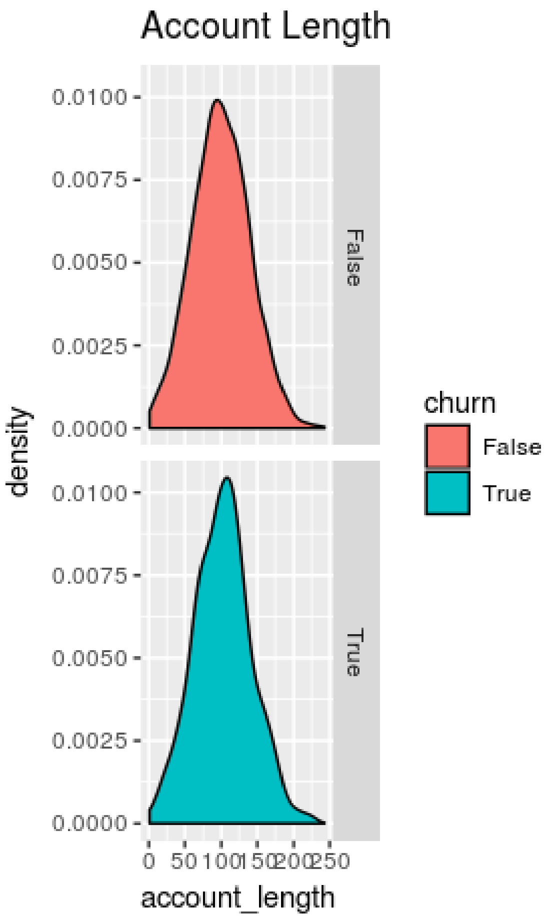
Project Report: Assignment 3

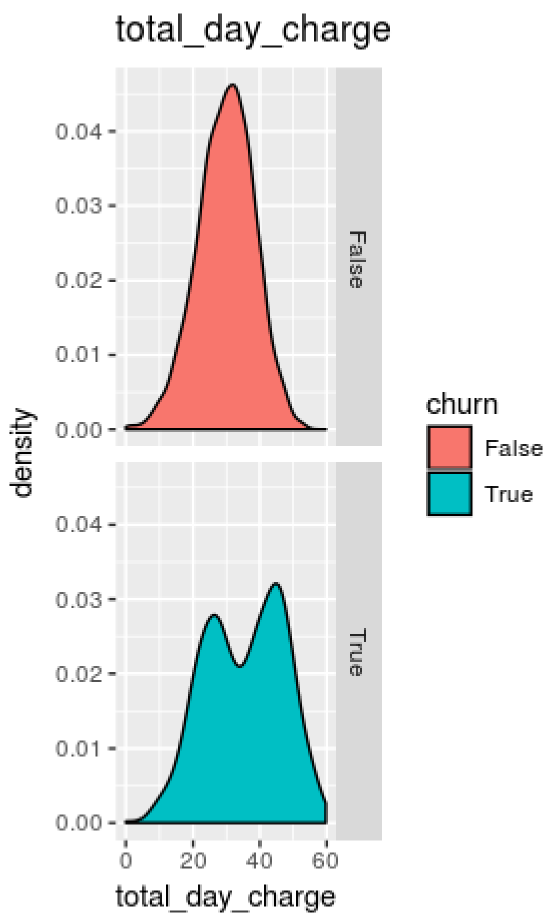
Customer Churn Prediction

Suyash Damle
15CS10057

Data Pruning/ Analysis of the attributes:

- The area code field is read into the data frame as numerical, where it should ideally be categorical. So, this field needs a change of type
- The data for phone numbers is more useful for the purpose of identification or verification of the provided data. Using it to train a classifier makes no sense, as it has *unique values for each observation*. This is verifiable by checking the number of level corresponding to the column "phone_numbers". This column has 5000 levels, for 5000 data values. Hence, the values are unique and useless for the classification task at hand. Thus, it is dropped out.
- Further checks are made during model training phase and required changes are made.





(Similar curves) *(Significantly different for some values)*

(Some samples of the type of curves obtained: attribute value densities for both churn types)

The curves are roughly bell-shaped. In the case of account length, visually, the graphs are similar for both positive and negative churns. This implies that the ***attribute account_length may not have much***

significance in the task of classification. The same pattern is followed by a number of other attributes: total_eve_charge, total_intl_charge.

Some other attributes, such as the total_day_charge, have distinctly different curves in True churn values than the False values. So, they are more likely to have some significance at some point for the task of classification at hand.

Division of dataset

The distiribution of data in the **original data** is as follows:

```
False   True
 4293    707
```

The train to test ratio is kept at 8 : 2
After proper symmetrical division of the data, with equal proportions of True and False values in both test and train sets:
(Using the *createDataPartition()* function from the *caret* package)

Train data:

```
False   True
 3435    566
```

Test data:

```
False   True
858    141
```

Evaluation metrics

- The confusion metrics could be easily plotted and related information obtained, in each case, using the ***confusionMatrix()*** function.

- The parameters requied for this are the test data classification and the predictions from the model.

  confusionMatrix (predictions, dataTest_y)

- The function also returns the following important values:
  - Accuracy
  - Sensitivity – the Recall
  - Specificity (True negative rate) – this is what is important for us, as the True (churn) is being considered as the negative class
  - Pos. Predicted Value – the Precision

The Naive Bayes Classifier:

Issues:

The naive bayes model takes *only categorical data.* So, to deal with the continuous numerical data abundant in this dataset, one needs to divide them into discrete classes. Two soluntions exist:

1. Manual discretization : Deciding some fixed values for each attribute and grouping data above and below (and between) these values separately.

2. Considering a probability density : One could assume that the attribute follows certain prob. distribution and then proceed:
    - assuming (say) gaussian distribution for all the attribute values in each class.
    Eg: we may assume that the “total_eve_charge” in True and False class for all the observations follows normal distribution.
    - from the training data, finding the mean and standard deviation of the distribution for both, True and False classes
    - for each of the attribute to be considered henceforth, finding the probability density value at the observation data, corresponding to both True and False classes. The one with higher pdf value is considered to be the valid classification for the observation point.
    So, if pdf is higher in False class, we put in category 1, else in category 2.

The used ***naiveBayes()*** function from ***e1071*** uses this second approach to deal with this kind of situation.

Analysis from the Naive Bayes model

The naive bayes model, once trained could be used for obtaining further insight into the data attributes: the model begins with the calculations of posterior probabilities of the form: P(X|Y), which are further expected to be useful in the calculation of P(Y|X) during the process of prediction.

The P(X|Y) data could be printed as:

model $ tables:

```
$state
          state
dataTrain_y          AK          AL          AR          AZ          CA          CO          CT          DC          DE          FL          GA          HI          IA
      False 0.016351119 0.026104418 0.017785427 0.019219736 0.007458405 0.020080321 0.018646013 0.018646013 0.018072289 0.017785427 0.017211704 0.018932874 0.015777395
      True  0.003241491 0.021069692 0.021069692 0.016207455 0.021069692 0.017828201 0.022690438 0.012965964 0.024311183 0.012965964 0.014586710 0.009724473 0.012965964
          state
dataTrain_y          ID          IL          IN          KS          KY          LA          MA          MD          ME          MI          MN          MO          MS
      False 0.025817556 0.018359151 0.017498566 0.017785427 0.020940906 0.016637980 0.019219736 0.018932874 0.019219736 0.020940906 0.024956971 0.020654045 0.019506598
      True  0.021069692 0.011345219 0.021069692 0.025931929 0.017828201 0.009724473 0.021069692 0.030794165 0.029173420 0.022690438 0.027552674 0.017828201 0.025931929
          state
dataTrain_y          MT          NC          ND          NE          NH          NJ          NM          NV          NY          OH          OK          OR          PA
      False 0.018646013 0.017211704 0.020367183 0.016924842 0.019219736 0.020080321 0.018646013 0.017498566 0.022088353 0.023522662 0.016924842 0.021227768 0.015777395
      True  0.027552674 0.021069692 0.014586710 0.011345219 0.016207455 0.034035656 0.016207455 0.022690438 0.027552674 0.024311183 0.017828201 0.027552674 0.011345219
          state
dataTrain_y          RI          SC          SD          TN          TX          UT          VA          VT          WA          WI          WV          WY
      False 0.019506598 0.019219736 0.018072289 0.017498566 0.020940906 0.023522662 0.025817556 0.021514630 0.017785427 0.022088353 0.028973035 0.024383247
      True  0.011345219 0.022690438 0.014586710 0.017828201 0.037277147 0.024311183 0.008103728 0.014586710 0.027552674 0.014586710 0.030794165 0.011345219

$account_length
          account_length
dataTrain_y     [,1]     [,2]
      False 100.2786 39.64771
      True  101.5689 39.85080
```

From this data, following conclusions are possible:

- The posterior probabilities, P(state| churn) for most states do not differ much
  Eg: P (state= AK | churn = False) ~ P(state=AR | churn = False)
  This indicates that users of these states are all equally likely to remain with our services

- In some states, the difference is significant
  Eg: P (state= AK | churn = True) >> P(state=AL | churn = True)
  In such states, the user is more (or less) likely to continue with the /service as compared to other states

```
$international_plan
          international_plan
dataTrain_y         no        yes
      False 0.93628164 0.06371836
      True  0.73239437 0.26760563

$voice_mail_plan
          voice_mail_plan
dataTrain_y        no       yes
      False 0.7122491 0.2877509
      True  0.8397887 0.1602113

$number_vmail_messages
          number_vmail_messages
dataTrain_y     [,1]     [,2]
      False 8.351092 13.78803
      True  5.021201 11.85650

$total_day_minutes
          total_day_minutes
dataTrain_y     [,1]     [,2]
      False 175.6971 49.90298
      True  206.5226 67.74762

$total_day_calls
          total_day_calls
dataTrain_y      [,1]     [,2]
      False  99.86084 19.62373
      True  100.75265 20.33670

$total_day_charge
          total_day_charge
dataTrain_y     [,1]      [,2]
      False 29.86907  8.483467
      True  35.10926 11.517274

$total_eve_minutes
          total_eve_minutes
dataTrain_y     [,1]     [,2]
      False 198.9172 50.38457
      True  212.8516 50.72032
```

and so on...

Final result:

```
Confusion Matrix and Statistics

          Reference
Prediction False True
     False   829   75
     True     29   66

               Accuracy : 0.8959
                 95% CI : (0.8753, 0.9141)
    No Information Rate : 0.8589
    P-Value [Acc > NIR] : 0.0002898

                  Kappa : 0.5028
 Mcnemar's Test P-Value : 1.021e-05

            Sensitivity : 0.9662
            Specificity : 0.4681
         Pos Pred Value : 0.9170
         Neg Pred Value : 0.6947
             Prevalence : 0.8589
         Detection Rate : 0.8298
   Detection Prevalence : 0.9049
      Balanced Accuracy : 0.7171

       'Positive' Class : False
```

The Decision Tree Classifier

- The decision tree classifier is trained using the ***rpart()*** (CART algorithm) function from the ***e1071*** package.
- Splitting is done on the basis of information gain and gini index – both

Using information gain as a heuristic for splitting:

```
Confusion Matrix and Statistics

          Reference
Prediction  False  True
     False    845    37
     True      13   104

                 Accuracy : 0.9499
                   95% CI : (0.9345, 0.9626)
     No Information Rate : 0.8589
     P-Value [Acc > NIR] : < 2.2e-16

                    Kappa : 0.7778
 Mcnemar's Test P-Value : 0.001143

              Sensitivity : 0.9848
              Specificity : 0.7376
           Pos Pred Value : 0.9580
           Neg Pred Value : 0.8889
               Prevalence : 0.8589
           Detection Rate : 0.8458
    Detection Prevalence : 0.8829
       Balanced Accuracy : 0.8612

        'Positive' Class :  False
```

Using Gini Index as heuristic:

```
Confusion Matrix and Statistics

          Reference
Prediction  False  True
     False    852    45
     True       6    96

                 Accuracy : 0.9489
                   95% CI : (0.9334, 0.9618)
     No Information Rate : 0.8589
     P-Value [Acc > NIR] : < 2.2e-16

                    Kappa : 0.7619
 Mcnemar's Test P-Value : 1.032e-07

              Sensitivity : 0.9930
              Specificity : 0.6809
           Pos Pred Value : 0.9498
           Neg Pred Value : 0.9412
               Prevalence : 0.8589
           Detection Rate : 0.8529
    Detection Prevalence : 0.8979
       Balanced Accuracy : 0.8369

        'Positive' Class :  False
```

So, ***information gain is the choice, as it offers better specificity, which is what we need***.

- The created tree demonstrates which attributes give best result at the time of splitting. The visualization of the created tree has been attached with the codes and the report, in a pdf document.

The Support Vector Machine (SVM) Models:

Two different kernels were tried :

Linear

```
              Confusion Matrix and Statistics

            Reference
Prediction  False  True
     False    858   129
     True       0    12

                  Accuracy : 0.8709
                    95% CI : (0.8485, 0.891)
     No Information Rate : 0.8589
     P-Value [Acc > NIR] : 0.1477

                     Kappa : 0.1378
 Mcnemar's Test P-Value : <2e-16

              Sensitivity : 1.00000
              Specificity : 0.08511
           Pos Pred Value : 0.86930
           Neg Pred Value : 1.00000
               Prevalence : 0.85886
           Detection Rate : 0.85886
    Detection Prevalence : 0.98799
       Balanced Accuracy : 0.54255

        'Positive' Class :  False
```

-> So, the linear SVM model (without scaling of attributes) *performs quite poorly in terms of the specificity.*

Polynomial

- The polynomial kernel, having a large number of parameters offers the option of ***tuning the parameters.*** This technique lets the user enter the range of values (or, discrete values) to be checked.
- The system itself checks the classifier over all the values (*grid search)* and determine the optimal values.
- Two distinct runs of optimization were performed, for different parameter combinations :

Optimizing the polynomial degree:

```
model_svm_poly=tune.svm(churn~.,data=(dataTrain[2:21]),cost=5,kernel="polynomial",
degree=c(2,3,4,5,6))
```

Optimal degree found : 2

```
Confusion Matrix and Statistics

          Reference
Prediction False True
     False   851  111
     True      7   30

               Accuracy : 0.8819
                 95% CI : (0.8602, 0.9012)
    No Information Rate : 0.8589
    P-Value [Acc > NIR] : 0.01866

                  Kappa : 0.2958
 Mcnemar's Test P-Value : < 2e-16

            Sensitivity : 0.9918
            Specificity : 0.2128
         Pos Pred Value : 0.8846
         Neg Pred Value : 0.8108
             Prevalence : 0.8589
         Detection Rate : 0.8519
   Detection Prevalence : 0.9630
      Balanced Accuracy : 0.6023

       'Positive' Class : False
```

Optimizing degree of polynomial and the “gamma” parameter:

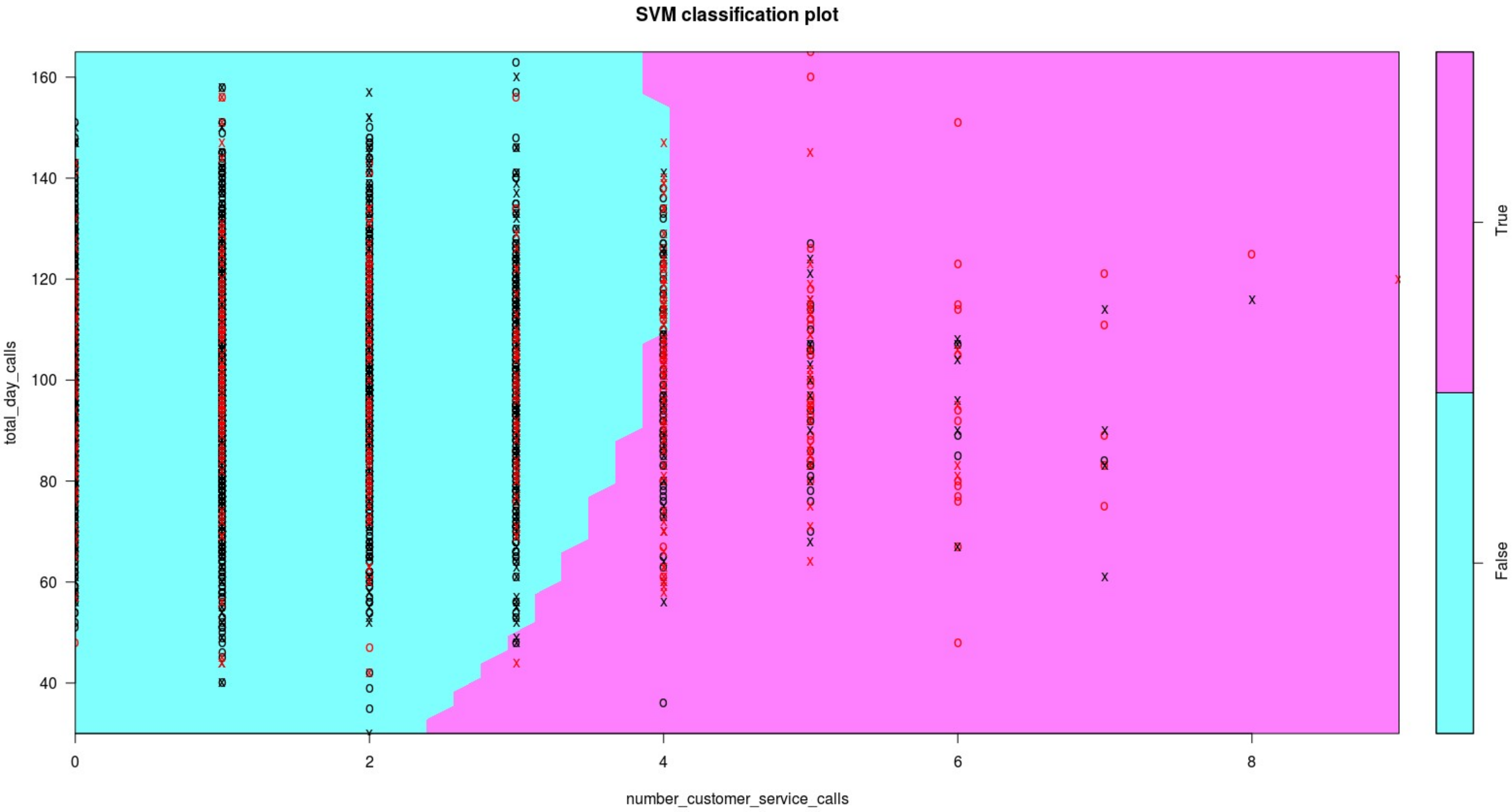
model_svm_poly=tune.svm(churn~.,data=(dataTrain[2:21]),cost=5,kernel="polynomial", degree=c(2,3,4),gamma=c(0.01,0.1,1))

Optimized parameters : degree = 3
gamma = 0.1

```
Confusion Matrix and Statistics

          Reference
Prediction False True
     False   817   61
     True     41   80

               Accuracy : 0.8979
                 95% CI : (0.8774, 0.916)
    No Information Rate : 0.8589
    P-Value [Acc > NIR] : 0.0001368

                  Kappa : 0.5523
 Mcnemar's Test P-Value : 0.0599338

            Sensitivity : 0.9522
            Specificity : 0.5674
         Pos Pred Value : 0.9305
         Neg Pred Value : 0.6612
             Prevalence : 0.8589
         Detection Rate : 0.8178
   Detection Prevalence : 0.8789
      Balanced Accuracy : 0.7598

       'Positive' Class : False
```

Decision Boundary visualization:

After the training of the classifier, the decision boundary could be visualized wrt the training data using the function:
plot(model_svm_poly$best.model,data=dataTrain[2:21],formula=total_day_calls~number_customer_service_calls)



The variables in the x and y axis could be easily changed by simple replacement in the formula

- Running tuning process on more than 2 parameters takes a very long time to terminate and hence, SVM could not be checked and optimized any further. However, as the number of variations are large, some combination(s) may perform much better than the above results.

Conclusion

The performances of all the explored models could be summarized as below:

MODEL	ACCURACY	RECALL	PRECISION	SPECIFICITY
Naive Bayes	89.60%	96.60%	91.70%	46.80%
Decision Tree (Binary, information gain)	94.99%	98.48%	95.80%	73.76%
SVM (Linear)	87.90%	100.00%	86.93%	8.50%
SVM (Polynomial – degree = 2, gamma=1)	88.19%	99.18%	88.46%	21.28%
SVM (Polynomial – degree = 3, gamma=0.1)	89.79%	95.22%	93.05%	56.74%

Based on the above data, one could conclude that the **decision tree classifier with the information gain heuristic and the support vector machine with polynomial kernel perform the best** and the performance of SVM (presently slighly lagging behind) may beat it; however careful tuning of the parameters is required for that purpose.