About Dataset

Customer churn refers to the phenomenon where customers discontinue their relationship or subscription with a company or service provider. It represents the rate at which customers stop using a company's products or services within a specific period. Churn is an important metric for businesses as it directly impacts revenue, growth, and customer retention.

The dataset contains

- age
- gender
- tenure
- usage frequency
- support calls
- payment delay
- subscription type
- contract length
- total spends
- last interaction

Link – https://www.kaggle.com/datasets/muhammadshahidazeem/customer-churn-dataset

Importing data

Code:

```
data <- read.csv("D:/AIUB/CSE SEM 10/DATA WAREHOUSING AND DATA MINING/final project/data/training.csv")
head(data)
```

output:

```
  CustomerID Age Gender Tenure Usage.Frequency Support.Calls Payment.Delay Subscription.Type
1          2  30 Female     39              14             5            18          Standard
2          3  65 Female     49               1            10             8             Basic
3          4  55 Female     14               4             6            18             Basic
4          5  58   Male     38              21             7             7          Standard
5          6  23   Male     32              20             5             8             Basic
6          8  51   Male     33              25             9            26           Premium
  Contract.Length Total.Spend Last.Interaction Churn
1          Annual         932               17     1
2         Monthly         557                6     1
3       Quarterly         185                3     1
4         Monthly         396               29     1
5         Monthly         617               20     1
6          Annual         129                8     1
```

Exploratory data analysis

Code:

```
str(data)
```

Output:

```
'data.frame':   440833 obs. of  12 variables:
 $ CustomerID       : int  2 3 4 5 6 8 9 10 11 12 ...
 $ Age              : int  30 65 55 58 23 51 58 55 39 64 ...
 $ Gender           : chr  "Female" "Female" "Female" "Male" ...
 $ Tenure           : int  39 49 14 38 32 33 49 37 12 3 ...
 $ Usage.Frequency  : int  14 1 4 21 20 25 12 8 5 25 ...
 $ Support.Calls    : int  5 10 6 7 5 9 3 4 7 2 ...
 $ Payment.Delay    : int  18 8 18 7 8 26 16 15 4 11 ...
 $ Subscription.Type: chr  "Standard" "Basic" "Basic" "Standard" ...
 $ Contract.Length  : chr  "Annual" "Monthly" "Quarterly" "Monthly" ...
 $ Total.Spend      : num  932 557 185 396 617 129 821 445 969 415 ...
 $ Last.Interaction : int  17 6 3 29 20 8 24 30 13 29 ...
 $ Churn            : int  1 1 1 1 1 1 1 1 1 1 ...
```

We don't need customer ID. So, we deleting customer ID column

Code:

```
data <- subset(data, select = -CustomerID)

head(data)
```

Output:

```
  Age Gender Tenure Usage.Frequency Support.Calls Payment.Delay Subscription.Type Contract.Length Total.Spend Last.Interaction
1  30 Female     39              14             5            18          Standard          Annual         932               17
2  65 Female     49               1            10             8             Basic         Monthly         557                6
3  55 Female     14               4             6            18             Basic       Quarterly         185                3
4  58   Male     38              21             7             7          Standard         Monthly         396               29
5  23   Male     32              20             5             8             Basic         Monthly         617               20
6  51   Male     33              25             9            26           Premium          Annual         129                8
  Churn
1     1
2     1
3     1
4     1
5     1
6     1
```

Checking for duplicate or missing values in the dataset

Code:

```
sum(duplicated(data))


sum(is.na(data))
```

Output:

```
> sum(duplicated(data))
[1] 0
> sum(is.na(data))
[1] 8
```

8 missing values found. So, we decided to delete the rows containing missing values

Code:

```
data <- na.omit(data)


sum(is.na(data))
```

Output:

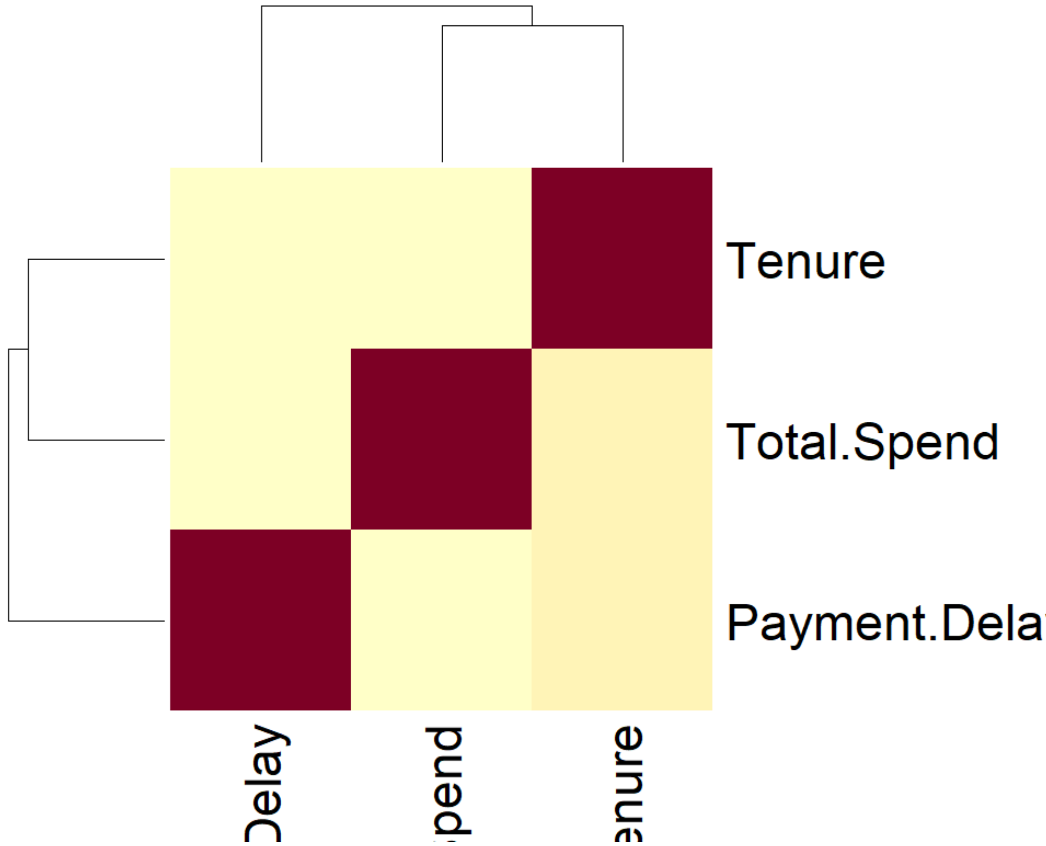
```
> sum(is.na(data))
[1] 0
```

Showing correlation between total money spend, payment delay and Tenure column

Code:

```
subset(as.matrix(Filter(is.numeric, na.exclude(distinct(data)))),
       select = c(Total.Spend,Payment.Delay, Tenure)) %>% cor() %>% heatmap()
```

Output:



Train Test split

Code:

```
sample <- sample(c(TRUE, FALSE), nrow(data), replace=TRUE, prob=c(0.7,0.3))
train  <- data[sample, ]
test   <- data[!sample, ]
```

Decision Tree with Gini index

Training the model and Testing

Code:

```
DT_Model_gini <- rpart(formula = Churn ~., data = train, method = "class", parms = list(split = "gini"))

test_pred_gini = predict(DT_Model_gini, newdata= test, type = "class")
```

Showing accuracy

Code:

```
sum(test_pred_gini == test$Churn) / nrow(test)
```

Output:

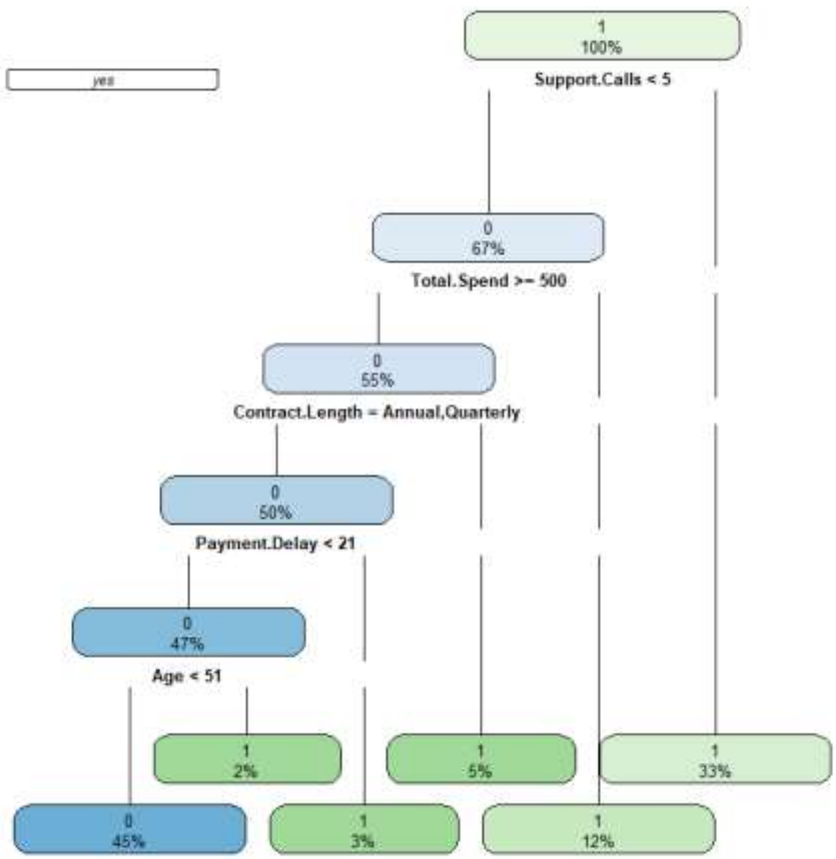
```
[1] 0.9756737
```

Showing How many rules are generated

Code:

```
rpart.plot(rpart(formula = Churn ~., data = train,
                 method = "class", parms = list(split = "gini")), extra = 100)
```

Output:



Showing the confusion matrix

Code:

```
confusion_matrix <- table(test_pred_gini, test$Churn)


print(confusion_matrix)
```

Output:

```
test_pred_gini     0     1
             0 56701  2813
             1   395 71965
```

Showing Precision, recall and F1 score

Code:

```
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
precision <- diag(confusion_matrix) / rowSums(confusion_matrix)
recall <- diag(confusion_matrix) / colSums(confusion_matrix)
f1_score <- 2 * (precision * recall) / (precision + recall)

report <- data.frame(Class = rownames(confusion_matrix),
                     Precision = precision,
                     Recall = recall,
                     F1_Score = f1_score)

print(report)
```

Output:

```
  Class Precision    Recall  F1_Score
0     0 0.9527338 0.9930818 0.9724895
1     1 0.9945412 0.9623820 0.9781973
```

Decision Tree with Information gain

Training the model and Testing

Code:

```
DT_Model_info <- rpart(formula = Churn ~., data = train, method = "class", parms = list(split = "information"))

test_pred_info = predict(DT_Model_info, newdata= test, type = "class")
```

Showing accuracy

Code:

```
sum(test_pred_info == test$Churn) / nrow(test)
```

Output:

```
0.9871923
```

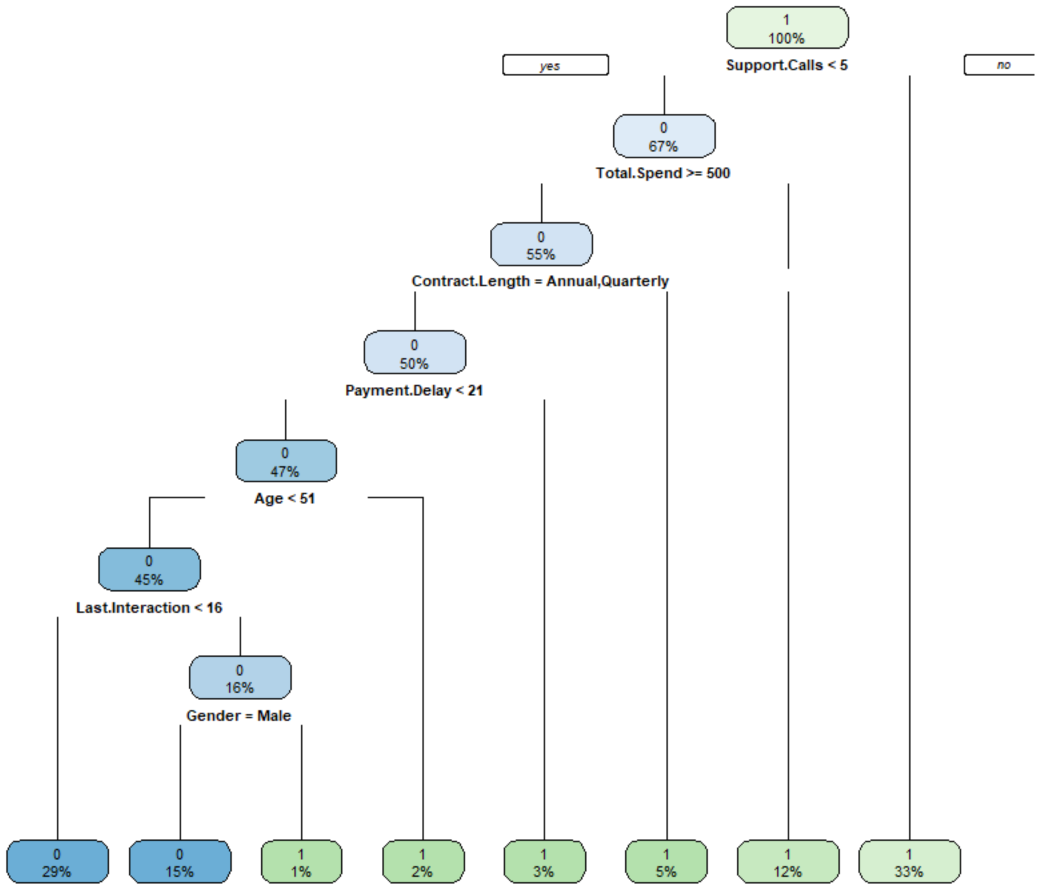
Code:

Showing How many rules are generated

Code:

```
rpart.plot(rpart(formula = Churn ~., data = train,
                 method = "class", parms = list(split = "information")), extra = 100)
```

Output:



Showing confusion matrix

Code:

```
confusion_matrix <- table(test_pred_info, test$Churn)

print(confusion_matrix)
```

Output:

```
test_pred_info     0     1
             0 56701  1294
             1   395 73484
```

Showing Precision, recall and F1 score

Code:

```
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
precision <- diag(confusion_matrix) / rowSums(confusion_matrix)
recall <- diag(confusion_matrix) / colSums(confusion_matrix)
f1_score <- 2 * (precision * recall) / (precision + recall)

report <- data.frame(Class = rownames(confusion_matrix),
                     Precision = precision,
                     Recall = recall,
                     F1_Score = f1_score)

print(report)
```

Output:

```
  Class Precision    Recall  F1_Score
0     0 0.9776877 0.9930818 0.9853247
1     1 0.9946534 0.9826954 0.9886383
```

Summary

	accuracy	Precision	Recall	F1	TP	TN	FP	FN	No. of rules
Gini	97%	0 95% 1 99%	0 99% 1 96%	0 97% 1 97%	56701	71965	395	2813	6
Information	98%	0 97% 1 99%	0 99% 1 98%	0 98% 1 98%	56701	73484	395	1294	8

Accuracy: This indicates the overall correctness of the decision tree's predictions on the dataset. The Gini-based tree has an accuracy of 97%, while the Information Gain-based tree has an accuracy of 98%. This suggests that both trees are performing well.

Precision: Precision is a measure of how many of the positive predictions made by the model are actually correct. For class 0 and class 1, both trees have a precision of 99%, meaning that the vast majority of the positive predictions are accurate for both classes.

Recall: Recall measures how well the model is able to identify all positive instances. For class 0, both trees have a recall of 95%, while for class 1, the recall is 96% for the Gini-based tree and 98% for the Information Gain-based tree. This indicates that the Information Gain-based tree is better at correctly identifying positive instances of class 1.

F1-Score: The F1-Score is a balanced measure that takes both precision and recall into account. For class 0, both trees have an F1-Score of 99%, while for class 1, the F1-Score is 97% for the Gini-based tree and 98% for the Information Gain-based tree. Again, the Information Gain-based tree has a slight advantage in class 1.

True Positives (TP): The number of instances that are correctly predicted as positive.

True Negatives (TN): The number of instances that are correctly predicted as negative.

False Positives (FP): The number of instances that are predicted as positive but are actually negative.

False Negatives (FN): The number of instances that are predicted as negative but are actually positive.

Number of Rules: This column indicates the number of rules extracted from each decision tree. The Gini-based tree generates 6 rules, while the Information Gain-based tree generates 8 rules. This suggests that the Information Gain-based tree have a more complex structure with more conditions for making decisions. As a result, we can say the Gini based tree is computationally less expensive

In summary, both decision trees seem to perform well, with the Information Gain-based tree having a slightly better accuracy, recall, and F1-Score for class 1 predictions. But the Gini based tree is computationally less expensive compare to information gain-based tree