Project
Churn reduction

Sankepally Vikram Reddy

19/05/2018

Contents

- **Finding threshold**
- **Applying threshold to the predicted probabilities**
- **Confusion matrix**

Chapter 6:

Results

- **Model summary**
- **ROC curve for each model**
- **Accuracy,Sensitivity and specificity**
- **Comparison of all models in R and Python**

Chapter 7:

Code

- **R code**
- **Python code**

Chapter 1:

Introduction:

Problem Statement

In this project I was given to predict Churn (loss of customers to competition) as it is a problem for companies because it is more expensive to acquire a new customer than to keep your existing one from leaving.

This problem statement is targeted at enabling churn reduction using analytics concepts.

Data:

I have been provided train and test data with various predictor variables and a target variable

It includes

- **Predictor variables**

1. "state"
2. "area.code"
3. "international.plan"
4. "number.vmail.messages"
5. "total.day.calls"
6. "total.eve.minutes"
7. "total.intl.minutes"
8. "total.intl.charge"
9. "Phone.number"
10. "voice.mail.plan"
11. "total.day.minutes"
12. "total.day.charge"
13. "total.eve.calls"
14. "total.night.minutes"
15. "total.night.charge"
16. "total.intl.calls"
17. "number.customer.service.calls"
18. "account.length"
19. "total.eve.charge"
20. "Total.eve.charge"

- **Target Variable**

1. "Churn"

Size of the data provided

Training data=3333 rows,21 columns
Test data=1667 rows,21 columns

Overview of the data

	State	account.length	area.code	phone.number	international.plan
1	KS	128	415	382-4657	no
2	OH	107	415	371-7191	no
3	NJ	137	415	358-1921	no
4	OH	84	408	375-9999	yes
5	OK	75	415	330-6626	yes
6	AL	118	510	391-8027	yes

	voice.mail.plan	number.vmail. messages	total.day.minutes
1	yes	25	265.1
2	yes	26	161.6
3	no	0	243.4
4	no	0	299.4
5	no	0	166.7
6	no	0	223.4

	total.day.calls	total.day.charge	total.eve.minutes	total.eve.calls
1	110	45.07	197.4	99
2	123	27.47	195.5	103
3	114	41.38	121.2	110
4	71	50.90	61.9	88
5	113	28.34	148.3	122
6	98	37.98	220.6	101

	Total.eve.charge	total.night.minutes	total.night.calls
1	16.78	244.7	91
2	16.62	254.4	103
3	10.30	162.6	104
4	5.26	196.9	89
5	12.61	186.9	121
6	18.75	203.9	118

	total.night.charge	total.intl.minutes	total.intl.calls
1	11.01	10.0	3
2	11.45	13.7	3
3	7.32	12.2	5

4	8.86	6.6	7	
5	8.41	10.1	3	
6	9.18	6.3	6	

	total.intl.charge	number.customer.service.calls	Churn
1	2.70	1	False.
2	3.70	1	False.
3	3.29	0	False.
4	1.78	2	False.
5	2.73	3	False.
6	1.70	0	False.

Chapter 2:

Exploratory data analysis

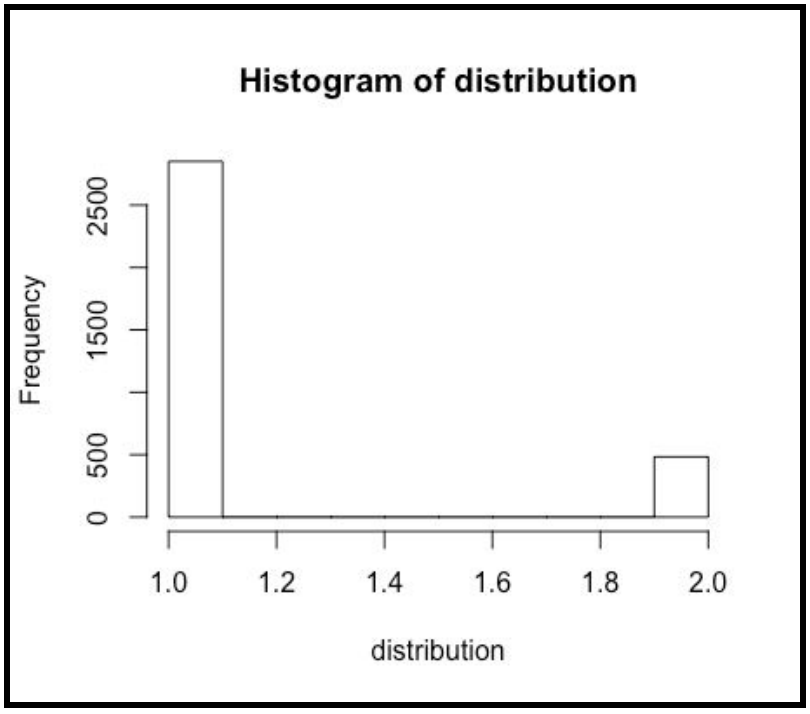
Steps I have done

- Missing values
- Class distribution
- Correlation plot

Missing values:

There are no missing values or empty values in the data,so there is no need of any imputation in the data.This lessens the burden in data preprocessing

Class distribution:



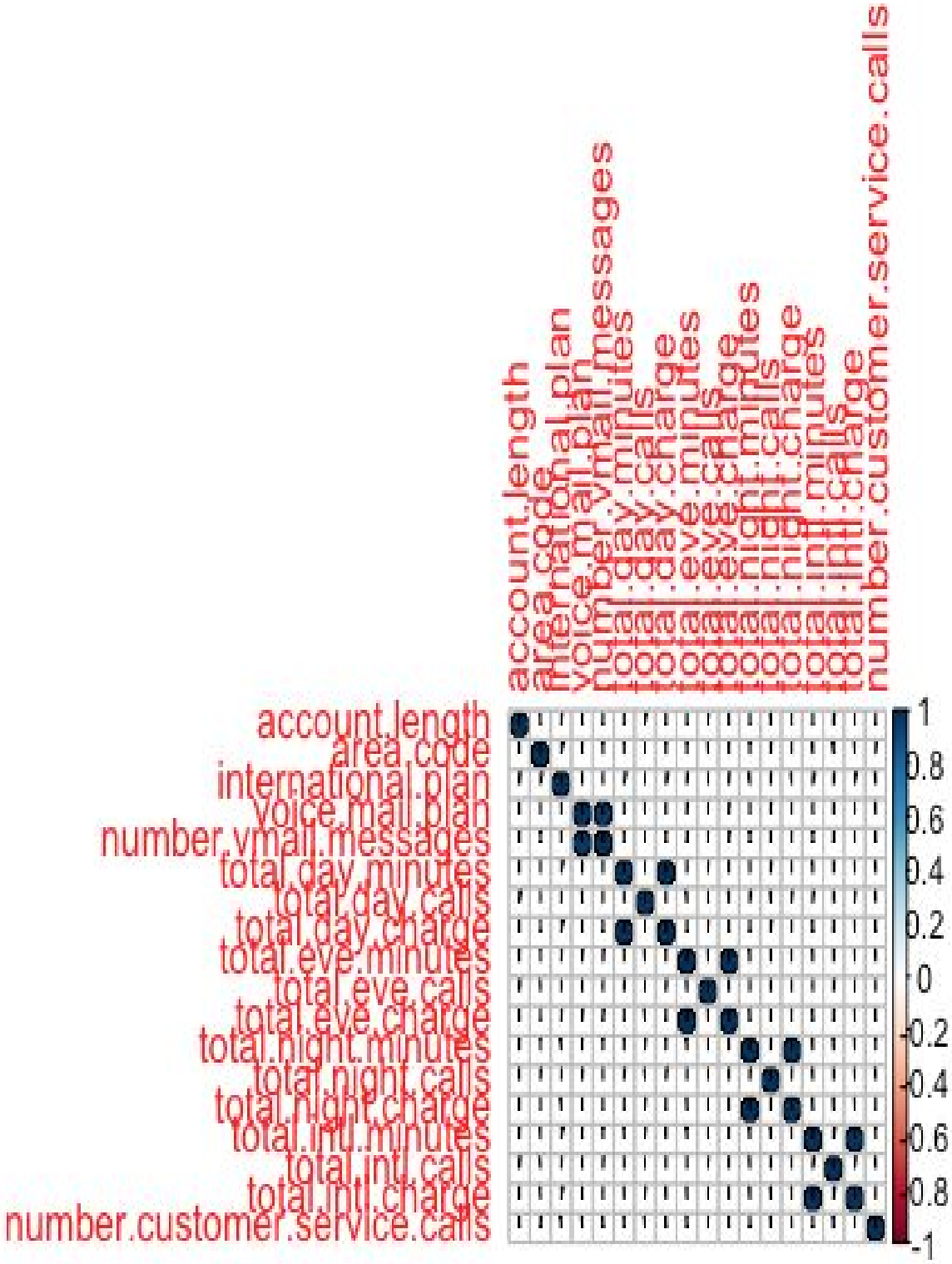
Number of True and False in the target variable

False. True.

2850 483

This tells that there is a problem of class imbalance.I will tell you in further chapters about how to deal with class imbalance

Correlation plot



As it is not possible to paste all the correlation matrix.From the correlation matrix and corr plot we can say that

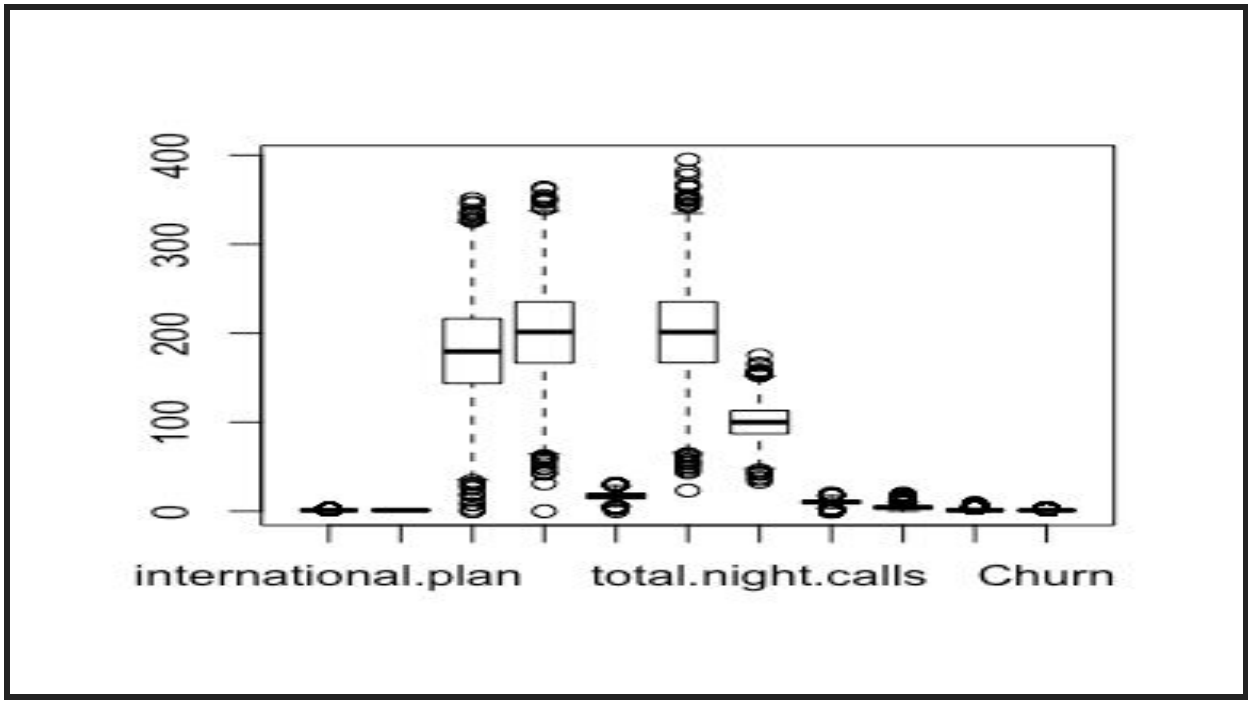
Predictor variables	**pearson correlation coefficient**
Number.vmail.messages ~ voice.mail.plan	0.956
Total.day.charge ~ total.day.minutes	0.999
Total.night.charge ~ total.night.minutes	0.9999
total.intl.charge ~ total.intl.minutes	0.9999

So I am removing

- Number.vmail.messages,
- total.day.charge,
- Total.night.charge,
- Total.intl.charge

From the data set because variables which has high pearson correlation coefficient will lead to un reliable results in the model
Further explanation will explained in next chapter.

Outlier analysis:
An outlier is a data point that is distant from other similar points. They may be due to variability in the measurement or may indicate experimental errors.
If possible, outliers should be excluded from the data set. However, detecting that anomalous instances might be very difficult, and is not always possible.



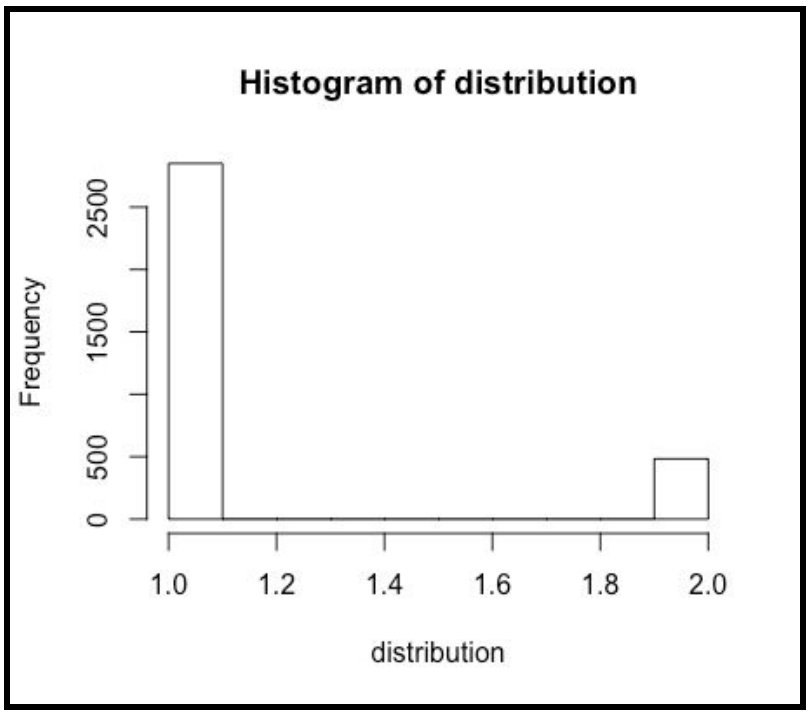
From the above graph we can see there exist some outliers in the data.In the next chapter i will explain about how to dealing with outliers

Chapter 3:

Data preprocessing:

Dealing with imbalanced data

As we can see that there is class imbalance problem



False. True.
2850 483

In order to avoid that issue i have use ROSE sampling

Rose:Generation of synthetic data by Randomly Over Sampling

Creates a sample of synthetic data by enlarging the features space of minority and majority class examples. Operationally, the new examples are drawn from a conditional kernel density estimate of the two classes

ROSE uses smoothed bootstrapping to draw artificial samples from the feature space neighbourhood around the minority class.

After applying ROSE sampling to the data
The distribution of classes
False True
1721 1612

Now the classes are evenly distributed

Removal of highly correlated variable

As we have seen in the above chapter ,the correlation plot displays the variables with their pearson correlation coefficient

We should remove the variables because

In a more general situation, when you have two independent variables that are very highly correlated, you definitely should remove one of them because you run into the multicollinearity conundrum and your regression model's regression coefficients related to the two highly correlated variables will be unreliable. Also, in plain English if two variables are so highly correlated they will obviously impart nearly exactly the same information to your regression model.
But, by including both we are actually weakening the model. we are not adding incremental information. Instead, we are infusing your model with noise. Not a good thing.

Predictor variables	**pearson correlation coefficient**
Number.vmail.messages ~ voice.mail.plan	0.956
Total.day.charge ~ total.day.minutes	0.999
Total.night.charge ~ total.night.minutes	0.9999
total.intl.charge ~ total.intl.minutes	0.9999

So I am removing

- Number.vmail.messages,
- total.day.charge,
- Total.night.charge,
- Total.intl.charge

Converting to the required data types:

Before passing the data to the model we have to make sure that they are converted to the required data types
Type of data:
'data.frame': 3333 obs. of 21 variables:
$ state : **Factor** w/ 51 levels "AK","AL","AR",..: 17 36 32 36 37 2 20 25 19 50 ...
$ account.length : **int** 128 107 137 84 75 118 121 147 117 141 ...
$ area.code : **int** 415 415 415 408 415 510 510 415 408 415 ...
$ phone.number : **Factor** w/ 3333 levels " 327-1058"," 327-1319",..: 1927 1576 1118 1708 111 2254 1048 81 292 118 ...
$ international.plan : **Factor** w/ 2 levels " no"," yes": 1 1 1 2 2 2 1 2 1 2 ...
$ voice.mail.plan : **Factor** w/ 2 levels " no"," yes": 2 2 1 1 1 1 2 1 1 2 ...
$ number.vmail.messages : **int** 25 26 0 0 0 0 24 0 0 37 ...
$ total.day.minutes : **num** 265 162 243 299 167 ...

$ total.day.calls : **int** 110 123 114 71 113 98 88 79 97 84 ...
$ total.day.charge : **num** 45.1 27.5 41.4 50.9 28.3 ...
$ total.eve.minutes : **num** 197.4 195.5 121.2 61.9 148.3 ...
$ total.eve.calls : **int** 99 103 110 88 122 101 108 94 80 111 ...
$ total.eve.charge : **num** 16.78 16.62 10.3 5.26 12.61 ...
$ total.night.minutes : **num** 245 254 163 197 187 ...
$ total.night.calls : **int** 91 103 104 89 121 118 118 96 90 97 ...
$ total.night.charge : **num** 11.01 11.45 7.32 8.86 8.41 ...
$ total.intl.minutes : **num** 10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...
$ total.intl.calls : **int** 3 3 5 7 3 6 7 6 4 5 ...
$ total.intl.charge : **num** 2.7 3.7 3.29 1.78 2.73 1.7 2.03 1.92 2.35 3.02 ...
$ number.customer.service.calls: **int** 1 1 0 2 3 0 3 0 1 0 ...
$ Churn : **Factor** w/ 2 levels " False."," True.": 1 1 1 1 1 1 1 1 1 1 ...

From the above data we should convert
International.plan
Voice.mail.plan
Churn variables
To factor levels
$ international.plan: Factor w/ 2 levels "1","2**"**
$ voice.mail.plan : Factor w/ 2 levels "1","2"
$ Churn: Factor w/ 2 levels "1","2":

Removing the predictor variables:

From the given dataset we should remove the variables which does not add any value to the model .

We will find out by passing all the variables to the model and eliminate the statistically insignificant predictor variables

state
account.length
area.code
international.plan ***
voice.mail.plan ***
total.day.minutes ***
total.day.calls

total.eve.minutes **
total.eve.calls
total.eve.charge **
total.night.minutes *
total.night.calls .
total.intl.minutes *
total.intl.calls ***
number.customer.service.calls ***

Signif. codes: 0 '' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1***

So we can say that the variables who do not have stars beside them are not statistically significant ,that means they are not useful in the model so that we can remove them
They are

- **State**
- **Area code**
- **total.day.calls**
- **total.eve.calls**
- **account.length**

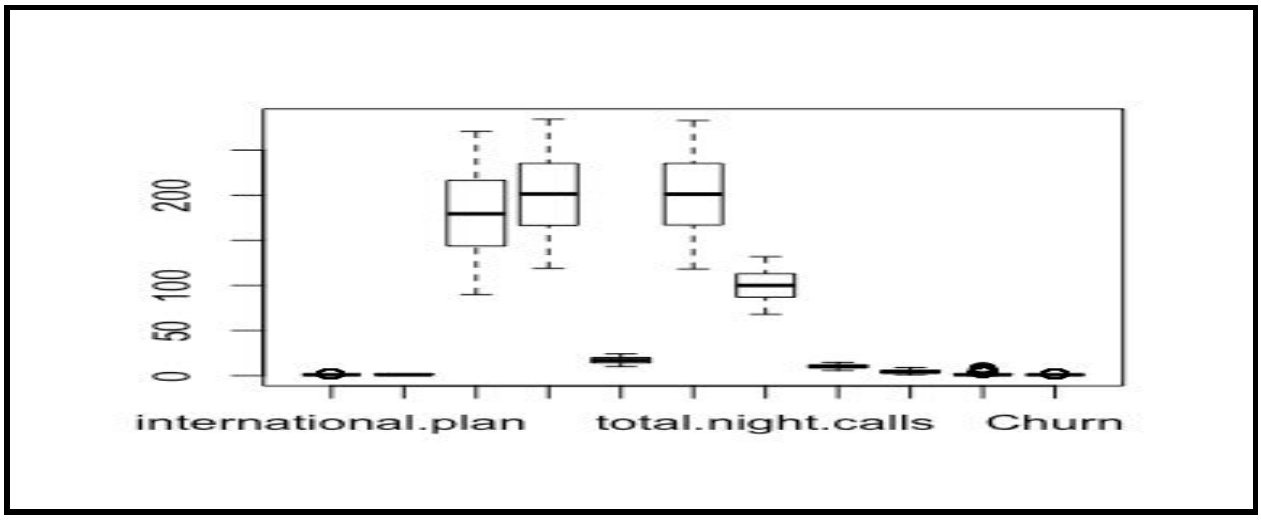
Dealing with outliers
Outliers are one of those statistical issues that everyone has in their data. Most parametric statistics, like means, standard deviations, and correlations, and every statistic based on these, are highly sensitive to outliers. And since the assumptions of common statistical procedures, like linear regression and ANOVA, are also based on these statistics, outliers can really mess up your analysis.
Despite all this, , it is NOT acceptable to drop an observation just because it is an outlier. They can be legitimate observations and are sometimes the most interesting ones. It's important to investigate the nature of the outlier before deciding.
1. Drop the outlier records.
In the case of Bill Gates, or another true outlier, sometimes it's best to completely remove that record from your dataset to keep that person or event from skewing your analysis.
2. Cap your data.
Another way to handle true outliers is to cap them. For example, if you're using income, you might find that people above a certain income level behave in the same way as those with a lower income. In this case, you can cap the income value at a level that keeps that intact.
3. Assign a new value.
If an outlier seems to be due to a mistake in your data, you try imputing a value. Common imputation methods include using the mean of a variable, or utilizing a regression model to predict the missing value.

4. Try a transformation.

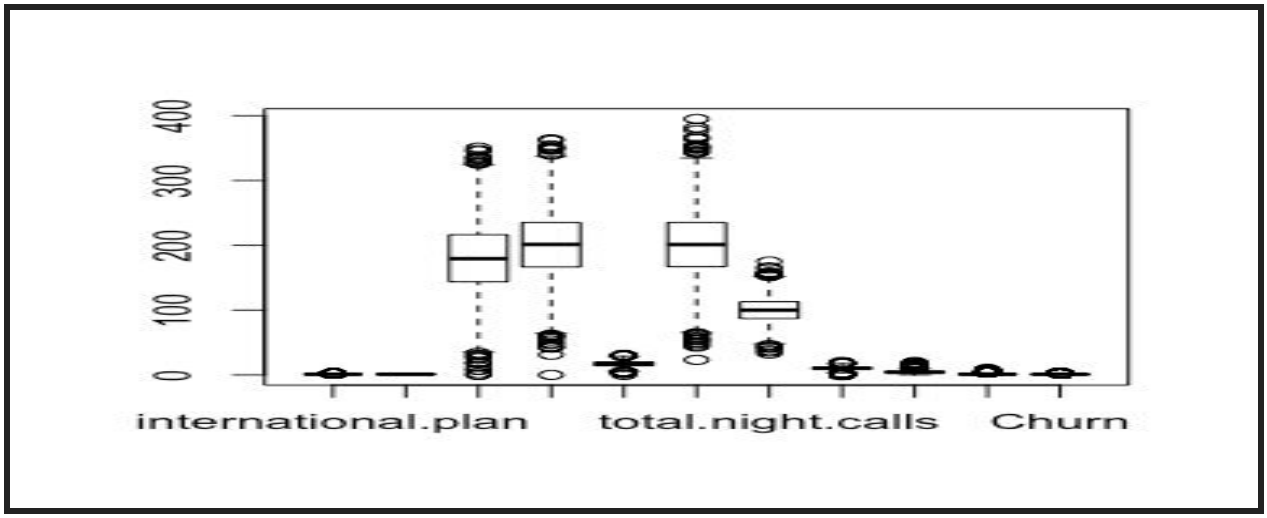
A different approach to true outliers could be to try creating a transformation of the data rather than using the data itself. For example, try creating a percentile version of your original field and working with that new field instead.

I have used the most widely used approach to replace outliers
I have used quantile function which has
The smallest observation corresponds to a probability of 0 and the largest to a probability of 1.
I have replace the observation with probability less than 0.05 with the observation probability of 0.05
Similarly the observation which has probability greater than 0.95 is replaced with the observation having probability of 0.95

After doing the data becomes



Before



Chapter 4:

Modelling:

Model selection

As it comes under the classification problem .we should choose classification algorithms
There are

1. **Logistic Regression**
2. **Naive Bayes**
3. **Stochastic Gradient Descent**
4. **K-Nearest Neighbours**
5. **Decision Tree**
6. **Random Forest**
7. **Support Vector Machine**

Out of these 7 classification algorithms i am using 4 popular classification algoritms
For our classification problem
They are

1. Logistic regression
2. Naive Bayes
3. Random Forest
4. K-Nearest neighbours

K fold cross validation

K-Fold Cross Validation is a common type of cross validation that is widely used in data science projects

K-fold cross validation is performed as per the following steps:

1. Partition the original training data set into k equal subsets. Each subset is called a fold. Let the folds be named as f1, f2, …, fk .
2. For i = 1 to i = k
    - Keep the fold fi as Validation set and keep all the remaining k-1 folds in the Cross validation training set.
    - Train your machine learning model using the cross validation training set and calculate the accuracy of your model by validating the predicted results against the validation set.
3. Estimate the accuracy of your machine learning model by **averaging the accuracies derived in all the k cases of cross validation.**

In the k-fold cross validation method, all the entries in the original training data set are used for both training as well as validation. Also, each entry is used for validation just once.

Generally, the value of k is taken to be 10, but it is not a strict rule, and k can take any value**.**

Applying different models

1.Logistic regression

Definition: Logistic regression is a machine learning algorithm for classification. In this algorithm, the probabilities describing the possible outcomes of a single trial are modelled using a logistic function.
Advantages: Logistic regression is designed for this purpose (classification), and is most useful for understanding the influence of several independent variables on a single outcome variable.
Disadvantages: Works only when the predicted variable is binary, assumes all predictors are independent of each other, and assumes data is free of missing values.

2.**Naive Bayes**

Definition: Naive Bayes algorithm based on Bayes' theorem with the assumption of independence between every pair of features. Naive Bayes classifiers work well in many real-world situations such as document classification and spam filtering.
Advantages: This algorithm requires a small amount of training data to estimate the necessary parameters. Naive Bayes classifiers are extremely fast compared to more sophisticated methods.
Disadvantages: Naive Bayes is is known to be a bad estimator.

3.K-Nearest Neighbours

Definition: Neighbours based classification is a type of lazy learning as it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the k nearest neighbours of each point.

Advantages: This algorithm is simple to implement, robust to noisy training data, and effective if training data is large.

Disadvantages: Need to determine the value of K and the computation cost is high as it needs to computer the distance of each instance to all the training samples.

4.Random Forest

Definition: Random forest classifier is a meta-estimator that fits a number of decision trees on various sub-samples of datasets and uses average to improve the predictive accuracy of the model and controls over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement.

Advantages: Reduction in over-fitting and random forest classifier is more accurate than decision trees in most cases.

Disadvantages: Slow real time prediction, difficult to implement, and complex algorithm.

Chapter 5:

Evaluation:

Predicting probabilities

It has two probabilities
One probability determining the even will happen and the other probability determining the event will not happen
The sum of probabilities will be one in total for for an event

ROC curve
A Receiver Operating Characteristic (ROC) Curve is a way to compare diagnostic tests. It is a plot of the true positive rate against the false positive rate.*
A ROC plot shows:

- The relationship between sensitivity and specificity. For example, a decrease in sensitivity results in an increase in specificity.
- Test accuracy; the closer the graph is to the top and left-hand borders, the more accurate the test. Likewise, the closer the graph to the diagonal, the less accurate the test. A perfect test would go straight from zero up the the top-left corner and then straight across the horizontal.
- The likelihood ratio; given by the derivative at any particular cutpoint.

ROC analysis provides tools to select possibly optimal models and to discard suboptimal ones independently from (and prior to specifying) the cost context or the class distribution. ROC analysis is related in a direct and natural way to cost/benefit analysis of diagnostic decision making.

Test accuracy is also shown as the area under the curve (which you can calculate using integral calculus). The greater the area under the curve, the more accurate the test.
A perfect test has an **area under the ROC curve** (AUROCC) of 1. The diagonal line in a ROC curve represents perfect chance.

In other words, a test that follows the diagonal has no better odds of detecting something than a random flip of a coin. The area under the diagonal is .5 (half of the area of the graph).
Therefore, a useless test (one that has no better odds than chance alone) has a AUROCC of .5.

Finding threshold

we will need to choose a threshold appropriate for your goal. The tradeoff with this is that you will need to decrease one of the TPR (true positive rate, or sensitivity), or TNR (true negative rate, or specificity) in order to increase the other - there is no way around this.

So, depending on our problem, we might e.g. happen to need a low false positive rate (FPR = 1 - TNR), which in turn will require you to have a high TNR - so this will definitely depend on details of your problem.

Having said this, to choose a threshold we will usually look at both the ROC curve and the distribution of TPR and TNR over the threshold.

Those should provide the required information for you to choose a reasonable tradeoff

So in our problem, for about equal TPR and TNR, we can choose a threshold around 0.5. If we would want to e.g. have very low FPR you would want to choose a lower threshold instead.

Applying threshold to the predicted probabilities

After choosing a threshold, we can use the predicted class probabilities to immediately determine the predicted class:

For completeness: predicted class probabilities from your model are made either a "positive" prediction (usually above the threshold) or a "negative" prediction (usually below the threshold) by this.

We can represent the effects of changing our threshold graphically by using a receiver operating characteristic, or ROC, curve. These graphs were initially developed for radar engineering but have found wide use in statistics for their ability to display this type of information.

The ROC graph plots **sensitivity on the y-axis** and **(1-specificity) on the x-axis**.

We put a point on the graph for each threshold value of the test, plotting the sensitivity and specificity of the test at that value. Connecting those points creates a curve - the ROC curve.

Confusion matrix

In predictive analytics, a table of confusion (sometimes also called a confusion matrix), is a table with two rows and two columns that reports the number of false positives, false negatives, true positives, and true negatives.

This allows more detailed analysis than mere proportion of correct classifications (accuracy). Accuracy is not a reliable metric for the real performance of a classifier, because it will yield misleading results if the data set is unbalanced

Chapter 6:

Model summary:

Logistic regression

```
Call:
NULL

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6288  -0.8637  -0.3917   0.9082   2.4307

Coefficients:
                              Estimate Std. Error z value Pr(>|z|)
(Intercept)                 -4.7488102  0.3561543 -13.334  < 2e-16 ***
international.plan2          2.2528916  0.1307439  17.231  < 2e-16 ***
voice.mail.plan2            -0.7231215  0.0993709  -7.277 3.41e-13 ***
total.day.minutes            0.0104370  0.0006496  16.066  < 2e-16 ***
total.eve.charge             0.0523234  0.0084613   6.184 6.26e-10 ***
total.night.minutes          0.0022150  0.0007516   2.947 0.003209 **
total.night.calls            0.0003172  0.0018479   0.172 0.863705
total.intl.minutes           0.0439363  0.0130108   3.377 0.000733 ***
total.intl.calls            -0.0434244  0.0146063  -2.973 0.002949 **
number.customer.service.calls  0.4734579  0.0262868  18.011  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4617.0  on 3332  degrees of freedom
Residual deviance: 3617.4  on 3323  degrees of freedom
AIC: 3637.4

Number of Fisher Scoring iterations: 4
```

Random forest

```
Random Forest


3333 samples
  9 predictor
```

2 classes: 'False', 'True'

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 2999, 3000, 3000, 3000, 3000, 3000, ...

Resampling results across tuning parameters:

mtry	ROC	Sens	Spec
2	0.8929956	0.8402205	0.7977571
5	0.8893092	0.8408019	0.7934284
9	0.8853641	0.8355727	0.7921708

ROC was used to select the optimal model using the largest value.

The final value used for the model was mtry = 2.

	Length	Class	Mode
call	5	-none-	call
type	1	-none-	character
predicted	3333	factor	numeric
err.rate	1500	-none-	numeric
confusion	6	-none-	numeric
votes	6666	matrix	numeric
oob.times	3333	-none-	numeric
classes	2	-none-	character
importance	9	-none-	numeric
importanceSD	0	-none-	NULL
localImportance	0	-none-	NULL
proximity	0	-none-	NULL
ntree	1	-none-	numeric
mtry	1	-none-	numeric
forest	14	-none-	list
y	3333	factor	numeric
test	0	-none-	NULL

inbag 0 -none- NULL

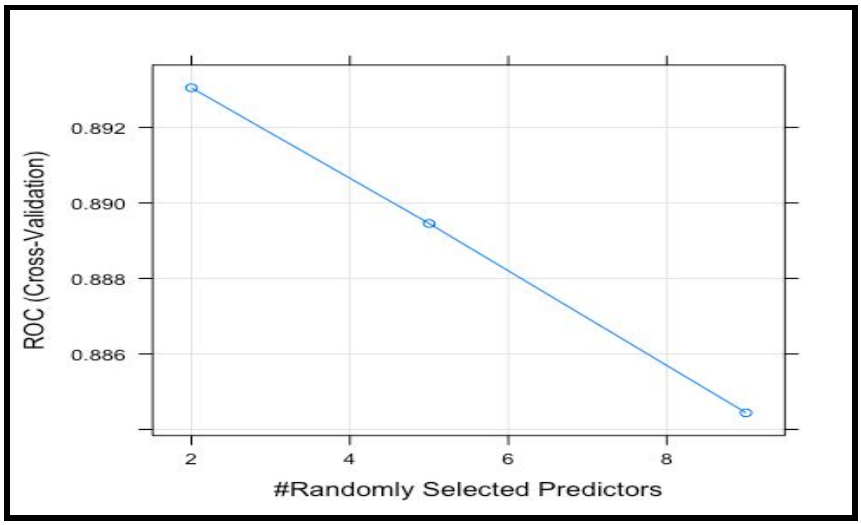
xNames 9 -none- character

problemType 1 -none- character

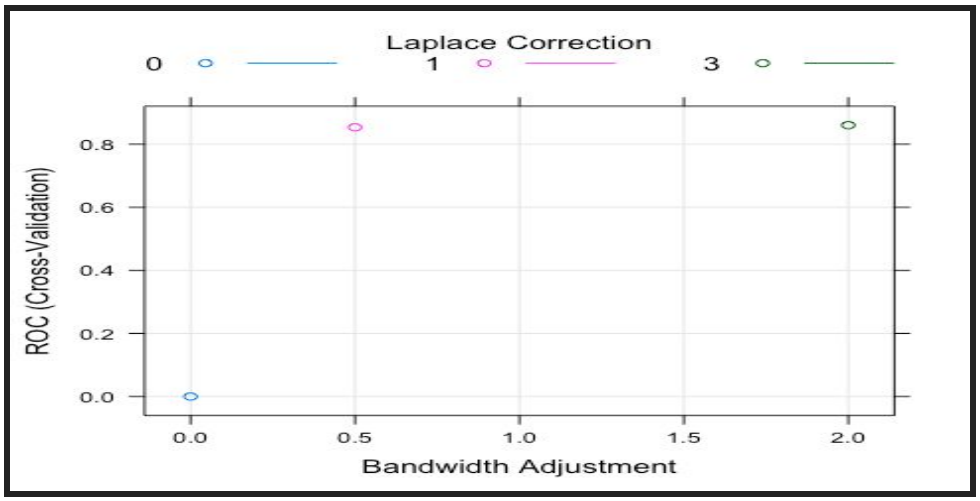
tuneValue 1 data.frame list

obsLevels 2 -none- character

param 1 -none- list



Naive bayes



Naive Bayes

3333 samples
  9 predictor
  2 classes: 'False', 'True'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3000, 3000, 2999, 2999, 3000, 3000, ...
Resampling results:

ROC	Sens	Spec
0.8493557	0.8750639	0.6805076

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'usekernel' was held constant at a value of FALSE
Tuning parameter 'adjust' was
 held constant at a value of FALSE

	Length	Class	Mode
apriori	2	table	numeric
tables	9	-none-	list
levels	2	-none-	character
call	5	-none-	call
x	9	data.frame	list
usekernel	1	-none-	logical
varnames	9	-none-	character
xNames	9	-none-	character
problemType	1	-none-	character
tuneValue	3	data.frame	list
obsLevels	2	-none-	character
param	0	-none-	list

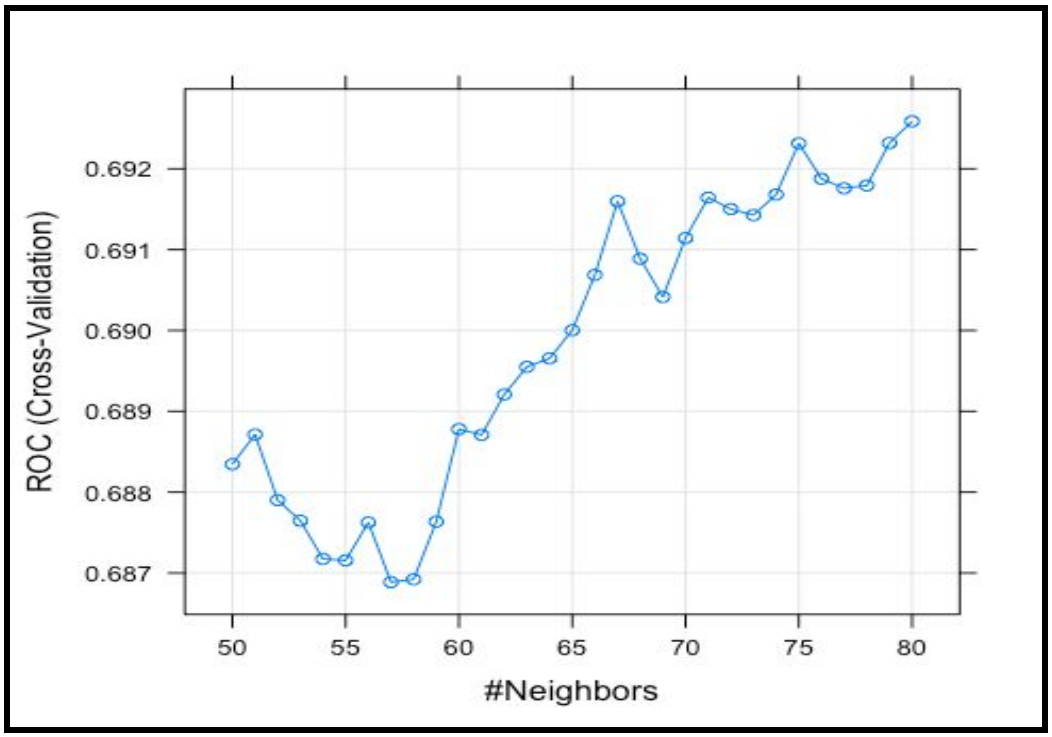
Here in this model i am getting warnings

These warnings are not a big deal; one of my test data points falls outside of the range of one of the distributions (probably the conditional). There's not much to do about that.

This is not an error or indication that the code is 'wrong', it is just information to let us know that one of your observations is producing some unusual probabilities - something you may want to examine in either your data or modeling approach.

How ever it is giving me the accuracy of 82 % which is good

KNN model



k-Nearest Neighbors

3333 samples
9 predictor
2 classes: 'False', 'True'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 2998, 3000, 3000, 3000, 2999, 3000, ...
Resampling results across tuning parameters:

k	ROC	Sens	Spec
1	0.5741986	0.6037303	0.5446668
2	0.6074361	0.6164942	0.5428188
3	0.6243220	0.6600954	0.5434323
4	0.6449716	0.6647164	0.5508665
5	0.6529256	0.7013543	0.5514838
6	0.6581500	0.6996034	0.5545970
7	0.6602843	0.7199388	0.5434131
8	0.6664527	0.7147164	0.5514723
9	0.6681302	0.7251680	0.5434016
10	0.6689696	0.7176065	0.5421670
11	0.6715313	0.7362011	0.5328694
12	0.6705800	0.7216729	0.5341040
13	0.6699673	0.7437626	0.5235220
14	0.6729913	0.7385233	0.5353194
15	0.6736765	0.7443406	0.5359290

ROC was used to select the optimal model using the largest value.
The final value used for the model was k = 15.

	Length	Class	Mode
learn	2	-none-	list
k	1	-none-	numeric

theDots 0 -none- list

xNames 9 -none- character

problemType 1 -none- character

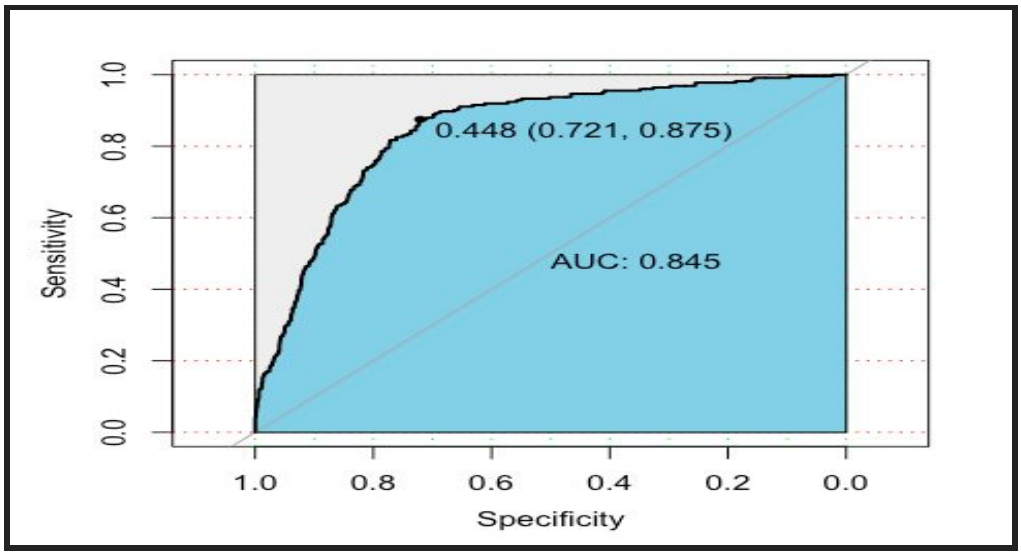
tuneValue 1 data.frame list

obsLevels 2 -none- character

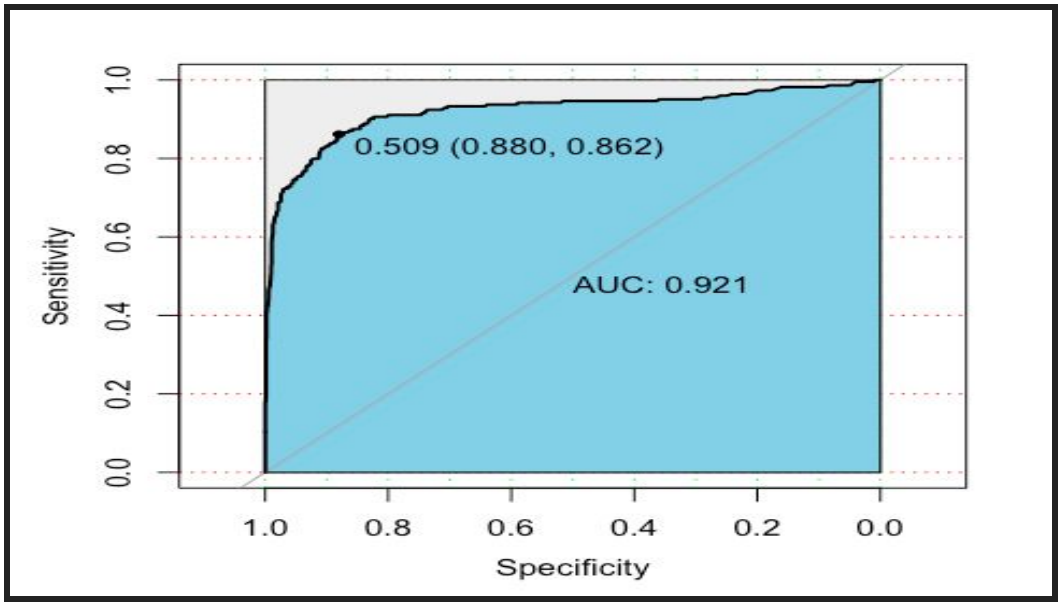
param 0 -none- list

ROC curve for each model

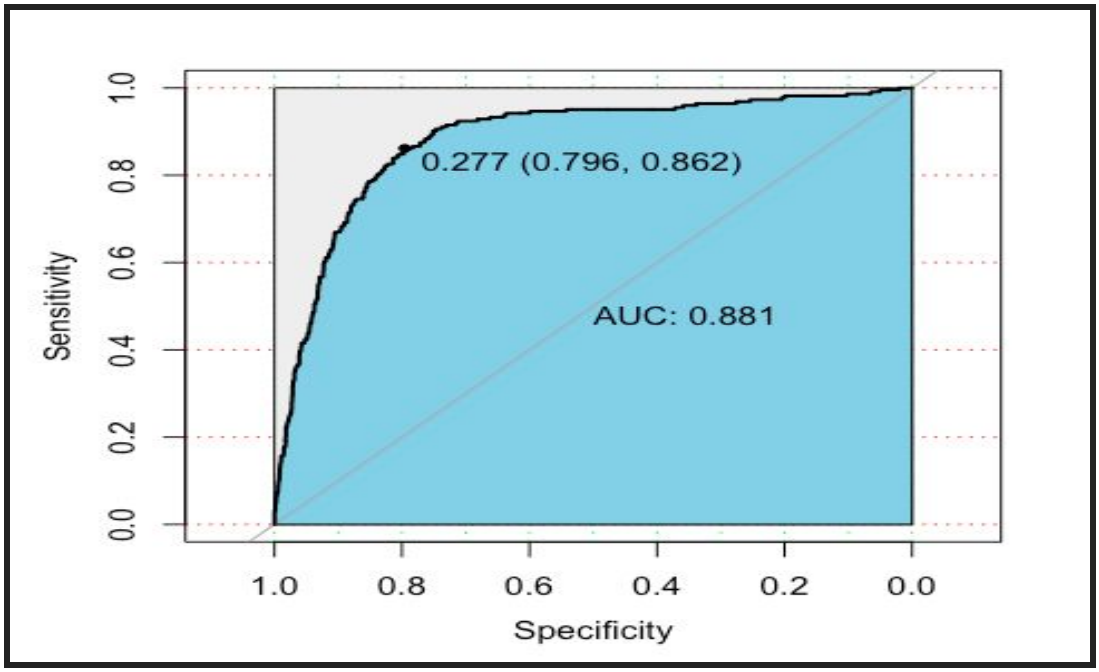
Logistic regression



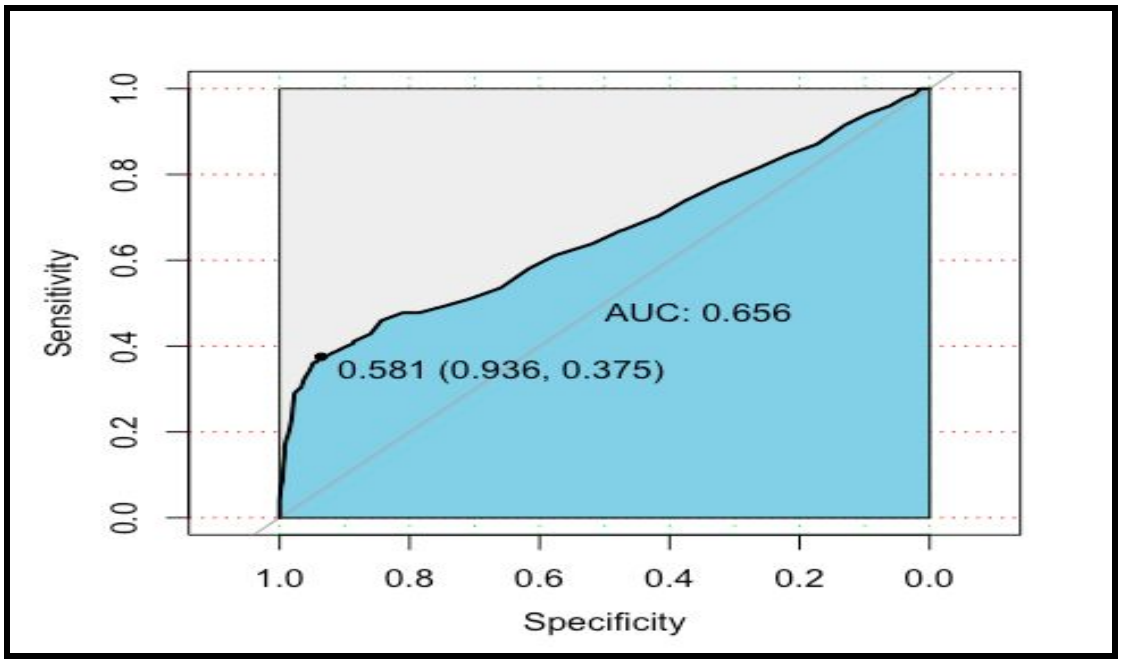
Random forest



Naive bayes:



KNN model



Accuracy:

Logistic regression

```
"threshold"
 0.490341
Confusion Matrix and Statistics

          Reference
Prediction  False.  True.
   False.    1113    330
   True.       41    183

               Accuracy : 0.7774
                 95% CI : (0.7567, 0.7972)
    No Information Rate : 0.6923
    P-Value [Acc > NIR] : 5.158e-15

                  Kappa : 0.3808
 Mcnemar's Test P-Value : < 2.2e-16

     Sensitivity : 0.9645
     Specificity : 0.3567
```

Random forest

```
"threshold"
 0.497

 Reference
Prediction  False.  True.
   False.    1263    180
   True.       30    194

               Accuracy : 0.874
                 95% CI : (0.8571, 0.8896)
    No Information Rate : 0.7756
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5779
       Mcnemar's Test P-Value : < 2.2e-16

     Sensitivity : 0.9768
     Specificity : 0.5187
```

Naive bayes

```
"threshold"
 0.2915878
Reference
Prediction  False.  True.
   False.    1175    268
   True.       36    188

            Accuracy : 0.8176
              95% CI : (0.7982, 0.8359)
   No Information Rate : 0.7265
   P-Value [Acc > NIR] : < 2.2e-16

                 Kappa : 0.4547
 Mcnemar's Test P-Value : < 2.2e-16

          Sensitivity : 0.9703
          Specificity : 0.4123
```

KNN

```
“threshold"
 0.4844907

Confusion Matrix and Statistics

          Reference
Prediction  False.  True.
   False.    1170    273
   True.      117    107

             Accuracy : 0.766
               95% CI : (0.745, 0.7862)
   No Information Rate : 0.772
   P-Value [Acc > NIR] : 0.7312

                 Kappa : 0.2229
 Mcnemar's Test P-Value : 4.203e-15

          Sensitivity : 0.9091
          Specificity : 0.2816
```

Comparison of models in R and Python

Observations from models in R

With outliers

Model	Threshold	Accuracy	Area under ROC
Logistic regression	0.490341	0.7774 =77.7%	0.845
Random forest	0.497	0.874 =87.4%	0.921
Naive bayes	0.2915878	0.8176 =81.7%	0.881
KNN algorithm	0.4844907	0.766=76.6%	0.686

Without outliers

Model	Threshold	Accuracy	Area under ROC
Logistic regression	0.4955664	0.7786 =77.8%	0.843
Random forest	0.52	0.8884 =88.8%	0.913
Naive bayes	0.3290708	0.8284 =82.4%	0.873
KNN algorithm	0.4525789	0.7097 =70.9%	0.701

From the above comparison we can say that **random forest algorithm** is performing well on test data.
It is the best among four models

Observations from models in Python

With outliers

Model	**Threshold**	**Accuracy**	**Area under ROC**
Logistic regression	0.490341	0.752=75.2%	0.820605633106
Random forest	0.168	0.870=87%	0.918471871597
Naive bayes	0.112829844327	0.815=81.5%	0.852616696367
KNN algorithm	0.333333333333	0.870=87%	0.682794401544

Without outliers

Model	**Threshold**	**Accuracy**	**Area under ROC**
Logistic regression	0.137582677315	0.757648470306	0.820633476883
Random forest	0.168	0.866826634673	0.920309560935
Naive bayes	0.122283585447	0.826634673065	0.860614048114
KNN algorithm	0.15	0.80323935213	0.709600225225

From the above comparison we can say that **random forest algorithm** is performing well on test data.
It is the best among four models

Chapter 7:

R code

```
#loading data
train=read.csv('Train_data.csv',na.strings=c("",NA),stringsAsFactors
= T,sep=',')
test=read.csv('Test_data.csv',na.strings=c("",NA),stringsAsFactors =
T)
#missing or empty values
sapply(train,function(x) sum(is.na(x)))
sapply(test,function(x) sum(is.na(x)))
#class imbalance
hist(as.numeric(as.factor(train$Churn)))

#correlation plot
corm=train
corm=corm[-4]#removing state variable
corm=corm[-1]#removing phone number
corm[3:4]=as.integer(unlist(corm[3:4]))
matrix=cor(corm[1:18])
head(print(matrix))
library(corrplot)
corrplot(matrix, method="pie")
#data pre processing
train$international.plan=as.factor(unclass(train$international.plan))
train$voice.mail.plan=as.factor(unclass(train$voice.mail.plan))
train$Churn=as.factor(unclass(train$Churn))
#deleting columns which are not
#useful in prediction(highly correlated)
library(dplyr)
train=select(train,-number.vmail.messages,-total.day.charge,
             -total.night.charge,-total.intl.charge)
#removing columns which are not statistically significant
#reason mentioned in report(to increase accuracy)
train=train[-4]#removinf customer phone number
train=select(train,-state,-area.code,-account.length,
-total.day.calls, -total.eve.calls )


#data preprocessing on test data
test$international.plan=as.factor(unclass(test$international.plan))
```

```
test$voice.mail.plan=as.factor(unclass(test$voice.mail.plan))
test=test[-4]#removing  phone number columns which  not useful


#removing columns which are highly correlated
test=select(test,-number.vmail.messages,-total.day.charge,
            -total.night.charge,-total.intl.charge)
#removing columns which are not statistically significant from test
dat
#as we have removed the same from train data

test=select(test,-state,-area.code,-account.length, -total.day.calls,
-total.eve.calls )
# removing outliers and replacing them with quantile 1 and quantile 2
#with respective to their position
boxplot(train)
outlier_removed_data=train
for (value in
c('total.eve.minutes','total.day.minutes','total.eve.charge',

'total.night.minutes','total.night.calls','total.intl.minutes',
                'total.intl.calls')){
qn = quantile(outlier_removed_data[value], c(0.05, 0.95), na.rm =
TRUE)
print(qn)
outlier_removed_data[value][outlier_removed_data[value]<qn[1]]=qn[1]
outlier_removed_data[value][outlier_removed_data[value]>qn[2]]=qn[2]
}
boxplot(outlier_removed_data)



#dealing with imbalanced train data(with outliers)
table(train$Churn)
prop.table(table(train$Churn))
install.packages("ROSE")
library(ROSE)
train_data <- ROSE(Churn ~ ., data = train, seed = 1)$data
train_data$Churn=ifelse(train_data$Churn==1,"False","True")

table(train_data$Churn)
prop.table(table(train_data$Churn))
```

```
#without outliers
outlier_removed_data=ROSE(Churn ~ ., data = outlier_removed_data,
seed = 1)$data
outlier_removed_data$Churn=ifelse(outlier_removed_data$Churn==1,"Fals
e","True")
```

```
#function for finding optimum thershold and calculating accuracy
myfunction=function(model){
  print(model)
  #summary of the model
  print(summary(model))
  #predicting classes
  pred_class=predict(model,test[1:10])
   print(table(pred_class,test$Churn))
  #predicting probabilities
  pred=predict(model,test[1:10],type='prob')
  a=data.frame(prob=pred[,2], obs=test$Churn)
  library(pROC)
  #plotting ROC
  modelroc = roc(a$obs,a$prob)
  plot(modelroc, print.auc=TRUE, auc.polygon=T,
grid=c(0.1,0.2),grid.col=c('green','red'),max.auc.polygon=TRUE,
auc.polygon.col="skyblue", print.thres=TRUE)
  ##adjust optimal cut-off threshold for class probabilities
  threshold = coords(modelroc,x="best",best.method =
"closest.topleft")[[1]]
  print("threshold")
  print(threshold)
  #get optimal cutoff threshold
  predCut = factor( ifelse(pred[, 2] < threshold, " False.", "
True.") )
  #confusion matrix
  confusionMatrix(test$Churn,predCut)
}
```

```
#######logistic regression#####
library(caret)
train_control_log<- trainControl(method="cv", number=10,classProbs =
TRUE,summaryFunction = twoClassSummary)
```

```
#model
#no tuning parameters for regressin and classification
Logmodel<- train(Churn~., data=train_data,
trControl=train_control_log, method="glm",metric="ROC")
#calling function
myfunction(Logmodel)
summary(Logmodel)
######model with out outliers###
Logmodel<- train(Churn~., data=outlier_removed_data,
trControl=train_control_log, method="glm",metric="ROC")
#calling function
myfunction(Logmodel)
```

```
# #####random forest#####
library(caret)
# define training control
train_control_RF<- trainControl(method="cv", number=10,classProbs =
TRUE,summaryFunction = twoClassSummary)
# tuning the model
newGrid = expand.grid(mtry = c(2,15,20))
#model

RFmodel<- train(Churn~., data=train_data, trControl=train_control_RF,
method="rf",ntree=500,
                metric="ROC",grid=newGrid)
#calling function
myfunction(RFmodel)
####model with out outliers###
RFmodel<- train(Churn~., data=outlier_removed_data,
trControl=train_control_log, method="rf",
                ntree=500,metric="ROC",grid=newGrid)
#calling function
myfunction(RFmodel)
```

```
#######naive bayes#######
train_control_NB<- trainControl(method="cv", number=10,classProbs =
TRUE,summaryFunction = twoClassSummary)
```

```
# train the model
grid <- data.frame(fL=c(0,1.0,3.0), usekernel = TRUE,
adjust=c(0,0.5,2.0))
nb_tune <- data.frame(usekernel =TRUE, fL = 0)
#model
NBmodel=train(Churn~., data=train_data, trControl=train_control_NB,
method="nb",metric="Accuracy",tuneGrid = grid)
#calling function

myfunction(NBmodel)
####model with out outliers###
NBmodel<- train(Churn~., data=outlier_removed_data,
trControl=train_control_log,
method="nb",metric="ROC",tuneGrid=nb_grid)
#calling function
myfunction(NBmodel)


#explanation:
#The problem is that we have  predictors and will be multiplying a lot
of stuff
#between [0, 1] together and that can cause numerical problems
#So, my guess as to the problem is that Naive Bayes isn't really the
right tool for the job here #Numerically, the probabilities are all
going towards zero
#These warnings are not a big deal; one of my  test data points falls
outside of the range of one of #the distributions (probably the
conditional).
#There's not much to do about that.
#This is not an error or indication that the code is 'wrong', it is
just information to let us know that #one of our observations is
producing some unusual probabilities - something we may want to
#examine in either  data or modeling approach.

#How ever it is giving me the accuracy of 82 % which is good



####### KNN algorithm ######
train_control_KNN<- trainControl(method="cv", number=10,classProbs =
TRUE,summaryFunction = twoClassSummary)
# tuning the model
```

```
grid   = expand.grid(k =50:80)
#model
KNNmodel=train(Churn~., data=train_data, trControl=train_control_KNN,
method="knn",metric="ROC",tuneGrid=grid)
plot(KNNmodel)
#calling function
myfunction(KNNmodel)
####model with out outliers###
KNNmodel<- train(Churn~., data=outlier_removed_data,
trControl=train_control_KNN,
method="knn",metric="ROC",tuneGrid=Knn_grid)
#calling function
myfunction(KNNmodel)
```

Python code

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue May 15 18:03:56 2018

@author: vikramreddy
"""
#loading requied libraries
import pandas as pd
import seaborn as sns
from imblearn.over_sampling import ADASYN
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.cross_validation import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import roc_curve, auc
from sklearn.utils import shuffle
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

#loading data
train=pd.read_csv('Train_data.csv')
test=pd.read_csv('Test_data.csv')

#check for missing values
train.describe()
train.isnull().values.ravel().sum()
#correlation plot of predictor variables
cor=train.iloc[:,0:18]
sns.heatmap(cor.corr(),cmap=sns.diverging_palette(220, 10,
as_cmap=True))



# creating function for data preprocessing
def data_cleaning(dat):
    #removing the variables which has high correlation
```

```
    dat=dat.drop(['number vmail messages', 'total day charge',
                  'total night charge','total intl charge'], axis=1)

    #data pre processing on
    #removing columns which are not useful
    dat=dat.drop(['state', 'phone number'], axis=1)
    dat['international plan'] = pd.factorize(dat['international
plan'])[0]
    dat['voice mail plan'] = pd.factorize(dat['voice mail plan'])[0]
    dat['Churn'] = pd.factorize(dat['Churn'])[0]
    return dat


#calling data_cleaning
train=data_cleaning(train)
test=data_cleaning(test)
#imbalance check in output
train['Churn'].hist()



#outlier detection
plt.boxplot(train)
columns=['account length','total day minutes','total day
calls','total eve minutes','total eve calls',
         'total eve charge','total night minutes','total night
calls','total intl minutes',
         'total intl calls']
#replacing variables which lie below the probability of 0.05 with the
observation having probability of 0.05 and
#replacing variables which lie above the probability of 0.95 with the
observation having probability of 0.95
down=train.quantile(0.05)
up=train.quantile(0.95)
pd.options.mode.chained_assignment = None
data_without_outlier=train
for column in columns:

data_without_outlier[data_without_outlier[column]<down[column]][colum
n]=down[column]
```

```
data_without_outlier[data_without_outlier[column]>up[column]][column]
=up[column]
```

```
#over sampling for data with outliers
sm = ADASYN()
x=pd.DataFrame(train.iloc[:,0:14])
y=pd.DataFrame(train.iloc[:,14])
x1,y1 = sm.fit_sample(x,y.values.ravel())
train=pd.DataFrame(x1,columns=list(x))
train['Churn']=y1
#checking class distribution after applyning over sampling(data with
outliers)
train['Churn'].hist()
#shuffling data
train=shuffle(train)
test=shuffle(test)
```

```
#oversampling of data without outliers
sm_clean = ADASYN()
x_clean=pd.DataFrame(data_without_outlier.iloc[:,0:14])
y_clean=pd.DataFrame(data_without_outlier.iloc[:,14])
x2,y2 = sm_clean.fit_sample(x_clean,y_clean.values.ravel())
clean_train=pd.DataFrame(x1,columns=list(x_clean))
clean_train['Churn']=y2
#checking class distribution after applyning over sampling for data
without outliers
clean_train['Churn'].hist()
#shuffling data
clean_train=shuffle(clean_train)
test=shuffle(test)

#creating a function for cross validation
def cross_valid(model,test,train):
    model.fit(train.iloc[:,0:14],train['Churn'])
    #prediction of model without using cross validation
    pred=model.predict(test.iloc[:,0:14])
```

```
    print("confusion matrix  of predictions(not probability
predictions) and model with out using cross validation")
    print(confusion_matrix(test['Churn'], pred))
    #predictions using cross validation
    predicted = pd.DataFrame(cross_val_predict(model,
test.iloc[:,0:14],test['Churn'],cv=10))
    print("accuracy of cross validation predictions(not probability
predition)")
    print(metrics.accuracy_score(test['Churn'], predicted))
    #finding the score for each cross validation
    accuracy = cross_val_score(model,
test.iloc[:,0:14],test['Churn'], cv=10,scoring='roc_auc')
    print("mean accuracy score of  10 cross validations")
    print (accuracy.mean())
    #predicting probabilities from cross validation
    predicted_proba = pd.DataFrame(cross_val_predict(model,
test.iloc[:,0:14],test['Churn'],cv=10,method='predict_proba'))
    print("predicted probabilities from cross validation")
    print(predicted_proba)
    fpr, tpr, thresholds =roc_curve(test['Churn'],
predicted_proba.iloc[:,1])
    roc_auc = auc(fpr, tpr)
    print("area under the curve")
    print(roc_auc)
    #determing threshold
    # The optimal cut off would be where tpr is high and fpr is low
    # tpr - (1-fpr) is zero or near to zero is the optimal cut off
point
    i = np.arange(len(tpr)) # index for df
    roc = pd.DataFrame({'fpr' : pd.Series(fpr, index=i),'tpr' :
pd.Series(tpr, index = i), '1-fpr' : pd.Series(1-fpr, index = i),
'tf' : pd.Series(tpr - (1-fpr), index = i), 'thresholds' :
pd.Series(thresholds, index = i)})
    thresh=roc.iloc[(roc.tf-0).abs().argsort()[0:1]]
    threshold=thresh[['thresholds']]
    print("threshold obtained from ROC curve")
    print(threshold.iloc[0,0])
    #mapping threshold to the predicted probabilities
    output= predicted_proba.iloc[:,1].map(lambda x:1  if x >
threshold.iloc[0,0] else 0)
    print("confusion matrix for predictions(after applying threshold
to probabilities) and test data ")
    print(confusion_matrix(test['Churn'], output))
```

```
    print("accuracy of cross validation predictions( probability
prediction)")

    print(metrics.accuracy_score(test['Churn'], output))
    #plotting Roc curve
    fig, ax = plt.subplots()
    plt.plot(roc['tpr'])
    plt.plot(roc['1-fpr'], color = 'red')
    plt.xlabel('1-False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic')
    ax.set_xticklabels([])
    return ""

#logistic regression
logreg=LogisticRegression()
#calling the function for data with outliers
cross_valid(logreg,test,train)
#calling the function for data without outliers(clean data)
cross_valid(logreg,test,clean_train)



#random forest
rf = RandomForestClassifier(n_estimators=500)
#calling the function for data with outliers
cross_valid(rf,test,train)
#calling the function for data with out outliers(clean data)
cross_valid(rf,test,clean_train)

#naive bayes model
gnb = GaussianNB()
#calling the function for data with outliers
cross_valid(gnb,test,train)
#calling the function for data without outliers(clean data)
cross_valid(gnb,test,clean_train)

#kNN model
kNN=KNeighborsClassifier(n_neighbors=20, radius=5.0,
algorithm='auto')
#calling the function for data with outliers
cross_valid(kNN,test,train)
#calling the function for data without outliers(clean data)
```

```
cross_valid(kNN,test,clean_train)
```

References

- Machine learning by brett lanz
- Pattern recognition and machine learning book by christopher bishop
- Augmented startup videos
- R cookbook