Machine Learning Homework 1

Bingxue Li

16th, Feb, 2024

The purpose of this report is to evaluate and compare various machine learning methods for a classification problem using our labeled dataset and to select the best predictive model for our unlabeled dataset. To determine whether a news title is sarcastic or not, I trained the model using both linear and nonlinear classification methods, and then applied ensemble and screening techniques. Following the criterion of Accuracy (A.2), I selected the lasso-screened logistic model to make predictions on our unlabeled dataset. In a purely predictive task, our goal is to obtain $\hat{f}(x_i) \xrightarrow{p} E[y_i|x_i]$ rather than estimators' consistency or asymptotic performance. We evaluate predictive performance by targeting low conditional prediction error, $\mathbb{E}[(Y - \hat{f}(x_i))^2|x_i]$, which follows a bias-variance decomposition.

Linear Models I begin with linear classification models, characterized by a linear decision boundary (A.3). Table 1 reports the baseline logistic regression (A.4) with an accuracy level of 0.751, and Table 2 reports linear discriminant analysis, which reduces dimensionality by projecting predictors onto a lower-dimensional subspace while maximizing class separability. It achieves a higher accuracy score of 0.766. However, given the dataset's high-dimensional nature ($p = 314$), incorporating all predictors into logistic or LDA models is both computationally demanding and susceptible to overfitting. Therefore, I attempt the shrinkage method with lasso, ridge, and elastic net regressions (A.5, Table 3, 4). Among the shrinkage methods, lasso regression achieves the highest accuracy level of 0.777 when $\lambda = 0.006$, with 160 non-zero coefficient variables selected. The ridge algorithm does not drop any variables across the λ-grid we tune over, and the best accuracy level Ridge can achieve is 0.773, with a larger variance of prediction error estimator compared to Lasso.

Nonlinear Models Nonlinear models offer more flexible ways of separating classes through hyperplanes with curvature. KNN (A.6) defines decision boundaries based on the local neighborhood of fixed data points, achieving an accuracy level of 0.753 for $k = 9$ (Table 5), where 9 nearest neighbors are considered. I further applied decision tree classification, which has more interpretable decision boundaries and is less computationally demanding (considering the search over the whole train dataset for neighbors), achieving the best trained accuracy level of 0.706 with the decision tree presented in Table 6. Additionally, I considered the support vector classifier with both linear (computationally convenient) and radial kernels (more flexible), with the former achieving an accuracy of 0.732 and the latter as high as 0.784 (Table 7, 8).

Screening Method The screening method involves applying a feature selection mechanism to reduce the dimensionality of the dataset, followed by training specific models on the selected features to achieve the best predictive performance. I considered feature selection with PCA or Lasso and then applied three representative wrapper methods, including logistic regression (classical post-lasso), support vector machine (nonlinear classification), or random forest (ensemble method). The outputs of these six combinations are reported in Tables 12, 13, 14, 15, 16, and 17. Among the screened methods attempted, lasso-screened logistic regression achieved the highest accuracy level of 0.785. However, PCA-screened GLM, SVM, and RF only achieved accuracy levels around 0.6 and 0.64. There could be several reasons for this, such as PCA reducing the predictor space too radically and causing underfitting, or nonlinearities not being adequately accounted for after PCA screening.

Ensemble Method Ensemble techniques such as Bagging, Boosting, and Random Forest (A.8) combine predictions from multiple single models. I first tried tree-based ensemble methods, including bagging, stochastic gradient boosting, and purely random forest (Tables 9, 10, 11) as simple decision tree suffers from high variance. The tuning parameter 'mtry' in random forest indicates that the number of variables randomly sampled as candidates at each split is best set at 314, which makes it computationally demanding with barely any improvement in prediction compared with simple shrinkage methods. A natural candidate for ensembling could be combining elastic net regularization and tree-based models since they have complementary strengths. Glmnet is ill-posed for capturing nonlinearity in classification (decision boundary), handling collinearity between predictors, and preserving variability from all predictors in the dataset, all of which could lead to prediction bias if we merely rely on a glmnet-trained model and poor generalization to unlabeled data. However, I was not able to make the algorithm converge in my code, and random forest is computationally demanding. I also tried ensembling with glm and glmnet classes, but the benefits of ensembling are modest considering that they are both linear boundary models.

Model Selection I finally selected the screening model of Logistic screened with Lasso with the highest accuracy level of prediction, low variance of predicting error (checked through bootstrap resampling), and computational convenience (even though this is lower in Train/Test Split evaluation, implying a potential downward biased estimate of out-of-sample prediction error). I apply this to the test dataset to obtain the prediction. We should be cautious that strong assumptions that the train and test datasets are drawn from the same DGP are made for prediction error estimators to be consistent.

A Technical Appendix

A.1 Train/Test Split and Cross-Validation

In the setting of this task, we have a training dataset with outcome, S_i observable and a testing dataset where outcome is not observed. One might considering training and evaluating the model with whole training dataset, but doing this with a same dataset would give biased estimate of the error on the test set to be predicted.

Different from cross-validation within the same train dataset, by splitting the train dataset into train and test(validation) subsets throughout the task, we create a holdout set that the model has not seen during training process.

```
trainIndex <- createDataPartition(fulldf$is_sarcastic, p = .8,
                                  list = FALSE,
                                  times = 1)
head(trainIndex)
dfTrain <- fulldf[ trainIndex,]
dfTest  <- fulldf[-trainIndex,]
```

With the above data, I further obtain a train subset of size 2161, $\frac{4}{5}$ of the original train set, and a test subset of size 539 with simple splitting according to the outcome. In the end I evaluate the out-of-sample performance of predicting of the tuned model with a confusion matrix applied to the test subset.

A.2 Classification and Accuracy

As we deal with binary classification outcomes, the prediction with linear models follows:
1. For each values of outcome class $S = 0, 1$, compute the fitted output of conditional probability $\hat{f}_s(x)$.
2. Identify the largest component and classify accordingly: $\hat{S}(x) = \text{argmax}_{s\in\mathcal{S}}\, \hat{f}_s(x)$.
Instead of Mean Squared Errors, we adopt the criteria Accuracy, for a validation sample of size n with classification outcome available to compare with,

$$\text{Accuracy} = \frac{\sum_{i=1}^{n} \mathbb{1}\{\hat{S}_i(x_i) = S_i\}}{n} \tag{1}$$

We know that MSE is non-convex for binary classification, training Binary outcome model with MSE might not minimize the cost function. In addition, MSE best performs under Gaussian distribution, which is not the case with binary outcome. Throughout this exercise, I specify the regression such that `metric="Accuracy"`.

A.3 Decision Boundary

$$\log \frac{\mathbb{P}(S = 0|X = x)}{\mathbb{P}(S = 1|X = x)} = \beta_0 + \beta^T x \tag{2}$$

A decision boundary is the set of points where log-odds in 2 equals to 0, equivaluent to a linear hyperplane such that $\{x \mid \beta_0 + \beta^T x = 0\}$.

A.4 Baseline: Logistic Regression

$$\log \frac{\mathbb{P}(S = 0|X = x)}{\mathbb{P}(S = 1|X = x)} = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p \tag{3}$$

$$\mathbb{P}(S = 1|X = x) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}, p = 314 \text{ with word2vec cleaned data.} \tag{4}$$

Drawbacks: Computationally costly and concern with overfitting considering high-demensionality feature of our dataset, p=314 with n=2700. Merits: Nice interpretability (interpretability under Hastie,Tibshirani, and Friedma's definition, not in an econometrics point of view regarding consistency or causality).

A.5 Regularized Logistic Regression

Lasso: L1 Regularized Logistic Regression

$$\max_{\beta_0,\beta}\left\{\sum_{i=1}^{N}\Big[s_i(\beta_0+\beta^T x_i)-\log(1+e^{\beta_0+\beta^T x_i})\Big]-\lambda\sum_{j=1}^{p}|\beta_j|\right\} \tag{5}$$

Tuning Parameter: λ
Ridge: L2 Regularized Logistic Regression

$$\max_{\beta_0,\beta}\left\{\sum_{i=1}^{N}\Big[s_i(\beta_0+\beta^T x_i)-\log(1+e^{\beta_0+\beta^T x_i})\Big]-\lambda\sum_{j=1}^{p}\beta_j^2\right\} \tag{6}$$

Tuning Parameter: λ
Elastic Net

$$\max_{\beta_0,\beta}\left\{\sum_{i=1}^{N}\Big[s_i(\beta_0+\beta^T x_i)-\log(1+e^{\beta_0+\beta^T x_i})\Big]-\lambda\sum_{j=1}^{p}\big(\alpha\beta_j^2+(1-\alpha)|\beta_j|\big)\right\} \tag{7}$$

Tuning Parameter: λ, α.
I set the tuning parameter grid search with gird `lambda = seq(0.001, 0.1, by = 0.001)` for L1 and L2. For elastic net, set `tuneLength=5` which automatically search over a 5x5 default pairs.

A.6 Nonlinear Models

A linear decision boundary is unlikely to be optimal in many cases (Note Hastie,Tibshirani, and Friedma gives a potential scenario: The training data in each class (Sarcastic or not) came from a mixture of multiple low-variance Gaussian distributions, with individual means themselves distributed as Gaussian). If the true optimal decision boundary is nonlinear and disjoint, we should consider methods that accomodate this nonlinearity. Even though linear boundary is stable to fit with low variance, it might introduce more bias in the estimator considering the strong conditions it implies. However, depending on the inputted points around the boundaries, K-nearest neighborhood can produce wiggly boundaries with high variance of the estimator.
K-Nearest Neighborhood

$$\hat{S}(x)=\frac{1}{k}\sum_{x_i\in N_k(x)} s_i \tag{8}$$

where $N_k(x)$ is a neighborhood of x with k closest, defined under Euclidean distance, observations (s_i, x_i) around point x.
Classification Tree In the case of classification tree, we obtain the tree with recursive binary spliting just as regression case. Classification prediction is made based on the largest(larger) occurrence of class in the given region that the observation belongs to. To evaluate a particular split, the criteria considered is Gini index, for a fixed region r:

$$G=\sum_{k=1}^{K}\frac{\sum_i^N \mathbb{1}\{s_i=k\}}{\sum_{i=1}^N \mathbb{1}\{X_i\in r\}}\left\{1-\frac{\sum_i^N \mathbb{1}\{s_i=k\}}{\sum_{i=1}^N \mathbb{1}\{X_i\in r\}}\right\} \tag{9}$$

A.7 Screening Method

Feature Selection: Lasso-Screened

$$\hat{B}_\lambda^{\text{Lasso}}=\{j\in(1,\ldots,p);\hat{\beta}_{\lambda,j}^{\text{Lasso}}\neq 0\} \tag{10}$$

We take the B with the tuned, fixed λ and use corresponding set of x_{j_B} for a logistic regression. This algorithem applies Lasso as variable selection process to reduce dimensionality and run a post regression to

reduce bias.
Feature Selection: PCA-Screened
We want transform out high-dimensional data (p=314) to a low-dimensional on that captures as much of the information as possible (variability are better preserved that dumping the variables in lasso variable selection). Components are selected and constructed with `preProcess = "pca"` under **caret** package, or manually with `PCA(predictors)` under **FactoMineR** package, and I used the later for a Train/Test Split I manually constructed in the code. For fixed i, the first component follows:

$$\max_{\phi_{11},\ldots,\phi_{p1}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j1} x_{ij} \right)^2 \right\} \text{ subject to } \sum_{j=1}^{p} \phi_{j1}^2 = 1 \tag{11}$$

$$c_{i1} = \phi_{11} x_{i1} + \phi_{21} x_{i2} + \ldots + \phi_{p1} x_{ip} \tag{12}$$

As c component defined with linear combination that leverages over all possible x, the merit of variability (controlling for variance) of PCA compared with Lasso as feature selection is immediate.
Wrapper Method For learning model following the screened variable processing, I attempt baseline logistic regression, elastic net, and support vector machine to accounting for a baseline, an easily interpretable (again, in machine learning sense), and a flexible model.

A.8 Ensemble Method

Ensemble methods Starting from a decision tree based method, we consider bagging, boosting, and random forest to improve the performance of individual weak learner models by combining multiple weak learners them into a stronger model with different combining techniques.
Bagging We first construct B bootstrap sample (drawing with replacement) from the training data. Then, train $f^{*b}(x)$ with each bootstrap sample respectively, and averaging across these predictions for fixed x.

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x). \tag{13}$$

With classification problem, we should consider voting rather than averaging.

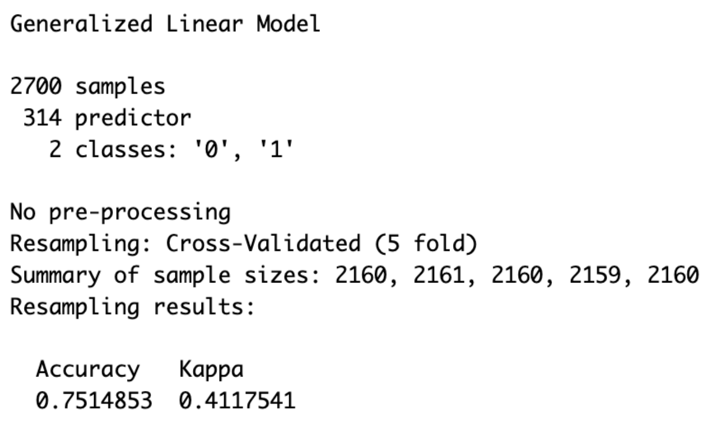
Manually Constructed Ensembling Model To account for both the merit of nonlinearity (flexibility) guaranteed by tree-based models like random forest, and the merit of reduction of dimension and easy interpretability of linear boundary models like `"glmnet"` class. I mannually ensemble the glmnet model with random forest model into one, but this does not converge in my code.

A.9 Resampling Method

In the code, I mainly try two types of resampling methods for evaluating the performance of the predictive model, cross-validation and bootstrap validation. The Bootstrap Validation creates several bootstrap samples from the train dataset and evaluate the model's performance over the bootstrap samples, which avoids loss in sample size due to resample with replacement but is computationally demanding and slower, see appendix for a computation time comparison.

B Output Appendix

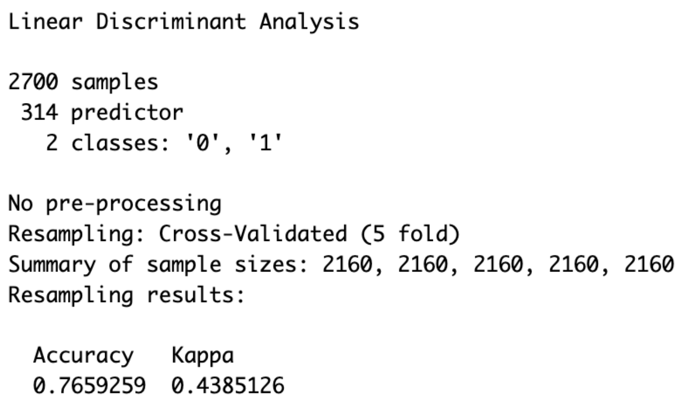
B.1 Baseline Logistic Regression



```
Generalized Linear Model

2700 samples
 314 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 2160, 2161, 2160, 2159, 2160
Resampling results:

  Accuracy   Kappa
  0.7514853  0.4117541
```

Figure 1: Linear Determinant Analysis

B.2 Linear Determinant Analysis



```
Linear Discriminant Analysis

2700 samples
 314 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 2160, 2160, 2160, 2160, 2160
Resampling results:

  Accuracy   Kappa
  0.7659259  0.4385126
```

Figure 2: Linear Determinant Analysis

B.3 Regularized Logistic Regression

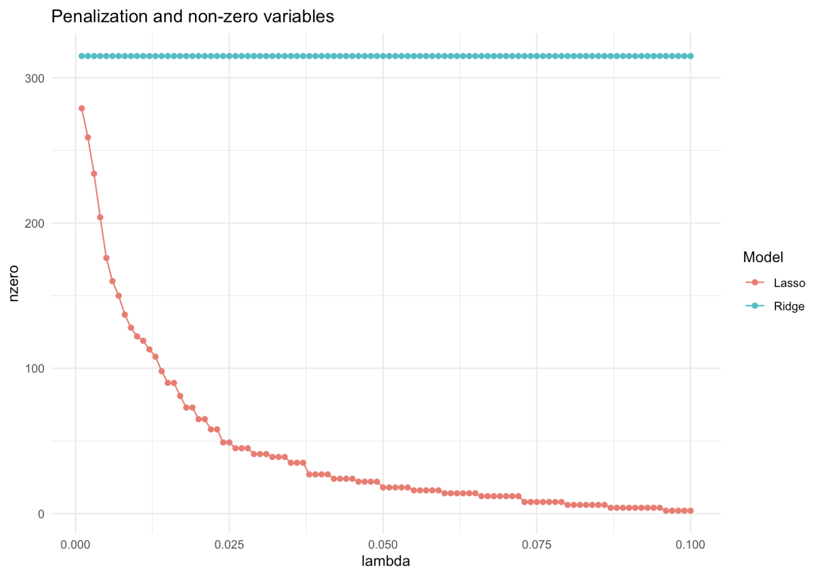


Figure 3: Lasso and Ridge: Number of Variable Selected

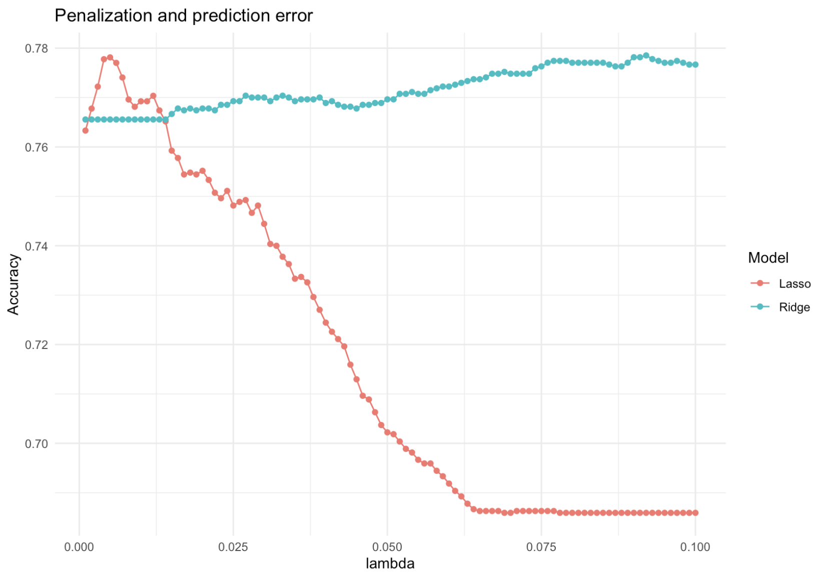
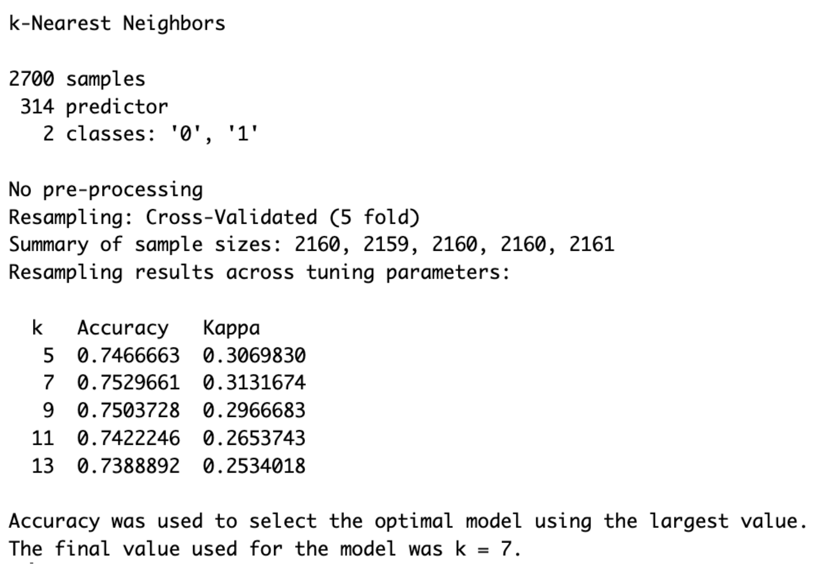


Figure 4: Lasso and Ridge: Accuracy Level Tuning

B.4 Nonlinear: K-nearest Neighborhood



```
k-Nearest Neighbors

2700 samples
 314 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 2160, 2159, 2160, 2160, 2161
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.7466663  0.3069830
   7  0.7529661  0.3131674
   9  0.7503728  0.2966683
  11  0.7422246  0.2653743
  13  0.7388892  0.2534018

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 7.
```

Figure 5: K-nearest Neighborhood

B.5 Nonlinear: Decision Tree

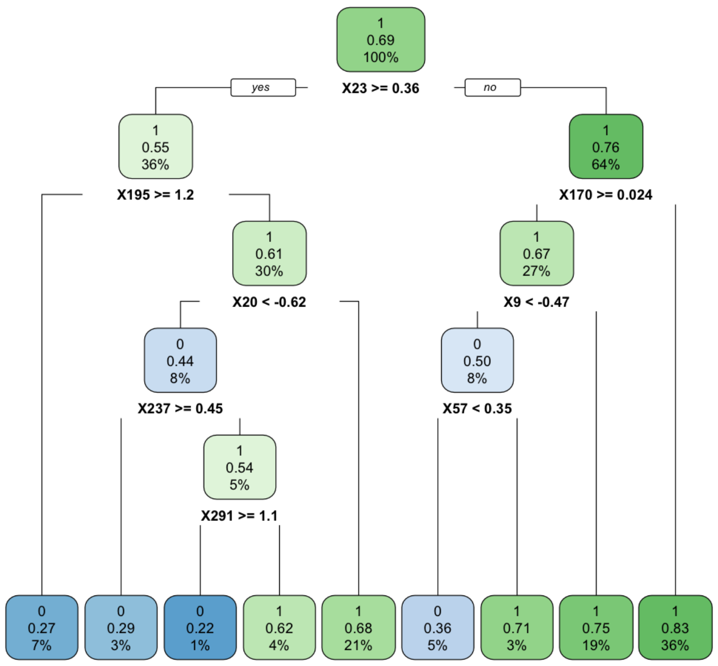
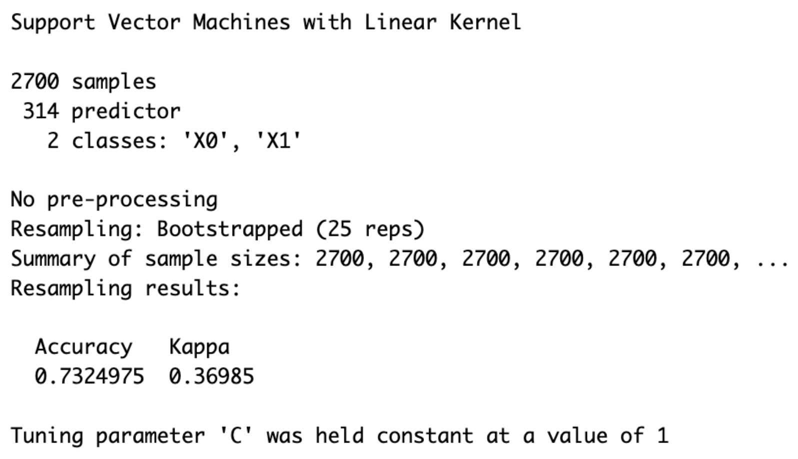


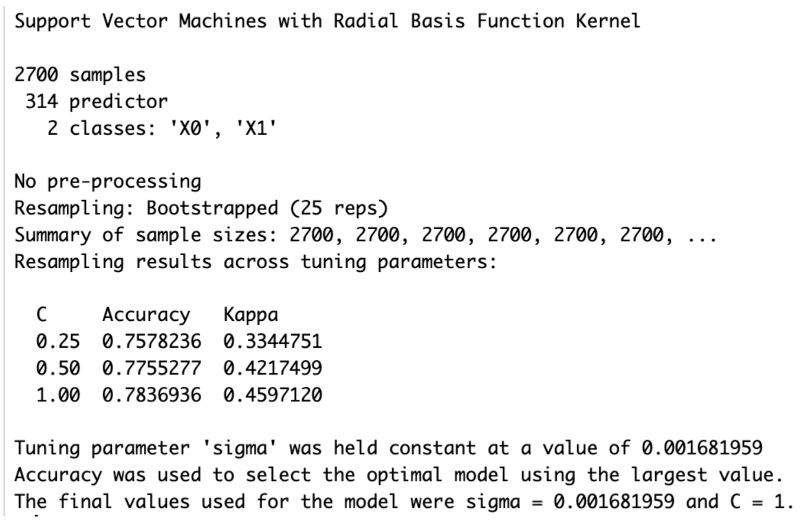
Figure 6: Decision Tree

B.6 Nonlinear: Support Vector Machine



```
Support Vector Machines with Linear Kernel

2700 samples
 314 predictor
   2 classes: 'X0', 'X1'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 2700, 2700, 2700, 2700, 2700, 2700, ...
Resampling results:

  Accuracy   Kappa
  0.7324975  0.36985

Tuning parameter 'C' was held constant at a value of 1
```

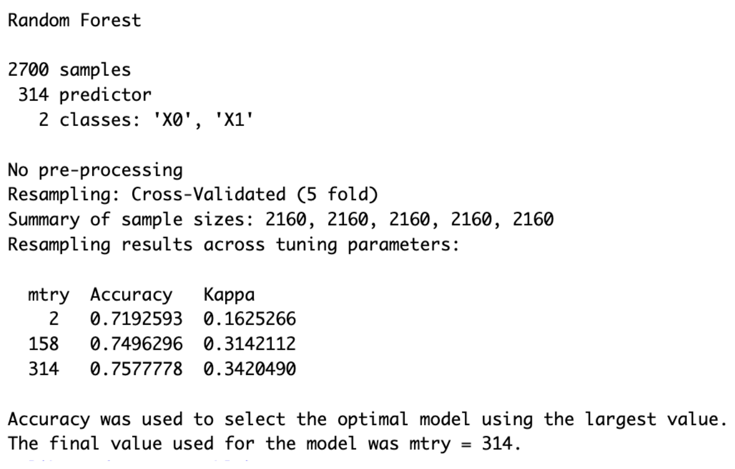
Figure 7: Support Vector Machine with Linear Kernel



```
Support Vector Machines with Radial Basis Function Kernel

2700 samples
 314 predictor
   2 classes: 'X0', 'X1'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 2700, 2700, 2700, 2700, 2700, 2700, ...
Resampling results across tuning parameters:

  C     Accuracy   Kappa
  0.25  0.7578236  0.3344751
  0.50  0.7755277  0.4217499
  1.00  0.7836936  0.4597120

Tuning parameter 'sigma' was held constant at a value of 0.001681959
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.001681959 and C = 1.
```

Figure 8: Support Vector Machine with Radial Kernel

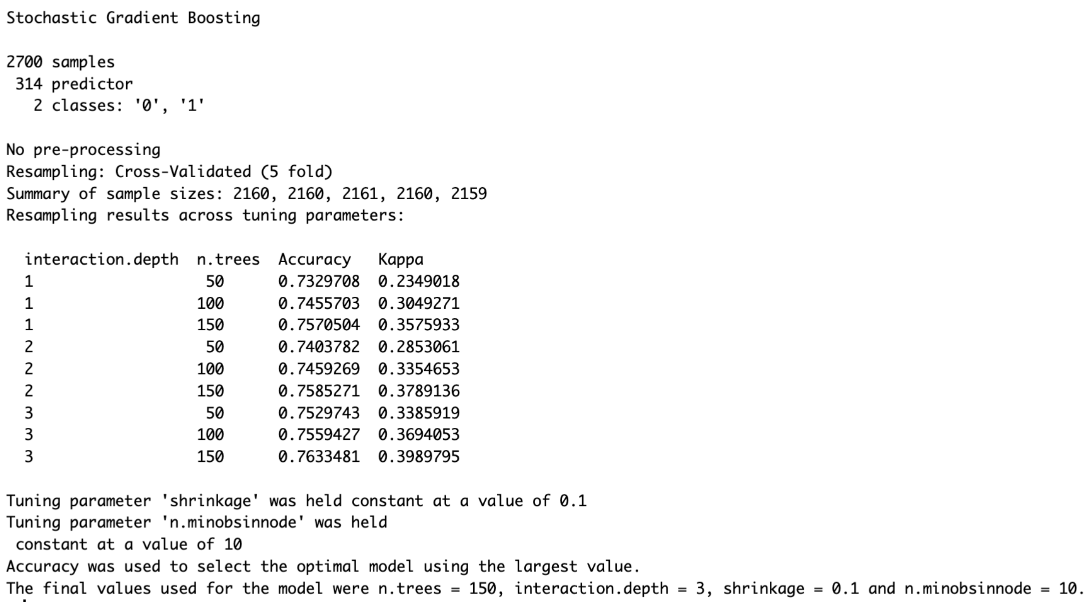
B.7 Ensemble Method: Random Forest



```
Random Forest

2700 samples
 314 predictor
   2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 2160, 2160, 2160, 2160, 2160
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
    2   0.7192593  0.1625266
  158   0.7496296  0.3142112
  314   0.7577778  0.3420490

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 314.
```

Figure 9: Random Forest

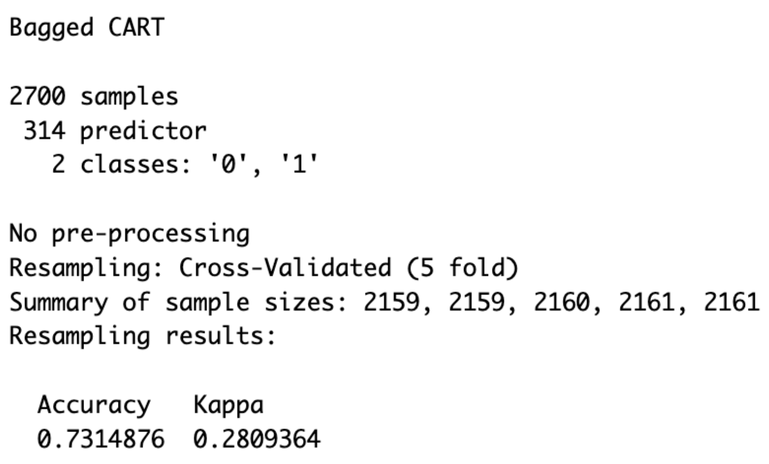
B.8 Ensemble Method: Stochastic Gradient Boosting



```
Stochastic Gradient Boosting

2700 samples
 314 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 2160, 2160, 2161, 2160, 2159
Resampling results across tuning parameters:

  interaction.depth  n.trees  Accuracy   Kappa
  1                   50      0.7329708  0.2349018
  1                  100      0.7455703  0.3049271
  1                  150      0.7570504  0.3575933
  2                   50      0.7403782  0.2853061
  2                  100      0.7459269  0.3354653
  2                  150      0.7585271  0.3789136
  3                   50      0.7529743  0.3385919
  3                  100      0.7559427  0.3694053
  3                  150      0.7633481  0.3989795

Tuning parameter 'shrinkage' was held constant at a value of 0.1
Tuning parameter 'n.minobsinnode' was held
 constant at a value of 10
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were n.trees = 150, interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

Figure 10: Stochastic Gradient Boosting

B.9 Ensemble Method: Bagging

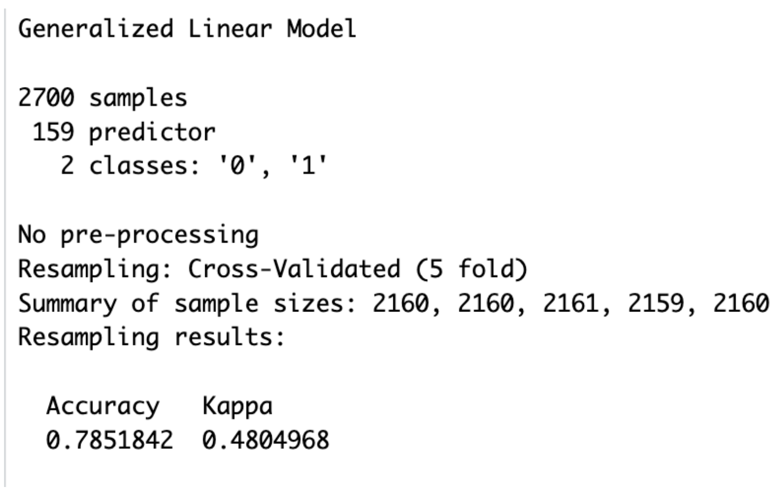


```
Bagged CART

2700 samples
 314 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 2159, 2159, 2160, 2161, 2161
Resampling results:

  Accuracy   Kappa
  0.7314876  0.2809364
```

Figure 11: Bagging

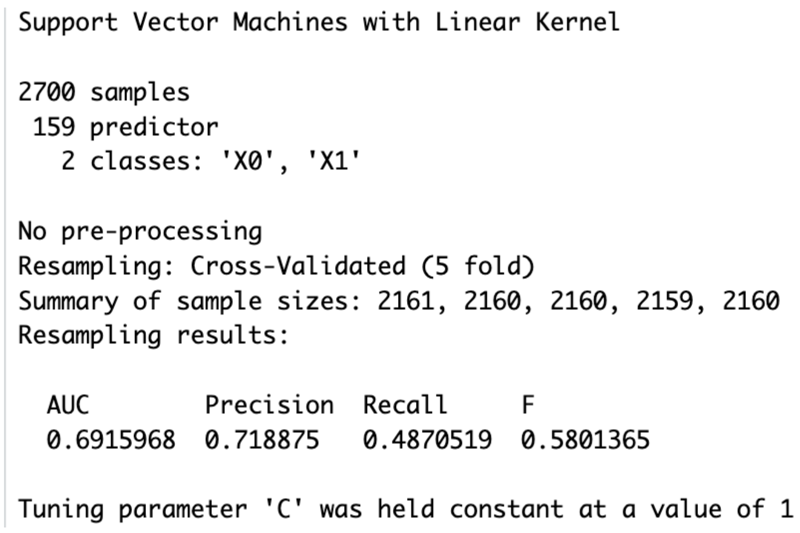
B.10 Ensemble Method: Selected Pool

B.11 Screening Method: Lasso/PCA



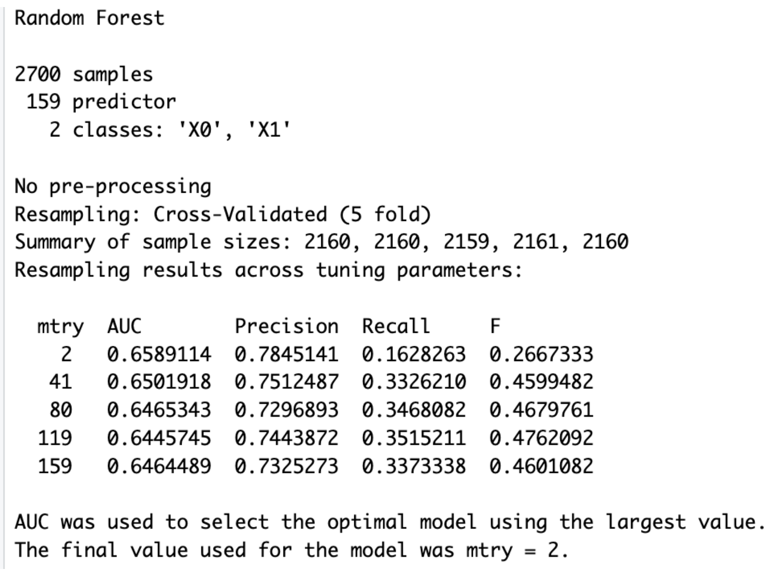
```
Generalized Linear Model

2700 samples
 159 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 2160, 2160, 2161, 2159, 2160
Resampling results:

  Accuracy   Kappa
  0.7851842  0.4804968
```

Figure 12: GLM Screened with Lasso



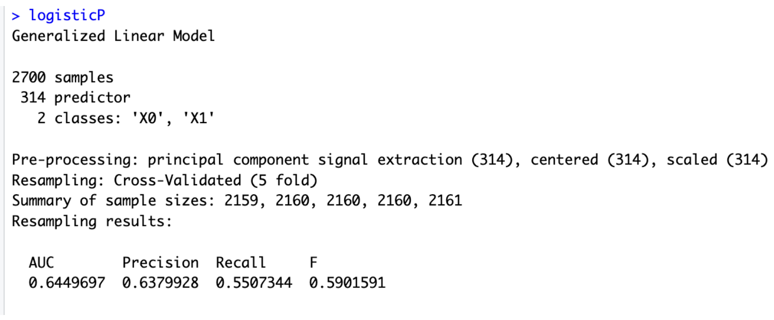
```
Support Vector Machines with Linear Kernel

2700 samples
 159 predictor
   2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 2161, 2160, 2160, 2159, 2160
Resampling results:

  AUC        Precision  Recall     F
  0.6915968  0.718875   0.4870519  0.5801365

Tuning parameter 'C' was held constant at a value of 1
```

Figure 13: Support Vector Machine Screened with Lasso



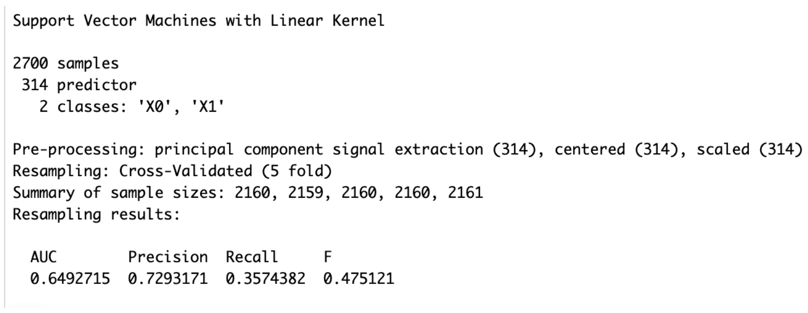
```
Random Forest

2700 samples
 159 predictor
   2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 2160, 2160, 2159, 2161, 2160
Resampling results across tuning parameters:

  mtry  AUC        Precision  Recall     F
    2   0.6589114  0.7845141  0.1628263  0.2667333
   41   0.6501918  0.7512487  0.3326210  0.4599482
   80   0.6465343  0.7296893  0.3468082  0.4679761
  119   0.6445745  0.7443872  0.3515211  0.4762092
  159   0.6464489  0.7325273  0.3373338  0.4601082

AUC was used to select the optimal model using the largest value.
The final value used for the model was mtry = 2.
```

Figure 14: Random Forest Screened with Lasso



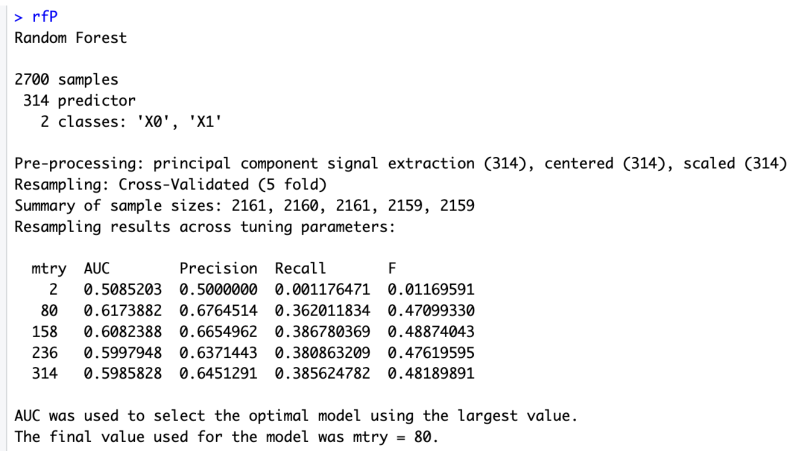
```
> logisticP
Generalized Linear Model

2700 samples
 314 predictor
   2 classes: 'X0', 'X1'

Pre-processing: principal component signal extraction (314), centered (314), scaled (314)
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 2159, 2160, 2160, 2160, 2161
Resampling results:

  AUC        Precision  Recall     F
  0.6449697  0.6379928  0.5507344  0.5901591
```

Figure 15: GLM Screened with PCA



```
Support Vector Machines with Linear Kernel

2700 samples
 314 predictor
   2 classes: 'X0', 'X1'

Pre-processing: principal component signal extraction (314), centered (314), scaled (314)
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 2160, 2159, 2160, 2160, 2161
Resampling results:

  AUC        Precision  Recall     F
  0.6492715  0.7293171  0.3574382  0.475121
```

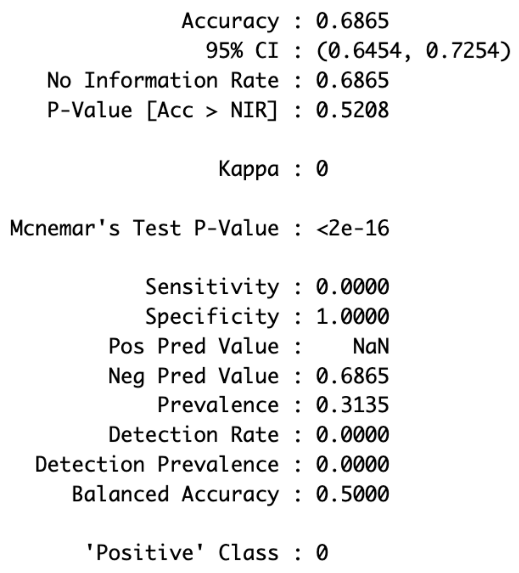
Figure 16: Support Vector Machine Screened with PCA



```
> rfP
Random Forest

2700 samples
 314 predictor
   2 classes: 'X0', 'X1'

Pre-processing: principal component signal extraction (314), centered (314), scaled (314)
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 2161, 2160, 2161, 2159, 2159
Resampling results across tuning parameters:

  mtry  AUC        Precision  Recall       F
    2   0.5085203  0.5000000  0.001176471  0.01169591
   80   0.6173882  0.6764514  0.362011834  0.47099330
  158   0.6082388  0.6654962  0.386780369  0.48874043
  236   0.5997948  0.6371443  0.380863209  0.47619595
  314   0.5985828  0.6451291  0.385624782  0.48189891

AUC was used to select the optimal model using the largest value.
The final value used for the model was mtry = 80.
```

Figure 17: Random Forest Screened with PCA

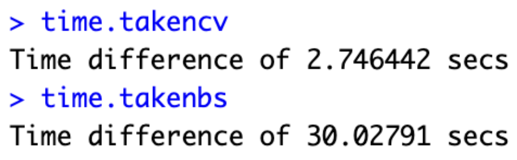
B.12 Cross-validation or Train/Test Split?



```
               Accuracy : 0.6865
                 95% CI : (0.6454, 0.7254)
    No Information Rate : 0.6865
    P-Value [Acc > NIR] : 0.5208

                  Kappa : 0

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

            Sensitivity : 0.0000
            Specificity : 1.0000
         Pos Pred Value :    NaN
         Neg Pred Value : 0.6865
             Prevalence : 0.3135
         Detection Rate : 0.0000
   Detection Prevalence : 0.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : 0
```

Figure 18: Train/Test Holdout Evaluated Accucracy of Post-Lasso Logistic

B.13 Resampling Method Choice



```
> time.takencv
Time difference of 2.746442 secs
> time.takenbs
Time difference of 30.02791 secs
```

Figure 19: Boostrap is Slower than Cross-validation