PROJECT 4

Telecom Customer Churn Prediction Assessment

Customer Churn is a burning problem for Telecom companies. In this project, we simulate one such case of customer churn where we work on a data of post-paid customers with a contract. The data has information about the customer usage behaviour, contract details and the payment details. The data also indicates which were the customers who cancelled their service. Based on this past data, we need to build a model which can predict whether a customer will cancel their service in the future or not.

You are expected to do the following :

1. **EDA (16 Marks)**
   - How does the data looks like, Univariate and bivariate analysis. Plots and charts which illustrate the relationships between variables (4 Marks)
   - Look out for outliers and missing values (4 Marks)
   - Check for multicollinearity & treat it (4 Marks)
   - Summarize the insights you get from EDA (4 Marks)
2. **Build Models and compare them to get to the best one (39 Marks)**
   - Logistic Regression (8 Marks)
   - KNN (8 Marks)
   - Naive Bayes (8 Marks) (is it applicable here? comment and if it is not applicable, how can you build an NB model in this case?)
   - Model Comparison using Model Performance metrics & Interpretation (15 Marks)
3. **Actionable Insights (5 marks)**
   - Interpretation & Recommendations from the best model

- 1. How does the data looks like, Univariate and bivariate analysis. Plots and charts which illustrate the relationships between variables (4 Marks)
- Look out for outliers and missing values (4 Marks)
- Check for multicollinearity & treat it (4 Marks)
- Summarize the insights you get from EDA (4 Marks)

Univariate Analysis Mean Median and Mode

R code:

View(cp_project)

names(cp_project)

dim(cp_project)

str(cp_project)

class(cp_project)

head(cp_project, n = 5)

tail(cp_project, n = 5)

summary(cp_project)

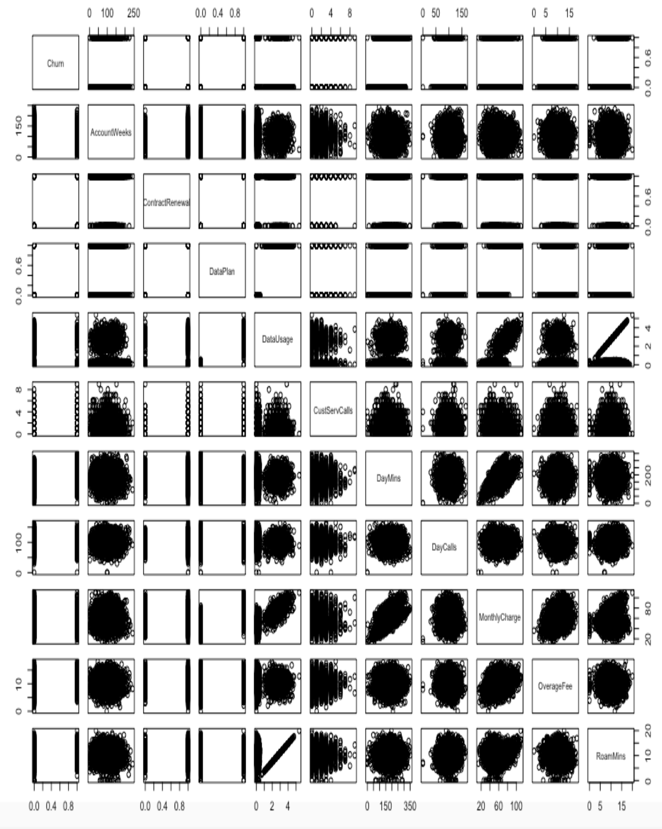
```
View(cp_project)
> names(cp_project)
 [1] "Churn"          "AccountWeeks"    "ContractRenewal" "DataPlan"
 [5] "DataUsage"       "CustServCalls"   "DayMins"        "DayCalls"
 [9] "MonthlyCharge"   "OverageFee"      "RoamMins"
> dim(cp_project)
[1] 3333   11
> str(cp_project)
Classes 'tbl_df', 'tbl' and 'data.frame': 3333 obs. of  11 variables:
 $ Churn          : num  0 0 0 0 0 0 0 0 0 0 ...
 $ AccountWeeks   : num  128 107 137 84 75 118 121 147 117 141 ...
 $ ContractRenewal: num  1 1 1 0 0 0 1 0 1 0 ...
 $ DataPlan       : num  1 1 0 0 0 0 1 0 0 1 ...
 $ DataUsage      : num  2.7 3.7 0 0 0 0 2.03 0 0.19 3.02 ...
```

```
 $ CustServCalls  : num  1 1 0 2 3 0 3 0 1 0 ...
 $ DayMins        : num  265 162 243 299 167 ...
 $ DayCalls       : num  110 123 114 71 113 98 88 79 97 84 ...
 $ MonthlyCharge  : num  89 82 52 57 41 57 87.3 36 63.9 93.2 ...
 $ OverageFee     : num  9.87 9.78 6.06 3.1 7.42 ...
 $ RoamMins       : num  10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...
> class(cp_project)
[1] "tbl_df"     "tbl"        "data.frame"
> head(cp_project, n = 5)
# A tibble: 5 x 11
  Churn AccountWeeks ContractRenewal DataPlan DataUsage CustServCalls
DayMins
  <dbl>        <dbl>           <dbl>    <dbl>     <dbl>         <dbl>   <dbl>
1     0          128               1        1       2.7             1    265.
2     0          107               1        1       3.7             1    162.
3     0          137               1        0       0               0    243.
4     0           84               0        0       0               2    299.
5     0           75               0        0       0               3    167.
# … with 4 more variables: DayCalls <dbl>, MonthlyCharge <dbl>,
#   OverageFee <dbl>, RoamMins <dbl>
> tail(cp_project, n = 5)
# A tibble: 5 x 11
  Churn AccountWeeks ContractRenewal DataPlan DataUsage CustServCalls
DayMins
  <dbl>        <dbl>           <dbl>    <dbl>     <dbl>         <dbl>   <dbl>
1     0          192               1        1      2.67             2    156.
2     0           68               1        0      0.34             3    231.
3     0           28               1        0      0                2    181.
4     0          184               0        0      0                2    214.
5     0           74               1        1      3.7              0    234.
# … with 4 more variables: DayCalls <dbl>, MonthlyCharge <dbl>,
#   OverageFee <dbl>, RoamMins <dbl>
> summary(cp_project)
     Churn         AccountWeeks   ContractRenewal     DataPlan
 Min.   :0.0000   Min.   :  1.0   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.0000   1st Qu.: 74.0   1st Qu.:1.0000   1st Qu.:0.0000
 Median :0.0000   Median :101.0   Median :1.0000   Median :0.0000
 Mean   :0.1449   Mean   :101.1   Mean   :0.9031   Mean   :0.2766
 3rd Qu.:0.0000   3rd Qu.:127.0   3rd Qu.:1.0000   3rd Qu.:1.0000
 Max.   :1.0000   Max.   :243.0   Max.   :1.0000   Max.   :1.0000
```

```
  DataUsage      CustServCalls      DayMins         DayCalls
 Min.   :0.0000   Min.   :0.000   Min.   :  0.0   Min.   :  0.0
 1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:143.7   1st Qu.: 87.0
 Median :0.0000   Median :1.000   Median :179.4   Median :101.0
 Mean   :0.8165   Mean   :1.563   Mean   :179.8   Mean   :100.4
 3rd Qu.:1.7800   3rd Qu.:2.000   3rd Qu.:216.4   3rd Qu.:114.0
 Max.   :5.4000   Max.   :9.000   Max.   :350.8   Max.   :165.0
 MonthlyCharge      OverageFee       RoamMins
 Min.   : 14.00   Min.   : 0.00   Min.   : 0.00
 1st Qu.: 45.00   1st Qu.: 8.33   1st Qu.: 8.50
 Median : 53.50   Median :10.07   Median :10.30
 Mean   : 56.31   Mean   :10.05   Mean   :10.24
 3rd Qu.: 66.20   3rd Qu.:11.77   3rd Qu.:12.10
 Max.   :111.30   Max.   :18.19   Max.   :20.00
```

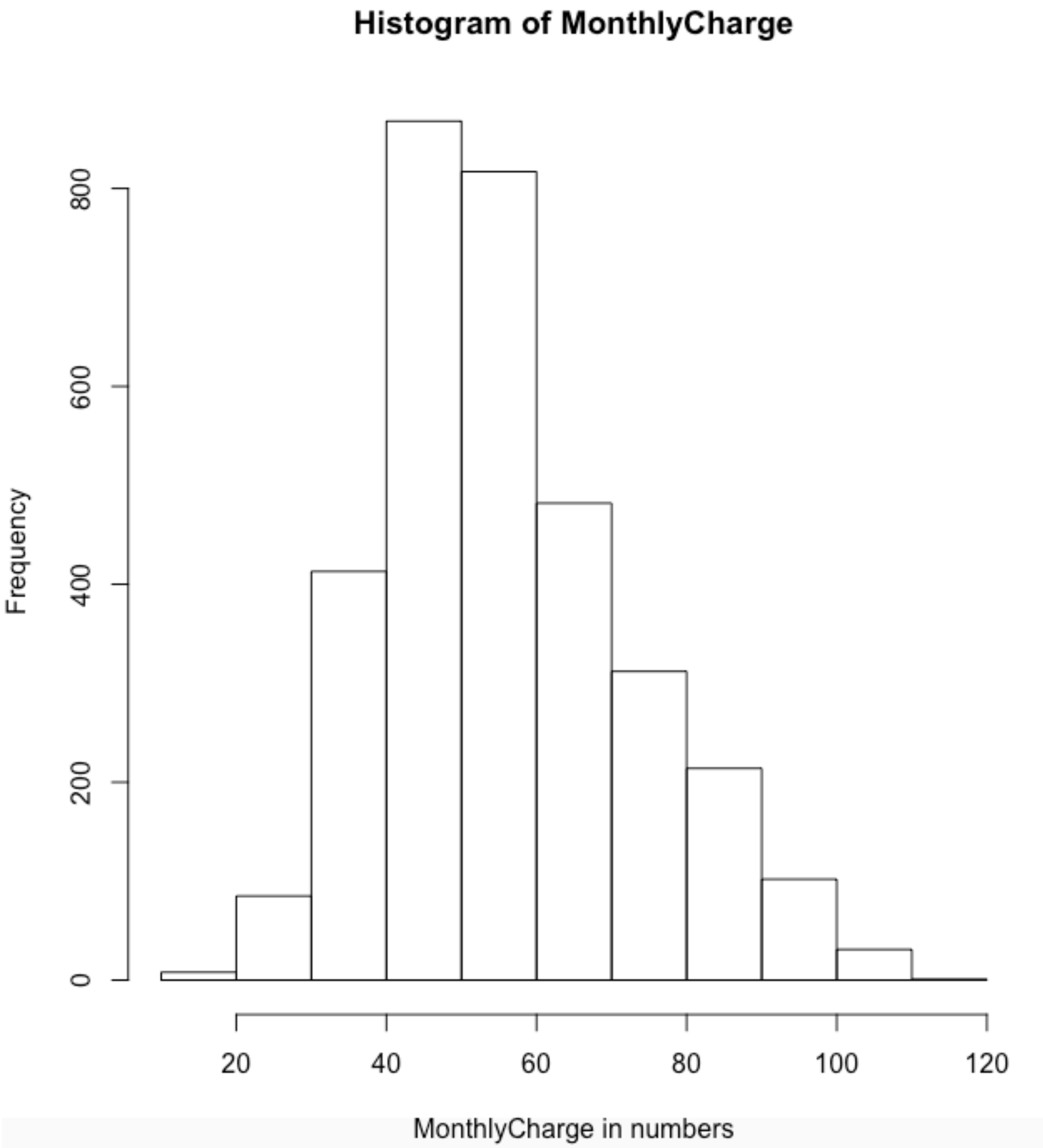
R code:

```
plot(cp_project$`DayMins (in seconds)`)
plot(cp_project)
```

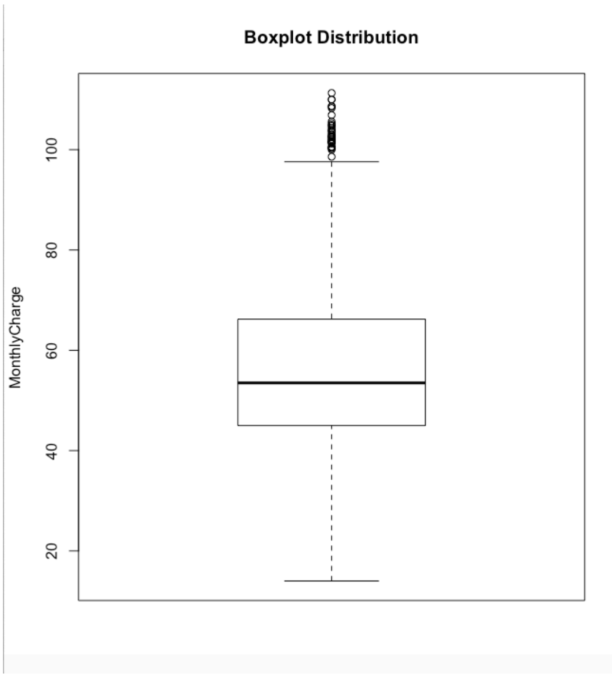


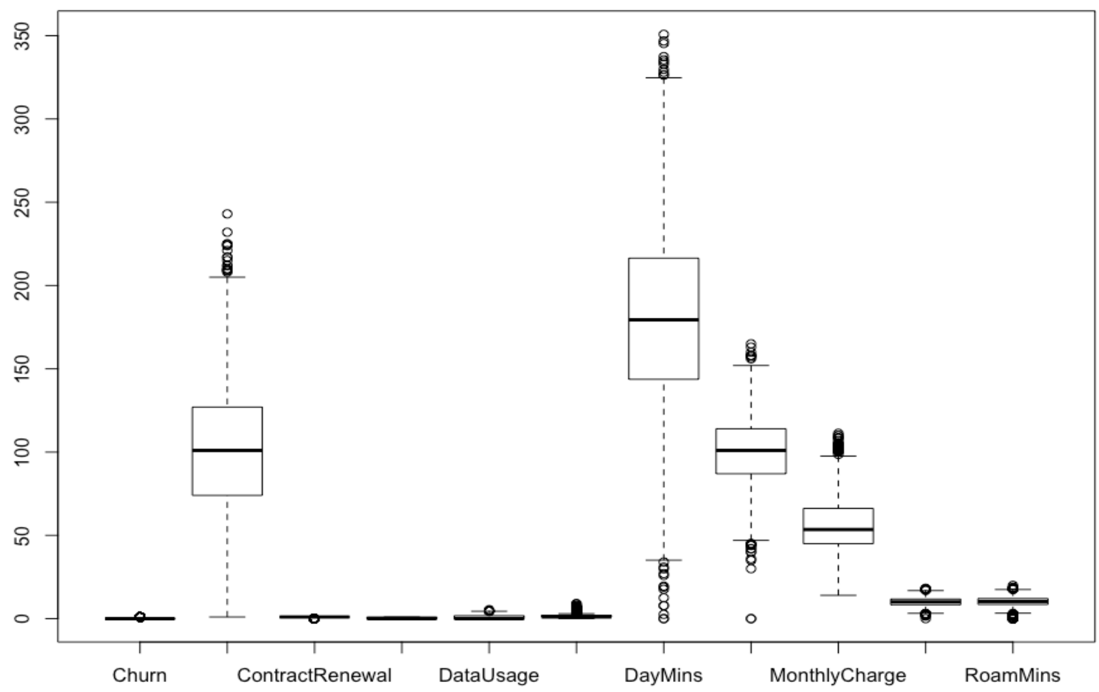
Histogram

```
hist(cp_project$MonthlyCharge,
    main = "Histogram of MonthlyCharge",
    xlab = "MonthlyCharge in numbers")
```



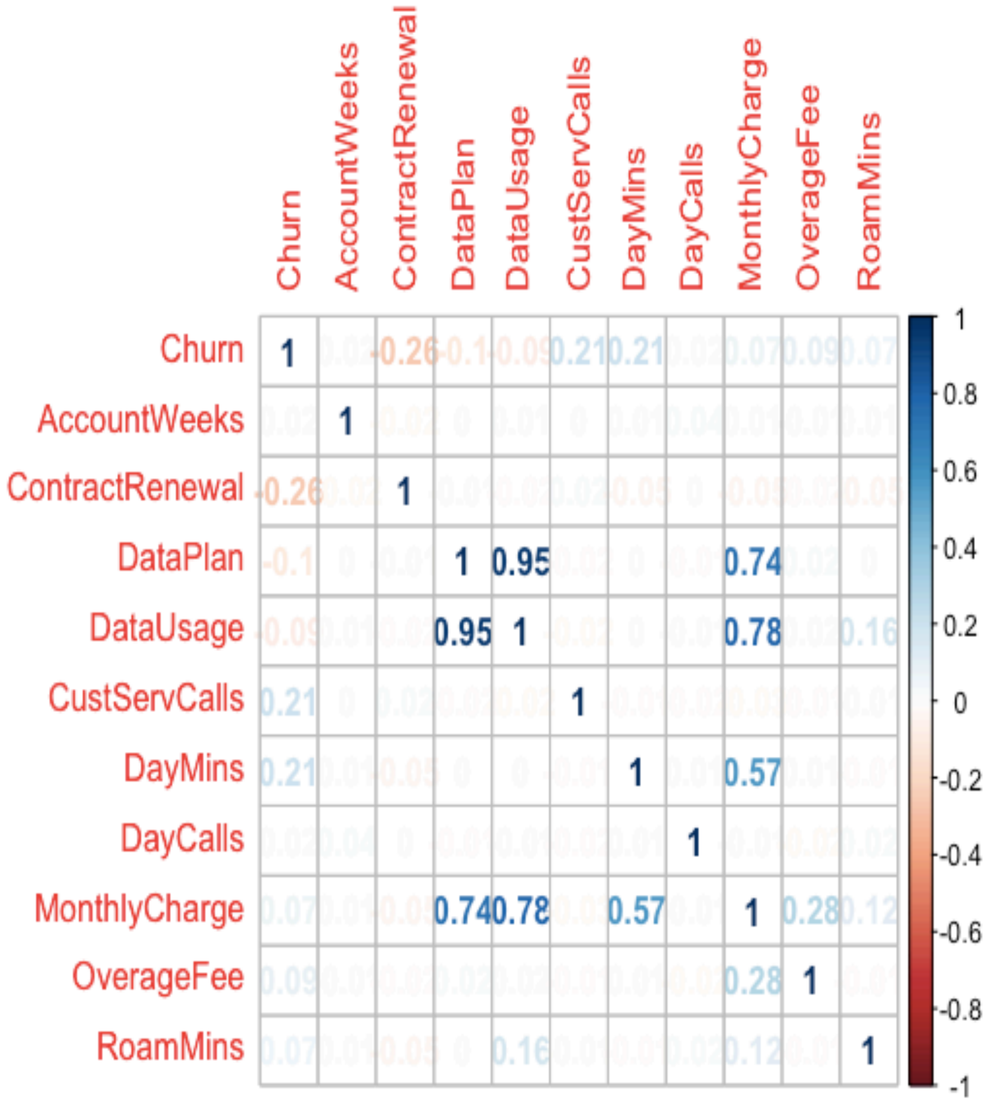
Boxplot:
boxplot(cp_project$`DayMins(in mins)`)
boxplot(cp_project$MonthlyCharge, ylab = 'MonthlyCharge', main ='Boxplot Distribution')
boxplot(cp_project)





Correlation:

```
numeric.var <- sapply(cp_project,is.numeric)
corr.matrix <- cor(cp_project[,numeric.var])
corrplot(corr.matrix, main="\n\nCorrelation Plot for Numerical Variables",
method = "number")
```



2. i) Logistic Regression

First Normal Regression:

R code:

```
Regression = lm(RoamMins~DayCalls+DayMins+OverageFee)
summary(Regression)
predict = predict(Regression)
data.frame(RoamMins, predict)
```

```
> Regression = lm(RoamMins~DayCalls+DayMins+OverageFee)
> summary(Regression)

Call:
lm(formula = RoamMins ~ DayCalls + DayMins + OverageFee)

Residuals:
    Min      1Q  Median      3Q     Max
-10.365  -1.745   0.045   1.846   9.837

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 10.1485134  0.3525142  28.789   <2e-16 ***
DayCalls     0.0029782  0.0024108   1.235    0.217
DayMins     -0.0005241  0.0008881  -0.590    0.555
OverageFee  -0.0115520  0.0190808  -0.605    0.545
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.792 on 3329 degrees of freedom
Multiple R-squared:  0.0006812,	Adjusted R-squared:  -0.0002194
F-statistic: 0.7564 on 3 and 3329 DF,  p-value: 0.5185

> predict = predict(Regression)
> data.frame(RoamMins, predict)
   RoamMins   predict
```

1	10.0	10.223155
2	13.7	10.317157
3	12.2	10.290454
4	6.6	10.167234
5	10.1	10.311965
6	6.3	10.195872
7	7.5	10.094882
8	7.1	10.241897
9	8.7	10.137616
10	11.2	10.134919
11	12.7	10.356825
12	9.1	10.333989
13	11.2	10.306267
14	12.3	10.185505
15	13.1	10.116289
16	5.4	9.990014
17	13.8	10.297242
18	8.1	10.262048
19	10.0	10.122737
20	13.0	10.206755
21	10.6	10.277165
22	5.7	10.282677
23	9.5	10.343994
24	7.7	10.318044
25	10.3	10.220506
26	15.5	10.149599
27	9.5	10.268933
28	14.7	10.205659
29	6.3	10.224420
30	11.1	10.310137
31	14.2	10.307982
32	10.3	10.226278
33	12.6	10.379742
34	11.8	10.223336
35	8.3	10.223169
36	14.7	10.145894
37	14.5	10.359128
38	10.0	10.141297
39	10.5	10.248978
40	11.1	10.197887

41	9.4	10.289751
42	14.6	10.193106
43	10.0	10.164318
44	9.2	10.237331
45	3.5	10.136917
46	8.5	10.251005
47	13.2	10.301235
48	7.4	10.368860
49	8.8	10.270986
50	11.0	10.355304
51	7.8	10.213745
52	6.8	10.080662
53	11.4	10.274675
54	9.3	10.258435
55	9.7	10.189607
56	10.2	10.112018
57	8.0	10.269899
58	5.8	10.385166
59	12.1	10.136563
60	12.0	10.232623
61	11.4	10.238525
62	11.6	10.151999
63	14.6	10.175804
64	12.6	10.219327
65	8.2	10.277465
66	6.2	10.193318
67	9.3	10.206497
68	8.3	10.240651
69	7.8	10.120746
70	13.8	10.257818
71	11.8	10.200217
72	12.1	10.210821
73	8.0	10.179347
74	7.3	10.345727
75	12.0	10.258753
76	6.1	10.205178
77	11.7	10.211200
78	8.2	10.420087
79	8.2	10.155774
80	15.0	10.261530

81	13.2	10.162597
82	12.6	10.227060
83	11.0	10.349370
84	9.8	10.168485
85	12.4	10.180105
86	8.6	10.233735
87	8.0	10.288939
88	12.0	10.267136
89	10.9	10.306205
90	13.9	10.253088
91	11.1	10.142132
92	8.9	10.229407
93	7.9	10.281262
94	9.5	10.189894
95	10.6	10.156006
96	9.8	10.271504
97	13.0	10.268945
98	8.7	10.183943
99	5.3	10.306544
100	9.8	10.112102
101	4.4	10.257871
102	14.6	10.353911
103	10.5	10.244484
104	12.5	10.266553
105	11.3	10.238716
106	11.8	10.109178
107	9.0	10.188013
108	9.8	10.381792
109	10.1	10.265865
110	9.6	10.210625
111	8.3	10.176640
112	12.6	10.270966
113	12.1	10.302259
114	13.3	10.284769
115	9.4	10.391958
116	20.0	10.163140
117	14.2	10.222457
118	9.4	10.212894
119	10.0	10.346342
120	8.7	10.271038

121	13.1	10.180993
122	7.2	10.258331
123	9.8	10.249272
124	11.6	10.351765
125	9.2	10.358450
126	12.0	10.065322
127	9.1	10.342075
128	6.4	10.334863
129	9.2	10.256552
130	9.5	10.285286
131	10.9	10.220448
132	6.1	10.342252
133	9.5	10.236857
134	7.1	10.213228
135	9.1	10.252865
136	11.2	10.313834
137	5.3	10.219574
138	12.0	10.236344
139	11.2	10.327706
140	10.2	10.254073
141	12.4	10.253527
142	10.5	10.118748
143	6.8	10.253258
144	11.7	10.046316
145	14.1	10.214478
146	14.3	10.235238
147	13.7	10.225043
148	11.7	10.140327
149	8.5	10.287602
150	11.1	10.208648
151	10.6	10.213902
152	10.1	10.329593
153	7.5	10.304474
154	6.9	10.220031
155	11.5	10.270198
156	9.8	10.231551
157	15.8	10.197715
158	13.7	10.307434
159	10.2	10.252408
160	9.6	10.337402

161	7.1	10.266500
162	12.0	10.222782
163	10.5	10.205652
164	12.2	10.325227
165	6.1	10.404266
166	12.1	10.171605
167	7.5	10.329910
168	10.9	10.205947
169	12.8	10.316951
170	6.3	10.178357
171	13.2	10.207806
172	10.6	10.226012
173	10.5	10.222870
174	14.1	10.243010
175	6.1	10.221329
176	11.1	10.228241
177	12.2	10.232458
178	11.5	10.216843
179	16.2	10.222383
180	0.0	10.221317
181	9.5	10.293533
182	11.9	10.317790
183	9.9	10.281153
184	14.6	10.195898
185	8.4	10.073302
186	10.8	10.312892
187	10.2	10.248526
188	10.9	10.085115
189	9.0	10.300657
190	9.1	10.243091
191	8.9	10.227220
192	9.5	10.327976
193	8.8	10.146141
194	13.4	10.310508
195	9.5	10.055420
196	6.8	10.198577
197	9.7	10.319221
198	10.7	10.075041
199	13.8	10.180394
200	13.0	10.193382

201	13.1	10.379549
202	11.2	10.164198
203	6.4	10.184103
204	6.8	10.296288
205	9.4	10.232403
206	12.1	10.197754
207	13.7	10.264080
208	10.8	10.218681
209	12.2	10.231539
210	15.8	10.225683
211	11.6	10.146705
212	11.9	10.196663
213	10.7	10.316035
214	12.2	10.409373
215	17.6	10.189752
216	11.5	10.228353
217	10.9	10.226558
218	4.7	10.162931
219	13.0	10.223408
220	7.1	10.328347
221	12.2	10.309616
222	10.2	10.300177
223	4.4	10.253056
224	8.9	10.266906
225	13.8	10.305970
226	2.7	10.231743
227	7.7	10.151825
228	9.6	10.173710
229	13.3	10.279274
230	11.9	10.114737
231	10.5	10.234872
232	11.0	10.203505
233	13.5	10.202634
234	10.9	10.221626
235	9.0	10.335883
236	10.2	10.271658
237	9.0	10.280464
238	9.8	10.341935
239	10.7	10.475582
240	9.4	10.206521

241	12.9	10.247103
242	12.3	10.283121
243	8.4	10.139670
244	7.1	10.169587
245	9.4	10.184638
246	9.5	10.313143
247	11.1	10.361813
248	10.2	10.229360
249	9.2	10.189470
250	11.8	10.246742
251	13.9	10.292703
252	14.4	10.211219
253	9.1	10.173479
254	9.5	10.254095
255	10.9	10.263963
256	14.1	10.311708
257	9.8	10.339634
258	14.5	10.311149
259	10.4	10.341970
260	8.7	10.166482
261	6.7	10.268740
262	15.4	10.257390
263	11.5	10.258058
264	12.5	10.260142
265	8.3	10.062386
266	11.4	10.125543
267	8.4	10.198643
268	13.5	10.242550
269	4.5	10.330101
270	9.9	10.329663
271	14.6	10.222983
272	7.7	10.334310
273	8.0	10.132592
274	13.0	10.221808
275	10.0	10.224842
276	9.8	10.205342
277	11.1	10.175726
278	6.5	10.263668
279	10.9	10.170126
280	10.5	10.316832

281	13.0	10.246592
282	10.4	10.199628
283	12.2	10.287636
284	9.0	10.086177
285	6.7	10.261545
286	15.6	10.237703
287	8.8	10.329744
288	14.5	10.181731
289	14.1	10.225315
290	5.3	10.121411
291	8.0	10.261290
292	9.7	10.349341
293	5.9	10.277197
294	10.3	10.265544
295	9.8	10.237072
296	9.5	10.280015
297	10.1	10.207659
298	11.9	10.307853
299	6.6	10.293376
300	6.6	10.229731
301	11.9	10.233012
302	5.9	10.171526
303	11.2	10.178344
304	9.1	10.212387
305	10.3	10.302266
306	9.1	10.206326
307	8.5	10.181339
308	11.4	10.163939
309	11.4	10.237449
310	8.9	10.261393
311	13.2	10.215718
312	9.7	10.215232
313	10.9	10.148290
314	9.8	10.342271
315	18.9	10.259098
316	12.4	10.410553
317	7.7	10.241624
318	7.6	10.135007
319	5.0	10.190165
320	9.4	10.163557

321	6.2	10.141676
322	12.9	10.215475
323	10.0	10.246044
324	11.3	10.359926
325	13.4	10.221510
326	7.1	10.300588
327	11.4	10.274602
328	9.5	10.211687
329	12.5	10.306293
330	14.4	10.190880
331	7.9	10.158906
332	9.5	10.109124
333	12.2	10.327730
334	9.3	10.201864
335	7.5	10.237681
336	8.6	10.244816
337	10.6	10.111431
338	7.0	10.305141
339	7.6	10.181720
340	14.6	10.115017
341	9.1	10.122500
342	10.8	10.246757
343	14.0	10.251520
344	0.0	10.304747
345	13.3	10.293308
346	7.2	10.298777
347	12.2	10.341222
348	10.5	10.291700
349	13.1	10.336343
350	12.8	10.218950
351	11.3	10.244303
352	10.1	10.130066
353	5.3	10.299951
354	14.7	10.217792
355	13.2	10.127776
356	12.7	10.328308
357	11.3	10.176324
358	8.5	10.199978
359	9.2	10.311405
360	5.8	10.241520

361	8.8	9.991282
362	11.3	10.226652
363	12.0	10.300375
364	11.3	10.227462
365	10.9	10.202041
366	10.1	10.062911
367	9.1	10.180508
368	18.0	10.339396
369	7.6	10.298286
370	16.0	10.186158
371	10.3	10.154022
372	10.6	10.244176
373	12.4	10.137936
374	14.8	10.221155
375	9.2	10.162156
376	10.6	10.268922
377	11.2	10.249983
378	6.7	10.283069
379	11.5	10.146633
380	6.8	10.237626
381	14.7	10.262732
382	14.7	10.297966
383	5.7	10.231516
384	3.7	10.103550
385	7.2	10.294410
386	10.7	10.237096
387	8.9	10.103969
388	8.5	10.223787
389	10.7	10.221916
390	10.2	10.239502
391	11.1	10.291709
392	8.7	10.328219
393	12.4	10.212505
394	9.4	10.142548
395	10.8	10.141606
396	9.7	10.213764
397	7.8	10.264706
398	2.0	10.171035
399	8.5	10.362025
400	10.6	10.198998

401	12.0	10.271313
402	10.6	10.337931
403	9.9	10.282729
404	11.2	10.300817
405	7.5	10.150868
406	9.3	10.188013
407	6.8	10.240872
408	8.5	10.358398
409	10.3	10.258231
410	4.8	10.263848
411	8.4	10.241283
412	10.4	10.228277
413	5.4	10.125485
414	7.0	10.232348
415	10.0	10.217375
416	8.7	9.999398
417	5.0	10.261054
418	9.8	10.171983
419	16.0	10.178824
420	7.5	10.155013
421	9.3	10.262610
422	15.3	10.250038
423	12.5	10.241464
424	10.3	10.306316
425	11.3	10.284768
426	10.9	10.330354
427	12.5	10.403050
428	9.6	10.162125
429	11.2	10.215082
430	12.4	10.133515
431	13.3	10.164625
432	11.4	10.183698
433	12.8	10.145027
434	11.8	10.281654
435	8.6	10.180285
436	11.2	10.192038
437	8.0	10.241005
438	8.3	10.127033
439	13.5	10.153294
440	6.3	10.248770

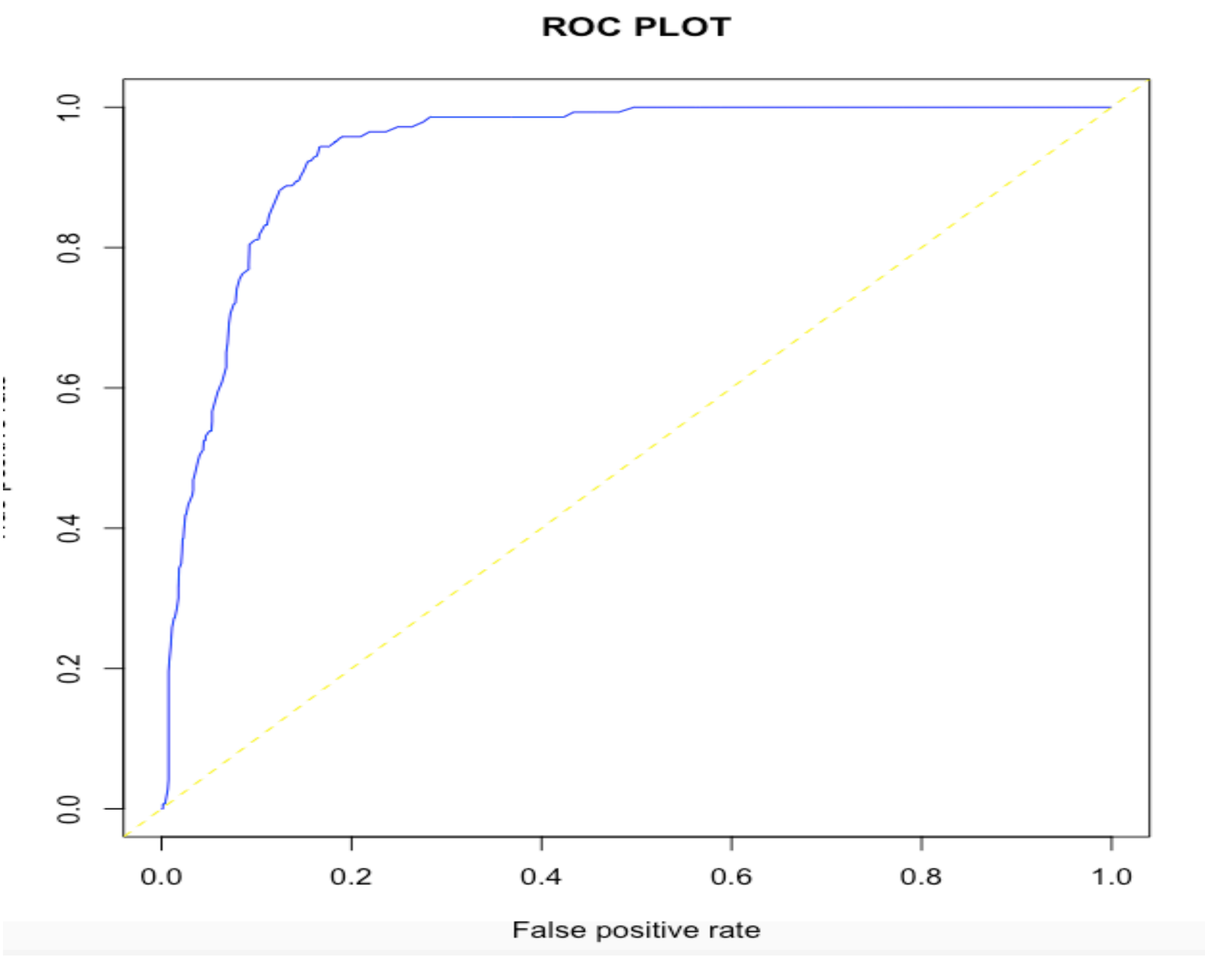
441	12.3	10.336197
442	12.4	10.190565
443	6.8	10.290504
444	12.6	10.294877
445	9.6	10.267283
446	11.1	10.275525
447	9.6	10.285187
448	6.9	10.218440
449	12.2	10.310337
450	6.3	10.128131
451	12.5	10.243502
452	9.8	10.206673
453	8.3	10.271171
454	14.3	10.115326
455	11.1	10.228144
456	14.8	10.237282
457	9.3	10.254813
458	9.7	10.207116
459	6.0	10.177555
460	11.0	10.339918
461	9.6	10.265092
462	9.6	10.263697
463	10.1	10.354876
464	5.9	10.188942
465	8.5	10.173057
466	13.6	10.294732
467	10.5	10.231821
468	11.6	10.250324
469	11.1	10.500458
470	17.2	10.086964
471	10.6	10.362313
472	9.5	10.310970
473	6.3	10.381963
474	6.2	10.091423
475	14.8	10.200985
476	9.9	10.316587
477	11.7	10.235704
478	7.6	10.309695
479	8.1	10.374123
480	11.2	10.261122

```
481    11.6 10.228490
482     5.3 10.327397
483     8.1 10.311994
484    13.3 10.204897
485    11.0 10.291354
486     6.7 10.324664
487    12.8 10.204734
488    10.5 10.170304
489     0.0 10.275279
490    12.3 10.290898
491    12.8 10.247646
492    14.3 10.170971
493     9.4 10.266026
494     5.9 10.234528
495     8.2 10.215895
496    11.1 10.302015
497     8.0 10.350774
498    11.9 10.166454
499     9.7 10.158444
500     7.5 10.186540
```

Logistic Regression:

```
Regression = lm(DataPlan~DayCalls+DayMins+OverageFee)
summary(Regression)
predict = predict(Regression)
data.frame(DataPlan, predict)
library(lmtest)
logit = glm(DataPlan~DayCalls+DayMins+OverageFee, data = cp_project, family
= binomial)
lrtest(logit)
install.packages("pscl")
library(pscl)
pR2(logit)
summary(logit)
odds = exp(coef(logit))
odds
Probability = odds/(1+odds)
Probability
predict(logit, type="response")
pred = fitted(logit)
```

```
data.frame(DataPlan,pred)
gg1 = floor(pred + 0.50)
gg1
table(Actual=DataPlan, prediction = gg1)
library(ROCR)
pred1 = prediction(gg1, DataPlan)
perf1 = performance(pred1, "tpr" , "fpr")
plot(perf1, col ="Blue", main = "ROC PLOT")
abline(0,1,lty =8, col ="yellow")
auc = performance(pred1, "auc")
auc = auc@y.valeus
auc
```



2 ii)

KNN:

```
str(cp_project)
data = cp_project[,-1]
norm = function(x) {(x-min(x)) /(max(x)-min(x))}
norm.data = as.data.frame(lapply(data[,-1], norm))
View(data)
View(norm.data)
usable.data = cbind(data[,1], norm.data)
str(usable.data)
View(usable.data)
library(caTools)
spl = sample.split(usable.data$`data[, 1]`, SplitRatio = 0.7)
train = subset(usable.data, spl == T)
test = subset(usable.data,spl == F)
dim(train)
dim(test)
library(class)
pred = knn(train[-1], test[-1], train[,1], k=48)
table.knn = table(test[,1], pred)
table.knn
sum(diag(table.knn))/sum(table.knn)

pred = knn(train[-1], test[-1], train[,1], k=48)
table.knn = table(test[,1], pred)
table.knn
sum(diag(table.knn))/sum(table.knn)
```

Knn value= 90.27%

2.iii) In this case we won't be able to Naïve Bayes algorithm as the data provided is numeric and for Naïve Bayes the data provided must be huge and categorical. Only KNN Model can be used to build. In order to build Naïve Bayes, the data must be really huge and factorial but in this case the data provided is numerical. So, in this case we won't be able to build Naïve Bayes algorithm.

Naïve Bayes:

```
NB = naiveBayes(`data[, 1]` ~., data = train)
predNB = predict(NB, test, type = "class")
tab.NB = table(test[,1], pred)
tab.NB
sum(diag(tab.NB)/sum(tab.NB))
```

NB Value= 90.27

2. iv) Model Comparison:

```
library(caret)
library(klaR)
train_control = trainControl(method = "cv" , number = 20)
modelKNN = train(`data[, 1]`~., data = train, trControl = train_control, method
="knn")
summary(modelKNN)
print(modelKNN)

k-Nearest Neighbors

2336 samples
   9 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 2101, 2102, 2102, 2103, 2102, 2103, ...
Resampling results across tuning parameters:

  k  RMSE      Rsquared     MAE
  5  43.74326  0.004801990  35.20971
  7  42.98471  0.006386241  34.81261
  9  42.57680  0.009996889  34.28788

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was k = 9.

folds <- createFolds(factor(train$`data[, 1]`), k = 10, list = FALSE)
folds
```

After comparing with both models, I can conclude that KNN method is much detail oriented

3. **<u>Actionable Insights (5 marks)</u>**

- Interpretation & Recommendations from the best model

The recommended Model is KNN as it is highly effective as we can see it has a good percentage of 90.27% as compared to NB model. As in this situation NB is not possible to perform. In order to perform Naïve Bayes Model we will have to convert them into numerical character.