CHURN PREDICTION MODEL

Step 1: preprocessing, cleaning data, and descriptive statistics.

```
% Importing the data, text file format and encoding file specified.
opts = delimitedTextImportOptions("NumVariables", 14, "Encoding", "UTF-8");
```

Specifying the column names and variables types (within the original data, all the variables are of the same type).

```
opts.VariableNames = ["AnonymousCustomerID", "CallFailure", "Complains", "SubscriptionLength",
opts.VariableTypes = ["double", "double", "double", "double", "double", "double", "double", "do
```

Importing data

```
churn_data = readtable("Churn_Dataset.csv", opts);
churn_data = rmmissing(churn_data);
```

Calling variables and converting Complains, AgeGroup, TariffPlan, Status, and Churn into categorical vars (this is done for further plotting).

```
AnonymousCustomerID      = churn_data.AnonymousCustomerID;
CallFailure              = churn_data.CallFailure;
Complains               = categorical(churn_data.Complains);
SubscriptionLength      = churn_data.SubscriptionLength;
ChargeAmount            = churn_data.ChargeAmount;
SecondsofUse            = churn_data.SecondsOfUse;
FrequencyofUse          = churn_data.FrequencyOfUse;
FrequencyofSMS          = churn_data.FrequencyOfSMS;
DistinctCalledNumbers   = churn_data.DistinctCalledNumbers;
AgeGroup                = categorical(churn_data.AgeGroup);
TariffPlan              = categorical(churn_data.TariffPlan);
Status                  = categorical(churn_data.Status);
Churn                   = categorical(churn_data.Churn);
CustomerValue           = churn_data.CustomerValue;
```

Descriptive Statistics | continuous variables (except AnonymousCustomerID)

2) CallFailure

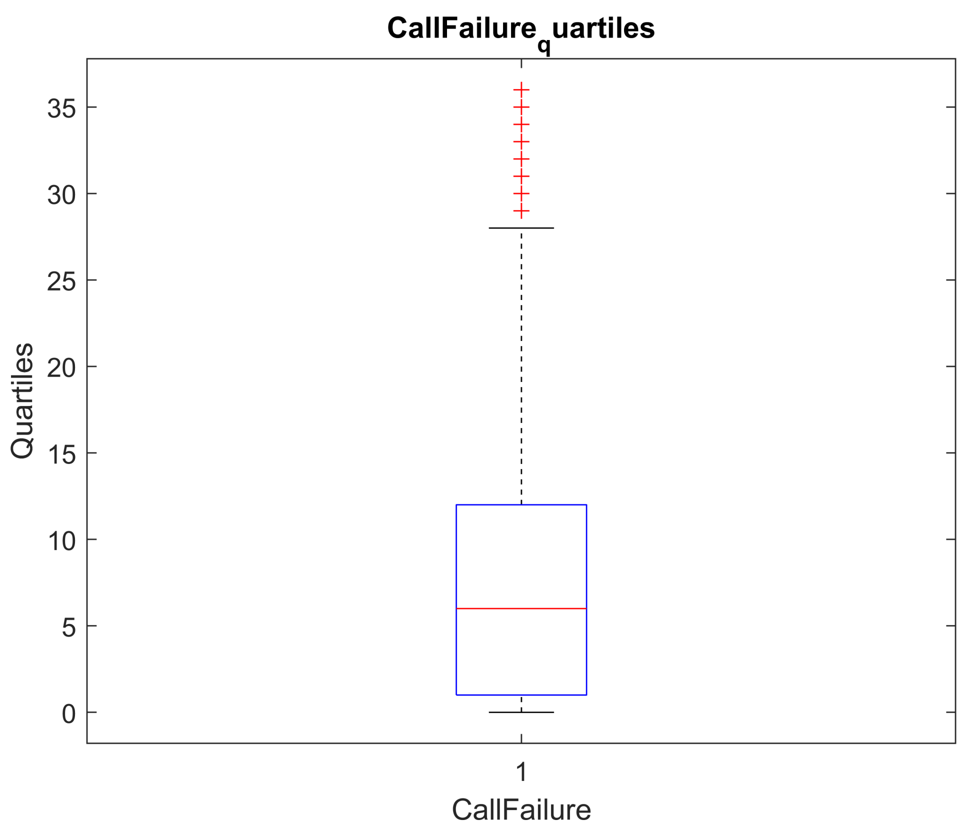
```
CallFailure_mean        = mean(CallFailure);
CallFailure_median      = median(CallFailure);
CallFailure_mode        = mode(CallFailure);
CallFailure_minimum     = min(CallFailure);
CallFailure_maximum     = max(CallFailure);
CallFailure_quartiles   = [0.25:0.25:1; quantile(CallFailure, 0.25:0.25:1)];


% Cell array for CallFailure stats
CallFailure_stats = {'Mean', CallFailure_mean; ...
         'Median', CallFailure_median; ...
         'Mode', CallFailure_mode; ...
```

```
          'Minimum', CallFailure_minimum; ...
          'Maximum', CallFailure_maximum; ...
          'Quartiles', CallFailure_quartiles}
```

CallFailure_stats = 6×2 cell

	1	2
1	'Mean'	7.6279
2	'Median'	6
3	'Mode'	0
4	'Minimum'	0
5	'Maximum'	36
6	'Quartiles'	[0.2500,0.5000,0.7500,1;1,...

```
% Boxplot - CallFailure_quartiles
figure(1);
boxplot(CallFailure)
xlabel('CallFailure')
ylabel('Quartiles')
title('CallFailure_quartiles')
```



4) SubscriptionLength

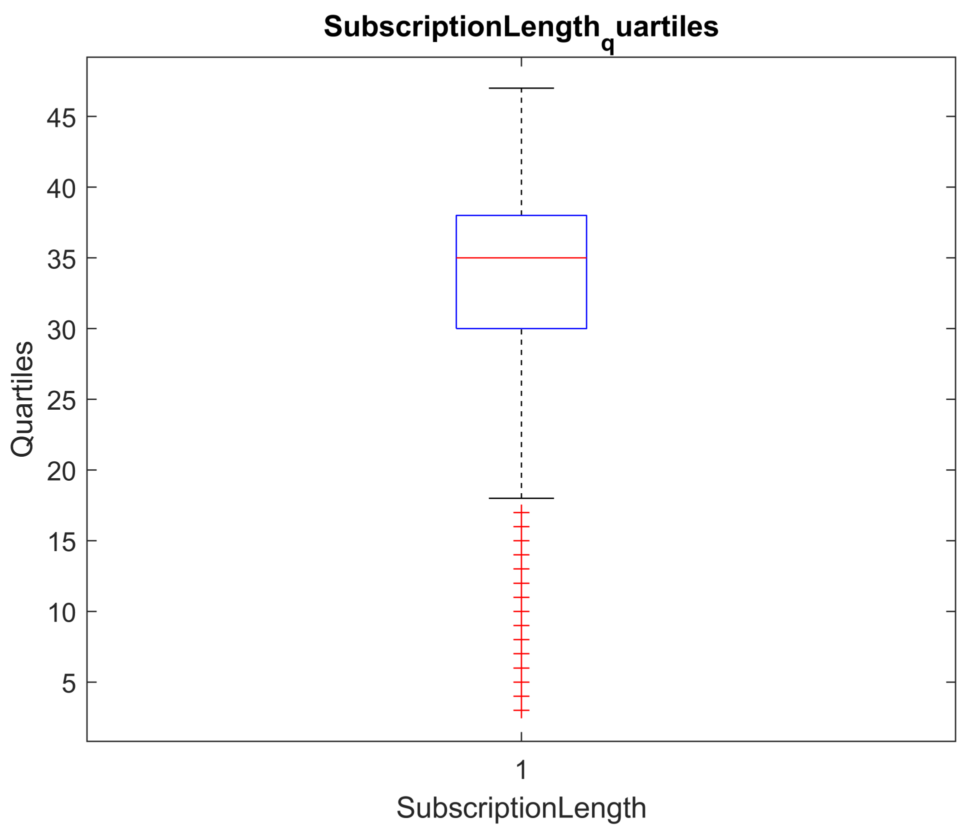
```
SubscriptionLength_mean         = mean(SubscriptionLength);
SubscriptionLength_median       = median(SubscriptionLength);
```

```
SubscriptionLength_mode        = mode(SubscriptionLength);
SubscriptionLength_minimum     = min(SubscriptionLength);
SubscriptionLength_maximum     = max(SubscriptionLength);
SubscriptionLength_quartiles   = [0.25:0.25:1; quantile(SubscriptionLength, 0.25:0.25:1)];


% Cell array for SubscriptionLength stats
SubscriptionLength_stats = {'Mean', SubscriptionLength_mean; ...
         'Median', SubscriptionLength_median; ...
         'Mode', SubscriptionLength_mode; ...
         'Minimum', SubscriptionLength_minimum; ...
         'Maximum', SubscriptionLength_maximum; ...
         'Quartiles', SubscriptionLength_quartiles}
```

SubscriptionLength_stats = 6×2 cell

	1	2
1	'Mean'	32.5419
2	'Median'	35
3	'Mode'	36
4	'Minimum'	3
5	'Maximum'	47
6	'Quartiles'	[0.2500,0.5000,0.7500,1;30,3...

```
% Boxplot - SubscriptionLength_quartiles
figure(2);
boxplot(SubscriptionLength)
xlabel('SubscriptionLength')
ylabel('Quartiles')
title('SubscriptionLength_quartiles')
```



5) ChargeAmount

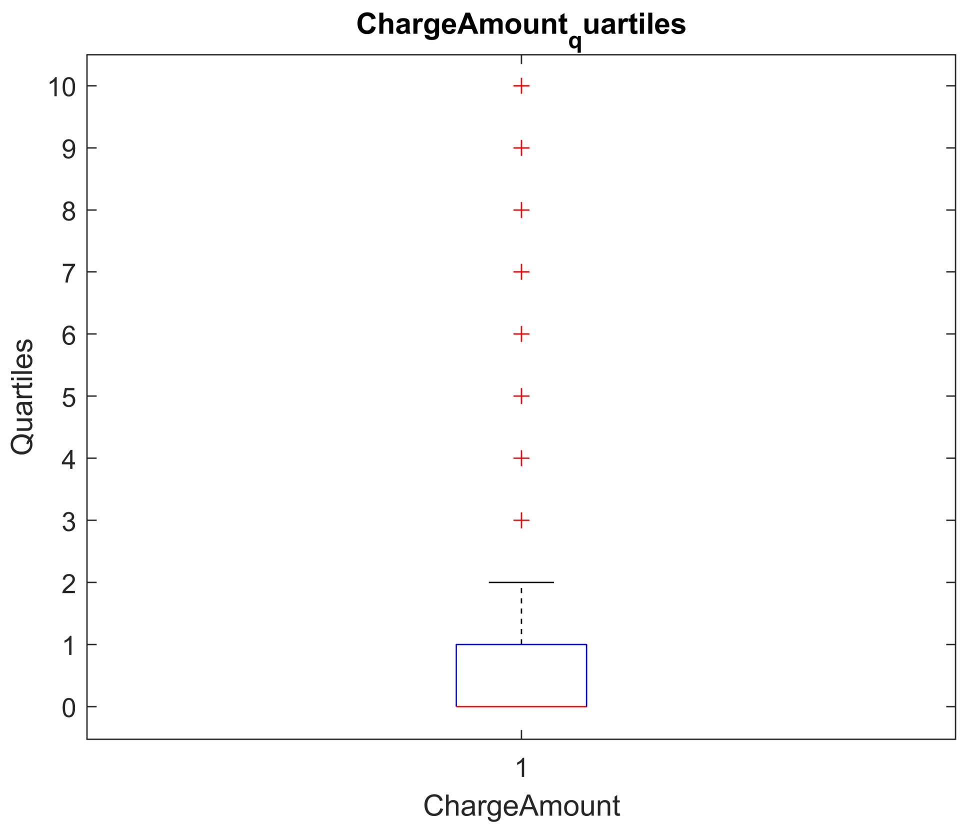
```
ChargeAmount_mean         = mean(ChargeAmount);
ChargeAmount_median       = median(ChargeAmount);
ChargeAmount_mode         = mode(ChargeAmount);
ChargeAmount_minimum      = min(ChargeAmount);
ChargeAmount_maximum      = max(ChargeAmount);
ChargeAmount_quartiles    = [0.25:0.25:1; quantile(ChargeAmount, 0.25:0.25:1)];


% Cell array for ChargeAmount stats
ChargeAmount_stats = {'Mean', ChargeAmount_mean; ...
         'Median', ChargeAmount_median; ...
         'Mode', ChargeAmount_mode; ...
         'Minimum', ChargeAmount_minimum; ...
         'Maximum', ChargeAmount_maximum; ...
         'Quartiles', ChargeAmount_quartiles}
```

ChargeAmount_stats = 6×2 cell

	1	2
1	'Mean'	0.9429
2	'Median'	0
3	'Mode'	0
4	'Minimum'	0
5	'Maximum'	10

	1	2
6	'Quartiles'	[0.2500,0.5000,0.7500,1;0...

```
% Boxplot - ChargeAmount_quartiles
figure(3);
boxplot(ChargeAmount)
xlabel('ChargeAmount')
ylabel('Quartiles')
title('ChargeAmount_quartiles')
```



6) SecondsofUse

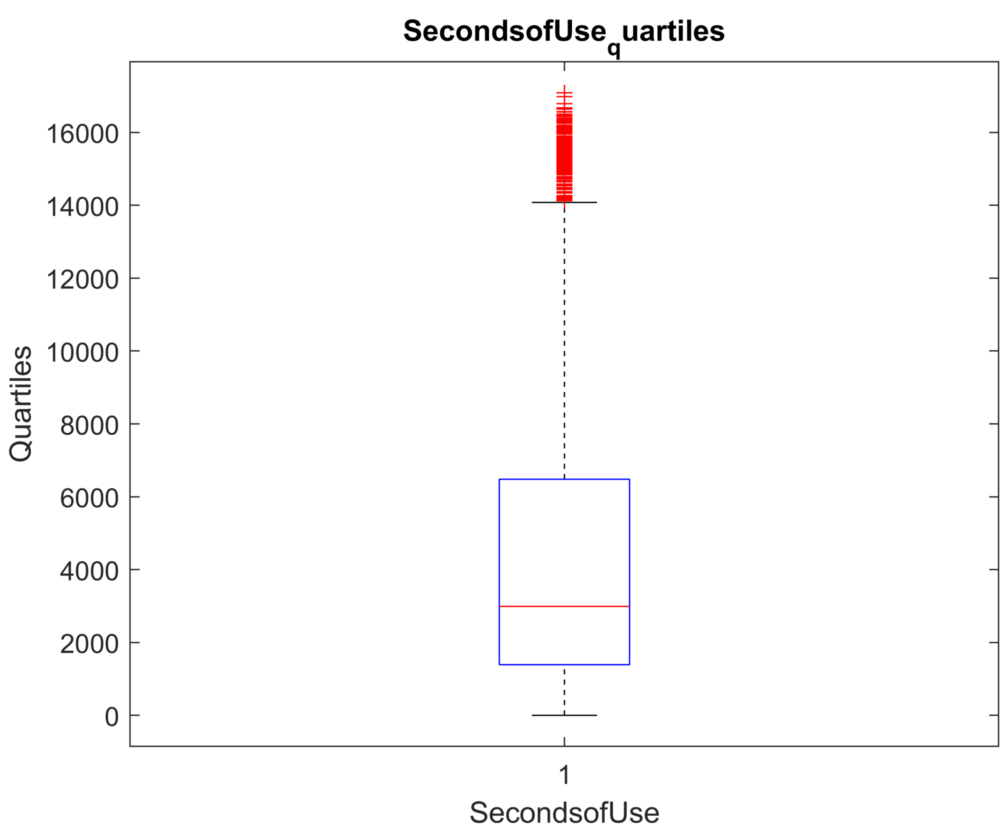
```
SecondsofUse_mean        = mean(SecondsofUse);
SecondsofUse_median      = median(SecondsofUse);
SecondsofUse_mode        = mode(SecondsofUse);
SecondsofUse_minimum     = min(SecondsofUse);
SecondsofUse_maximum     = max(SecondsofUse);
SecondsofUse_quartiles   = [0.25:0.25:1; quantile(SecondsofUse, 0.25:0.25:1)];


% Cell array for SecondsofUse stats
SecondsofUse_stats = {'Mean', SecondsofUse_mean; ...
         'Median', SecondsofUse_median; ...
         'Mode', SecondsofUse_mode; ...
         'Minimum', SecondsofUse_minimum; ...
         'Maximum', SecondsofUse_maximum; ...
```

```
        'Quartiles', SecondsofUse_quartiles}
```

SecondsofUse_stats = 6×2 cell

	1	2
1	'Mean'	4472.5
2	'Median'	2990
3	'Mode'	0
4	'Minimum'	0
5	'Maximum'	17090
6	'Quartiles'	[0.2500,0.5000,0.7500,1;1390,2990,6...

```
% Boxplot - SecondsofUse_quartiles
figure(4);
boxplot(SecondsofUse)
xlabel('SecondsofUse')
ylabel('Quartiles')
title('SecondsofUse_quartiles')
```



7) FrequencyofUse

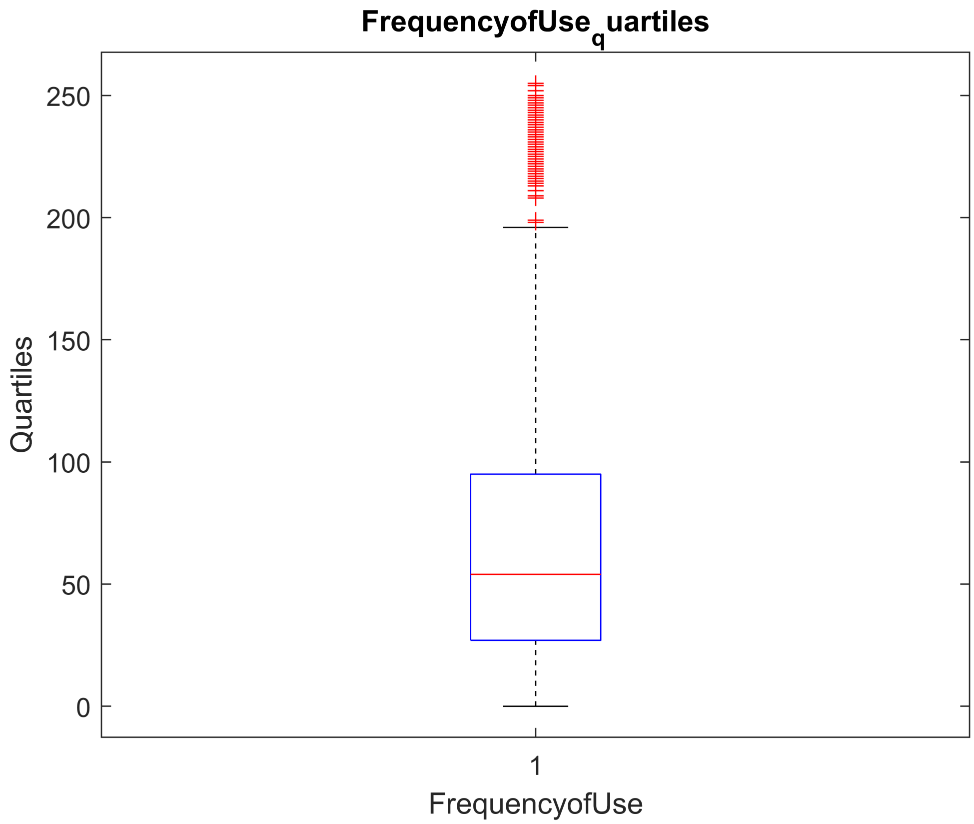
```
FrequencyofUse_mean        = mean(FrequencyofUse);
FrequencyofUse_median      = median(FrequencyofUse);
FrequencyofUse_mode        = mode(FrequencyofUse);
FrequencyofUse_minimum     = min(FrequencyofUse);
```

```
FrequencyofUse_maximum     = max(FrequencyofUse);
FrequencyofUse_quartiles   = [0.25:0.25:1; quantile(FrequencyofUse, 0.25:0.25:1)];


% Cell array for FrequencyofUse stats
FrequencyofUse_stats = {'Mean', FrequencyofUse_mean; ...
         'Median', FrequencyofUse_median; ...
         'Mode', FrequencyofUse_mode; ...
         'Minimum', FrequencyofUse_minimum; ...
         'Maximum', FrequencyofUse_maximum; ...
         'Quartiles', FrequencyofUse_quartiles}
```

FrequencyofUse_stats = 6×2 cell

	1	2
1	'Mean'	69.4606
2	'Median'	54
3	'Mode'	0
4	'Minimum'	0
5	'Maximum'	255
6	'Quartiles'	[0.2500,0.5000,0.7500,1;27,5...

```
% Boxplot - FrequencyofUse_quartiles
figure(5);
boxplot(FrequencyofUse)
xlabel('FrequencyofUse')
ylabel('Quartiles')
title('FrequencyofUse_quartiles')
```



8) FrequencyofSMS

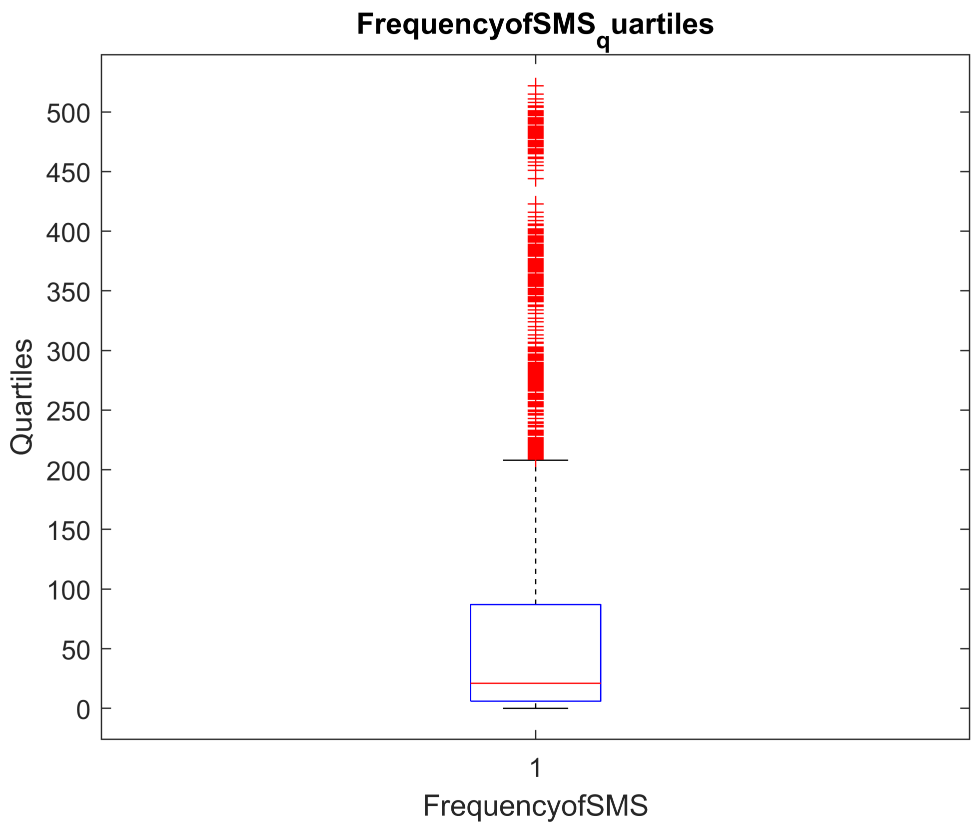
```
FrequencyofSMS_mean        = mean(FrequencyofSMS);
FrequencyofSMS_median      = median(FrequencyofSMS);
FrequencyofSMS_mode        = mode(FrequencyofSMS);
FrequencyofSMS_minimum     = min(FrequencyofSMS);
FrequencyofSMS_maximum     = max(FrequencyofSMS);
FrequencyofSMS_quartiles   = [0.25:0.25:1; quantile(FrequencyofSMS, 0.25:0.25:1)];


% Cell array for FrequencyofSMS stats
FrequencyofSMS_stats = {'Mean', FrequencyofSMS_mean; ...
         'Median', FrequencyofSMS_median; ...
         'Mode', FrequencyofSMS_mode; ...
         'Minimum', FrequencyofSMS_minimum; ...
         'Maximum', FrequencyofSMS_maximum; ...
         'Quartiles', FrequencyofSMS_quartiles}
```

FrequencyofSMS_stats = 6×2 cell

	1	2
1	'Mean'	73.1749
2	'Median'	21
3	'Mode'	0
4	'Minimum'	0
5	'Maximum'	522

	1	2
6	'Quartiles'	[0.2500,0.5000,0.7500,1;6,21...

```
% Boxplot - FrequencyofSMS_quartiles
figure(6);
boxplot(FrequencyofSMS)
xlabel('FrequencyofSMS')
ylabel('Quartiles')
title('FrequencyofSMS_quartiles')
```



9) DistinctCalledNumbers

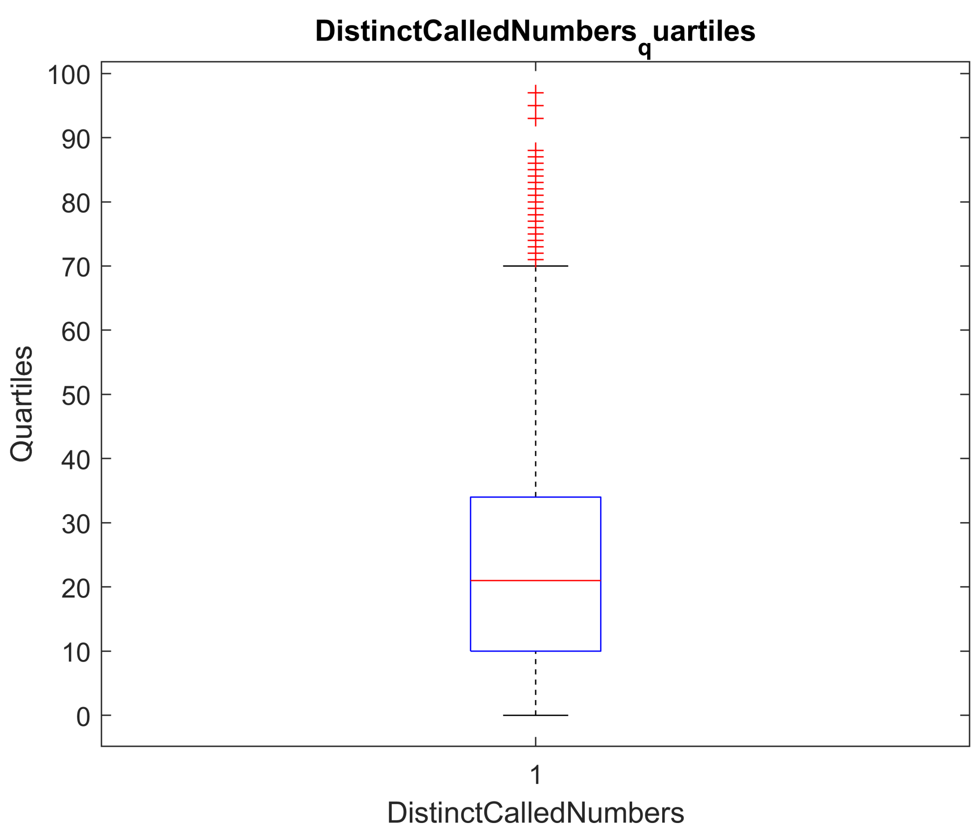
```
DistinctCalledNumbers_mean        = mean(DistinctCalledNumbers);
DistinctCalledNumbers_median      = median(DistinctCalledNumbers);
DistinctCalledNumbers_mode        = mode(DistinctCalledNumbers);
DistinctCalledNumbers_minimum     = min(DistinctCalledNumbers);
DistinctCalledNumbers_maximum     = max(DistinctCalledNumbers);
DistinctCalledNumbers_quartiles   = [0.25:0.25:1; quantile(DistinctCalledNumbers, 0.25:0.25:1)]


% Cell array for DistinctCalledNumbers stats
DistinctCalledNumbers_stats = {'Mean', DistinctCalledNumbers_mean; ...
         'Median', DistinctCalledNumbers_median; ...
         'Mode', DistinctCalledNumbers_mode; ...
         'Minimum', DistinctCalledNumbers_minimum; ...
         'Maximum', DistinctCalledNumbers_maximum; ...
```

```
        'Quartiles', DistinctCalledNumbers_quartiles}
```

DistinctCalledNumbers_stats = 6×2 cell

	1	2
1	'Mean'	23.5098
2	'Median'	21
3	'Mode'	0
4	'Minimum'	0
5	'Maximum'	97
6	'Quartiles'	[0.2500,0.5000,0.7500,1;10,2...

```
% Boxplot - DistinctCalledNumbers_quartiles
figure(7);
boxplot(DistinctCalledNumbers)
xlabel('DistinctCalledNumbers')
ylabel('Quartiles')
title('DistinctCalledNumbers_quartiles')
```



14) CustomerValue

```
CustomerValue_mean        = mean(CustomerValue);
CustomerValue_median      = median(CustomerValue);
CustomerValue_mode        = mode(CustomerValue);
CustomerValue_minimum     = min(CustomerValue);
```

```
CustomerValue_maximum      = max(CustomerValue);
CustomerValue_quartiles    = [0.25:0.25:1; quantile(CustomerValue, 0.25:0.25:1)];


% Cell array for CustomerValue stats
CustomerValue_stats = {'Mean', CustomerValue_mean; ...
         'Median', CustomerValue_median; ...
         'Mode', CustomerValue_mode; ...
         'Minimum', CustomerValue_minimum; ...
         'Maximum', CustomerValue_maximum; ...
         'Quartiles', CustomerValue_quartiles}
```

CustomerValue_stats = 6×2 cell

	1	2
1	'Mean'	170.9503
2	'Median'	120.675
3	'Mode'	0
4	'Minimum'	0
5	'Maximum'	817.65
6	'Quartiles'	[0.2500,0.5000,0.7500,1;47.0100,120.6750,241.9...

```
% Boxplot - CustomerValue_quartiles
figure(8);
boxplot(CustomerValue)
xlabel('CustomerValue')
ylabel('Quartiles')
title('CustomerValue_quartiles')
```

Std. dev.

```
CallFailure_std              = [std(CallFailure)];
SubscriptionLength_std    = [std(SubscriptionLength)];
ChargeAmount_std          = [std(ChargeAmount)];
SecondsofUse_std          = [std(SecondsofUse)];
FrequencyofUse_std        = [std(FrequencyofUse)];
FrequencyofSMS_std        = [std(FrequencyofSMS)];
DistinctCalledNumbers_std = [std(DistinctCalledNumbers)];
CustomerValue_std         = [std(CustomerValue)];

std_dev_cont   = [CallFailure_std(1), ...
             SubscriptionLength_std(1), ...
             ChargeAmount_std(1), ...
             SecondsofUse_std(1), ...
             FrequencyofUse_std(1), ...
             FrequencyofSMS_std(1), ...
             DistinctCalledNumbers_std(1), ...
             CustomerValue_std(1)];
```

Skewness and Kurtosis

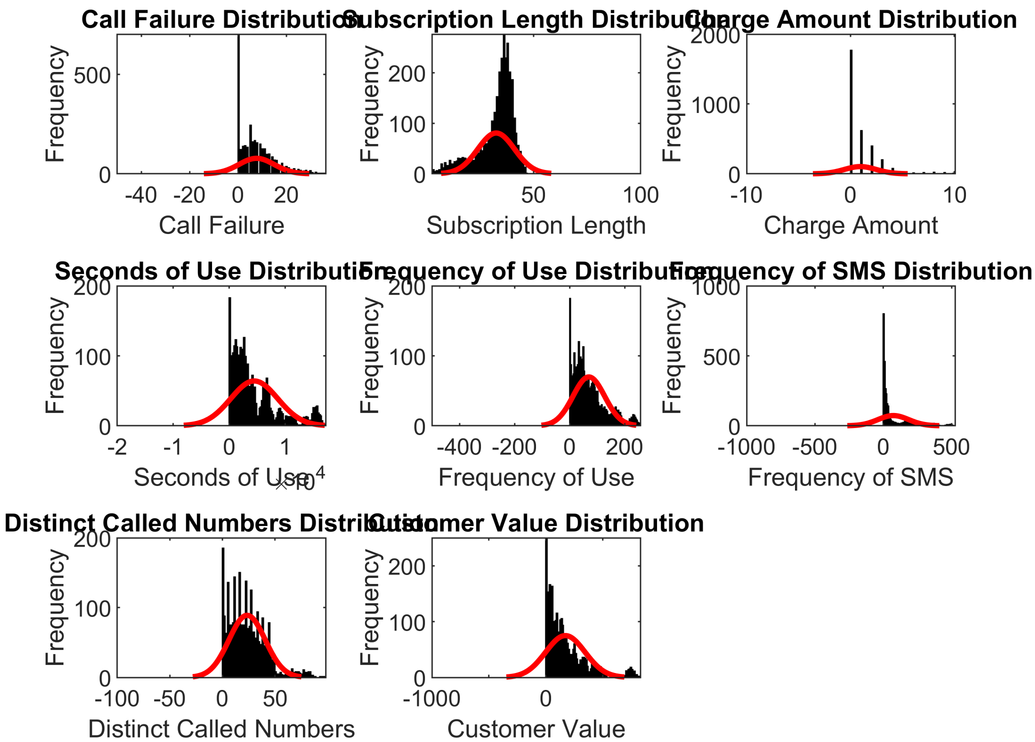
```
CallFailure_skew_kur           = [skewness(CallFailure), kurtosis(CallFailure)];
SubscriptionLength_skew_kur    = [skewness(SubscriptionLength), kurtosis(SubscriptionLength)];
ChargeAmount_skew_kur          = [skewness(ChargeAmount), kurtosis(ChargeAmount)];
SecondsofUse_skew_kur          = [skewness(SecondsofUse), kurtosis(SecondsofUse)];
FrequencyofUse_skew_kur        = [skewness(FrequencyofUse), kurtosis(FrequencyofUse)];
FrequencyofSMS_skew_kur        = [skewness(FrequencyofSMS), kurtosis(FrequencyofSMS)];
```

```
DistinctCalledNumbers_skew_kur = [skewness(DistinctCalledNumbers), kurtosis(DistinctCalledNumbe
CustomerValue_skew_kur         = [skewness(CustomerValue), kurtosis(CustomerValue)];

skew_cont = [CallFailure_skew_kur(1), SubscriptionLength_skew_kur(1), ...
        ChargeAmount_skew_kur(1), SecondsofUse_skew_kur(1), FrequencyofUse_skew_kur(1), ...
        FrequencyofSMS_skew_kur(1), DistinctCalledNumbers_skew_kur(1), CustomerValue_skew_kur(1
kurt_cont = [CallFailure_skew_kur(2), SubscriptionLength_skew_kur(2), ...
        ChargeAmount_skew_kur(2), SecondsofUse_skew_kur(2), FrequencyofUse_skew_kur(2), ...
        FrequencyofSMS_skew_kur(2), DistinctCalledNumbers_skew_kur(2), CustomerValue_skew_kur(2
```

Plotting distribution

```
figure(9)

% Call Failure
subplot(3,3,1)
histfit(CallFailure, 80)
xlabel('Call Failure')
ylabel('Frequency' )
title('Call Failure Distribution')

% Subscription Length
subplot(3,3,2)
histfit(SubscriptionLength, 80)
xlabel('Subscription Length')
ylabel('Frequency' )
title('Subscription Length Distribution')

% Charge Amount
subplot(3,3,3)
histfit(ChargeAmount, 80)
xlabel('Charge Amount')
ylabel('Frequency' )
title('Charge Amount Distribution')

% Seconds of Use
subplot(3,3,4)
histfit(SecondsofUse, 80)
xlabel('Seconds of Use')
ylabel('Frequency' )
title('Seconds of Use Distribution')

% Frequency of Use
subplot(3,3,5)
histfit(FrequencyofUse, 80)
xlabel('Frequency of Use')
ylabel('Frequency' )
title('Frequency of Use Distribution')

% Frequency of SMS
subplot(3,3,6)
histfit(FrequencyofSMS, 80)
xlabel('Frequency of SMS')
ylabel('Frequency' )
```

```
title('Frequency of SMS Distribution')

% Distinct Called Numbers
subplot(3,3,7)
histfit(DistinctCalledNumbers, 80)
xlabel('Distinct Called Numbers')
ylabel('Frequency' )
title('Distinct Called Numbers Distribution')

% Customer Value
subplot(3,3,8)
histfit(CustomerValue, 80)
xlabel('Customer Value')
ylabel('Frequency' )
title('Customer Value Distribution')
```

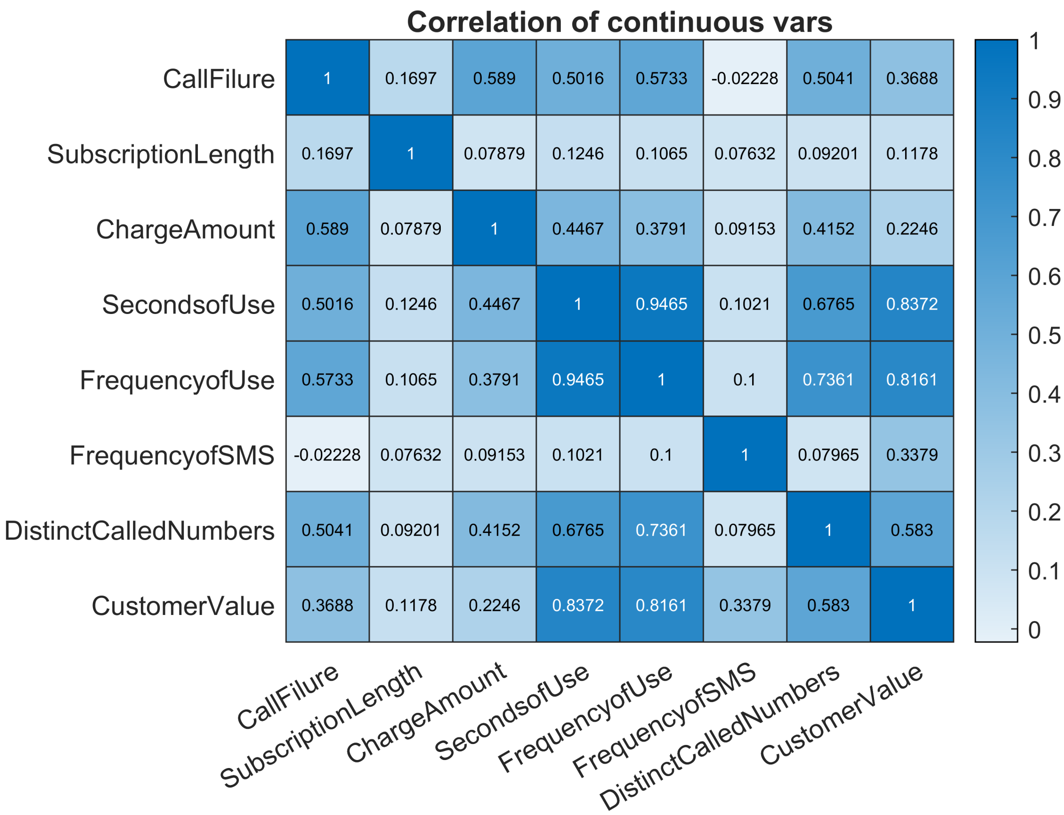


Correlation plot of cotinuous variables

- *Please, **DO NOT** integrate this section with the previous section!*
- *For some reason, if done, the correlation plot appears inside the distributions plot.*
- *Also figure number for corrplot "figure(500)" is a huge number **on purpose** cause earlier it was mixing with the distribution plots.*
- *Otherwise, you can follow the continuous figure numbers after **"figure (11)".** Thanks and apologies for any inconvenience.*

```
figure(500)
```

```
cont_var = [CallFailure, SubscriptionLength, ChargeAmount, SecondsofUse, ...
            FrequencyofUse, FrequencyofSMS, DistinctCalledNumbers, CustomerValue];
h1 = heatmap(corrcoef(cont_var));
labels = ["CallFilure", "SubscriptionLength", "ChargeAmount", "SecondsofUse", ...
          "FrequencyofUse", "FrequencyofSMS", "DistinctCalledNumbers", "CustomerValue"];
h1.XDisplayLabels = labels;
h1.YDisplayLabels = labels;
title('Correlation of continuous vars')
```



Descriptive Statistics | categorical variables

Frequency distributions

```
% Converting complains, AgeGroup, TariffPlan, Status, and Churn into categorical vars.
Complains_Freq_dist  = tabulate(churn_data.Complains);
AgeGroup_Freq_dist   = tabulate(churn_data.AgeGroup);
TariffPlan_Freq_dist = tabulate(churn_data.TariffPlan);
Status_Freq_dist     = tabulate(churn_data.Status);
Churn_Freq_dist      = tabulate(churn_data.Churn);
```

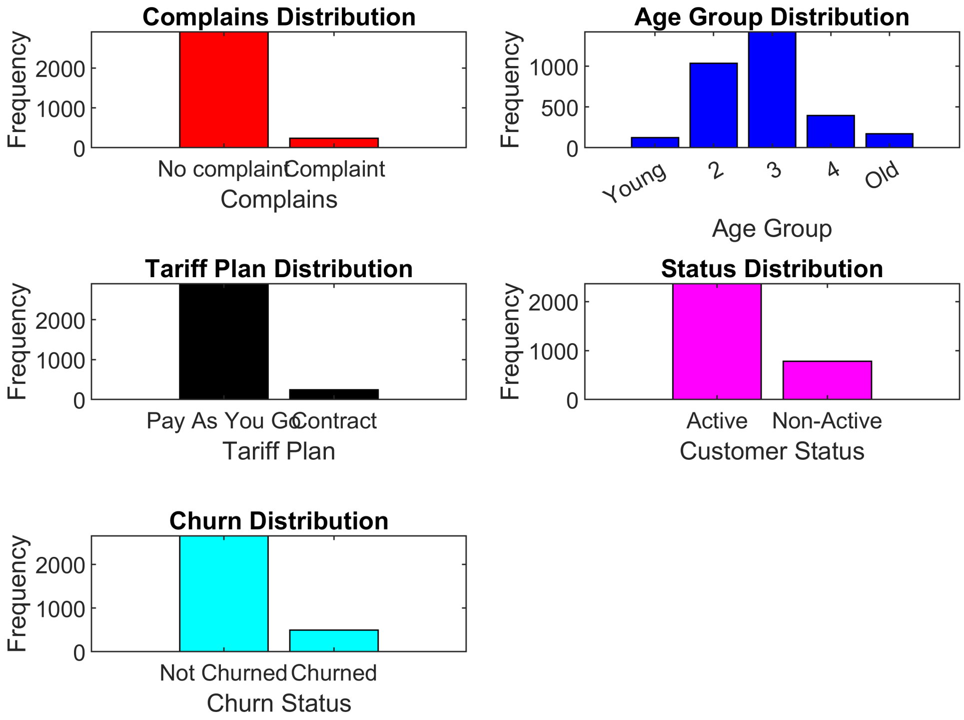
Std. dev.

```
Complains_std                 = [std(churn_data.Complains)];
AgeGroup_std                = [std(churn_data.AgeGroup)];
TariffPlan_std              = [std(churn_data.TariffPlan)];
```

```
Status_std                  = [std(churn_data.Status)];
Churn_std                   = [std(churn_data.Churn)];

std_dev_cat   = [Complains_std(1), ...
             AgeGroup_std(1), ...
             TariffPlan_std(1), ...
             Status_std(1), ...
             Churn_std(1)];
```

Plotting frequency distribution

```
figure(11)

% Complains
subplot(3,2,1)
bar(Complains_Freq_dist(:,1), Complains_Freq_dist(:,2), 'red')
xticklabels({'No complaint', 'Complaint'})
xlabel('Complains')
ylabel('Frequency')
title('Complains Distribution')

% Age Group
subplot(3,2,2)
bar(AgeGroup_Freq_dist(:,1), AgeGroup_Freq_dist(:,2), 'blue')
xticklabels({'Young', '2', '3', '4', 'Old'})
xlabel('Age Group')
ylabel('Frequency')
title('Age Group Distribution')

% Traffic Plan
subplot(3,2,3)
bar(TariffPlan_Freq_dist(:,1), TariffPlan_Freq_dist(:,2), 'black')
xticklabels({'Pay As You Go', 'Contract'})
xlabel('Tariff Plan')
ylabel('Frequency')
title('Tariff Plan Distribution')

% Status
subplot(3,2,4)
bar(Status_Freq_dist(:,1), Status_Freq_dist(:,2), 'magenta')
xticklabels({'Active', 'Non-Active'})
xlabel('Customer Status')
ylabel('Frequency')
title('Status Distribution')

% Churn
subplot(3,2,5)
bar(Churn_Freq_dist(:,1), Churn_Freq_dist(:,2), 'cyan')
xticklabels({'Not Churned', 'Churned'})
xlabel('Churn Status')
ylabel('Frequency')
title('Churn Distribution')
```



Step 2: Clustering analysis

```
% Extract the required variables for analysis part
CallFailure = churn_data.CallFailure;
Complains = churn_data.Complains;
SubscriptionLength = churn_data.SubscriptionLength;
ChargeAmount = churn_data.ChargeAmount;
SecondsOfUse = churn_data.SecondsOfUse;
FrequencyOfUse = churn_data.FrequencyOfUse;
FrequencyOfSMS = churn_data.FrequencyOfSMS;
DistinctCalledNumbers = churn_data.DistinctCalledNumbers;
AgeGroup = churn_data.AgeGroup;
TariffPlan = churn_data.TariffPlan;
Status = churn_data.Status;
Churn = churn_data.Churn;
CustomerValue = churn_data.CustomerValue;
```

Now, we do the clustering part,

```
% Select the variables for clustering the dataset
clustering_vars = [CallFailure, SubscriptionLength, ChargeAmount, SecondsOfUse, FrequencyOfUse,
```

```
% Normalize the variables before doing clustering
normalized_vars = zscore(clustering_vars);
```

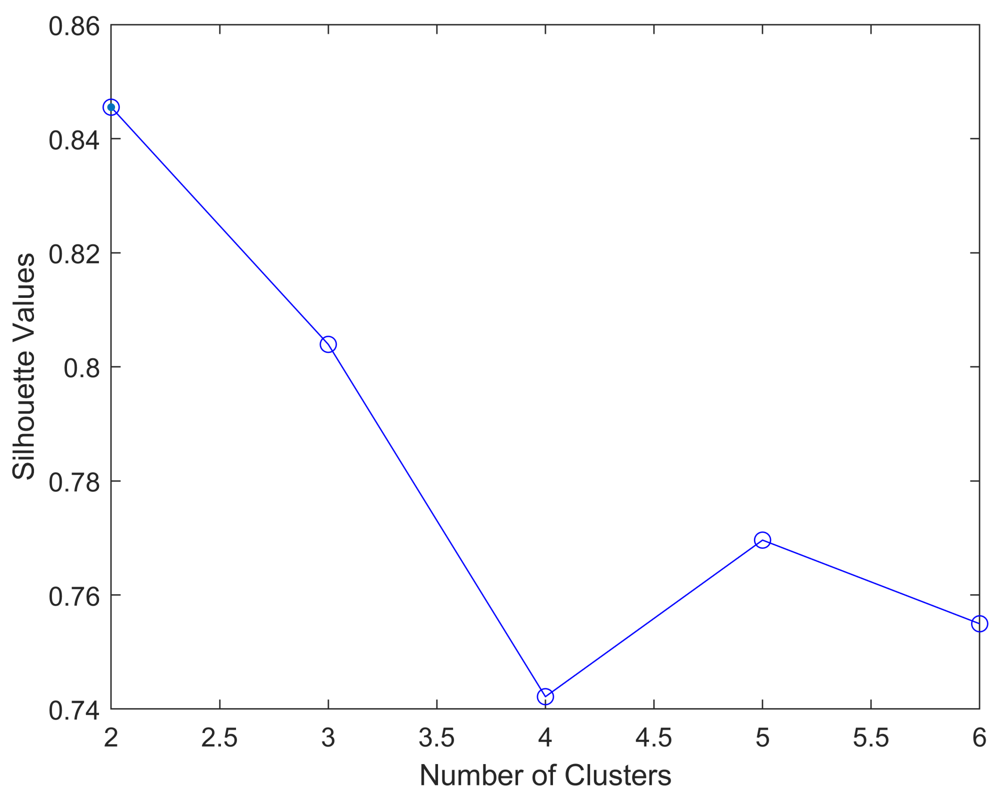
```
% Determination of optimal number of clusters
E = evalclusters(clustering_vars,'kmeans','silhouette','klist',[2:6])
```

```
E = 
  SilhouetteEvaluation with properties:

    NumObservations: 3150
         InspectedK: [2 3 4 5 6]
    CriterionValues: [0.8455 0.8039 0.7422 0.7696 0.7549]
           OptimalK: 2
```

```
kbest = E.OptimalK
```

```
kbest = 2
```

```
figure;
plot(E)
```




```
% Perform k-means clustering
[idx, centroids] = kmeans(normalized_vars, kbest);
```

```
% Assign cluster labels to the data to identify the cluster group
```

```
clustered_data = [churn_data, table(idx, 'VariableNames', {'Cluster'})]
```

clustered_data = 3150×15 table

...

	AnonymousCustomerID	CallFailure	Complains	SubscriptionLength
1	1	8	0	38
2	2	0	0	39
3	3	10	0	37
4	4	10	0	38
5	5	3	0	38
6	6	11	0	38
7	7	4	0	38
8	8	13	0	37
9	9	7	0	38
10	10	7	0	38
11	11	6	0	38
12	12	9	0	38
13	13	25	0	38
14	14	4	0	38
15	15	9	0	37
16	16	3	0	37
17	17	0	0	37
18	18	2	0	38
19	19	0	0	37
20	20	3	0	37
21	21	7	0	37
22	22	8	0	37
23	23	23	1	33
24	24	21	1	36
25	25	13	1	36
26	26	1	0	34
27	27	9	0	35
28	28	9	1	36
29	29	0	0	36
30	30	1	0	36
31	31	3	0	33

	AnonymousCustomerID	CallFailure	Complains	SubscriptionLength
32	32	0	1	36
33	33	0	0	36
34	34	25	0	31
35	35	6	0	34
36	36	2	0	36
37	37	3	0	35
38	38	6	0	36
39	39	4	0	27
40	40	4	0	26
41	41	11	0	27
42	42	0	0	26
43	43	10	0	25
44	44	2	0	25
45	45	2	0	18
46	46	16	0	17
47	47	12	0	15
48	48	8	0	16
49	49	2	0	9
50	50	3	0	9
51	51	11	0	40
52	52	3	0	41
53	53	13	0	39
54	54	13	0	40
55	55	6	0	40
56	56	14	0	40
57	57	7	0	40
58	58	16	0	39
59	59	10	0	40
60	60	10	0	40
61	61	9	0	40
62	62	12	0	40
63	63	28	0	40
64	64	7	0	40

	AnonymousCustomerID	CallFailure	Complains	SubscriptionLength
65	65	12	0	39
66	66	6	0	39
67	67	3	0	39
68	68	5	0	40
69	69	3	1	39
70	70	6	0	39
71	71	10	0	39
72	72	11	0	39
73	73	26	1	35
74	74	24	1	38
75	75	16	0	38
76	76	4	0	36
77	77	12	0	37
78	78	12	1	38
79	79	3	0	38
80	80	4	0	38
81	81	6	0	35
82	82	3	0	38
83	83	3	0	38
84	84	28	0	33
85	85	9	1	36
86	86	5	0	38
87	87	6	0	37
88	88	9	0	38
89	89	7	0	29
90	90	7	0	28
91	91	14	0	29
92	92	3	0	28
93	93	13	0	27
94	94	5	0	27
95	95	5	0	20
96	96	19	0	19
97	97	15	0	17

	AnonymousCustomerID	CallFailure	Complains	SubscriptionLength
98	98	11	0	18
99	99	5	0	11
100	100	6	1	11

⋮

```
% Display the clustered data result
disp(clustered_data);
```

AnonymousCustomerID	CallFailure	Complains	SubscriptionLength	ChargeAmount	SecondsOfUse	Frequen
1	8	0	38	0	4370	7
2	0	0	39	0	318	
3	10	0	37	0	2453	6
4	10	0	38	0	4198	6
5	3	0	38	0	2393	5
6	11	0	38	1	3775	8
7	4	0	38	0	2360	3
8	13	0	37	2	9115	12
9	7	0	38	0	13773	16
10	7	0	38	1	4515	8
11	6	0	38	0	5918	9
12	9	0	38	0	2238	5
13	25	0	38	3	15140	22
14	4	0	38	1	3095	2
15	9	0	37	0	15485	18
16	3	0	37	1	6500	8
17	0	0	37	0	875	1
18	2	0	38	0	710	1
19	0	0	37	0	0	
20	3	0	37	0	7508	12
21	7	0	37	1	11465	15
22	8	0	37	1	6718	7
23	23	1	33	0	955	4
24	21	1	36	8	10435	9
25	13	1	36	1	5818	9
26	1	0	34	0	2840	2
27	9	0	35	0	2990	4
28	9	1	36	0	2268	4
29	0	0	36	0	133	
30	1	0	36	0	1668	2
31	3	0	33	1	6785	9
32	0	1	36	0	628	
33	0	0	36	1	338	
34	25	0	31	3	16075	23
35	6	0	34	3	6965	11
36	2	0	36	0	3718	8
37	3	0	35	0	1665	4
38	6	0	36	0	2780	5
39	4	0	27	1	1315	3
40	4	0	26	0	1568	3
41	11	0	27	0	700	2
42	0	0	26	0	825	1
43	10	0	25	0	1225	3
44	2	0	25	0	3225	6
45	2	0	18	0	1695	2
46	16	0	17	1	6393	14

47	12	0	15	0	8933	17
48	8	0	16	3	2853	4
49	2	0	9	1	4390	4
50	3	0	9	1	1488	1
51	11	0	40	1	4430	7
52	3	0	41	1	378	
53	13	0	39	1	2513	6
54	13	0	40	1	4258	6
55	6	0	40	1	2453	6
56	14	0	40	2	3835	8
57	7	0	40	1	2420	4
58	16	0	39	3	9175	12
59	10	0	40	1	13833	17
60	10	0	40	2	4575	8
61	9	0	40	1	5978	9
62	12	0	40	1	2298	5
63	28	0	40	4	15200	22
64	7	0	40	2	3155	3
65	12	0	39	1	15545	18
66	6	0	39	2	6560	8
67	3	0	39	1	935	1
68	5	0	40	1	770	1
69	3	1	39	1	60	
70	6	0	39	1	7568	13
71	10	0	39	2	11525	15
72	11	0	39	2	6778	7
73	26	1	35	1	1015	5
74	24	1	38	9	10495	9
75	16	0	38	2	5878	10
76	4	0	36	1	2900	2
77	12	0	37	1	3050	4
78	12	1	38	1	2328	4
79	3	0	38	1	193	
80	4	0	38	1	1728	2
81	6	0	35	2	6845	10
82	3	0	38	1	688	1
83	3	0	38	2	398	
84	28	0	33	4	16135	24
85	9	1	36	4	7025	11
86	5	0	38	1	3778	8
87	6	0	37	1	1725	4
88	9	0	38	1	2840	5
89	7	0	29	2	1375	3
90	7	0	28	1	1628	4
91	14	0	29	1	760	2
92	3	0	28	1	885	1
93	13	0	27	1	1285	3
94	5	0	27	1	3285	6
95	5	0	20	1	1755	2
96	19	0	19	2	6453	14
97	15	0	17	1	8993	17
98	11	0	18	4	2913	4
99	5	0	11	2	4450	4
100	6	1	11	2	1548	2
101	5	0	36	0	4310	6
102	0	0	37	0	258	
103	7	0	35	0	2393	5
104	7	0	36	0	4138	6
105	0	0	36	0	2333	5
106	8	0	36	0	3715	7
107	1	0	36	0	2300	3
108	10	0	35	1	9055	11
109	4	0	36	0	13713	16
110	4	0	36	0	4455	8

111	3	0	36	0	5858	9
112	6	1	36	0	2178	5
113	22	0	36	2	15080	22
114	1	0	36	0	3035	2
115	6	0	35	0	15425	17
116	0	0	35	0	6440	8
117	0	0	35	0	815	1
118	0	0	36	0	650	1
119	0	0	35	0	0	
120	0	0	35	0	7448	12
121	4	0	35	0	11405	15
122	5	0	35	0	6658	7
123	20	0	31	0	895	4
124	18	0	34	7	10375	9
125	10	0	34	0	5758	9
126	0	0	32	0	2780	1
127	6	0	33	0	2930	3
128	6	1	34	0	2208	4
129	0	1	34	0	0	
130	0	0	34	0	1608	2
131	0	0	31	0	6725	9
132	0	0	34	0	568	
133	0	0	34	0	0	
134	22	0	29	2	16015	23
135	3	0	32	2	6905	11
136	0	0	34	0	3658	7
137	0	0	33	0	1605	4
138	3	0	34	0	2720	5
139	1	0	25	0	1255	3
140	1	0	24	0	1508	3
141	8	0	25	0	640	2
142	0	1	24	0	765	
143	7	0	23	0	1165	2
144	0	0	23	0	3165	5
145	0	0	16	0	1635	2
146	13	0	15	0	6333	13
147	9	0	13	0	8873	16
148	5	0	14	2	2793	4
149	0	0	7	0	4330	3
150	0	0	7	0	1428	1
151	9	0	42	1	4450	7
152	1	1	43	0	398	
153	11	0	41	1	2533	6
154	11	0	42	1	4278	6
155	4	0	42	1	2473	6
156	12	0	42	2	3855	8
157	5	0	42	1	2440	4
158	14	0	41	3	9195	12
159	8	0	42	1	13853	17
160	8	0	42	2	4595	8
161	7	0	42	1	5998	9
162	10	0	42	0	2318	5
163	26	0	42	4	15220	22
164	5	0	42	2	3175	2
165	10	0	41	1	15565	18
166	4	0	41	2	6580	8
167	1	0	41	0	955	1
168	3	0	42	0	790	1
169	1	0	41	0	80	
170	4	0	41	1	7588	12
171	8	0	41	2	11545	15
172	9	0	41	2	6798	7
173	24	1	37	0	1035	4
174	22	1	40	9	10515	9

175	14	0	40	2	5898
176	2	0	38	1	2920
177	10	0	39	0	3070
178	10	1	40	0	2348
179	1	0	40	0	213
180	2	0	40	1	1748
181	4	0	37	2	6865
182	1	1	40	0	708
183	1	0	40	2	418
184	26	0	35	4	16155
185	7	1	38	4	7045
186	3	0	40	1	3798
187	4	0	39	0	1745
188	7	0	40	1	2860
189	5	0	31	2	1395
190	5	0	30	1	1648
191	12	0	31	0	780
192	1	0	30	0	905
193	11	0	29	1	1305
194	3	0	29	1	3305
195	3	0	22	1	1775
196	17	0	21	2	6473
197	13	0	19	1	9013
198	9	0	20	4	2933
199	3	0	13	2	4470
200	4	0	13	2	1568
201	7	0	34	0	4290
202	0	0	35	0	0
203	9	0	33	0	2373
204	9	0	34	0	4118
205	2	0	34	0	2313
206	10	0	34	0	3695
207	3	0	34	0	2280
208	12	0	33	1	9035
209	6	0	34	0	13693
210	6	0	34	0	4435
211	5	0	34	0	5838
212	8	0	34	0	2158
213	24	0	34	2	15060
214	3	0	34	0	3015
215	8	0	33	0	15405
216	2	0	33	0	6420
217	0	0	33	0	795
218	1	0	34	0	630
219	0	0	33	0	0
220	2	0	33	0	7428
221	6	0	33	0	11385
222	7	0	33	0	6638
223	22	0	29	0	875
224	20	0	32	7	10355
225	12	1	32	0	5738
226	0	0	30	0	2760
227	8	1	31	0	2910
228	8	0	32	0	2188
229	0	1	32	0	0
230	0	0	32	0	1588
231	2	0	29	0	6705
232	0	0	32	0	0
233	0	0	32	0	0
234	24	0	27	2	15995
235	5	0	30	2	6885
236	1	0	32	0	3638
237	2	0	31	0	1585
238	5	0	32	0	2700

239	3	0	23	0	1235	3
240	3	0	22	0	1488	3
241	10	0	23	0	620	2
242	0	0	22	0	0	
243	9	0	21	0	1145	3
244	1	0	21	0	3145	5
245	1	0	14	0	1615	2
246	15	0	13	0	6313	13

Now the reporting part here,

```
% Count the number of customers in each cluster
[cluster_ids, ~, cluster_counts] = unique(idx);
cluster_counts = accumarray(cluster_counts, 1);
disp(cluster_counts);
```

```
        2302
         848
```

```
% Calculate mean and standard deviation for each cluster
data = clustered_data(:, 2:end-1);
data = table2array(data);
cluster = clustered_data.Cluster;

% Calculate descriptive statistics for each cluster
[mean_Data, std_Data] = grpstats(data, cluster, {'mean', 'std'});

% Display the statistics
for i = 1:size(mean_Data, 1)
    fprintf('Cluster %d:\n', i);
    for j = 1:size(data, 2)
        fprintf('Variable %d:\n', j);
        fprintf('Mean: %.2f\n', mean_Data(i, j));
        fprintf('Standard Deviation: %.2f\n\n', std_Data(i, j));
    end
end
```

```
Cluster 1:
Variable 1:
Mean: 5.51
Standard Deviation: 5.60
Variable 2:
Mean: 0.09
Standard Deviation: 0.29
Variable 3:
Mean: 32.07
Standard Deviation: 8.61
Variable 4:
Mean: 0.50
Standard Deviation: 0.87
Variable 5:
Mean: 2411.23
Standard Deviation: 1769.13
Variable 6:
Mean: 41.64
Standard Deviation: 27.80
```

```
Variable 7:
Mean: 60.57
Standard Deviation: 108.78
Variable 8:
Mean: 16.81
Standard Deviation: 13.04
Variable 9:
Mean: 2.83
Standard Deviation: 0.85
Variable 10:
Mean: 1.03
Standard Deviation: 0.18
Variable 11:
Mean: 1.34
Standard Deviation: 0.47
Variable 12:
Mean: 0.21
Standard Deviation: 0.41
Variable 13:
Mean: 98.08
Standard Deviation: 79.15
Cluster 2:
Variable 1:
Mean: 13.38
Standard Deviation: 8.10
Variable 2:
Mean: 0.03
Standard Deviation: 0.17
Variable 3:
Mean: 33.83
Standard Deviation: 8.35
Variable 4:
Mean: 2.14
Standard Deviation: 2.14
Variable 5:
Mean: 10067.92
Standard Deviation: 3757.83
Variable 6:
Mean: 144.99
Standard Deviation: 48.38
Variable 7:
Mean: 107.40
Standard Deviation: 114.38
Variable 8:
Mean: 41.70
Standard Deviation: 13.67
Variable 9:
Mean: 2.82
Standard Deviation: 0.99
Variable 10:
Mean: 1.20
Standard Deviation: 0.40
Variable 11:
Mean: 1.00
Standard Deviation: 0.00
Variable 12:
Mean: 0.01
Standard Deviation: 0.10
Variable 13:
Mean: 368.76
Standard Deviation: 200.41
```

```
figure(12)
```

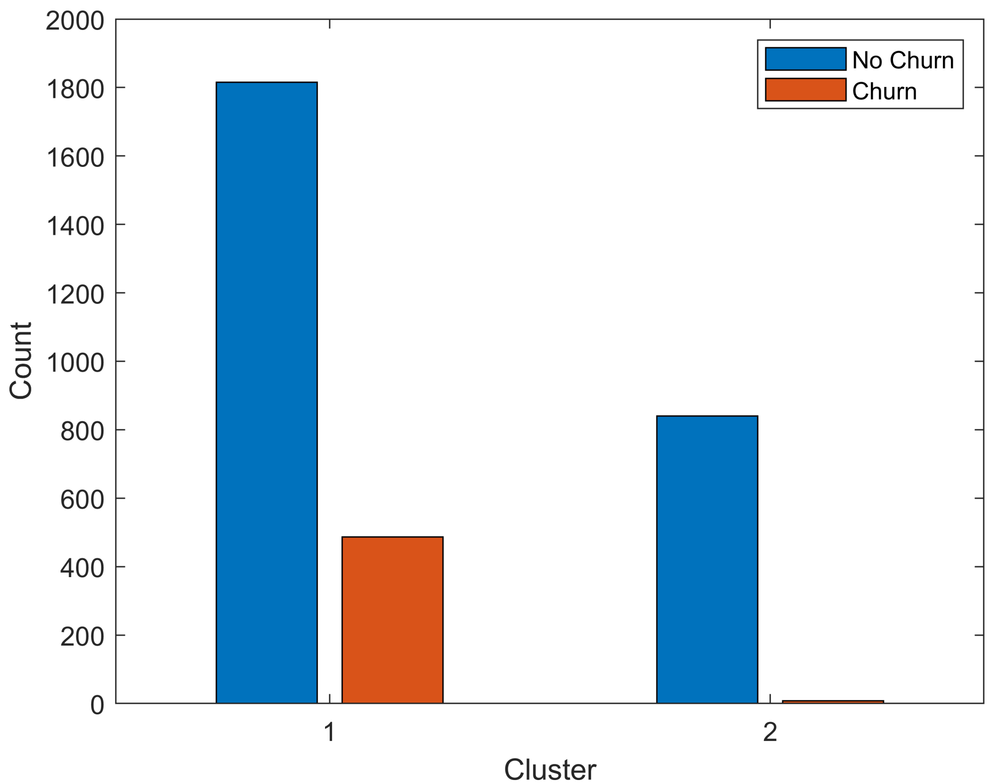
```
churn = clustered_data.Churn;
cluster = clustered_data.Cluster;
% Count occurrences of churn within each cluster
unique_Clusters = unique(cluster);
```

```
unique_Clusters = 2×1
     1
     2
```

```
unique_Churn = unique(churn);
```

```
unique_Churn = 2×1
     0
     1
```

```
churnCounts = zeros(numel(unique_Clusters), numel(unique_Churn));

for i = 1:numel(unique_Clusters)
    for j = 1:numel(unique_Churn)
        churnCounts(i, j) = sum(cluster == unique_Clusters(i) & churn == unique_Churn(j));
    end
end
% Bar plot
bar(churnCounts, 'grouped');
xlabel('Cluster');
ylabel('Count');
legend("No Churn", "Churn")
```



Step 3: Model Development

- Normalizing the dataset
- Splitting data into Training, Testing and Validation data
- Creating Lasso Logistic Regression and SVM and Classification tree.
- Evaluation of the classification model.

```
% Set the random number generator to its default state
rng('default');

% Creating Vectors for target Variable
Y = churn_data.Churn;
X = churn_data{:,{'AgeGroup', 'CallFailure', 'ChargeAmount', 'Complains',...
    'CustomerValue', 'DistinctCalledNumbers', 'FrequencyOfSMS', 'FrequencyOfUse',...
    'SecondsOfUse', 'SubscriptionLength', 'Status', 'TariffPlan'}};

% Normalize the features using z-score normalization
X_norm = zscore(X);

% Splitting the dataset in Training and Testing Data
n = length(Y);
c = cvpartition(n,"holdout",0.3);
Xtrain = X_norm(training(c),:);
Xtest = X_norm(test(c),:);
Ytrain = Y(training(c),:);
Ytest = Y(test(c),:);

% Splitting the training data into training and validation data using cross validation
n2 = length(Ytrain);
cv2 = cvpartition(n2,'KFold', 4);
```

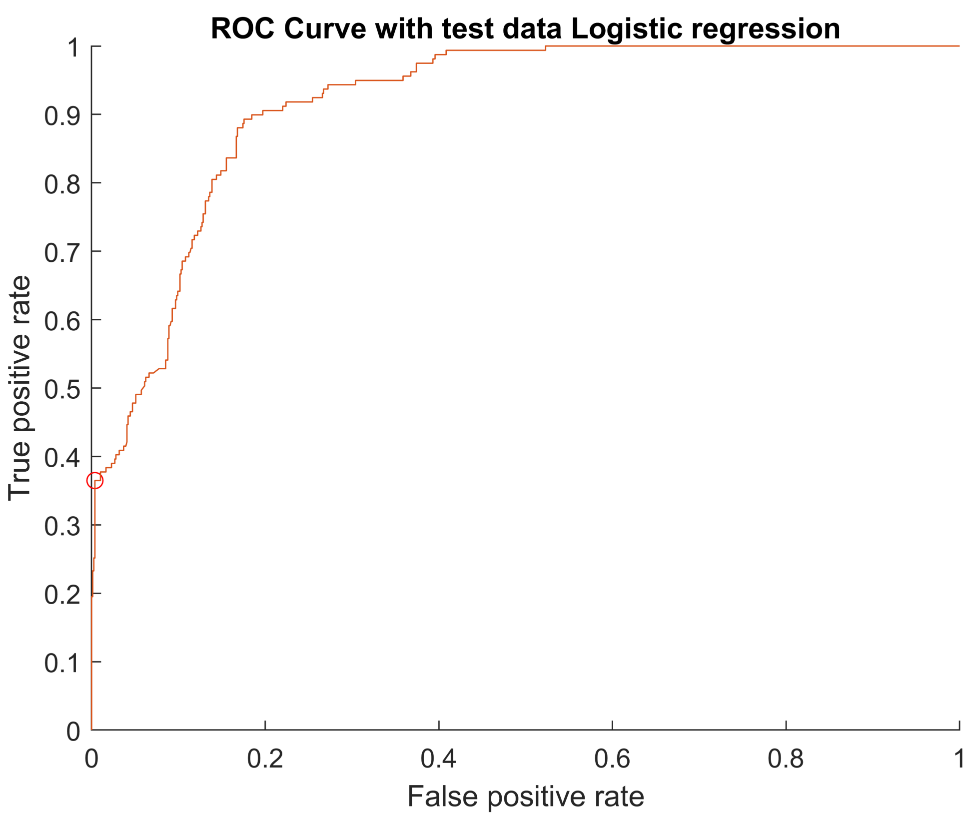
```
% Logistic regression Model
glm1 = fitglm(Xtrain,Ytrain)
```

```
glm1 = 
Generalized linear regression model:
    y ~ 1 + x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10 + x11 + x12
    Distribution = Normal

Estimated Coefficients:
```

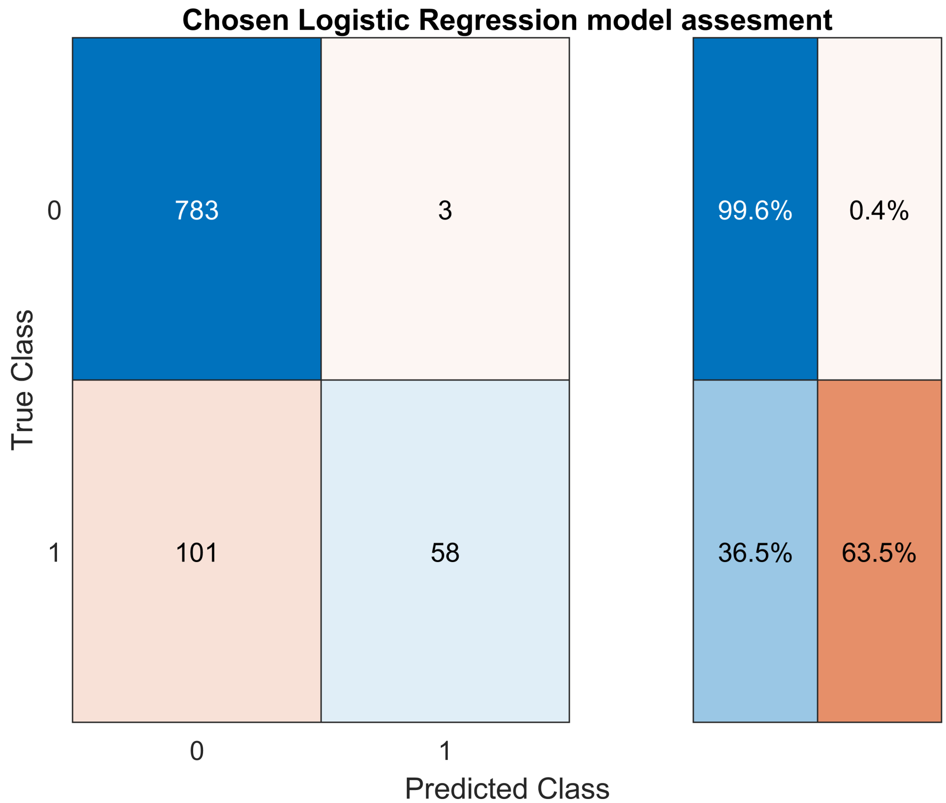
	Estimate	SE	tStat	pValue
(Intercept)	0.15405	0.0056833	27.106	8.6573e-140
x1	-0.011495	0.0088655	-1.2966	0.19492
x2	0.048868	0.0096141	5.083	4.0302e-07
x3	-0.025992	0.010177	-2.5541	0.010713
x4	0.14898	0.0060053	24.809	5.8568e-120
x5	0.013646	0.018828	0.72475	0.46868
x6	-0.023194	0.0088612	-2.6175	0.0089189

```
    x7              -0.022262    0.0073993    -3.0086      0.0026543
    x8              -0.097201     0.024355     -3.991     6.7966e-05
    x9               0.046727     0.027728     1.6852       0.092096
    x10             -0.022814     0.006141    -3.7151     0.00020824
    x11               0.10279     0.007753     13.258     1.1985e-38
    x12            -0.0073154    0.0069583    -1.0513        0.29322


2205 observations, 2192 error degrees of freedom
Estimated Dispersion: 0.071
F-statistic vs. constant model: 152, p-value = 8.14e-277
```

```
ptest1 =predict(glm1,Xtest);
[xxLR,yyLR,Tresholds_LR,auctest_LR, OPTROCPT1] = perfcurve(Ytest,ptest1,'1');
figure;
hold on
plot(OPTROCPT1(1),OPTROCPT1(2),'ro')
plot(xxLR,yyLR)
xlabel('False positive rate'); ylabel('True positive rate');
title('ROC Curve with test data Logistic regression')
hold off
```




```
ThresholdForLR = Tresholds_LR((xxLR==OPTROCPT1(1))&(yyLR==OPTROCPT1(2)));
ptest1Binom = double(ptest1>=ThresholdForLR);

% Logistic Regression
confusion_matrix_LR = confusionchart(Ytest, ptest1Binom,'RowSummary','row-normalized');
title("Chosen Logistic Regression model assesment")
```




```
TN_LR = 783; FP_LR = 3; TP_LR = 58; FN_LR = 101; Total = 945;
```

```
%Compute accuracy
accuracy_LR= (TP_LR + TN_LR)/Total
```

```
accuracy_LR = 0.8899
```

```
% Compute precision, recall, and F1 score
precision_LR = TP_LR / (TP_LR + FP_LR)
```

```
precision_LR = 0.9508
```

```
recall_LR = TP_LR / (TP_LR + FN_LR)
```

```
recall_LR = 0.3648
```

```
F1score_LR = 2 * precision_LR * recall_LR / (precision_LR + recall_LR)
```

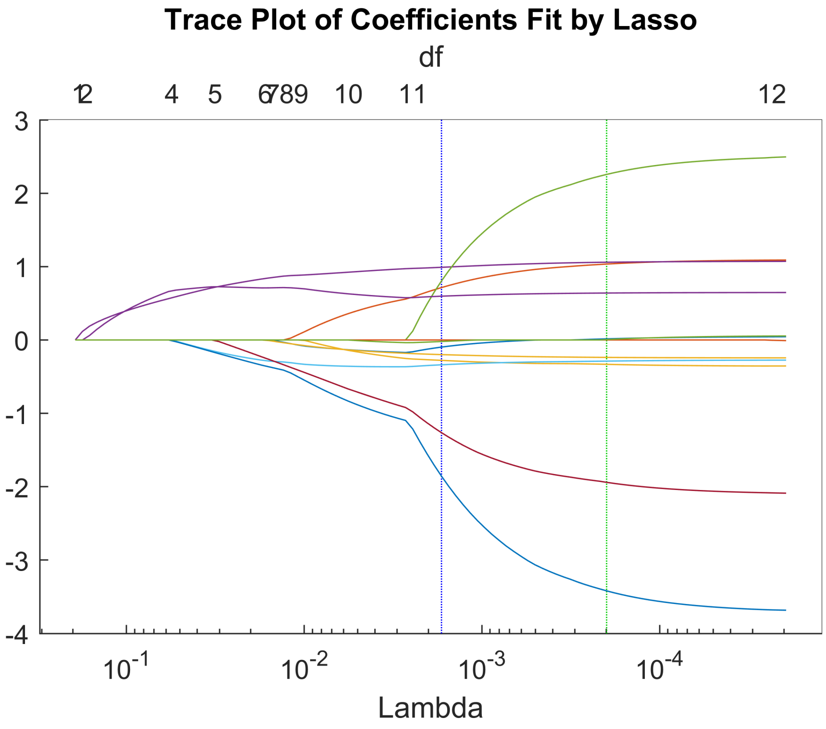
```
F1score_LR = 0.5273
```

```
% Training lasso logistic regression models
[B4,FitInfo4] = lassoglm(Xtrain,Ytrain,'binomial','CV',cv2);
```

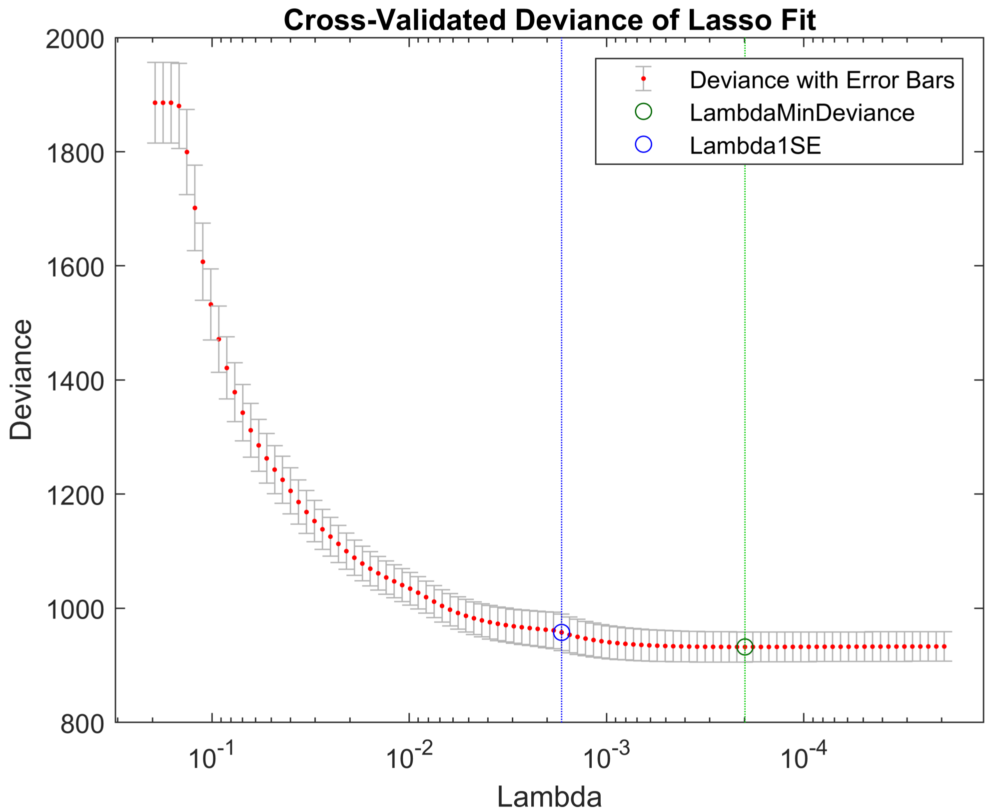
```
% Number of Features
idxIndex1SE = FitInfo4.Index1SE;
Number_of_predictors_lamnda1SE = FitInfo4.DF(idxIndex1SE);
idxIndexMinDev = FitInfo4.IndexMinDeviance;
Number_of_predictors_lamndaMinDev = FitInfo4.DF(idxIndexMinDev);
fprintf("If you use LambdaMinDev, we will have %d features"...
    + newline + "If you use  Lambda1SE, we will have %d features", Number_of_predictors_lamndaM
```

```
If you use LambdaMinDev, we will have 11 features
If you use  Lambda1SE, we will have 11 features
```

```
% Plotting the lasso plots
lassoPlot(B4,FitInfo4,'PlotType','Lambda','XScale','log');
```




```
lassoPlot(B4,FitInfo4,'plottype','CV');
legend('show')
```




```
% Retraining the Model
optimal_lambda = FitInfo4.Lambda1SE;
[Bfinal_Lamnda1SE,fitInfofinal_Lamnda1SE] = lassoglm(Xtrain,Ytrain,'binomial','Lambda',optimal_
B0_final1SE= fitInfofinal_Lamnda1SE.Intercept;
coef_final_Lamnda1SE = [B0_final1SE; Bfinal_Lamnda1SE];
test_predictions_lasso_Lamnda1SE = glmval(coef_final_Lamnda1SE,Xtest,'logit');
```

```
% Training SVM model
svm1 = fitcsvm(Xtrain,Ytrain,'KernelFunction','RBF','KernelScale','auto', 'CVPartition', cv2);
svm2 = fitcsvm(Xtrain,Ytrain,'KernelFunction','polynomial', 'CVPartition', cv2);
svm3 = fitcsvm(Xtrain,Ytrain,'KernelFunction','linear', 'CVPartition', cv2);
% Accuracy of models
cross_validation_error_svm1 = kfoldLoss(svm1);
cross_validation_accuracy_svm1 = 1 - cross_validation_error_svm1
```

```
cross_validation_accuracy_svm1 = 0.9451
```

```
cross_validation_error_svm2 = kfoldLoss(svm2);
cross_validation_accuracy_svm2 = 1- cross_validation_error_svm2
```

```
cross_validation_accuracy_svm2 = 0.9587
```

```
cross_validation_error_svm3 = kfoldLoss(svm3);
cross_validation_accuracy_svm3 = 1- cross_validation_error_svm3
```

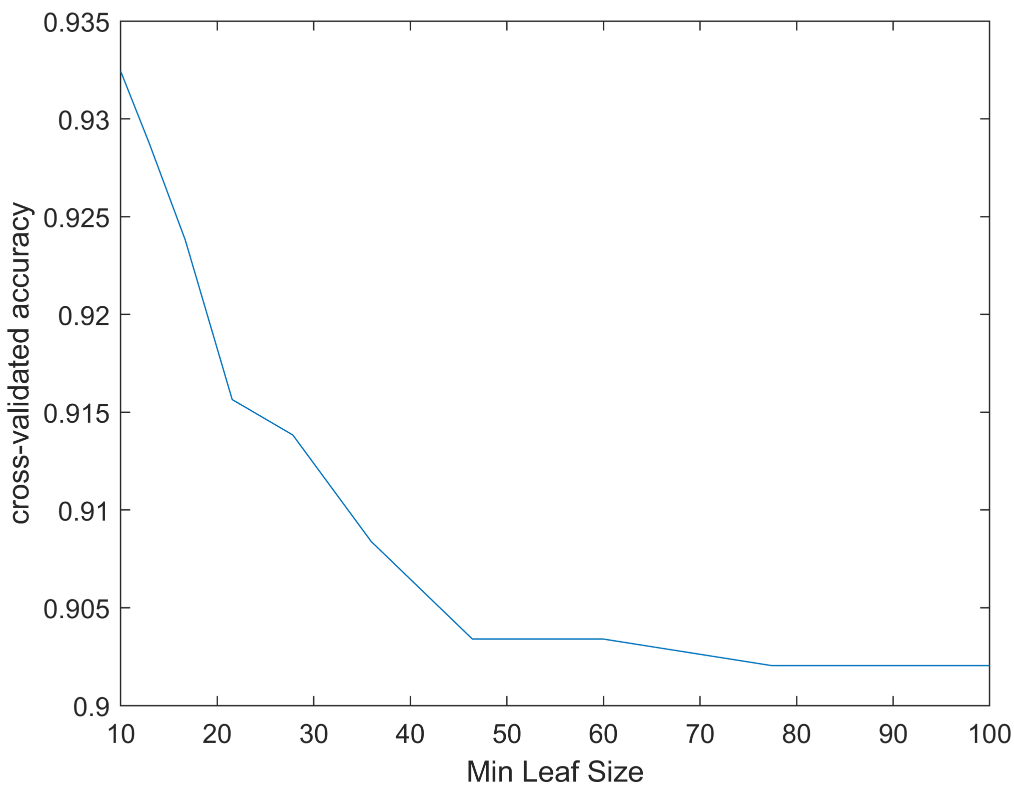
```
cross_validation_accuracy_svm3 = 0.8989
```

```
% Final SVM model
disp("The cross-validation accuracy in model 2 is higher"...
    + newline + "so the final model is created using "...
    + newline + "Polynomial Kernel Function")
```

```
The cross-validation accuracy in model 2 is higher
so the final model is created using
Polynomial Kernel Function
```

```
svm_final = fitcsvm(Xtrain,Ytrain,'KernelFunction','Polynomial');
```

```
% Training Classification tree model
leafs = logspace(1,2,10);
N = numel(leafs);
accuracy_mdl = zeros(N,1);
for n=1:N
    t = fitctree(Xtrain, Ytrain,'CVPartition', cv2, 'MinLeafSize',leafs(n));
    accuracy_mdl(n) = 1 - kfoldLoss(t);
end
plot(leafs,accuracy_mdl);
xlabel('Min Leaf Size');
ylabel('cross-validated accuracy');
```



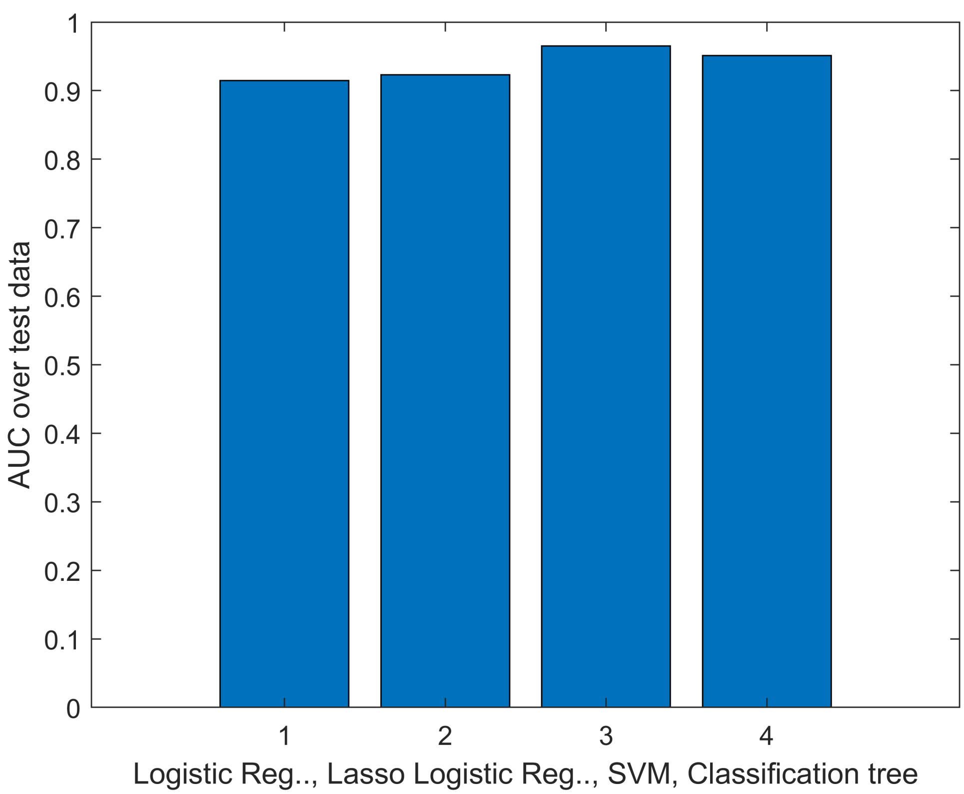

```
[max_acc, idx] = max(accuracy_mdl);
```

```
optimal_min_leaf_size = leafs(idx);
fprintf('Optimal minimum leaf size = %f, with cross-validated accuracy = %f\n', optimal_min_lea
```

```
Optimal minimum leaf size = 10.000000, with cross-validated accuracy = 0.932426
```

```
final_model_tree =fitctree(Xtrain, Ytrain,'MinLeafSize',optimal_min_leaf_size);
```

```
% AUC curve
% SVM model
ScoreSVMModel = fitPosterior(svm_final);
[predictions_svm,score_svm] = predict(ScoreSVMModel,Xtest);
[Xsvm,Ysvm,Tsvm,AUCsvm] = perfcurve(Ytest,score_svm(:,2),'1');

% Tree model
[predictions_tree,score_tree] = predict(final_model_tree,Xtest);
[Xtree,Ytree,Tree,AUCtree] = perfcurve(Ytest,score_tree(:,2),'1');

% Lasso logistic regression
[xxtest_lasso_Lamnda1SE,yytest_lasso_Lamnda1SE,Tresholds_lasso_Lamnda1SE,auctest_lasso_Lamnda1S

% AUC for all model
auctest_all=[auctest_LR; auctest_lasso_Lamnda1SE; AUCsvm; AUCtree];
bar (auctest_all)
xlabel('Logistic Reg.., Lasso Logistic Reg.., SVM, Classification tree'); ylabel('AUC over test
```

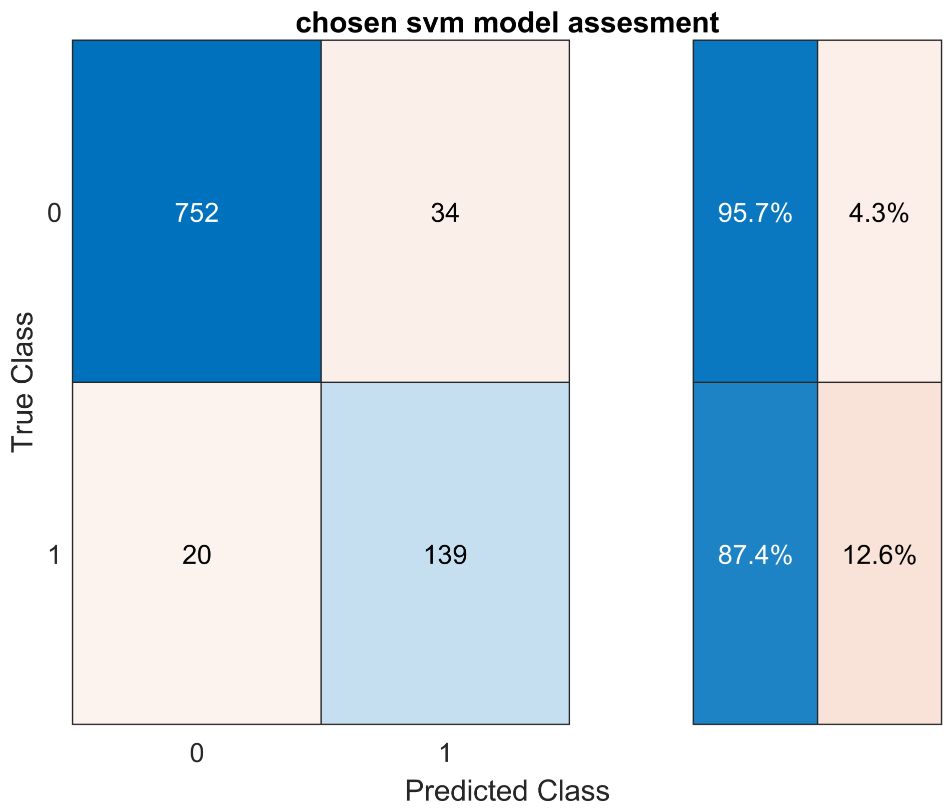



```
% Evaluation of model
```

```
fprintf("The AUC of Lasso Logistic Regression, SVM, Classification tree are %.4f, %.4f and %.4
    + newline + "Classification tree and SVM performs better than Lasso Logistic Regression.".
    + newline + "SVM performs slightly better than Classification tree.", auctest_all(1), aucte
```

```
The AUC of Lasso Logistic Regression, SVM, Classification tree are 0.9145, 0.9230 and 0.9652 respectively
Classification tree and SVM performs better than Lasso Logistic Regression.
SVM performs slightly better than Classification tree.
```

```
% Confusion matrix and F1 score
% SVM
predicitions_over_test_data_svm = predict(svm_final,Xtest);
confusion_matrix_svm = confusionchart(Ytest,predicitions_over_test_data_svm,'RowSummary','row-r
title("chosen svm model assesment")
```




```
TN_svm = 752; FP_svm = 34; TP_svm = 139; FN_svm = 20; Total = 945;
accuracy_svm = (TP_svm + TN_svm)/Total
```

```
accuracy_svm = 0.9429
```

```
precision_svm = TP_svm/ (TP_svm + FP_svm)
```

```
precision_svm = 0.8035
```

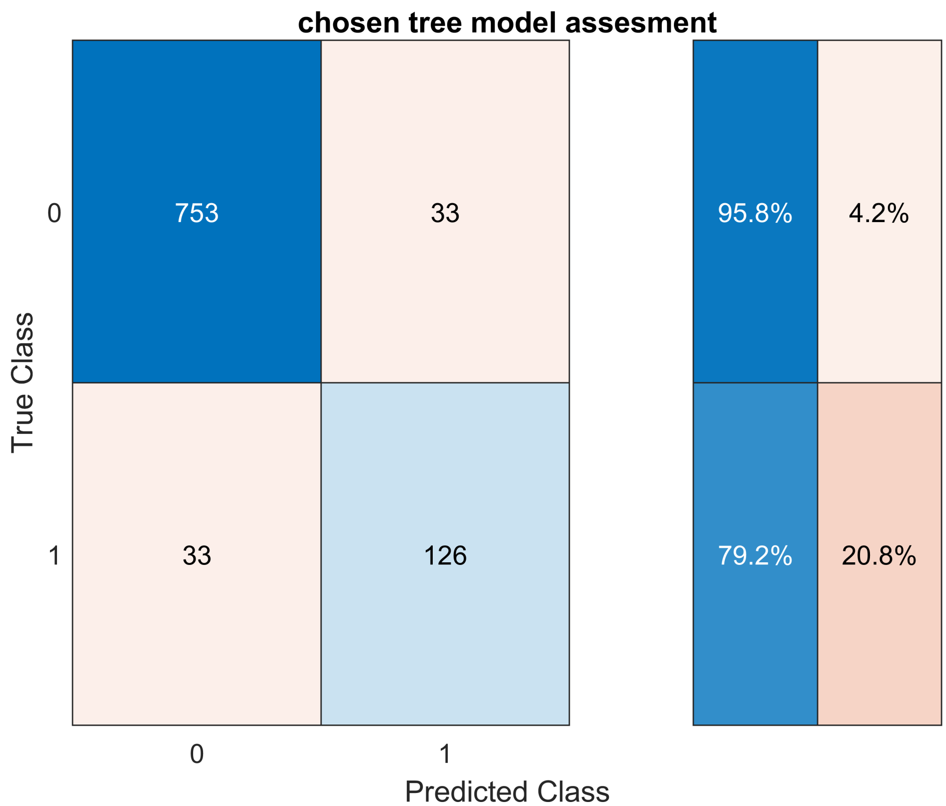
```
recall_svm = TP_svm / (TP_svm + FN_svm)
```

```
recall_svm = 0.8742
```

```
F1score_svm = 2 * precision_svm * recall_svm / (precision_svm + recall_svm)
```

```
F1score_svm = 0.8373
```

```
% Tree classification
predicitions_over_test_data_tree= predict(final_model_tree,Xtest);
confusion_matrix_tree = confusionchart(Ytest, predicitions_over_test_data_tree,'RowSummary','ro
title("chosen tree model assesment")
```




```
TN_tree = 753; FP_tree = 33; TP_tree = 126; FN_tree = 33; Total = 945;

% Compute accuracy
accuracy_tree= (TP_tree + TN_tree)/Total
```

```
accuracy_tree = 0.9302
```

```

% Compute precision, recall, and F1 score
precision_tree = TP_tree / (TP_tree + FP_tree)
```

```
precision_tree = 0.7925
```

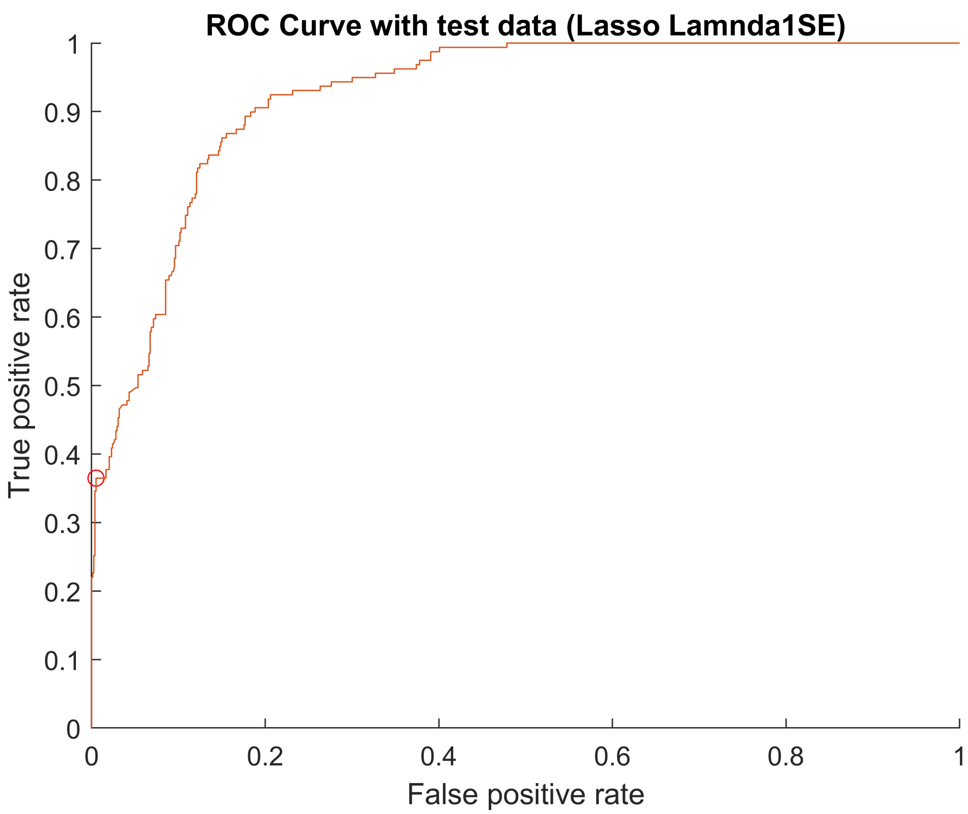
```
recall_tree = TP_tree / (TP_tree + FN_tree)
```

```
recall_tree = 0.7925
```

```
F1score_tree = 2 * precision_tree * recall_tree / (precision_tree + recall_tree)
```

```
F1score_tree = 0.7925
```

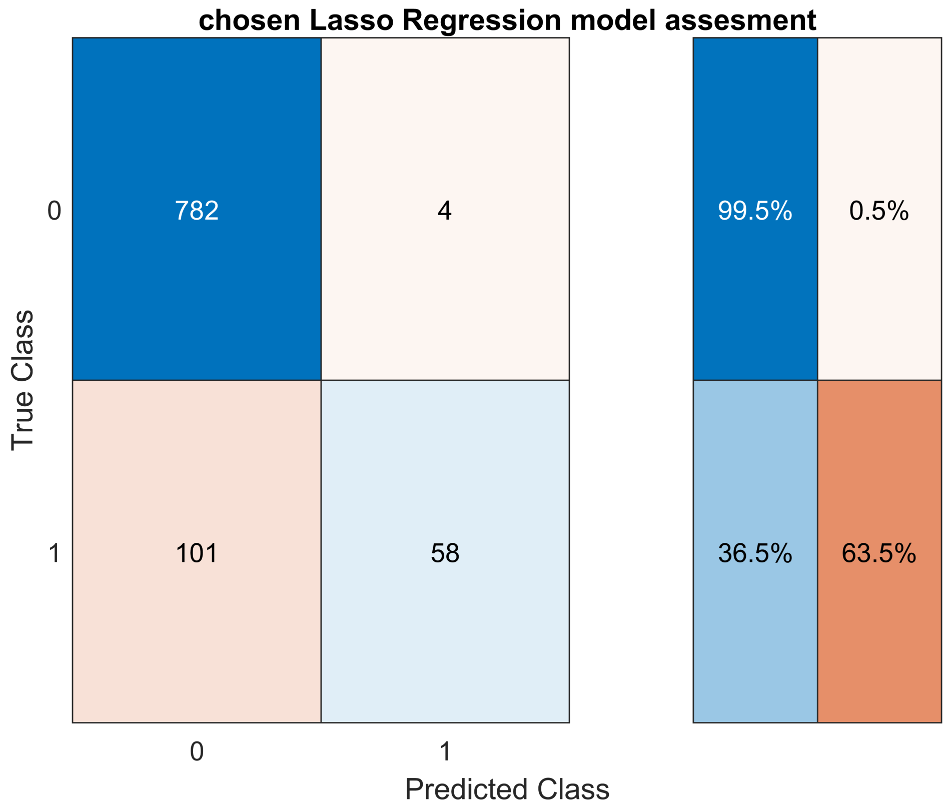
```
% Lasso Logistic regression
figure;
hold on
plot(OPTROCPT_lasso_Lamnda1SE(1),OPTROCPT_lasso_Lamnda1SE(2),'ro')
plot(xxtest_lasso_Lamnda1SE,yytest_lasso_Lamnda1SE)
xlabel('False positive rate'); ylabel('True positive rate');
title('ROC Curve with test data (Lasso Lamnda1SE)')
hold off
```




```
Threshold_lasso_1SE =Tresholds_lasso_Lamnda1SE((xxtest_lasso_Lamnda1SE==OPTROCPT_lasso_Lamnda1S
```

```
Threshold_lasso_1SE = 0.6050
```

```
predictions_lasso_regression = double(test_predictions_lasso_Lamnda1SE >= Threshold_lasso_1SE);

confusion_matrix_lasso = confusionchart(Ytest,predictions_lasso_regression ,'RowSummary','row-r
title("chosen Lasso Regression model assesment")
```




```
TN_lasso = 782; FP_lasso = 4; TP_lasso = 58; FN_lasso = 101; Total = 945;

%Compute accuracy
accuracy_lasso= (TP_lasso + TN_lasso)/Total
```

```
accuracy_lasso = 0.8889
```

```

% Compute precision, recall, and F1 score
precision_lasso = TP_lasso / (TP_lasso + FP_lasso)
```

```
precision_lasso = 0.9355
```

```
recall_lasso = TP_lasso / (TP_lasso + FN_lasso)
```

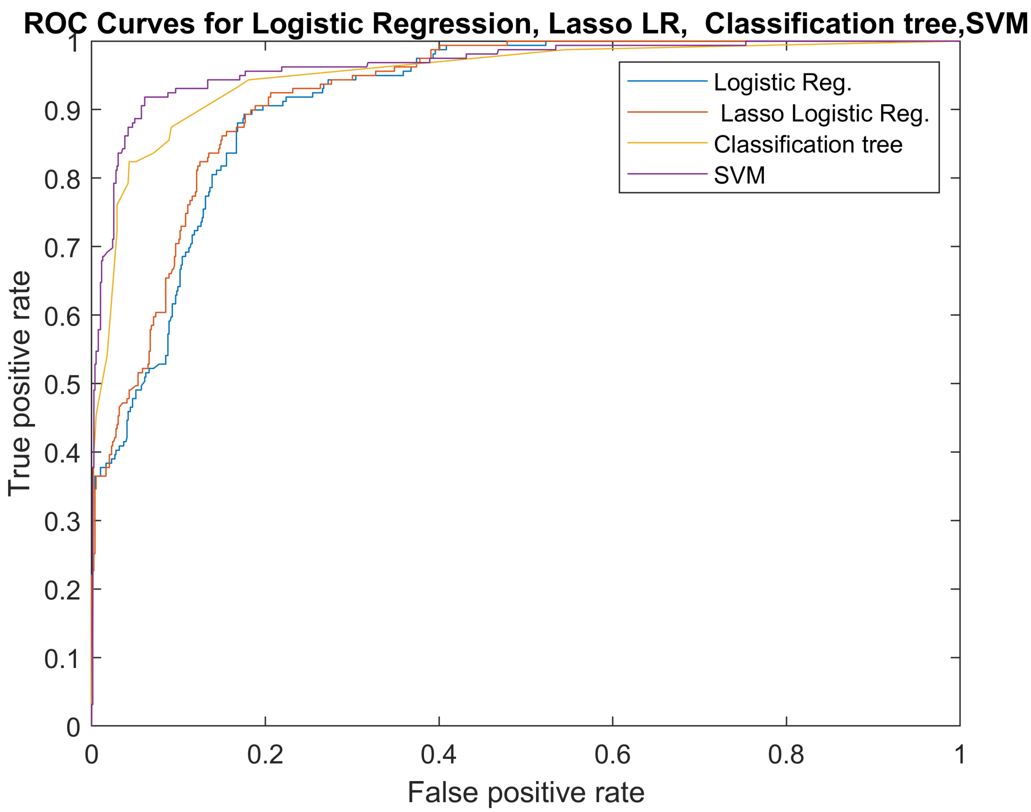
```
recall_lasso = 0.3648
```

```
F1score_lasso = 2 * precision_lasso * recall_lasso / (precision_lasso + recall_lasso)
```

```
F1score_lasso = 0.5249
```

```
% Visual Evaluation through ROC Plot
plot(xxLR, yyLR)
hold on
plot(xxtest_lasso_Lamnda1SE,yytest_lasso_Lamnda1SE)
hold on
```

```
plot(Xtree,Ytree)
hold on
plot(Xsvm,Ysvm)
legend('Logistic Reg.',' Lasso Logistic Reg.','Classification tree', 'SVM')
xlabel('False positive rate'); ylabel('True positive rate');
title('ROC Curves for Logistic Regression, Lasso LR,  Classification tree,SVM')
hold off
```




```
% Evaluation of Models based on F1 score
disp("Based on the F1-scores you provided, the SVM model has the best performance over the test
    + newline + "followed by the Classification Tree and Lasso Logistic Regression model with L
```

```
Based on the F1-scores you provided, the SVM model has the best performance over the test data,
followed by the Classification Tree and Lasso Logistic Regression model with Lambda1SE model respectively.
```