Classification on Collegiate Ranking Data

Comparing Classification Algorithms for Predicting Improvement in College Rank

Ian Johnson

Southern Methodist University

October 16, 2016

Contents

Executive Summary

This report explores the classification task of identifying which universities will increase in collegiate rankings based on their performance scores over time. A number of classification algorithms are used, including decision trees, K-nearest-neighbors, and both linear and non-linear support vector machines. Random forests are used to identify the most important variables in the classification task, and it is determined that the most recent performance scores are the strongest prediction of improvement scores. The best classifier produced is a non-linear support vector machine with a Gaussian (RBF) kernel.

1 Data Preparation

For the purpose of our analysis, we will propose the following problem:

Given the rankings and scores of universities in 2005 and 2015, predict whether or not the universities will improve in overall world ranking between those two years.

In order to classify schools as improving or not, a number of additions and transformations must occur. First, most obviously, each university needs to be assigned an actual (measured) value of improvement. Subsequently, a subset of the attributes from the THE, Shanghai, and CWUR datasets must be chosen to be used for the classifier [1],[2],[3]. Finally, missing data must be imputed or removed.

1.1 Directly Measuring Improvement

In order to directly measure the improvement of a university, we define improvement as a decrease in Shanghai world rank from 2005 to 2015. Each university which appears in the Shanghai dataset in either 2005 or 2015 is added to a table of universities which includes columns of the university's Shanghai scores from both 2005 and 2015. The world rank scores of the university and 2005 and 2015 are then compared to see if the ranking of the university improved over the 10-year period. The improvement (or lackthereof) of a university is recorded in an appended column in the matrix, called "improved," which is encoded as TRUE or FALSE and will be used as a class label for our classification algorithms.

1.2 Selecting a Subset of Attributes for Classification

Because we choose to isolate the years 2005 and 2015, the Shanghai dataset is an obvious choice of attributes to perform classification on, as it includes data from both of those years. The THE dataset includes some additional data which includes direct measurements of various attributes of universities such as number of students and student-to-staff ratio. These data columns are deliberately excluded, so that classification can be performed strictly based on actual scores. By excluding directly measured measurements, we can classify based on perceived quality of a university without arbitrarily defining a directly measured attribute as being good or bad.

1.3 Missing Data

By merging the Shanghai datasets from 2005 and 2015, we introduce a number of universities which occur in one year, but not in the other. Imputing data for such universities would likely produce reasonable results for the non-class attributes of each university. However, all such universities will be excluded because imputing the class attribute (improved) of each university could significantly skew our model, as well as the perceived accuracy of our model.

1.4 Dataset Used for Classification

The following columns are included in the final classification datset. All columns are in the table twice (once for 2005 and once for 2015), with the exception of *university_name* and *improved.*

Attribute	Data Scale	Description
university_name	*nominal*	The name of the university
total_score	*ratio*	The Shanghai Ranking total score, used for ranking
alumni	*ratio*	Alumni score based on the number of alumni winning nobel prizes and fields medals
award	*ratio*	Metric for the number of staff winning nobel prizes and fields medals
hici	*ratio*	Metric for the number of highly-cited researchers at the university
ns	*ratio*	Metric for the number of papers published in *Nature and Science*
pub	*ratio*	Metric for the number of papers indexed in *Science Citation Index-Expanded* and *Social Science Citation Index*
pcp	*ratio*	Weighted scores of above five indicators, divided by number of full time academic staff
year	*interval*	The year that this ranking occurred
improved	*nominal*	Whether or not the university improved its ranking between 2005 and 2015

2 Modeling

We create five different models from five different classification algorithms. Each algorithm is tuned with hyper-parameters in an effort to produce the best model. To evaluate each model, a random subsample of the dataset will be used to train the model and the remaining data (a disjoint subset) will be used to evaluate the model. Because we're using a static dataset (no more data will be added in the future that we need to classify, since this is all historical data), this novel cross-validation strategy is reasonable, because our best metric for evaluating model performance is measuring how well it classifies the data we already have.

2.1 Decision Tree Classifier

Our first novel approach to building a classifier is to use a decision tree algorithm. Figure 2.1 shows the tree. Each node in the tree represents a 'decision' in the classifier. As a new item enters the decision tree classifier, it will be cascaded down the tree, being checked at every node. When it reaches a leaf node, the tree assigns it a class. Decision tree classification is performed using the CRAN package "caret" [4].

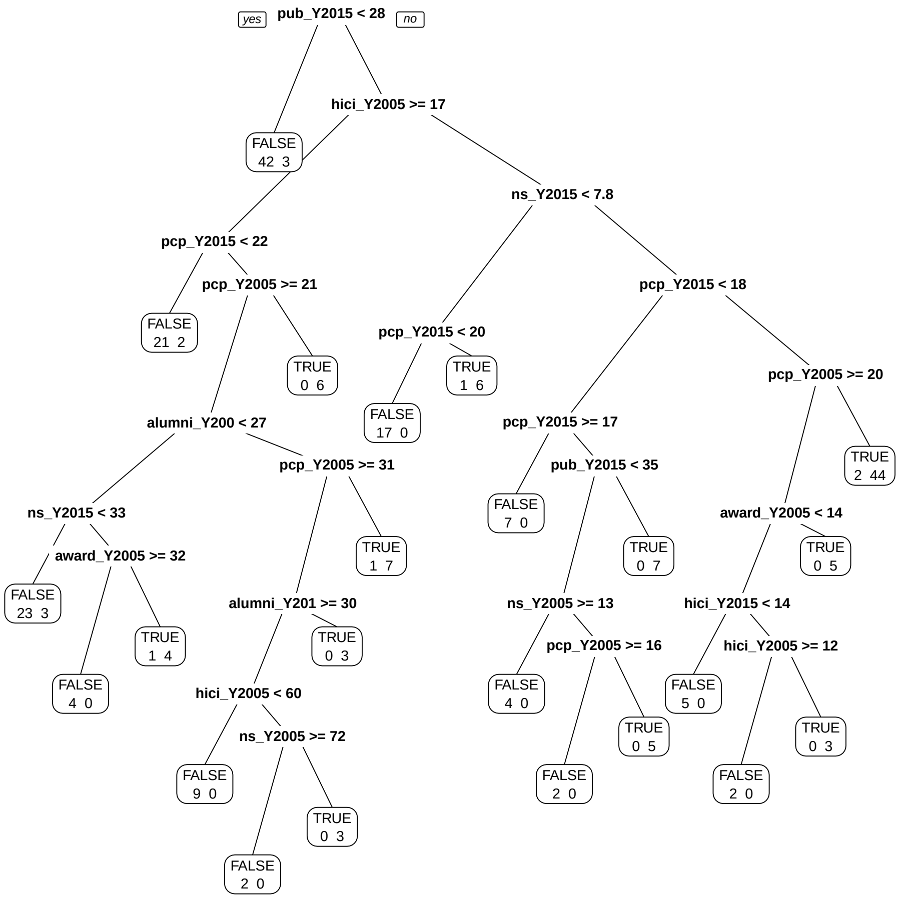

Figure 2.1 - A decision tree to classify university rank improvement

Each of the leaf nodes in Figure 2.1 shows the split of improvement scores at the given leaf of the tree. The strongest possible classifier will have a complete split at each leaf (all of the values will either be true or false at any given leaf node). However, if there is a complete split at each leaf, it's very possible that the tree was overfit to the training dataset. Hyperparameters were supplied to this tree to prevent such an occurence.

```
                   Overall
alumni_Y2005      2.986667
alumni_Y2015      3.050420
award_Y2005       6.111111
hici_Y2005       14.900559
hici_Y2015        1.800000
ns_Y2005          4.997403
ns_Y2015         10.478300
pcp_Y2005        14.722430
pcp_Y2015        19.900335
pub_Y2015        15.853314
```

```
total_score_Y2005  0.000000
pub_Y2005          0.000000
total_score_Y2015  0.000000
award_Y2015        0.000000
```

Figure 2.2 - The variable importances for the decision tree in Figure 2.1

Figure 2.2 shows that the most important factors for the decision tree are pcp and pub from 2015. (These are the overall Shanghai scores and the publication scores, respectively). Interestingly, it appears that in general, the 2015 scores are much more important in the decision tree. The decision tree algorithm seems, therefore, to indicate that the most important data in identifying if a school has improved is to look at its most recent scores.

To evaluate the strength of the decision tree model, we create a confusion matrix and evaluate a number of model strength statistics in Figure 2.3

```
Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE    53   18
     TRUE     31   33

               Accuracy : 0.637
                 95% CI : (0.5499, 0.718)
    No Information Rate : 0.6222
    P-Value [Acc > NIR] : 0.39768

                  Kappa : 0.2648
 Mcnemar's Test P-Value : 0.08648

            Sensitivity : 0.6471
            Specificity : 0.6310
         Pos Pred Value : 0.5156
         Neg Pred Value : 0.7465
             Prevalence : 0.3778
         Detection Rate : 0.2444
   Detection Prevalence : 0.4741
      Balanced Accuracy : 0.6390

       'Positive' Class : TRUE
```

Figure 2.3 - Evaluation of the decision tree model

Figure 2.3 shows that the decision tree produced 31 type I errors and 18 type II errors. It had an accuracy of .637 (note that accuracy *is* a meaningful metric in this scenario, because we have a near-even split of classes in the dataset), which leaves lots of room for improvement in forthcoming models. The kappa value of the model is 0.2648, which is far from optimal. However, this decision tree is a good starting point for classification.

2.2 Association Rule Based Classifier

Our second approach to classification will be a rule-based classifier. We will use an association rule based classification algorithm. Association rule-based classification is typically strongest for high-dimensional data, which this dataset is not; however, we will evaluate it nonetheless, because it stands to reason that certain pairings of 2005 and 2015 scores result in improvement or lackthereof. Association rule-based classifiers excel at identifying trends such as these. In order to build such a model, all data must be discretized. For the model shown in Figure 2.4, every data column was discretized using frequency-based discretization from CRAN package "arules" [6]. Association rule-based classification will be performed using CRAN package "arulesCBA" [5].

```
    lhs                          rhs                support confidence     lift
[1] {pub_Y2005=[10.1, 34),
     award_Y2015=0.0,
     ns_Y2015=[ 0.0, 12.2),
     pub_Y2015=[ 7.8, 36.8)} => {improved=FALSE}     0.2  0.8333333 1.426941
```

Figure 2.4 - The rule base for the association rule classifier

Figure 2.4 shows the rule base for the association rule classifier. The first rule identifies *pub* scores from 2005 and *award*, *ns*, and *pub* scores from 2015 as important factors in the classification. This continues the trend set forth by the decision tree model where features from 2015 are generally more important for classification than features from 2005.

Unfortunately, the algorithm only produced a single rule for classification, and the rule has very little support (it represents only a small sample of the dataset). Therefore, the identified attributes are likely not particularly meaningful, and it is unlikely that the classifier will produce meaningful results. Nonetheless, the model is evaluated in Figure 2.5.

```
Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE    81   49
     TRUE      0    0

               Accuracy : 0.6231
                 95% CI : (0.5339, 0.7065)
    No Information Rate : 0.6231
    P-Value [Acc > NIR] : 0.539

                  Kappa : 0
 Mcnemar's Test P-Value : 7.025e-12

            Sensitivity : 1.0000
            Specificity : 0.0000
         Pos Pred Value : 0.6231
         Neg Pred Value :    NaN
             Prevalence : 0.6231
         Detection Rate : 0.6231
   Detection Prevalence : 1.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : FALSE
```

Figure 2.5 - Evaluation of the association rule model

Figure 2.5 shows that, as expected, the association rule classification model is completely worthless. While it has an accuracy of .6231, it has a kappa value of 0 because it assigns the same prediction to every single new data entry. This is not unexpected for a low-dimensional dataset such as the one we use for analysis.

2.3 K-Nearest Neighbors Classifier

The third classification model will be built using the K-nearest-neighbors algorithm, which performs classification by calculating the euclidian distance from each new data entry to each training entry in an N-dimensional space, where N is the number of columns in the training data. It then takes the K nearest neighbors of the new data entry and assigns it a class based on the mode of the classes of the training data. Note that all attributes are normalized before using KNN, to prevent one attribute from artificially carrying more weight than another due to differing dynamic ranges. Figure 2.6 shows an evaluation of the KNN model for K=3.

The KNN classifer is built using CRAN package ”class” [7].

```
Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE    56   16
     TRUE     24   31

               Accuracy : 0.685
                 95% CI : (0.5967, 0.7645)
    No Information Rate : 0.6299
    P-Value [Acc > NIR] : 0.1154

                  Kappa : 0.3474
 Mcnemar's Test P-Value : 0.2684

            Sensitivity : 0.7000
            Specificity : 0.6596
         Pos Pred Value : 0.7778
         Neg Pred Value : 0.5636
             Prevalence : 0.6299
         Detection Rate : 0.4409
```

```
  Detection Prevalence : 0.5669
     Balanced Accuracy : 0.6798

      'Positive' Class : FALSE
```

Figure 2.6 - Evaluation of the association rule model

Figure 2.6 shows that the KNN model made 24 type I errors and 16 type II errors, for an accuracy of 0.685 and a kappa of 0.3474. This noticeably outperforms the decision tree model, but still leaves significant room for improvement. Interestingly, both the KNN model and the decision tree model produced many more Type I errors than type II errors. This misclassification trend may be of interest if it continues in the subsequent models.

Because KNN has a very explicit hyper-parameter, K, it is meaningful to compare the results of the classifier with different values of K. Figure 2.7 shows the accuracy and kappa scores of the KNN model for a few values of K. Note that all values of K are odd, because an even value of K could result in a random class assignment (calculating the mode of an even number of elements for binary classification can lead to 50-50 splits).

K	Accuracy	Kappa
1	0.693	0.3717
3	0.708	0.4089
5	0.685	0.3474
7	0.669	0.2907
9	0.677	0.3224

Figure 2.7 - Accuracy and kappa for various values of K

Although KNN does not provide an explicit way of measuring the importance of any particular variable, we can still evaluate the working theory that the 2015 attributes are more important than the 2005 attributes by performing KNN on the dataset with only the 2015 attributes. Figure 2.8 shows the results of performing KNN with a K value of 5 with only the 2015 attributes.

```
Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE    52   16
     TRUE     28   31

               Accuracy : 0.6535
                 95% CI : (0.564, 0.7357)
    No Information Rate : 0.6299
    P-Value [Acc > NIR] : 0.32531

                  Kappa : 0.2941
 Mcnemar's Test P-Value : 0.09725
```

```
            Sensitivity : 0.6500
            Specificity : 0.6596
         Pos Pred Value : 0.7647
         Neg Pred Value : 0.5254
             Prevalence : 0.6299
         Detection Rate : 0.4094
   Detection Prevalence : 0.5354
      Balanced Accuracy : 0.6548

       'Positive' Class : FALSE
```

Figure 2.8 shows that excluding the 2005 data from the improvement classification for KNN only reduces accuracy to 0.6535 (from 0.685) and kappa to 0.2363 (from 0.3474). Clearly, the 2005 data carries some weight in the KNN classifier. However, the KNN classification still outperforms the decision tree classifier even without any of the 2005 data. This supports the claim that the 2015 attributes are more relevant to classification than the 2005 attributes for this classification task.

2.4 Linear Support Vector Machine Classifier

Our next classification model will be built using a linear SVM (support vector machine). A linear SVM attempts to, in N dimensions, draw a linear N-1 dimensional boundary which separates the data into two classes. SVMs have an important hyper-parameter called C, or cost, which defines how strongly the SVM will fit to the training data. For low values of C, the SVM will try to build a highly generalized model, while high values of C will build a model strongly tuned to the intricacies (or noise) of the training data.

The linear SVM classifer is built using CRAN package ”kernlab” [8].

```
 Setting default kernel parameters

Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE    74   21
     TRUE      6   26

               Accuracy : 0.7874
                 95% CI : (0.706, 0.855)
    No Information Rate : 0.6299
    P-Value [Acc > NIR] : 9.908e-05

                  Kappa : 0.5119
 Mcnemar's Test P-Value : 0.007054

            Sensitivity : 0.9250
```

```
         Specificity : 0.5532
      Pos Pred Value : 0.7789
      Neg Pred Value : 0.8125
          Prevalence : 0.6299
      Detection Rate : 0.5827
Detection Prevalence : 0.7480
   Balanced Accuracy : 0.7391

    'Positive' Class : FALSE
```

Figure 2.9 - Evaluation of the linear SVM model

Figure 2.9 shows that the linear SVM produced 6 type I errors and 21 type II errors for an accurracy of 0.7874 and a kappa value of 0.5119. This is by far the best-performing classifier we've built so far. Moreover, it ended the trend of producing a high number of type I errors, which was an issue for the previous classifiers. This particular SVM model was produced with a C value of 0.01; however, C is an explicit hyper-parameter that can be tuned.

Figure 2.10 shows the accuracy and kappa values of the models generated using a set of costs (C values). As expected, higher cost values generally lead to lower accuracies, because the resulting SVM is overfitted to the training data.

Cost	Accuracy	Kappa
0.01	0.787	0.5119
0.05	0.756	0.4499
0.1	0.748	0.4295
0.5	0.756	0.4448
1.0	0.748	0.4295
5.0	0.748	0.4295

Figure 2.10 - Accuracy and kappa for various values of C

Much like KNN, SVMs allow very little insight into the importance of the variables in the dataset for the classification task. However, an additional variable importance metric will be introduced later, following a look at non-linear SVMs for classification.

2.5 Non-Linear Support Vector Machine Classifier

SVMs can be improved upon by adding non-linearity to the dividing 'line.' Various kernel methods allow for the projection of the dataset into higher-dimensional space, where non-linear differences in the original N-dimensional space may become linearly separable. A number of different SVM kernels exist, the most popular being the RBF kernel. Different kernel types will be explored more in later secions. After the data has been projected into additional dimensions, a non-linear N-dimensional decision function can be built which accepts new N-dimensional data and assigns it a class. Overfitting is a large concern with non-linear SVMs, especially with the RBF kernel. The RBF kernel can project data into an infinite-dimensional space, allowing for perfect classification of the training set. However, this will generate a highly noisy decision function which is tightly fitted to minor intricacies of the training set.

```
Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE    74   12
     TRUE      6   35

               Accuracy : 0.8583
                 95% CI : (0.7853, 0.9138)
    No Information Rate : 0.6299
    P-Value [Acc > NIR] : 1.006e-08

                  Kappa : 0.6878
 Mcnemar's Test P-Value : 0.2386

            Sensitivity : 0.9250
            Specificity : 0.7447
         Pos Pred Value : 0.8605
         Neg Pred Value : 0.8537
             Prevalence : 0.6299
         Detection Rate : 0.5827
   Detection Prevalence : 0.6772
      Balanced Accuracy : 0.8348

       'Positive' Class : FALSE
```

Figure 2.11 - Evaluation of the non-linear SVM model

Figure 2.11 shows that the non-linear SVM model yields 7 type I errors and 12 type II errors, for an accuracy of 0.8504 and a kappa value of 0.6719. Once again, this noticeably outperforms the previous models. This is expected, as SVMs with non-linear kernels are generally one of the most effective classification methods for many datasets and dataset sizes [9]. Once again, the type II errors outnumbered the type I errors for the non-linear SVM, so it seems that the trend in the first models toward type I errors was simply coincidental. The SVM evaluated in figure 2.8 was generated with a cost value of 5.

Figure 2.12 shows the accuracy and kappa values for various values of C (cost) for the RBF kernel SVM. For this kernel, it seems that there's a peak in accuracy and kappa around C = 5, and the accuracy and kappa decrease as C becomes greater than or less than 5.

Cost	Accuracy	Kappa
0.1	0.646	0.0530
0.5	0.780	0.5008
1	0.811	0.5911
5	0.850	0.6719
10	0.819	0.6099
50	0.803	0.5796

Figure 2.12 - Accuracy and kappa for various values of C

Because both linear and non-linear SVMs provide no insight into the importance of variables in the classification task, a supplimentary look into variable importance will be performed using random forests.

2.6 Random Forest Variable Importance Evaluation

A random forest importance measure of all variables is computed on the training portion of the dataset, and the results are shown in Figure 2.13. CRAN package ”randomForest” is used to create the random forest [10].

```
                  FALSE       TRUE MeanDecreaseAccuracy MeanDecreaseGini
alumni_Y2005  0.3939566  2.5824562             2.379334         4.229096
award_Y2005   9.6998752 -5.0685380             6.336197         3.533620
hici_Y2005   12.5835376 12.5896161            19.844724        13.072607
ns_Y2005      6.8646409 -1.1564982             5.835473         9.374086
pub_Y2005    11.9586938 -3.9829827             9.362715        10.793983
pcp_Y2005    11.8115391  6.5423973            15.541639        12.927753
alumni_Y2015  0.6463150  1.9962960             2.080835         5.200726
award_Y2015   7.0356634  0.7671450             7.396267         4.906967
hici_Y2015   11.5612008  0.8319753            11.752110        11.557287
ns_Y2015     19.0074145 -0.6251110            17.740425        14.815487
pub_Y2015    20.1146849  8.2147815            21.441046        16.786782
pcp_Y2015     9.8549678  1.8104305             9.724811        13.654336
```

Figure 2.13 - Variable importance evaluation using a random forest

Figure 2.13 shows 4 different scores for the importance of each variable (the weight of a FALSE value, the weight of a TRUE value, the effect on accuracy, and the effect on the Gini index). Figure 2.14 reduces Figure 2.13 into two items by taking the row means of each row, creating an aggregate score for each variable. Figure 2.15 takes the sum of the 2005 and 2015 variables from Figure 2.14 and compares them directly, showing that the 2015 attributes do, in fact, carry slightly more weight in this classification task than the 2005 attributes.

```
alumni_Y2005  award_Y2005   hici_Y2005     ns_Y2005    pub_Y2005    pcp_Y2005
    2.396211     3.625289    14.522621     5.229426     7.033102    11.705832
alumni_Y2015  award_Y2015   hici_Y2015     ns_Y2015    pub_Y2015    pcp_Y2015
    2.481043     5.026511     8.925643    12.734554    16.639324     8.761136
```

Figure 2.14 - Condensed variable importance evaluation using a random forest

Year	Weight
2005	45.38
2015	59.65

Figure 2.15 - Variable importance weights by year

3 Model Evaluation

The five models we produced provided insight into the classification task (identifying which universities will improve and which will not) and produced a maximum accuracy of 85% (using a non-linear support vector machine with the RBF kernel). We identify that 2015 scores are more important than 2005 scores in predicting whether or not a university improved between 2005 and 2015. This may be due to the way that Shanghai rankings were performed in those years, or this may be inherent to the generalized classification task of predicting improvement.

Stakeholders in the collegiate ranking problem may gain important insights from the classification process. **Students** looking at universities to attend can recognize that the most recent scores for universities outweigh the old ones considerably. Meanwhile, **governments** and **university leadership** know that their decision to invest in improving their universities should not be dependent on passed performance, because it's not a good predictor of whether or not the university will improve in the future.

4 A Deeper Look at Support Vector Machines

In section 2, we took advantage of the kernel method for SVMs to project our data into additional dimensions to produce a non-linear decision function for our SVM classifier. However, the only kernel we used was the RBF kernel, which is one of many popular non-linear SVM kernels. The following sections will explore some additional non-linear kernels for SVMs.

4.1 The Polynomial Kernel

The polynomial kernel uses polynomial functions to project the dataset into additional dimensions. Figure 3.1 shows the performance of the polynomial kernel on the improvement classification problem. Note that the same testing and training sets will be used for the additional SVM kernel classifiers as were used for the classifiers built in section 2.

```
 Setting default kernel parameters

Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE    70   22
     TRUE     10   25

               Accuracy : 0.748
                 95% CI : (0.6633, 0.8208)
    No Information Rate : 0.6299
    P-Value [Acc > NIR] : 0.003175

                  Kappa : 0.4295
 Mcnemar's Test P-Value : 0.051830
```

```
            Sensitivity : 0.8750
            Specificity : 0.5319
         Pos Pred Value : 0.7609
         Neg Pred Value : 0.7143
             Prevalence : 0.6299
         Detection Rate : 0.5512
   Detection Prevalence : 0.7244
      Balanced Accuracy : 0.7035

       'Positive' Class : FALSE
```

Figure 3.1 - Evaluation of the non-linear SVM model with a polynomial kernel

Figure 3.1 shows that the polynomial kernel yields 10 type I errors and 22 type II errors for an accuracy of 0.748 and a kappa value of 0.4295. While this outperforms the linear SVM, it is considerably less accurate than the RBF kernel.

4.2 The Hyperbolic Tangent Kernel

The hyperbolic tangent kernel transforms the data into a higher dimensional space using a hyperbolic tangent function. Figure 3.2 shows the performance of a non-linear SVM with the hyperbolic tangent kernel for this classification problem.

```
 Setting default kernel parameters

Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE    80   47
     TRUE      0    0

               Accuracy : 0.6299
                 95% CI : (0.5398, 0.7139)
    No Information Rate : 0.6299
    P-Value [Acc > NIR] : 0.5397

                  Kappa : 0
 Mcnemar's Test P-Value : 1.949e-11

            Sensitivity : 1.0000
            Specificity : 0.0000
         Pos Pred Value : 0.6299
         Neg Pred Value :    NaN
             Prevalence : 0.6299
         Detection Rate : 0.6299
   Detection Prevalence : 1.0000
```

```
      Balanced Accuracy : 0.5000

       'Positive' Class : FALSE
```

Figure 3.2 - Evaluation of the non-linear SVM model with a hyperbolic tangent kernel

Figure 3.2 shows that the hyperbolic tangent kernel performs poorly for this classification task. It produces exclusively FALSE predictions, yielding 47 type II errors. This is not a meaningful classifier.

4.3 The Anova Kernel

The Anova kernel is a variant of the RBF kernel which passes data through a polynomial transform before applying an RBF (Gaussian) kernel. To some degree, it can be considered a combination of the polynomial and RBF kernels. Figure 3.3 shows the performance of the Laplacian kernel for the university improvement classification problem.

```
 Setting default kernel parameters
Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE    50   20
     TRUE     30   27

               Accuracy : 0.6063
                 95% CI : (0.5157, 0.6918)
    No Information Rate : 0.6299
    P-Value [Acc > NIR] : 0.7414

                  Kappa : 0.1911
 Mcnemar's Test P-Value : 0.2031

            Sensitivity : 0.6250
            Specificity : 0.5745
         Pos Pred Value : 0.7143
         Neg Pred Value : 0.4737
             Prevalence : 0.6299
         Detection Rate : 0.3937
   Detection Prevalence : 0.5512
      Balanced Accuracy : 0.5997

       'Positive' Class : FALSE
```

Figure 3.3 - Evaluation of the non-linear SVM model with an ANOVA kernel

Figure 3.3 shows that the ANOVA kernel SVM classifier yields 30 type I and 20 type II errors, for an accuracy of 0.6063 and a kappa of 0.1911. This only narrowly outperforms random class assignment, and it is inferior to each of the non-SVM classifiers generated.

4.4 Conclusion on SVM Kernels

As expected, the RBF kernel was the most effective in performing this classification task. The other non-linear kernels used (ANOVA, Hyperbolic Tangent, and Polynomial) were not able to match the high accuracy and kappa values of the RBF kernel's classifier on the testing dataset. The linear kernel, interestingly, outperformed all of the non-linear kernels except the RBF kernel, suggesting that the underlying N-dimensional space is nearly linearly-separable with respect to the class variable (improved).

References

[1] THE Times Higher Education Rankings. *timeshighereducation.com*, THE World Rankings, 2016.

[2] Academic Ranking of World Universities. *shanghairanking.com*, Shanghai World Rankings, 2015.

[3] CWUR | Center for World University Rankings. *cwur.org*, Worlds Top Universities, Rankings by Country, 2015.

[4] Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca, Yuan Tang and Can Candan. (2016). caret: Classification and Regression Training. R package version 6.0-68. https://CRAN.R-project.org/package=caret

[5] Michael Hahsler, Christian Buchta, Bettina Gruen and Kurt Hornik (2016). arules: Mining Association Rules and Frequent Itemsets. R package version 1.5-0. https://CRAN.R-project.org/package=arules

[6] Ian Johnson (2016). arulesCBA: Classification Based on Association Rules. R package version 1.0.2.

[7] Venables, W. N. Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth Edition. Springer, New York. ISBN 0-387-95457-0

[8] Alexandros Karatzoglou, Alex Smola, Kurt Hornik, Achim Zeileis (2004). kernlab - An S4 Package for Kernel Methods in R. Journal of Statistical Software 11(9), 1-20. URL http://www.jstatsoft.org/v11/i09/

[9] Lu, D.S.; Weng, Q.H. A survey of image classification methods and techniques for improving classification performance. Int. J. Remote Sens. 2007, 28, 823âĂŞ870.

[10] A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News 2(3), 18–22.