40k are more prone to attrition compared to people of other incomes.



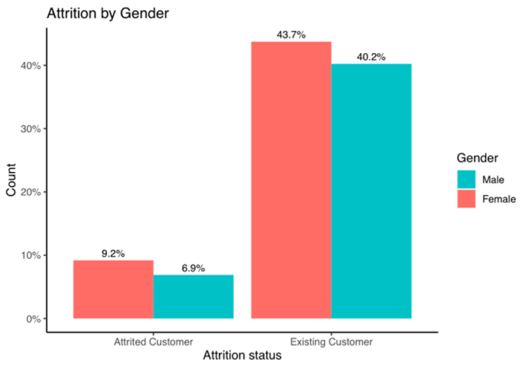


Fig 5: Attrition number by Card type

Fig 6: Attrition Status by Gender against existing customer

You can see through this graph that people with blue card type are more prone to attrition compared to people of other card types.

You can see through this graph that females (9.2%) are more prone to attrition compared to males (6.9%).

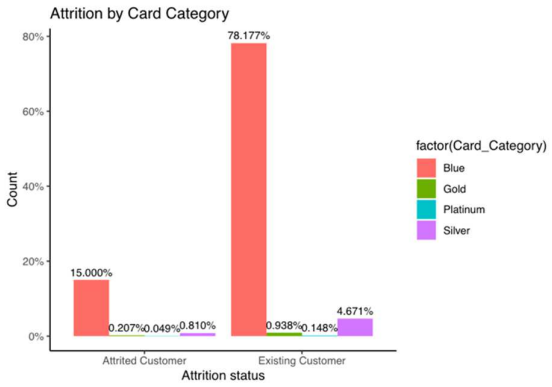


Fig 7: Attrition by Card Category

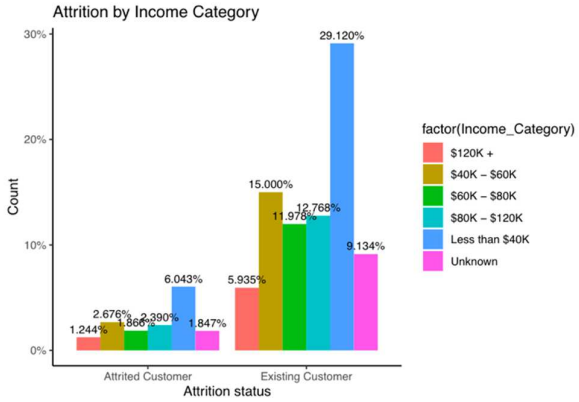
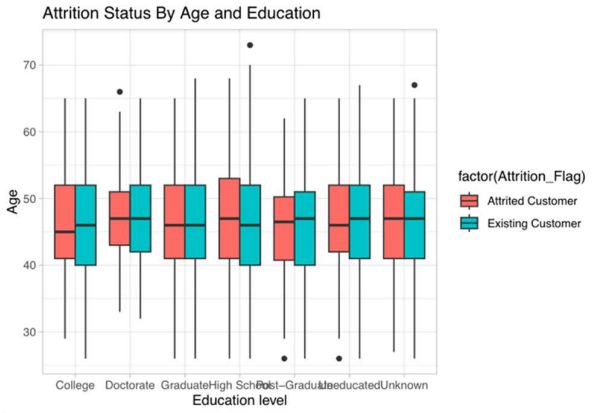


Fig 8: Attrition by Income Category

Comparison of the Attrition Status between Attrited Customer Existing customer is shown above.



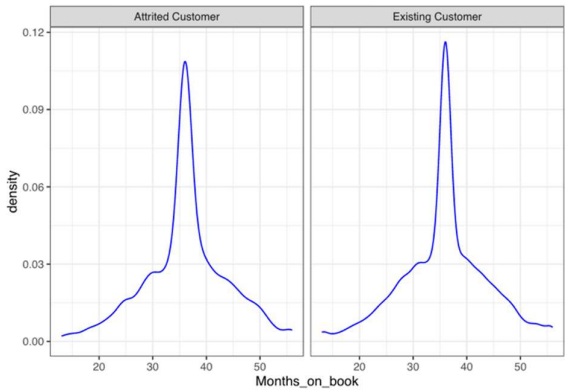
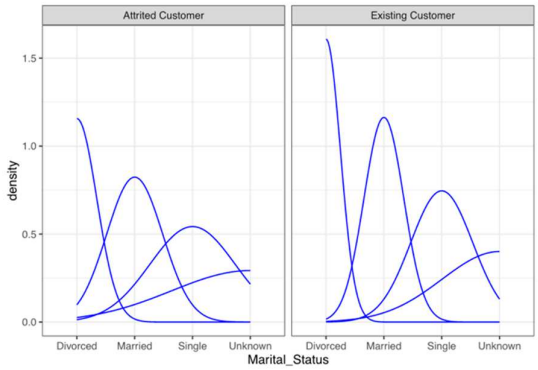
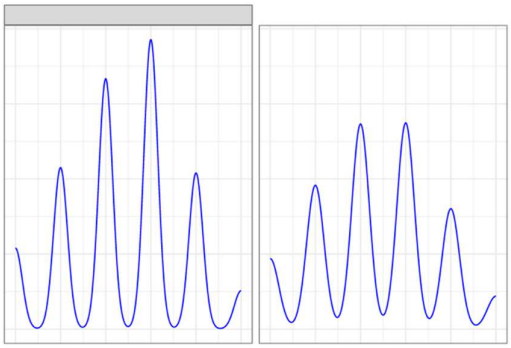


Fig 9- Attrition by Age and Education

Boxplot showing attrition status of the card holders based on age and education of the individual





Comparison of the Attrition Status between Attrited Customer and Existing customer is shown above.

V. Principal Component Analysis

Principal component analysis (PCA) is a technique for reducing the dimensionality of such datasets, increasing interpretability but at the same time minimizing information loss. It does so by creating new uncorrelated variables that successively maximize variance.

Although our dataset does not have a huge number of features in the traditional sense, we still wanted to use PCA to learn how it works and reduce the dimensions and to see if using PCA is a good strategy for when number of features are about 20.

```r
#PCA
churn.pca <- prcomp(scale(churn[,c(2,4,9:20)]), center = TRUE)
summary(churn.pca)
```

We have only added the 14 numerical data that were there in our data to generate 14 principal components. We have also made sure to normalize the data.

Before PCA, we standardize/normalize data. Usually, normalization is done so that all features are at the same scale. Normalization is important in PCA since it is a variance maximizing exercise. It projects your original data onto directions which maximize the variance.

In theory it is possible to apply PCA on discrete variables as well. This can be done by one hot encoding categorical variables and then applying PCA on it. However, it is not advised to do so. General rule of thumb is, if your variables don't belong on a coordinate plane, then do not apply PCA on them.

```{r}
#How much variation in the original data does PCA account for
churn.pca.var <- churn.pca$sdev^2
churn.pca.var.per <- round(churn.pca.var/sum(churn.pca.var)*100,1)
churn.pca.var.per
```

This snippet shows what percentage of variation each principal component is responsible for.

```
##  [1] 18.3 14.6 12.8 10.3  8.9  7.2  7.0  6.5  5.8  4.3  1.6  1.5  1.2  0.0
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5    PC6     PC7
## Standard deviation     1.6025 1.4301 1.3408 1.2024 1.11491 1.0019 0.99250
## Proportion of Variance 0.1834 0.1461 0.1284 0.1033 0.08879 0.0717 0.07036
## Cumulative Proportion  0.1834 0.3295 0.4579 0.5612 0.64998 0.7217 0.79203
##                           PC8     PC9    PC10    PC11    PC12    PC13
## Standard deviation     0.95112 0.89829 0.77448 0.47086 0.45909 0.40948
## Proportion of Variance 0.06462 0.05764 0.04284 0.01584 0.01505 0.01198
## Cumulative Proportion  0.85665 0.91429 0.95713 0.97297 0.98802 1.00000
##                           PC14
## Standard deviation     1.067e-15
## Proportion of Variance 0.000e+00
## Cumulative Proportion  1.000e+00
```

Next thing we will do is create a scree plot to find out how much variation in the original data does PCA account for.

```{r}
#Plotting PCA percentages
barplot(churn.pca.var.per, main="Screen Plot", xlab="Principal Component Analysis",
        names = c("PC1", "PC2", "PC3", "PC4", "PC5", "PC6", "PC7", "PC8", "PC9", "PC10",
                  "PC11", "PC12", "PC13","PC14"), ylab="Percent Variation", )

#PCA ends here
```

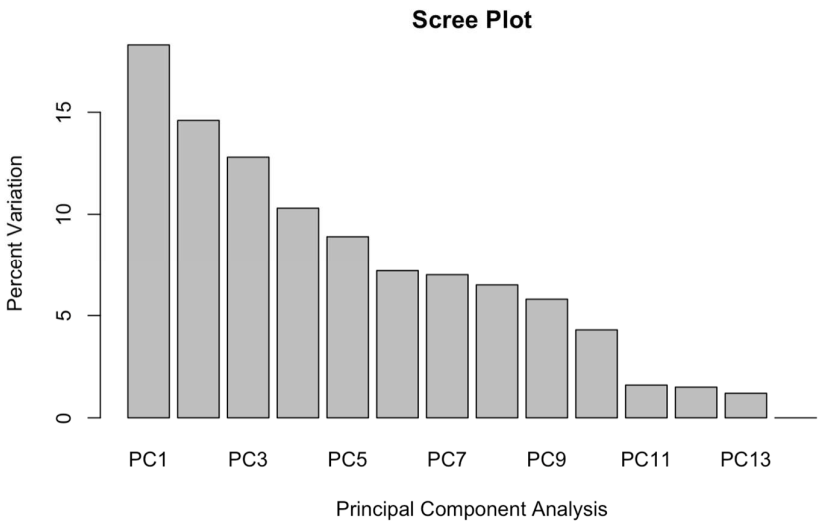


Fig 10 - Scree Plot of PCA

VI. Dimension Reduction

Upon examining the scree plot and cumulative proportion, we can observe that first 10 principal components are responsible for 95.713% of variance. Therefore, we have decided to drop the last 4 principal components while creating our model. This will reduce the model dimensions from 19 to 15.

```{r}
pc_data <- churn.pca$x[,1:10]
cat_data <- churn[,c(1,3,5:8)]
churn_pca <-data.frame(cat_data, pc_data)
```

Here, pc_data represents the data in principal components 1 through 10 and cat_data contains all the columns with categorical data in them. We have combined these two to create a single data frame by the name of 'churn_pca' on which we will create our models and compare the results with the dataframe containing just features and no principal components.

VII. Data Preprocessing

```{r}
churn_pca[sapply(churn_pca, is.character)]<- lapply(churn_pca[sapply(churn_pca, is.character)], as.factor)
summary(churn_pca)
```

```r
#Converting all features to categorical data
churn[sapply(churn, is.character)]<- lapply(churn[sapply(churn, is.character)], as.factor)
```

Here, we are transforming all the categorical features to factors. This is a necessary step before we could create models.

Categorical Variable Transformation: is turning a categorical variable to a numeric variable. Categorical variable transformation is mandatory for most of the machine learning models because they can handle only numeric values. It is also called encoding, or in text mining, embedding is also meant to handle similar situation but embedding is usually suppose to return numeric values containing semantics of original data. Categorical variable transformation is important for any models and its selection matters a lot for model performance.

Upon completing this step, we begin to split our datasets. From here on we will refer to **Dataset I for the churn_pca dataset** containing principal components and **Dataset II for the churn dataset** for regular data with all the features for clarity

Dataset I

```r
#Splitting the pca dataset
intrain_pca<- createDataPartition(churn_pca$Attrition_Flag, p=0.80, list = FALSE)
training_pca<- churn_pca[intrain_pca,]
testing_pca<- churn_pca[-intrain_pca,]
dim(training_pca); dim(testing_pca)
#summary(training_pca)
#summary(testing_pca)
```

```
[1] 8102   16
[1] 2025   16
```

Dataset II

```r
#Splitting the regular dataset
intrain_reg<- createDataPartition(churn$Attrition_Flag, p=0.80, list = FALSE)
training_reg<- churn[intrain_reg,]
testing_reg<- churn[-intrain_reg,]
dim(training_reg); dim(testing_reg)
#summary(training_reg)
#summary(testing_reg)
```

```
[1] 8102   20
[1] 2025   20
```

We have split our datasets in **80:20** ratio for training and testing data.

VIII. Models

Random Forest

Random forest is a Supervised Machine Learning Algorithm that works by combining results of multiple decision trees to reach one single result. It works well for both regression and classification problems. In random forest, instead of working on just one decision tree, it takes into consideration, each and every tree and aggregates them together to predict the most popular result. A singular decision tree is more prone to problems of bias and overfitting, as compared to multiple trees ensembled together in the random forest algorithm. Thus, they predict results with more accuracy, especially in cases where individual decision trees are correlated with each other.

Steps in random forest algorithm are:

Step 1: In Random Forest n number of random records are taken from the data set having k number of records.

Step 2: Independent decision trees are constructed for every sample.

Step 3: An output will be given by every decision tree.

Step 4: Depending on Majority Voting or Averaging for Classification and regression respectively, concluding output is considered.

In our case, dataset I predicted attrition with **91.26** accuracy and dataset II produced predictions with an accuracy score of **96.35**

Logistic Regression

Logistic Regression is a Supervised Learning technique. It's mainly used for predicting the categorical variable by mapping the independent input-output pairs.
The outcome is a categorical or discrete value like, true or false, right or wrong, 0 or 1 etc. Though, it doesn't give exact values. It gives a probabilistic value. One major distinction between Logistic Regression and Linear Regression is that Linear Regression predicts a regression line whereas Logistic Regression is used to solve classification problems.

Logistic Regression can provide probabilities and categorize new data by utilizing continuous and discrete datasets. It can be used to classify the observations by using different types of data. Then, it can easily predict which variable is the most effective which is used for the categorization.

In dataset I, we got the accuracy score of Logistic Regression to be **90.07** and **76.05** in dataset II.

SVM

Support Vector Machine is another Supervised Learning algorithms, which is widely used for regression as well as classification problems. Although it has more real-life use cases for classification than regression in Machine Learning. The way it works is, it generates a best line or decision boundary that is the maximum distance between data points of both classes. This helps us in assigning future data their respective class or category with ease and confidence. This model selects outliers as vectors while creating the hyperplane. These outlier vectors cases are called support vectors, and hence the name Support Vector Machine.

There are two types of SVMs:
1) Linear SVM: used for linearly separable data which means dataset can be partitioned in two using a single straight line.
2) Non-linear SVM: used for non-linear separable data which means data cannot be partitioned in two classes using a single straight line.

Example:
Consider a unique fruit that has some features of an apple and an orange. The model which can accurately identify if the fruit an orange or an apple can be created by SVM. The model will first be trained by inputting many various images of apples and oranges so that it can differentiate and learn about the distinctive features of oranges and apples. The model can then be tested with the unique fruit. As the support vector

generates a decision boundary between the two data (apples and oranges in this case) and choose extreme cases (support vectors), the extreme case of apples and oranges will be seem. With the help of the groundwork provided by the support vectors, the unique fruit can be categorized into an apple or an orange.

How does SVM help in predicting attrition?
SVM model can create hyperplanes among different categorical features from our dataset and train on it. It can use the most extreme case, and create a decision boundary. With the help of the groundwork provided by decision boundary, predict which category would be more suitable when we input our test data.
SVM's accuracy in database I came out to be **87.21** and **86.86** for database II.

Naive Bayes

It is another supervised learning model and is used for classification problems. It is based on Bayes theorem of conditional probability. This model is easy to setup and performs really well for large datasets. It's not just easy to use, but also powerful enough to compete with and even outperform some sophisticated classification methods. A few use cases of Naive Bayes Algorithm include: Sentimental analysis, spam filtration and classification of articles.

Naive Bayes gets its name because: Naive: It is called Naive as it presumes that occurrence of a specific feature is self-sustaining. Example: if the vegetable is identified on the basis of color, size, weight etc., then purple, spherical, 1oz is recognized as a Brinjal. Therefore, every feature independently gives an identity that it is a brinjal regardless of each other. Bayes: It is called Bayes as it follows Bayes' Theorem. In dataset I, the accuracy of this model is **89.78** and in dataset II, the accuracy is **89.33**.

Decision Tree

Decision Tree is a one more type of Supervised learning technique which is primarily used to solve classification problems, but in cases, can be used to solve Regression problems as well. The main idea of Decision Trees is to continuously to make decisions like yes or no based on certain rules to and split the dataset till the point each data point belonging to different class is isolated. This phenomenon creates a Tree like structure, hence the name. In this tree structure, internal nodes represent features, branches represent decision rules and each leaf node represents the outcome.

Reasons that make decision tree a viable classification model:
1) Since we humans also make decisions based upon our own certain set of rules, it mimics our thinking process while making a decision, and hence, it's easy to understand.
2) The tree-like structure makes it easy for a user who's reading the logic for the first time to understand what is going on.

In our case, decision tree model produced an accuracy of **88.74** on dataset I and **94.07** on dataset II.

IX. Model Evaluation and Interpretation

Confusion Matrix and Statistics

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

Random Forest

Dataset I

```
Confusion matrix:
                  Attrited Customer Existing Customer class.error
Attrited Customer               675               627  0.48156682
Existing Customer               109              6691  0.01602941
Confusion Matrix and Statistics

                  Reference
Prediction          Attrited Customer Existing Customer
  Attrited Customer               166                18
  Existing Customer               159              1682

               Accuracy : 0.9126
                 95% CI : (0.8994, 0.9245)
    No Information Rate : 0.8395
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6066

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

            Sensitivity : 0.51077
            Specificity : 0.98941
         Pos Pred Value : 0.90217
         Neg Pred Value : 0.91363
             Prevalence : 0.16049
         Detection Rate : 0.08198
   Detection Prevalence : 0.09086
      Balanced Accuracy : 0.75009

       'Positive' Class : Attrited Customer
```

Dataset II

```
Confusion matrix:
                  Attrited Customer Existing Customer class.error
Attrited Customer              1086               216  0.16589862
Existing Customer                82              6718  0.01205882
Confusion Matrix and Statistics

                  Reference
Prediction          Attrited Customer Existing Customer
  Attrited Customer               265                20
  Existing Customer                60              1680

               Accuracy : 0.9605
                 95% CI : (0.9511, 0.9686)
    No Information Rate : 0.8395
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8457

 Mcnemar's Test P-Value : 1.299e-05

            Sensitivity : 0.8154
            Specificity : 0.9882
         Pos Pred Value : 0.9298
         Neg Pred Value : 0.9655
             Prevalence : 0.1605
         Detection Rate : 0.1309
   Detection Prevalence : 0.1407
      Balanced Accuracy : 0.9018

       'Positive' Class : Attrited Customer
```

Logistic Regression

Dataset I

```
Confusion Matrix and Statistics

          target
y_pred       1    2
     1     165   41
     2     160 1659

               Accuracy : 0.9007
                 95% CI : (0.8869, 0.9134)
    No Information Rate : 0.8395
    P-Value [Acc > NIR] : 1.038e-15

                  Kappa : 0.5676

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

            Sensitivity : 0.50769
            Specificity : 0.97588
         Pos Pred Value : 0.80097
         Neg Pred Value : 0.91204
             Prevalence : 0.16049
         Detection Rate : 0.08148
   Detection Prevalence : 0.10173
      Balanced Accuracy : 0.74179

       'Positive' Class : 1
```

Dataset II

```
          target
y_pred       1    2
     1      44  204
     2     281 1496

               Accuracy : 0.7605
                 95% CI : (0.7413, 0.7789)
    No Information Rate : 0.8395
    P-Value [Acc > NIR] : 1.0000000

                  Kappa : 0.017

 Mcnemar's Test P-Value : 0.0005586

            Sensitivity : 0.13538
            Specificity : 0.88000
         Pos Pred Value : 0.17742
         Neg Pred Value : 0.84187
             Prevalence : 0.16049
         Detection Rate : 0.02173
   Detection Prevalence : 0.12247
      Balanced Accuracy : 0.50769

       'Positive' Class : 1
```

SVM

Dataset I

```
Confusion Matrix and Statistics

                  Reference
Prediction          Attrited Customer Existing Customer
  Attrited Customer                94                12
  Existing Customer               231              1688

               Accuracy : 0.88
                 95% CI : (0.865, 0.8938)
    No Information Rate : 0.8395
    P-Value [Acc > NIR] : 1.563e-07

                  Kappa : 0.3879

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

            Sensitivity : 0.28923
            Specificity : 0.99294
         Pos Pred Value : 0.88679
         Neg Pred Value : 0.87962
             Prevalence : 0.16049
         Detection Rate : 0.04642
   Detection Prevalence : 0.05235
      Balanced Accuracy : 0.64109

       'Positive' Class : Attrited Customer
```

Dataset II

```
Confusion Matrix and Statistics

                  Reference
Prediction          Attrited Customer Existing Customer
  Attrited Customer                62                 5
  Existing Customer               263              1695

               Accuracy : 0.8677
                 95% CI : (0.8521, 0.8821)
    No Information Rate : 0.8395
    P-Value [Acc > NIR] : 0.0002322

                  Kappa : 0.2766

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

            Sensitivity : 0.19077
            Specificity : 0.99706
         Pos Pred Value : 0.92537
         Neg Pred Value : 0.86568
             Prevalence : 0.16049
         Detection Rate : 0.03062
   Detection Prevalence : 0.03309
      Balanced Accuracy : 0.59391

       'Positive' Class : Attrited Customer
```

Naïve Bayes

Dataset I

Confusion Matrix and Statistics

Prediction	Reference: Attrited Customer	Reference: Existing Customer
Attrited Customer	146	28
Existing Customer	179	1672

Accuracy	0.8978
95% CI	(0.8838, 0.9106)
No Information Rate	0.8395
P-Value [Acc > NIR]	2.646e-14
Kappa	0.5329
Mcnemar's Test P-Value	< 2.2e-16
Sensitivity	0.44923
Specificity	0.98353
Pos Pred Value	0.83908
Neg Pred Value	0.90330
Prevalence	0.16049
Detection Rate	0.07210
Detection Prevalence	0.08593
Balanced Accuracy	0.71638
'Positive' Class	Attrited Customer

Dataset II

Confusion Matrix and Statistics

Prediction	Reference: Attrited Customer	Reference: Existing Customer
Attrited Customer	203	127
Existing Customer	122	1573

Accuracy	0.877
95% CI	(0.8619, 0.891)
No Information Rate	0.8395
P-Value [Acc > NIR]	1.156e-06
Kappa	0.5465
Mcnemar's Test P-Value	0.7999
Sensitivity	0.6246
Specificity	0.9253
Pos Pred Value	0.6152
Neg Pred Value	0.9280
Prevalence	0.1605
Detection Rate	0.1002
Detection Prevalence	0.1630
Balanced Accuracy	0.7750
'Positive' Class	Attrited Customer

Decision Tree

Dataset I

Confusion Matrix and Statistics

Prediction	Reference: Attrited Customer	Reference: Existing Customer
Attrited Customer	171	74
Existing Customer	154	1626

Accuracy	0.8874
95% CI	(0.8728, 0.9009)
No Information Rate	0.8395
P-Value [Acc > NIR]	5.090e-10
Kappa	0.536
Mcnemar's Test P-Value	1.678e-07
Sensitivity	0.52615
Specificity	0.95647
Pos Pred Value	0.69796
Neg Pred Value	0.91348
Prevalence	0.16049
Detection Rate	0.08444
Detection Prevalence	0.12099
Balanced Accuracy	0.74131
'Positive' Class	Attrited Customer

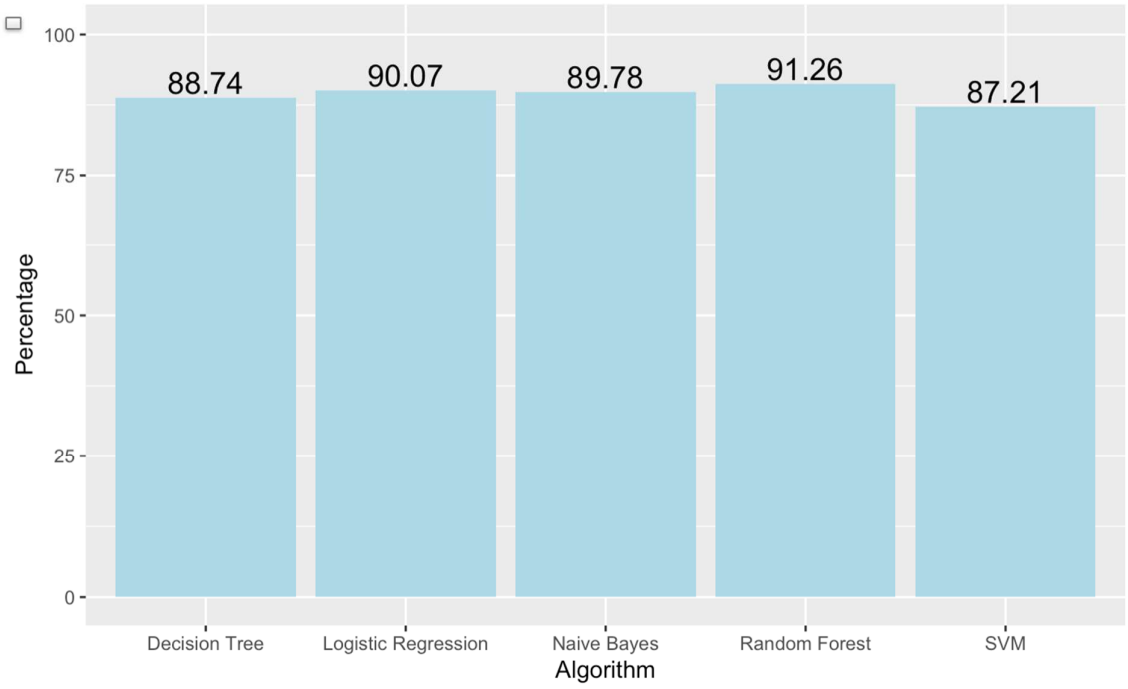
Dataset II

Confusion Matrix and Statistics

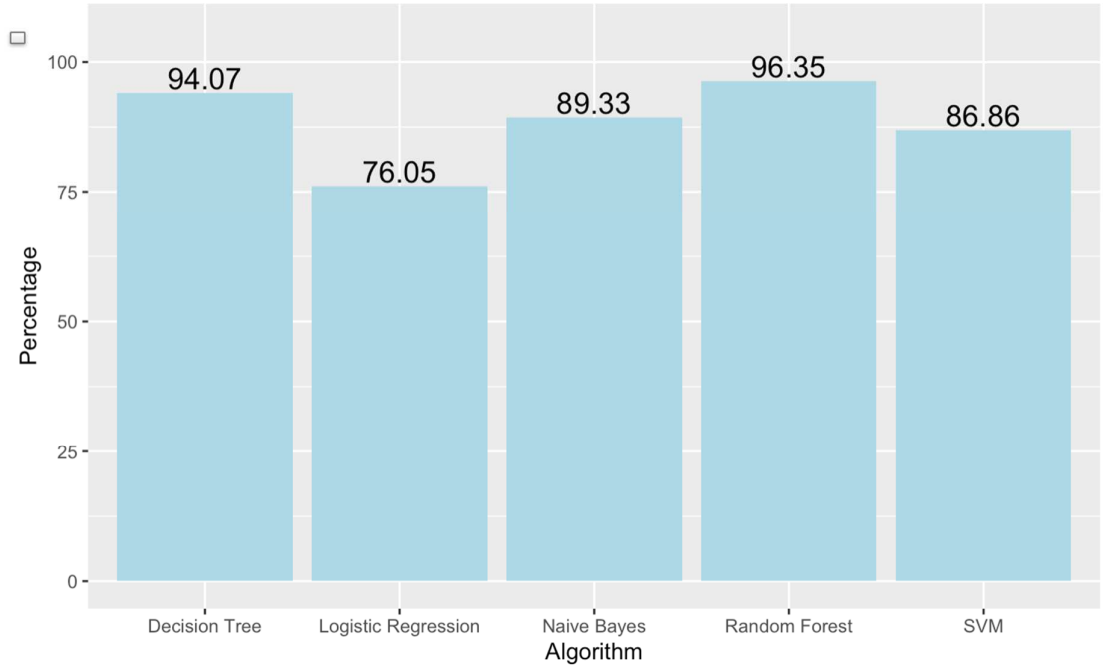
Prediction	Reference: Attrited Customer	Reference: Existing Customer
Attrited Customer	246	59
Existing Customer	79	1641

Accuracy	0.9319
95% CI	(0.92, 0.9424)
No Information Rate	0.8395
P-Value [Acc > NIR]	<2e-16
Kappa	0.7406
Mcnemar's Test P-Value	0.1058
Sensitivity	0.7569
Specificity	0.9653
Pos Pred Value	0.8066
Neg Pred Value	0.9541
Prevalence	0.1605
Detection Rate	0.1215
Detection Prevalence	0.1506
Balanced Accuracy	0.8611
'Positive' Class	Attrited Customer

Accuracy Percentage of All Models



Dataset I



Dataset II

X. Conclusion & Future Scope

We can conclude with certainty that 'Random Forest' will be the best model for predicting credit card customer attrition for both datasets since it predicted customer attrition with the highest accuracy in both datasets – with reduced dimensions and principal components, and the regular one.

We also observed that Naïve Bayes and SVM perform a tiny bit better with PCA data whereas the accuracy of Decision Tree and Random Forest dipped a bit when we used the PCA data. Also, as we observe the graphs, we can see few models performed a bit better with the transformed PCA data instead of the regular ones and vice versa. Dataset is of 20 dimensions (features), which might not be considered large enough to perform dimentionality reduction or even PCA, but performing PCA altered the model accuracy to an extend. Accuracy of all the models before PCA differed a lot from each other, but after PCA all the models gave the same accuracy in the range of 90%. This shows that all the models were able to train and fit the data correctly and capture the variablitiy of the features, which was not the case when we considered all the features from the dataset.

For future scope, we could implement more models and try deep learning and see if we get more accurate predictions. We can also build a web application and/or mobile application and sell our product to banks and financial institutions where they can make use of it to determine if they should issue a new credit card to a customer based on the details provided by him as well as the customers past credit card usage history and patterns.

XI. Challenges Faced

1. K Means- we encountered certain errors while preparing for kmeans clustering and we were also running short on time, hence we were not able to implement kmeans clustering.

2. SVM model takes more time to execute than expected. Currently, we are working with a database with about 10,000 rows, and the although the time taken in manageable. But, in case we have to work on an even bigger database in future, we will have to be careful about using this model.

3. Time management and coordination – Since all 3 of us have completely different other 2 subjects, and had projects and assignments in those subjects as well, it was a bit challenging working together and finding a suitable time available to all three. We overcame this by setting up virtual meetings, saving us transit times, so that we

had more time to focus on the actual content of the project and deliver it on time with all the deliverables.

4. We were unable to perform deep learning and see if it can be effectively used for predicting customer churn.

XII. References

1. *Dataset* *https://www.kaggle.com/code/varunbarath/credit-card-customers-bank-churners/data*
2. *Logistic-Regression* *https://www.javatpoint.com/logistic-regression-in-machine-learning*
3. *SVM* *https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm*
4. *Decision-tree* *https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm*
5. *https://towardsdatascience.com*
6. *https://medium.com*

XIII. Contributions

Raj Shah (Group Leader) – Data Gathering, Data Preparation, K-Means, SVM Model, Random Forest, Evaluation and Interpretation, Documentation and Presentation.

Ravi Teja – Data Cleaning, Data Exploration and Visualization, Decision Tree, Logistic Regression, Evaluation and Interpretation, Documentation and Presentation.

Akshay Singh – Data Preparation, Exploration and Visualization, Principal Component Analysis, Naïve Bayes, Logistic Regression Evaluation, Documentation and Presentation.