axa171831

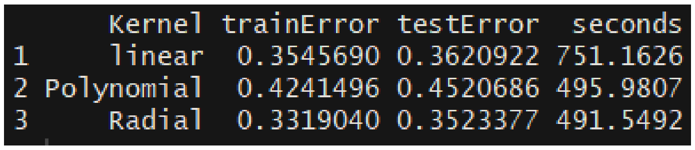
Dataset1: Online news popularity

The dataset consists of the number of shares on different content pages posted on Mashable.com. The predicted variables is the number of shares which has been changed to binary for classification. The predicted shares now means whether the article was in the top 50% in terms of shares or not. The predictor variables are very heterogeneous such as the number of words in the article title, the polarity of words, the rate of non-stop words and the article category among others.

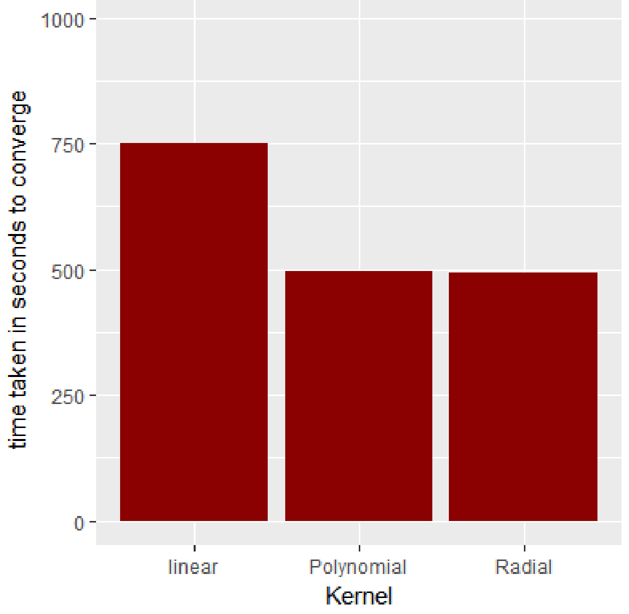
SVM

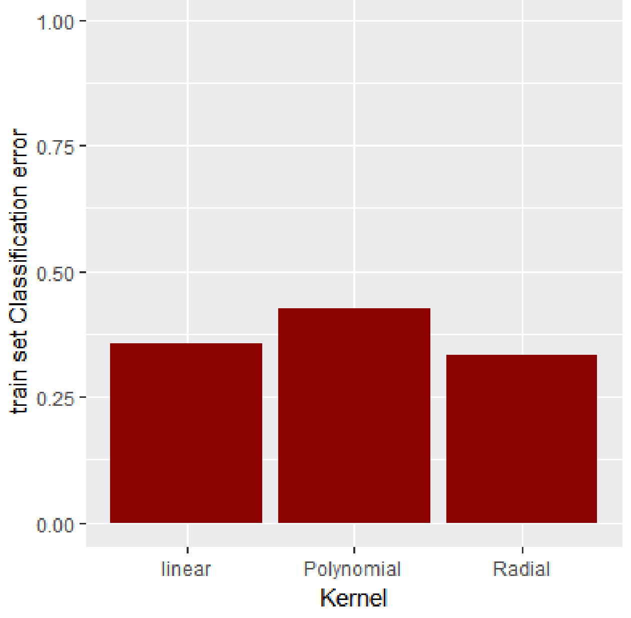
I tried three different kernels in SVM. SVM was taking a lot of time so I had to use a library supporting multi-core processing for SVM. A visual summary of all kernels is given below.

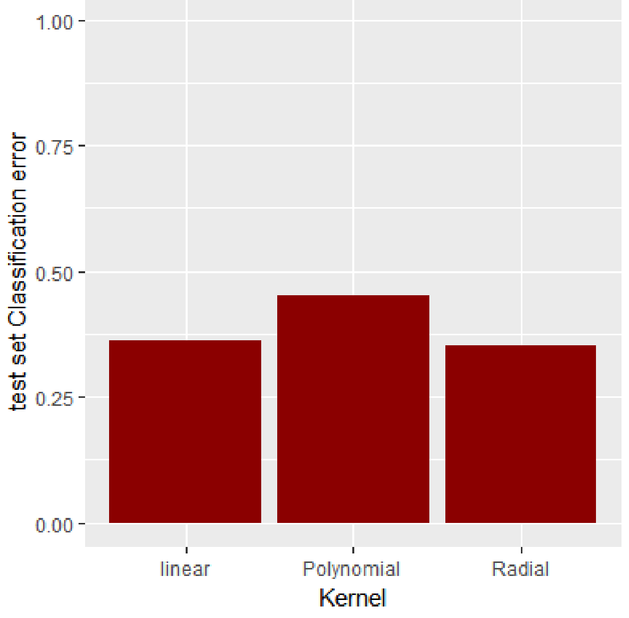
We can see that radial kernel performs best both in terms of the error rate, but also in terms of the time taken to converge. Polynomial kernel performed the worst in terms of error, which may be because I used a 6th degree kernel. Experimenting with this may have improved the model.



```
      Kernel trainError testError   seconds
1     linear  0.3545690 0.3620922 751.1626
2 Polynomial  0.4241496 0.4520686 495.9807
3     Radial  0.3319040 0.3523377 491.5492
```

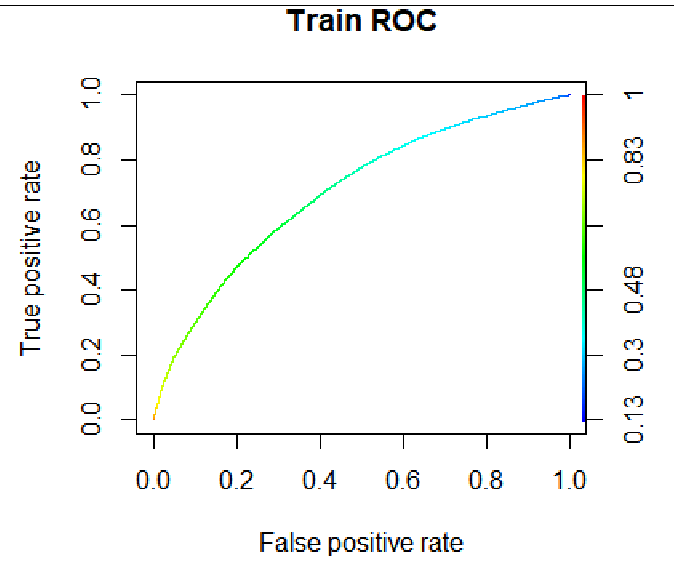
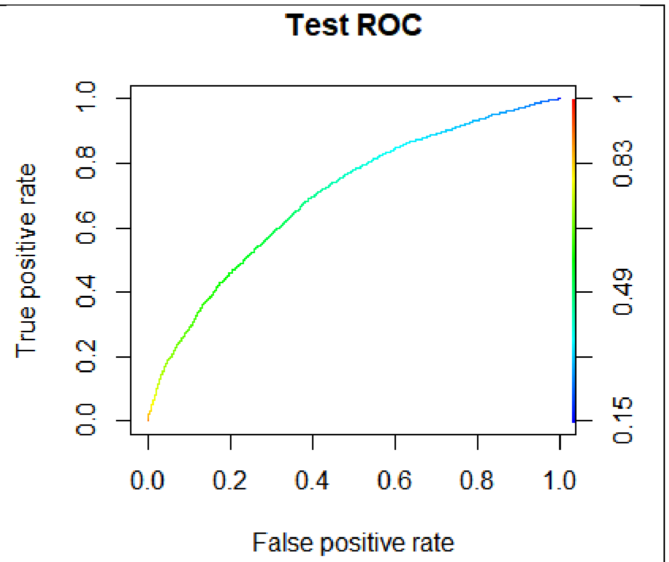
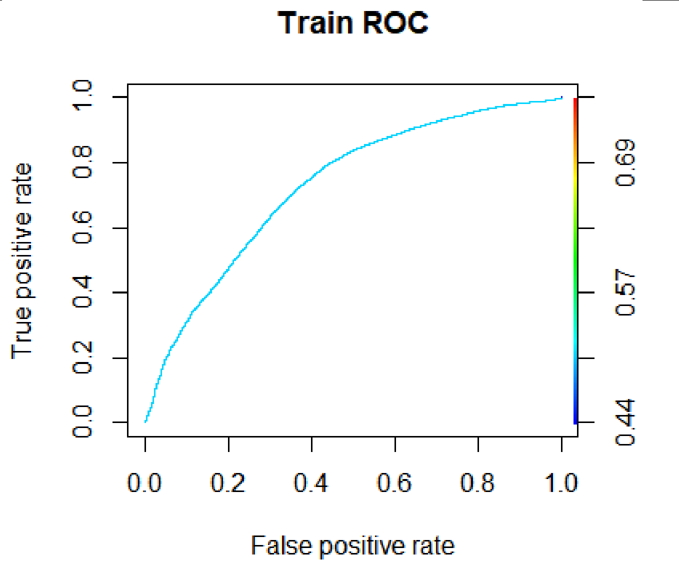
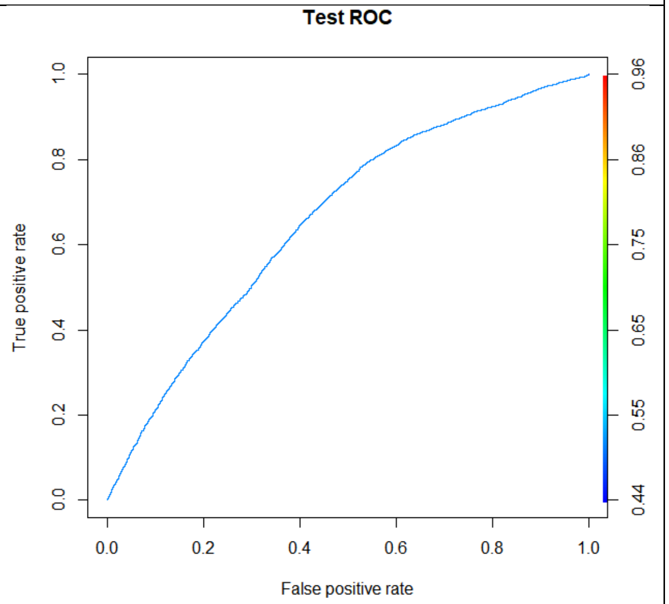
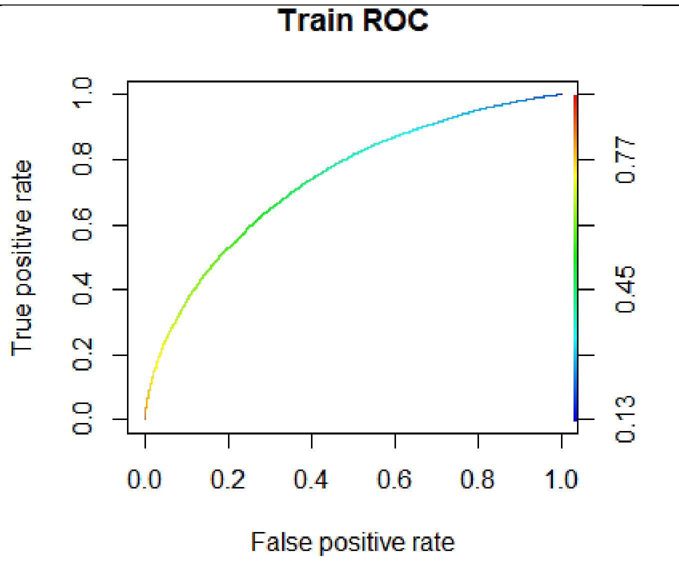
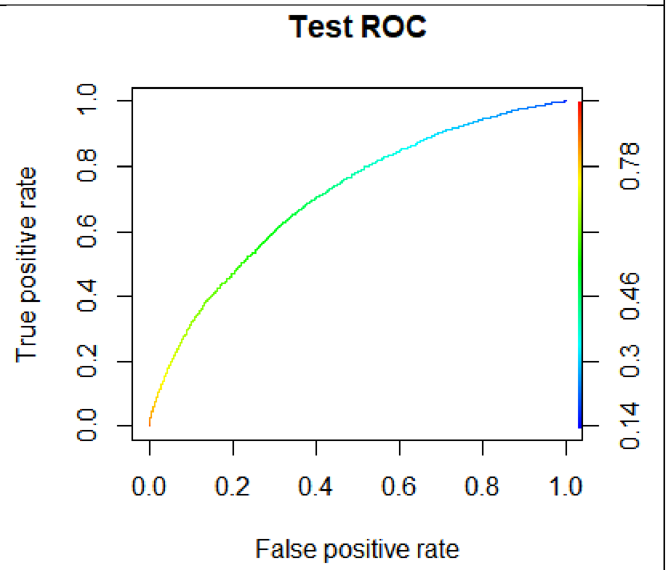






axa171831

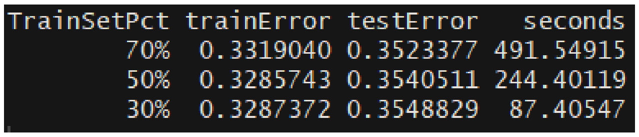
The ROC curves for all kernels are given below. We can see that Polynomial kernel has the least area under the curve and hence it performs worst.

Linear		
Polynomial		
Radial		

axa171831

Experimenting with training size set

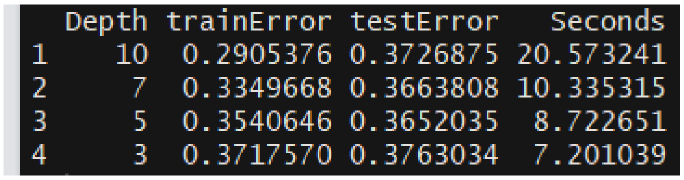
I tried changing the percentage of data in training size set to see if that causes any issues. The results are summarized to the right. We can see that because the data is so large, we can use a smaller dataset to save some time and still not lose any accuracy level.



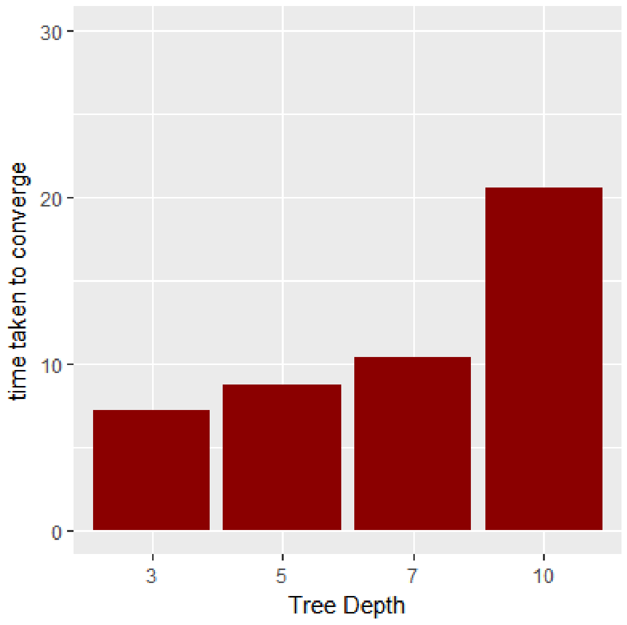
```
TrainSetPct trainError testError   seconds
        70%  0.3319040 0.3523377 491.54915
        50%  0.3285743 0.3540511 244.40119
        30%  0.3287372 0.3548829  87.40547
```

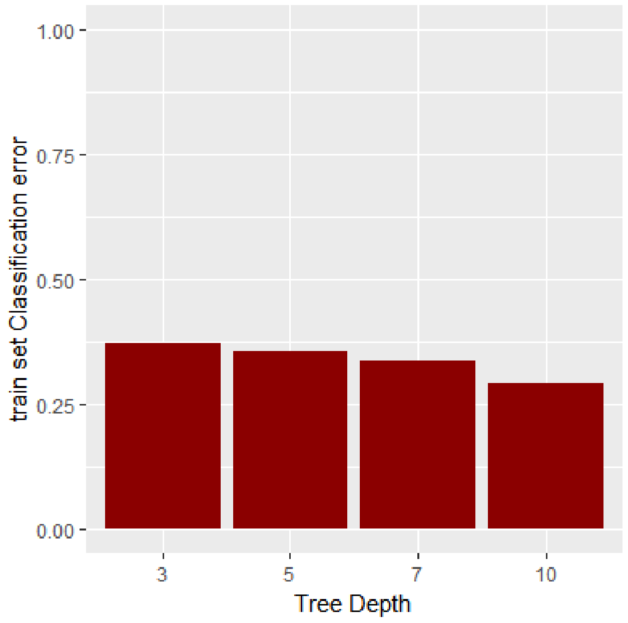
Classification Tree

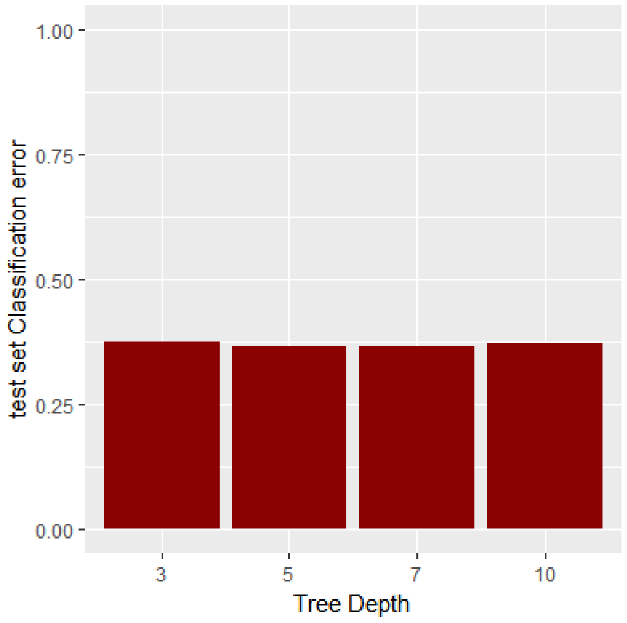
I trained different trees at varying depth starting with a depth of 10. The results are summarized below. We can see that there is a big difference between train and test error of the tree with depth 10. This is possible because this tree is overfitting the data too much. Trees with depth 5 and 7 do best. The splits were made on the basis of Gini index.



```
  Depth trainError testError   Seconds
1    10  0.2905376 0.3726875 20.573241
2     7  0.3349668 0.3663808 10.335315
3     5  0.3540646 0.3652035  8.722651
4     3  0.3717570 0.3763034  7.201039
```

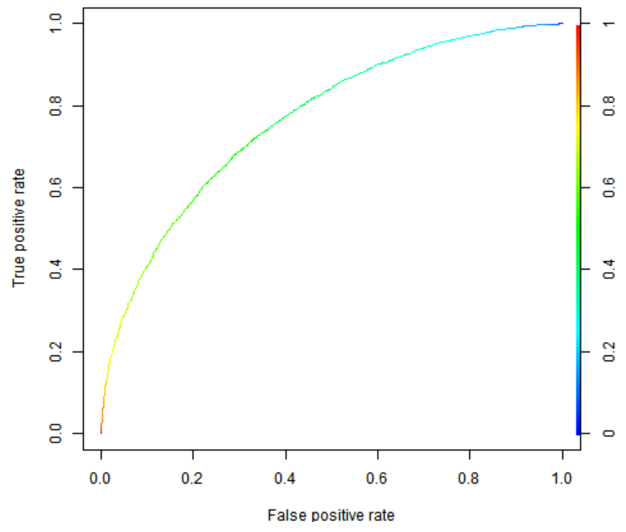
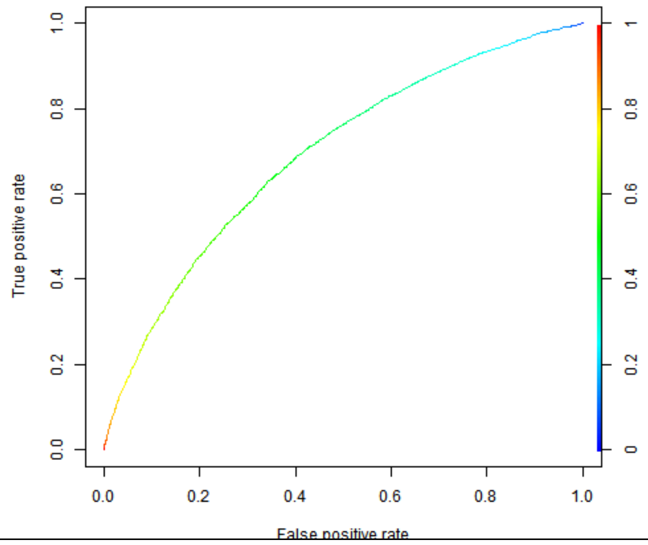
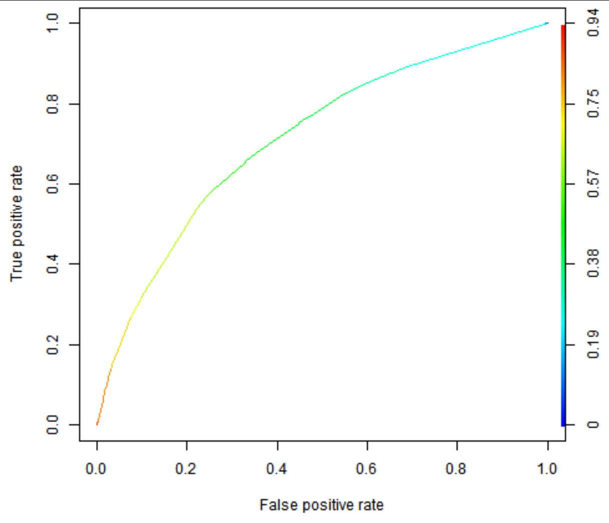
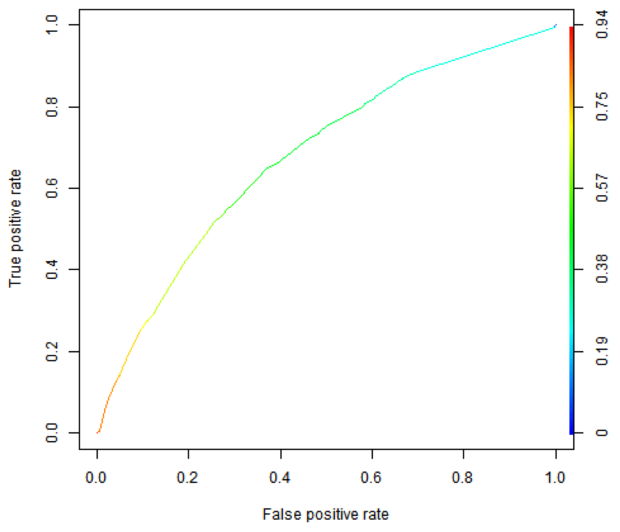
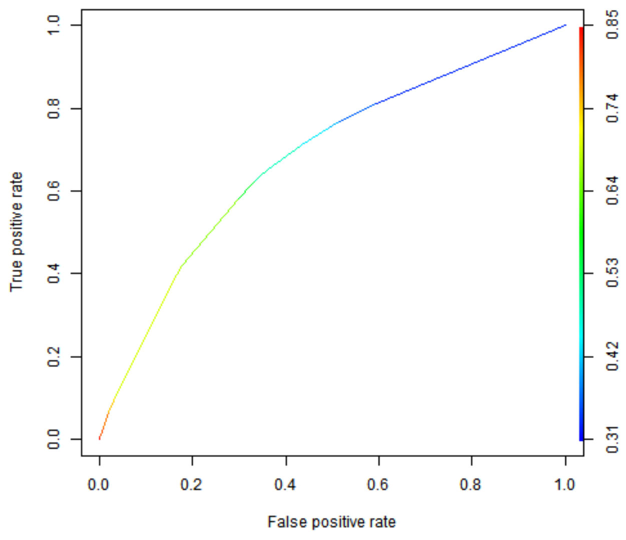
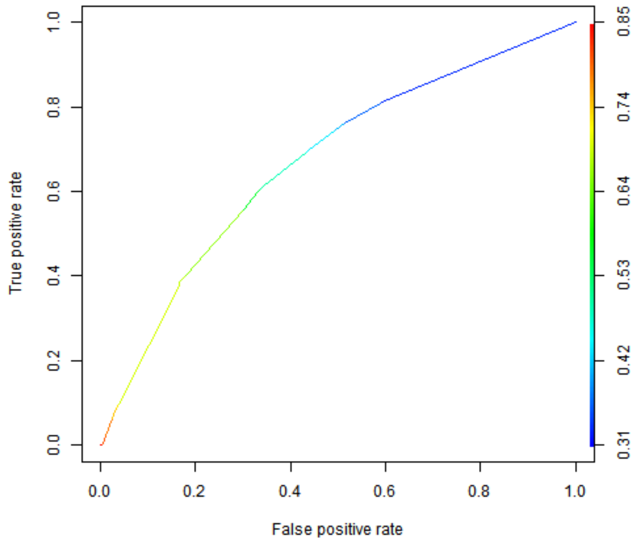






The ROC curves for all tree depths are given below. We can see that the ROC curve on test set does worse compared to train ROC curve.

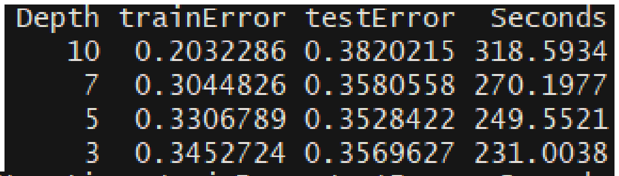
axa171831

Depth	Train ROC Curve	Test ROC Curve
10		
7		
5		

axa171831

Boosted Trees

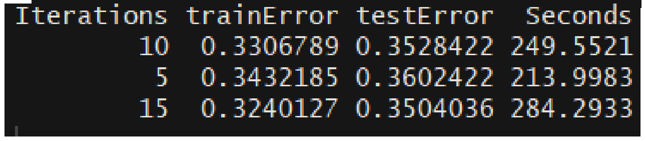
I tried boosted trees with different depths. Again, trees with a depth of 10 showed signs of overfitting. A depth of 5 was generalizing best.



```
Depth trainError testError   Seconds
   10  0.2032286 0.3820215 318.5934
    7  0.3044826 0.3580558 270.1977
    5  0.3306789 0.3528422 249.5521
    3  0.3452724 0.3569627 231.0038
```

Experimenting with Boosting iterations

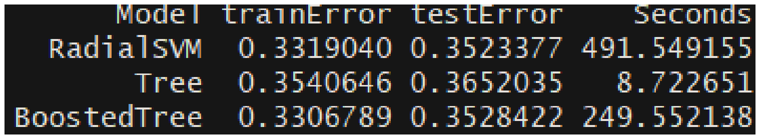
Next, I tried changing the number of iterations in boosting. Reducing the number of iterations from 10 to 5 reduced the accuracy considerably but increasing to 15 did not improve the model too much.



```
Iterations trainError testError  Seconds
        10  0.3306789 0.3528422 249.5521
         5  0.3432185 0.3602422 213.9983
        15  0.3240127 0.3504036 284.2933
```

Comparison of models

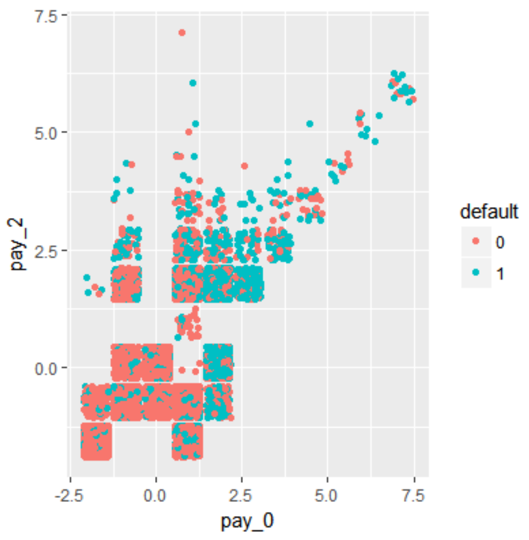
Boosted trees perform best but they take significantly longer than normal trees. It boils down to whether the extra bit of accuracy is worth the time.



```
      Model trainError testError    Seconds
  RadialSVM  0.3319040 0.3523377 491.549155
       Tree  0.3540646 0.3652035   8.722651
BoostedTree  0.3306789 0.3528422 249.552138
```

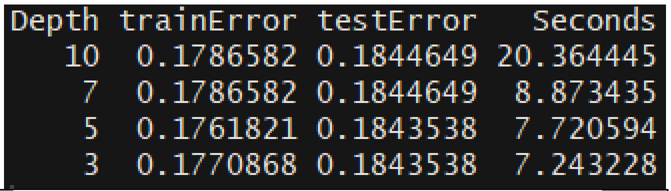
Dataset2: Credit card default

The dataset consists of default status of 30,000 individuals (~6600 defaulters) with attributes related to the total amount of loan, payment delays, gender, education, age, marital status and the amount billed/paid. While there isn't enough space here to reproduce all the descriptive, one graph was somewhat interesting. The chart shows the number of months payment is delayed for the months of September and august 2005. There is some separation in the graph between defaulters and non-defaulters which I hope to capture in my models. We will see if the models can capture more separation than the chart shown towards the right by using other variables in the model.

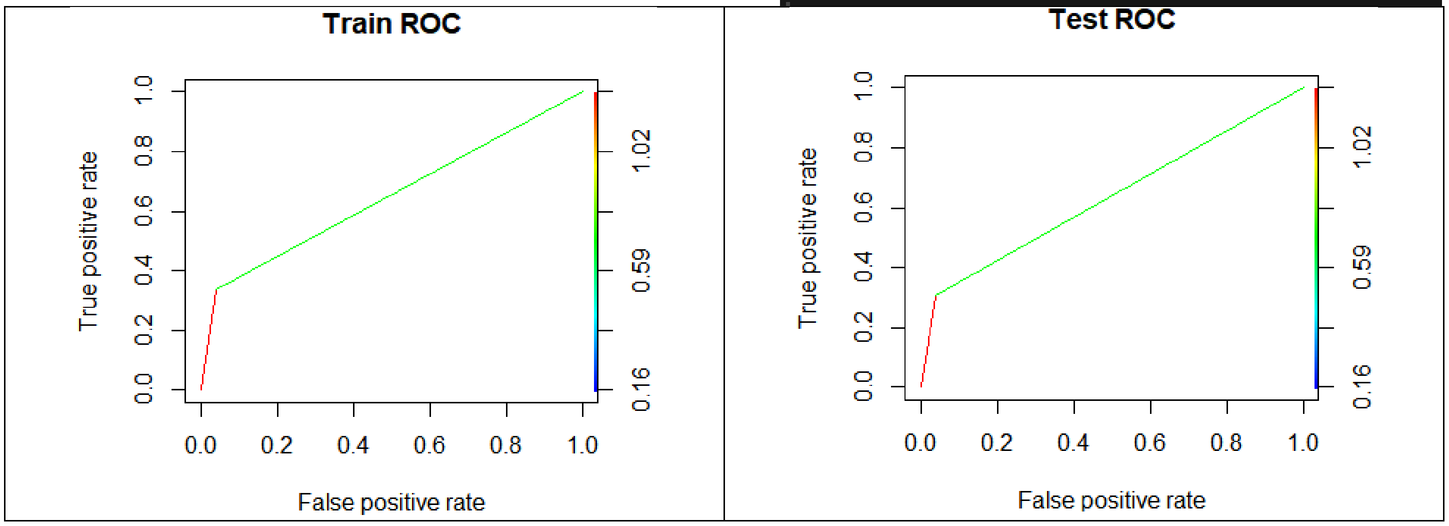


Classification Trees

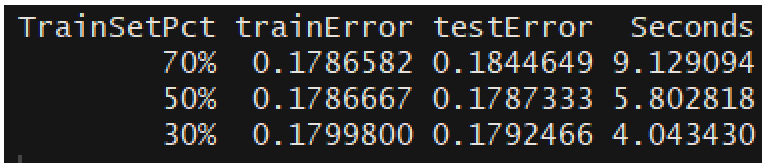
Classification trees showed an interesting behavior with the dataset. No matter how much I tried coercing them into making complex trees, the result was a tree with just one split. This is possibly because of the chart shown to the top right. The algorithm just splits based on pay_0 and doesn't deem suitable to make another split. It is preferring a shorter tree. The ROC curves are oddly shaped because of just one split as well



```
Depth trainError testError   Seconds
   10  0.1786582 0.1844649 20.364445
    7  0.1786582 0.1844649  8.873435
    5  0.1761821 0.1843538  7.720594
    3  0.1770868 0.1843538  7.243228
```

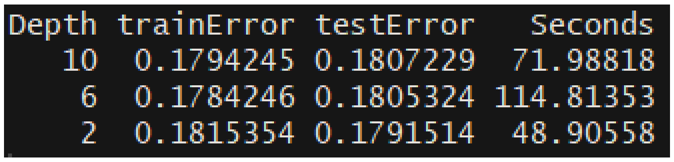


Another odd thing is that reducing the training set observation actually reduced the test set error rate slightly! I have seen such behavior for the first time. My guess would be that the algorithm is overfitting too much on the larger train set.


```
TrainSetPct trainError testError   Seconds
        70%  0.1786582 0.1844649 9.129094
        50%  0.1786667 0.1787333 5.802818
        30%  0.1799800 0.1792466 4.043430
```

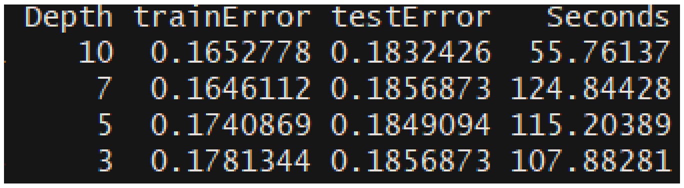
A different library?

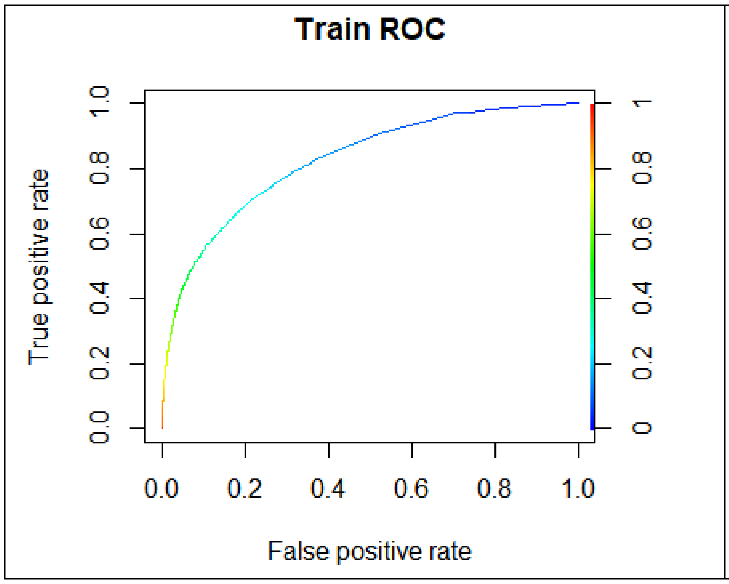
I also tried a different library (evtree) which in my experience uses as many variables it can to create models. Using that library didn't help the accuracy too much either.

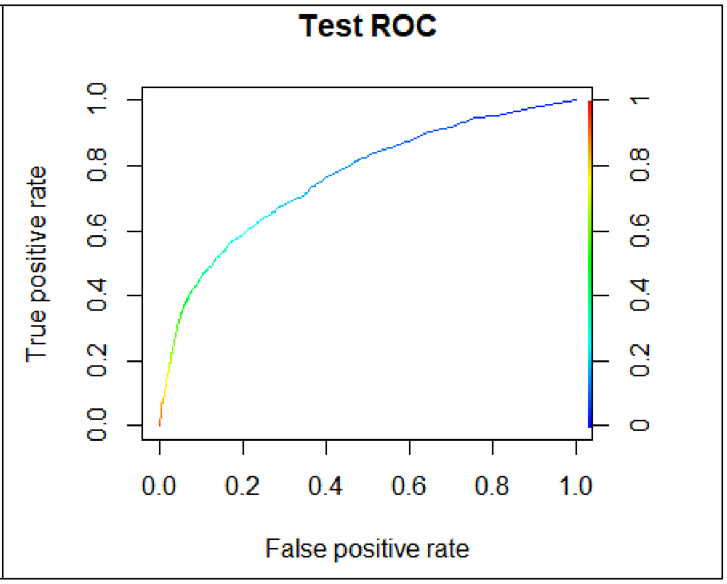

```
Depth trainError testError   Seconds
   10  0.1794245 0.1807229  71.98818
    6  0.1784246 0.1805324 114.81353
    2  0.1815354 0.1791514  48.90558
```

Boosted Trees

Interestingly, boosted trees were also not better at classification than normal trees. Boosted trees have more normal looking ROC curves though which possibly means that boosting is making use of other variables in the dataset too.


```
 Depth trainError testError   Seconds
    10  0.1652778 0.1832426  55.76137
     7  0.1646112 0.1856873 124.84428
     5  0.1740869 0.1849094 115.20389
     3  0.1781344 0.1856873 107.88281
```

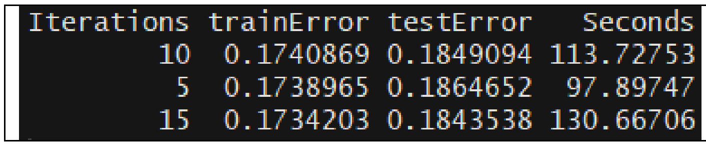


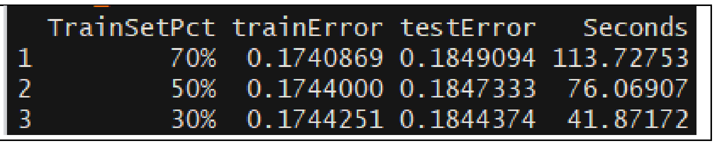


There is no significant difference in the training and testing set errors for boosted trees of varying depth.

Changing the number of boosting iterations and train set size

Changing the boosting iterations doesn't change much in term of accuracy. This is because the splits are still made on the basis of one or two variables. Choosing a smaller training set does affect it either.

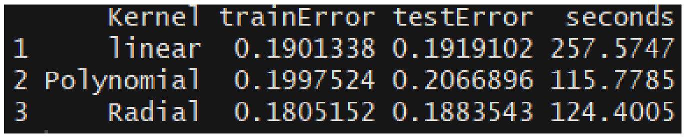

```
Iterations trainError testError   Seconds
        10  0.1740869 0.1849094 113.72753
         5  0.1738965 0.1864652  97.89747
        15  0.1734203 0.1843538 130.66706
```


```
  TrainSetPct trainError testError   Seconds
1         70%  0.1740869 0.1849094 113.72753
2         50%  0.1744000 0.1847333  76.06907
3         30%  0.1744251 0.1844374  41.87172
```

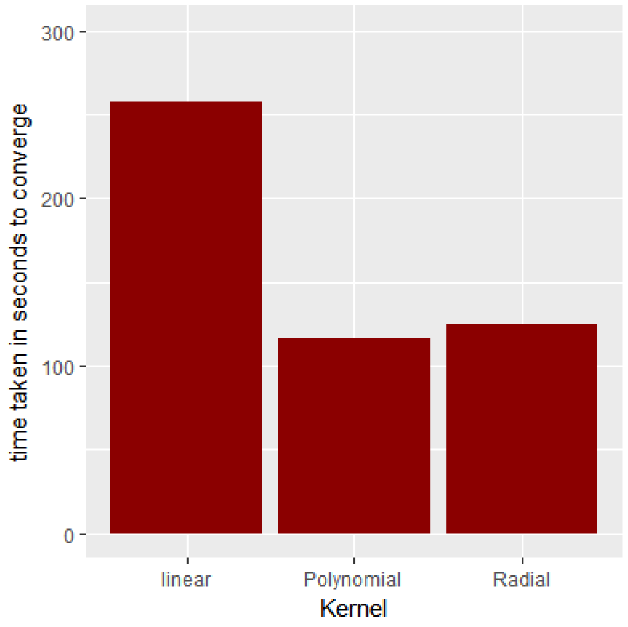
Can SVM do better?

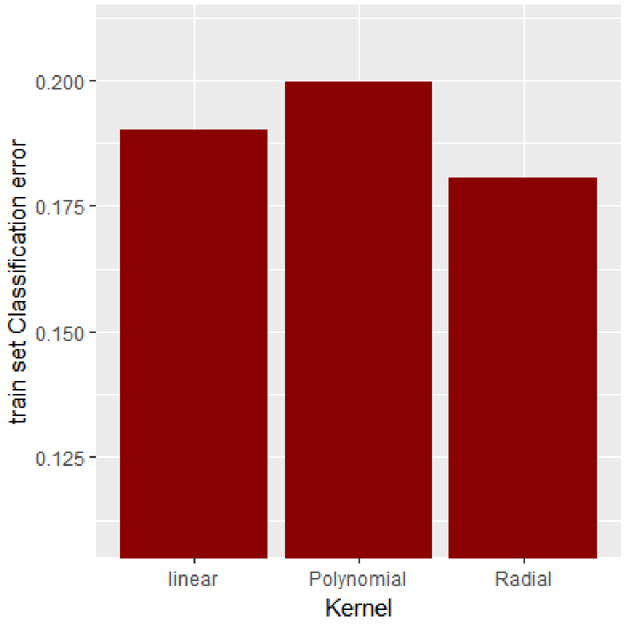
The three SVM kernels chosen were linear, radial and polynomial. Unlike decision trees, there was some difference between the three. Radial kernel again performed the best and polynomial the worst. Linear kernel again takes a lot of computational effort.

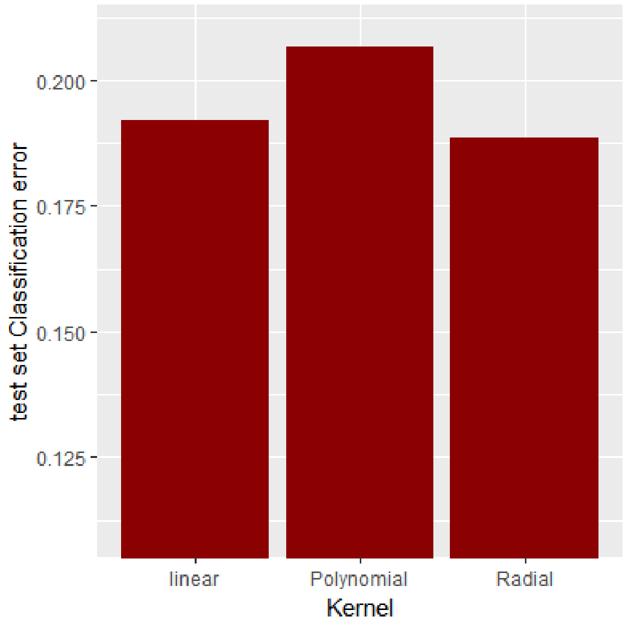
axa171831



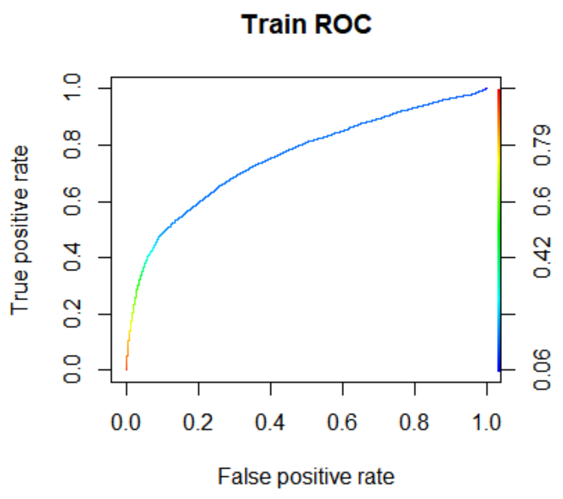
```
     Kernel  trainError  testError   seconds
1    linear   0.1901338  0.1919102  257.5747
2 Polynomial  0.1997524  0.2066896  115.7785
3    Radial   0.1805152  0.1883543  124.4005
```

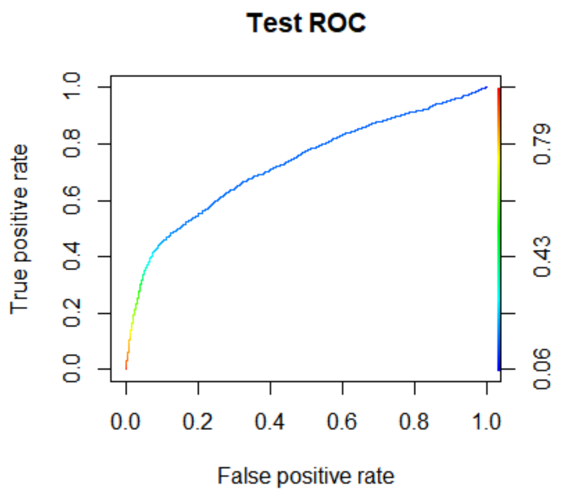






ROC curves for Radial kernel are given below

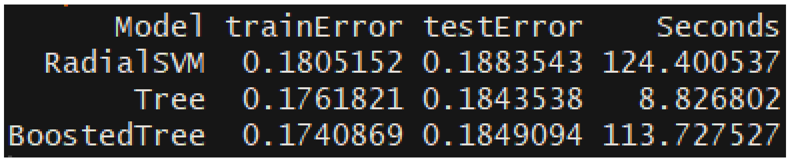




axa171831

Comparing different techniques

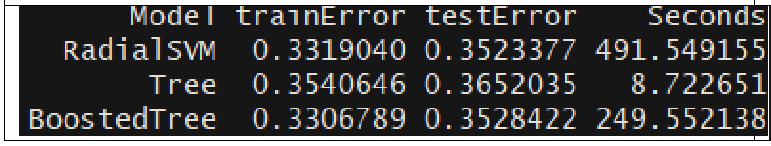
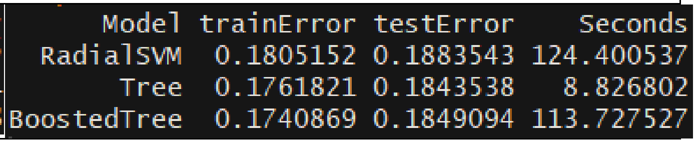
All three techniques are very close in terms of test set accuracy. Trees are more efficient in terms of computational power used.



```
       Model trainError testError    Seconds
   RadialSVM  0.1805152 0.1883543 124.400537
        Tree  0.1761821 0.1843538   8.826802
 BoostedTree  0.1740869 0.1849094 113.727527
```

Comparing the two datasets

The second dataset is much easier to predict with one or two variables making very good separation in the data. This is because of the business problem analyzed. People who are going to intentionally or unintentionally default on their credit card mostly do in the first few months. The dataset also tells us that it is possible for real life data to be classified with relative ease and we don't need very complicated models for it. I even tried a different real life dataset and I faced the same "problem" of accurate predictions with just one predictor variable. The dataset is also about after-the-fact. If we had data about things we do not know after the credit card approval and rather we have data about the attributes that are used predict default before a decision about a loan/credit approval is made, we would have a more complicated model.

Online News	Credit Card
	

```
       Model trainError testError    Seconds
   RadialSVM  0.3319040 0.3523377 491.549155
        Tree  0.3540646 0.3652035   8.722651
 BoostedTree  0.3306789 0.3528422 249.552138
```

```
       Model trainError testError    Seconds
   RadialSVM  0.1805152 0.1883543 124.400537
        Tree  0.1761821 0.1843538   8.826802
 BoostedTree  0.1740869 0.1849094 113.727527
```

The online news data is much more compute-intensive because of the inherent uncertainty of the business. We simply cannot know if some content on sites such as mashabale are going to be successful or not. That is the nature of social media. I have seen the exact comments posted on reddit.com with a difference of just a few minutes and the response to those comments were completely opposite. One person's comment was lauded while the other's comment was removed due to negative feedback.