Assignment #3

Rajati

Muxin Liang

Contents

Exercise # 1

LASSO and Boosting for Regression.

(a) Data set

Download Communities and Crime data. The first 5 predictors are non-predictive, and the last columns: ViolentCrimesPerPop: total number of violent crimes per 100K popuation, is the quantitative value to be predicted.

(b) Replace missing value

I used sklearn's imputer to replace missing values. Firstly, I used pandas replace to replace all the "?" into "NaN". Then, I used the most frequent values to represent all the NaN in data.

This process has been done in the Jupyter Notebook.

(c) Correlation matrix plot

The correlation matrix plot is shown below, colors that closer to yellow indicates features that have strong relationship in between.

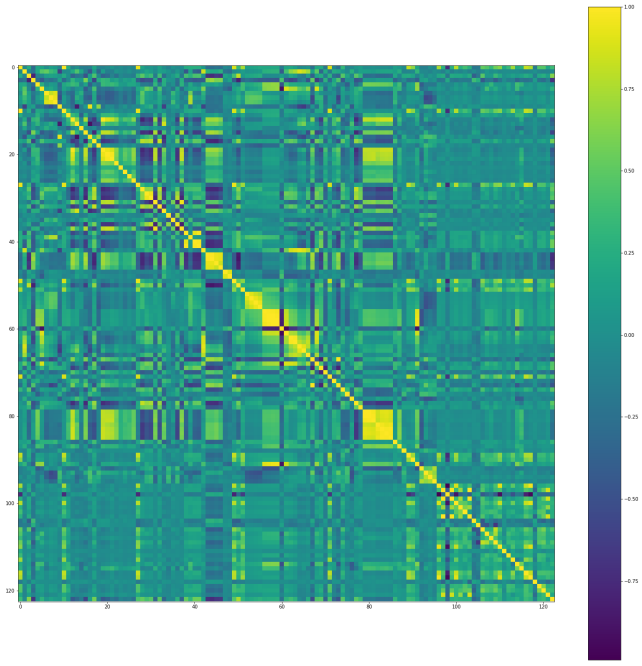


Figure 1: Correlation matrix of features

(d) Coefficient of Variation

Coefficient of Variation(CV) is calculated by:

$$C_v = \frac{\sigma}{\mu}, \text{ where } \sigma \text{ stands for standard deviation and } \mu \text{ stands for mean}$$

Detailed calculation for every parameter shown in the notebook.

(e) Scatter plots and box plots for highest-CV predictors

$\lfloor\sqrt{128}\rfloor = 11$, so I picked 11 features with highest CV and made those plots.

1. Pairwise scatter plot, the labels on axis are the index of predictors. As shown in the picture, it is hard to recognize significant predictors from observation.

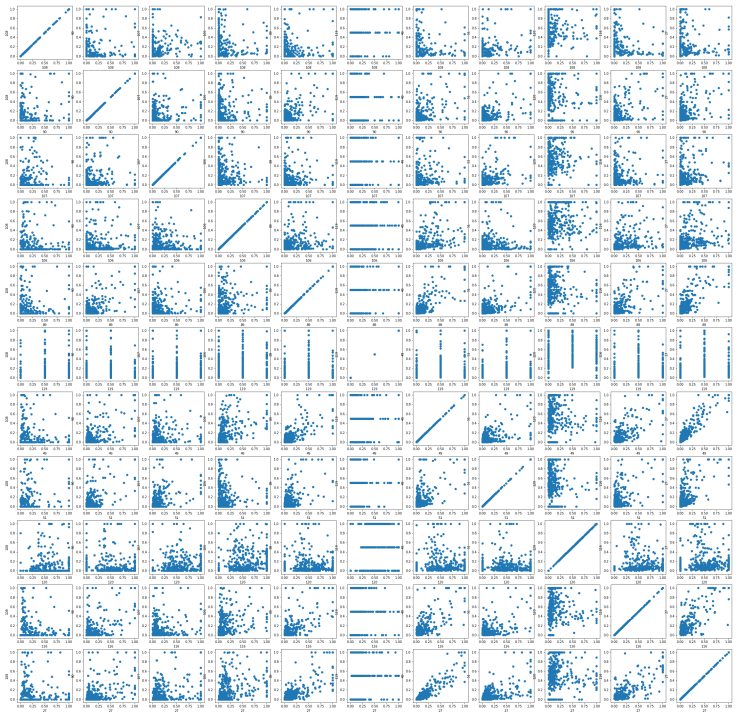


Figure 2: Pairwise scatter plots for 11 features with highest CV

2. Box plot:

   x-axis has the index of selected predictors and y-axis shows values of them.

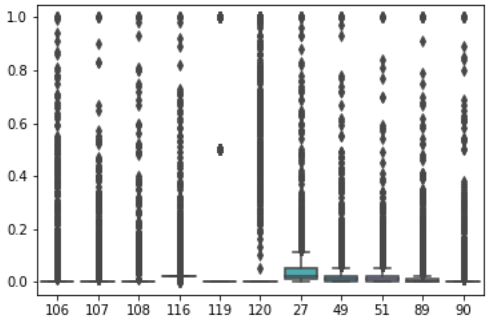


Figure 3: Box plot for 11 features with highest CV

(f) Fit linear model

We fit linear model using linear regression using all the predictors and get the resulted MSE = 0.6215 and $R^2 = 0.0180$, not good values for linear regression(Low R^2 score). We conclude that using all predictors to fit a linear model is not a good way for predicting this data.

Detailed codes in notebook.

(g) Ridge model

There are two steps in fitting a ridge model. Firstly, use cross validation to determine an optimal λ for regularization. Then use this λ to make prediction and measure test error.

We used 5-fold validation and the list of λ to choose from was:

$$[0.001, 0.01, 0.1, 0.3, 0.5, 1.0, 2.0, 5.0, 10.0]$$

The best result from cross validation gives $\lambda = 2$ with MSE = 0.01902

Therefore we use $\lambda = 2$ to fit test data and it has $R^2 = 0.6322$ and MSE = 0.0175.

Detailed code in Jupyter Notebook.

(h) Lasso model

There are two steps in fitting a Lasso model. Firstly, use cross validation to determine an optimal α for regularization. Then use this α to make prediction and measure test error.

We used 5-fold validation and the list of α to choose from was:

$$[0.001, 0.01, 0.1, 0.3, 0.5, 1.0, 2.0, 5.0, 10.0]$$

The best result from cross validation gives $\alpha = 0.0001$ with MSE = 0.0190

Therefore we use $\alpha = 0.0001$ to fit test data and it has $R^2 = 0.6297$ and MSE = 0.0176.

Then we try the normalized form of data. Use normalize from sklearn, we normalize train and test data and fit lasso model. This time, the best from validation gives $\alpha = 0.0001$ and MSE $= 0.0193$.

Therefore we use $\alpha = 0.0001$ to fit NORMALIZED test data and it has $R^2 = 0.633$ and MSE $= 0.0175$. A slightly better result comparing to not normalizing. Therefore we conclude there is improvement in accuracy for normalizing data

Detailed code in Jupyter Notebook.

(i) PCR model

There is no direct PCR in sklearn, but we can use PCA to do dimension reduction and then fit reduced-dimension data with linear model. By iterating from 1 to 122 predictors, we get a plot of number of predictors against MSE.

The best MSE happens when we choose number of reduced dimension = 91, and its MSE is 0.0197.

Detailed code in Jupyter Notebook.

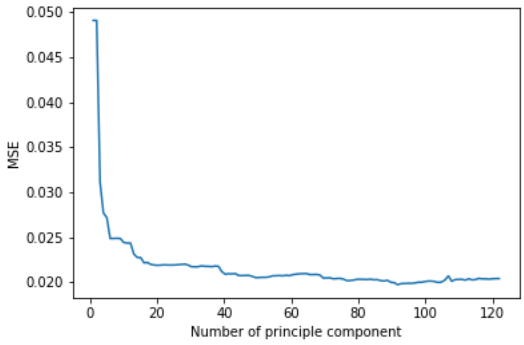


Figure 4: PCR results over different number of predictors

(j) XGBoost model tree

We use XGBoost package in python to finish this problem. And we use cross validation to choose the best α for regularization from the list:

$$[0.001, 0.01, 0.1, 0.3, 0.5, 1.0, 2.0, 5.0, 10.0]$$

The final result gives $\alpha = 0.5$ and the minimum MSE in cross validation is 0.1764.

Detailed code in Jupyter Notebook.

Exercise # 2

(a) Download dataset

Detailed code in Jupyter Notebook.

(b) Data preparation

1. Data imputation techniques: There are many ways of imputation to deal with missing data.

   Hot-deck: a missing value will be imputed from a randomly selected similar record. One form of hot-deck imputation is called "last observation carried forward" (LOCF) It sorts a dataset according to any of a number of variables. And it will use the cell value immediately prior to the missing data to impute the missing value. The shortcoming of this method is known to increase risk of increasing bias and potentially false conclusions.

   Cold-deck: it selects donors from another dataset to imput missing values.

   Mean Substitution: replacing any missing value with the mean of that variable for all other cases. Benefit: not changing the sample mean for that variable. Shortcoming: there will be no relationship between imputed variables and any other measured variables.(bad for multivariate analysis)

   Regression: fit a regression model on observed variables and use the fitted model to predict those missing values. Problem: will get greater precision in the imputed values than is warranted.

   Most frequent Substitution: replacing any missing value with the most frequent value of that variable for all other cases.

   I will use mean substitution to impute missing values.

2. Calculate CV:

   Coefficient of Variation(CV) is calculated by:

   $$C_v = \frac{\sigma}{\mu}, \text{ where } \sigma \text{ stands for standard deviation and } \mu \text{ stands for mean}$$

   Detailed calculation for every parameter shown in the notebook.

3. Correlation matrix:
   The correlation matrix plot is shown below, colors that closer to yellow indicates features that have strong relationship in between.

4. Scatter and box plots:
   $\lfloor\sqrt{170}\rfloor = 13$, so I picked 13 features with highest CV and made those plots. The plots looks sparse because there are some extremely big values. It is not possible to draw conclusions about significance of those features just by observing the scatter plots.

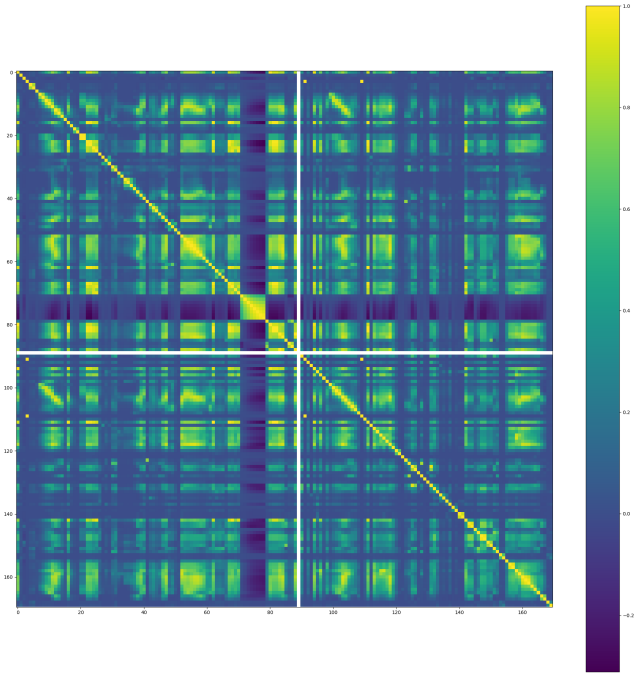


Figure 5: Correlation matrix of features

5. Numbers of pos and neg:
   In train data, there are 59000 "neg" and 1000 "pos". In test data there are 15625 "neg" and 375 "pos", and we calculate the probability of the result is positive in train and test

$$Prob(Y = pos) = 1.7\% \approx 0\% \text{ (train)}$$
$$Prob(Y = pos) = 2.3\% \approx 0\% \text{ (test)}$$

   Therefor, the dataset is very imbalanced in both train and test set.

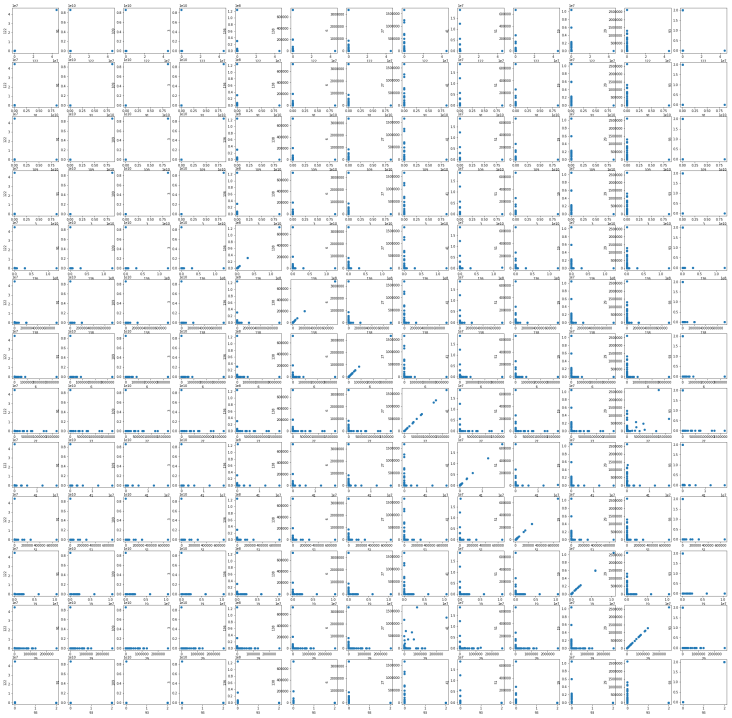


Figure 6: Pairwise scatter plots for 13 features with highest CV

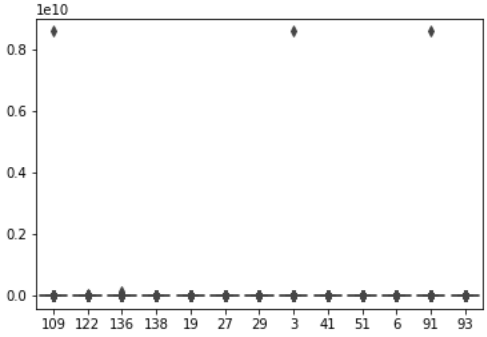


Figure 7: Box plot for 13 features with highest CV

(c) Train a random forest

I used sklearn's RandomForestClassifer to train the random forest. The detailed code can be seen from Jupyter Notebook.

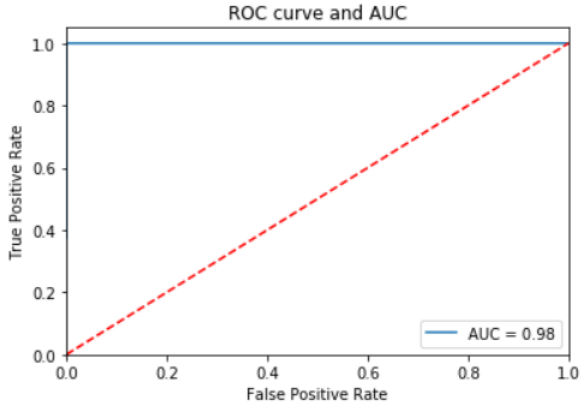
I put both train set and test set into prediction to get an overview of the classifier. Here I mark "pos" as 1 and "neg" as 0.

The following graphics contains those information, from top to bottom: ROC and AUC information, classification report, and confusion metrics.

The Out of bag error, as calculated by setting "OOB parameter" when training, gives $oob\_error = 0.93\%$. By observing confusion matrix, we can conclude that there are 31 "pos" cases was recognized as "neg" (False negative) and 2 "neg" recognized as "pos"(False positive) when using train data set for classification.

Misclassification in train set is calculated by equation:

$$M = \frac{(FP+FN)}{Total} = \frac{33}{60000} = 0.05\%$$



	precision	recall	f1-score	support
0	1.00	1.00	1.00	59000
1	1.00	0.97	0.98	1000
avg / total	1.00	1.00	1.00	60000

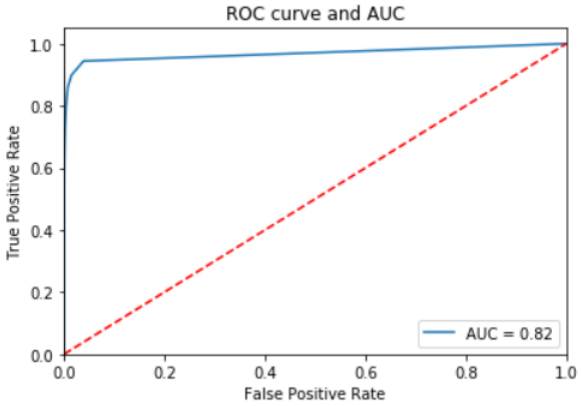
```
[[58998     2]
 [   31   969]]
```

Figure 8: Classification result of Ramdom Forest using train data

By observing confusion matrix, we can conclude that there are 132 "pos" cases was recognized as "neg" (False negative) and 18 "neg" recognized as "pos"(False positive) when using train data set for classification. Misclassification in train set is calculated by equation:

$$M = \frac{(FP+FN)}{Total} = \frac{150}{16000} = 0.94\%$$

The test error, as shown in figure, is about 1%. Comparing it to OOB error, we can find that they are quite close.



	precision	recall	f1-score	support
0	0.99	1.00	1.00	15625
1	0.93	0.65	0.76	375
avg / total	0.99	0.99	0.99	16000

```
[[15607    18]
 [  132   243]]
```

Figure 9: Classification result of Random Forest using test data

(d) Deal with imbalance in random forest

To deal with imbalance in random forest, one way is to adjust the weight of different class. This process can be done by adjust the *class_weight* parameter in sklearn's RandomForestClassifier.

The parameter I chose for weighting is class 0(neg) = 1 and class 1 (pos) = 10.

The following graphics contains those information, from top to bottom: ROC and AUC information, classification report, and confusion metrics.

By observing confusion matrix, we can conclude that there are 44 "pos" cases was recognized as "neg" (False negative) and 1 "neg" recognized as "pos"(False positive) when using train data set for classification. Misclassification in train set is calculated by equation:

$$M = \frac{(FP+FN)}{Total} = \frac{45}{60000} = 0.075\%$$

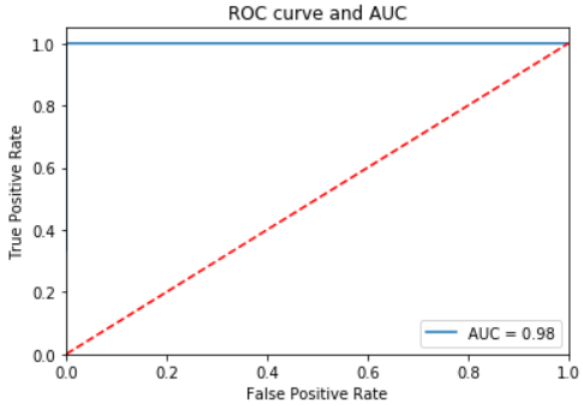
OOB error is about 0.94%

.

By observing confusion matrix, we can conclude that there are 149 "pos" cases was recognized as "neg" (False negative) and 13 "neg" recognized as "pos"(False positive) when using test data set for classification. Misclassification in test set is calculated by equation:

$$M = \frac{(FP+FN)}{Total} = \frac{149+13}{60000} = 1.01\%$$

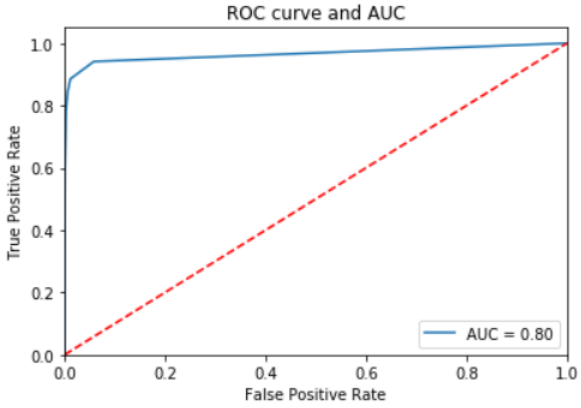
Test error(1%) is also very close to OOB error.



	precision	recall	f1-score	support
0	1.00	1.00	1.00	59000
1	1.00	0.96	0.98	1000
avg / total	1.00	1.00	1.00	60000

```
[[58999     1]
 [   44   956]]
```

Figure 10: Classification result of Random Forest using balanced train data



	precision	recall	f1-score	support
0	0.99	1.00	0.99	15625
1	0.95	0.60	0.74	375
avg / total	0.99	0.99	0.99	16000

```
[[15612    13]
 [  149   226]]
```

Figure 11: Classification result of Random Forest using balanced test data

(e) Model trees

This question requests us to train a logistic model tree. To do this, we need to call weka using python's package python-weka-wrapper and using JVM to do the training. To get python weka wrapper, we need to

get java together with javabirdge(installed by conda in python).

Firstly, we need to convert pd.DataFrames into data instances that used by weka. This involved in using function *ndarray_to_instance* and convert the labels from numeric to nomial. Since the dataset is really big and the training time can be really long, I only used $\frac{1}{10}$ of train and test data after resampling.

Then, we put the data into cross validation with 5-fold, 10-fold and LOOCV(fold = n). During training , I found that LOOCV costs too much time for my computer to finish the job, so I gave up LOOCV. And the results of comfusion matrix and ROC/AUC curve is shown in the figures below.

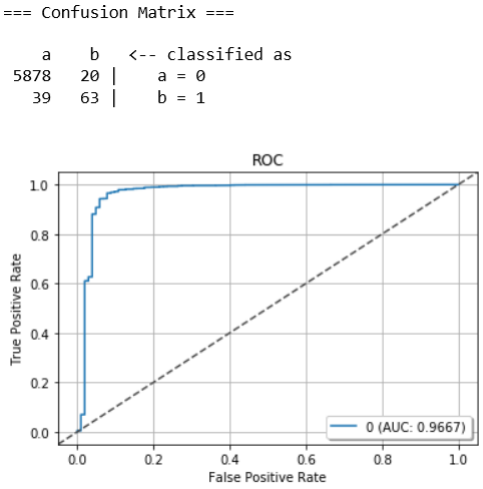


Figure 12: Confusion Matrix and ROC curve of 5-fold CV

By observing confusion matrix for 5-fold, we can conclude that there are 39 "pos" cases was recognized as "neg" (False negative) and 20 "neg" recognized as "pos"(False positive) among 6000 pieces of data.
Mean error rate for 5-fold cv is 0.983% and AUC result is shown on the graph. For detailed code and classification report, see Jupyter Notebook.

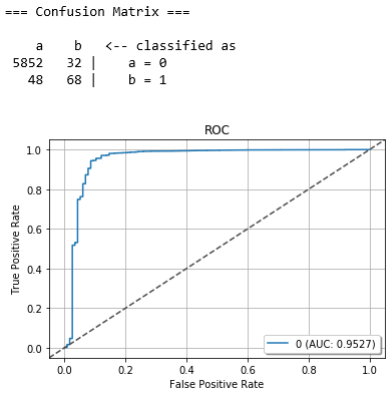


Figure 13: Confusion Matrix and ROC curve of 10-fold CV

By observing confusion matrix for 10-fold CV, we can conclude that there are 48 "pos" cases was recognized as "neg" (False negative) and 32 "neg" recognized as "pos"(False positive) among 6000 pieces of data.
Mean error rate for 10-fold cv is 1.333% and AUC result is shown on the graph. For detailed code and classification report, see Jupyter Notebook.

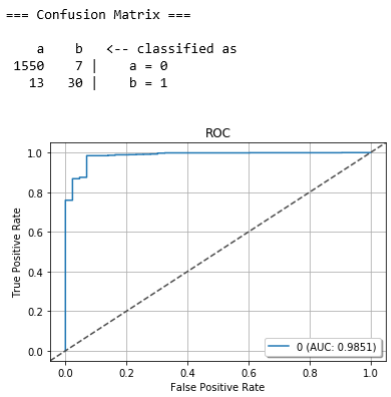


Figure 14: Confusion Matrix and ROC curve of testset

By observing confusion matrix for testset we can conclude that there are 13 "pos" cases was recognized as "neg" (False negative) and 7 "neg" recognized as "pos"(False positive) among 1600 pieces of data.
Mean error rate for testset is 1.25% and AUC result is shown on the graph. For detailed code and classification report, see Jupyter Notebook.

There is no significant different in error rate of testset and CV results. I guess the reason is that the class is really imbalanced in this dataset.

(f) SMOTE

I used imblearn's smote function to preprocess the data. Need to install imblearn from conda.

Then, we put the data into cross validation with 5-fold and 10-fold. And the results of comfusion matrix and ROC/AUC curve is shown in the figures below.

Since SMOTE will generate a number of samples, I chose a smaller subsample to make it faster. Size of original traindata is 1200 and for testdata is 320.

Firstly, use SMOTE to preprocess traindata and testdata. After processing, the number of label 0 and 1 are almost equal.

Then, convert data to instances and do training just like what we do in section (e). And the results of comfusion matrix and ROC/AUC curve is shown in the figures below.

By observing confusion matrix for 5-fold, we can conclude that there are 0 "pos" cases was recognized as "neg" (False negative) and 18 "neg" recognized as "pos"(False positive) 2358 pieces of data.
Mean error rate for 5-fold cv is 0.763% and AUC result is shown on the graph. For detailed code and classification report, see Jupyter Notebook.

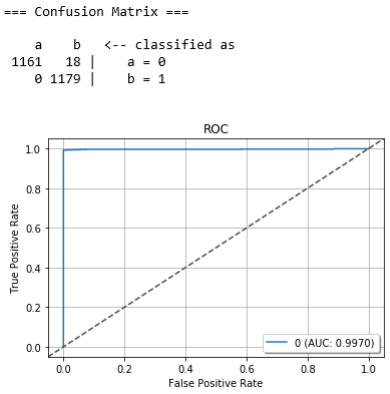


Figure 15: Confusion Matrix and ROC curve of 5-fold CV

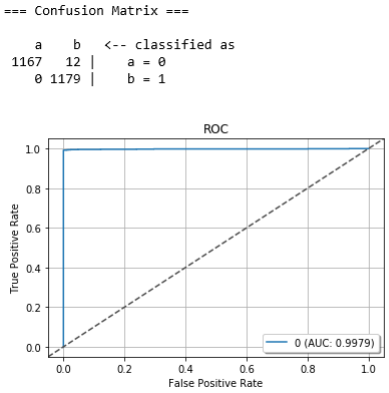


Figure 16: Confusion Matrix and ROC curve of 10-fold CV

By observing confusion matrix for 10-fold CV, we can conclude that there are 0 "pos" cases was recognized as "neg" (False negative) and 12 "neg" recognized as "pos"(False positive) among 6000 pieces of data.
Mean error rate for 10-fold cv is 0.509% and AUC result is shown on the graph. For detailed code and classification report, see Jupyter Notebook.

By observing confusion matrix for testset we can conclude that there are 0 "pos" cases was recognized as "neg" (False negative) and 4 "neg" recognized as "pos"(False positive) among 1600 pieces of data.
Mean error rate for testset is 0.635% and AUC result is shown on the graph. For detailed code and classification report, see Jupyter Notebook.

In conclusion, there are slightly improvements in accuracy after applying SMOTE to do data preprocsessing comparing to training on uncompensated data.

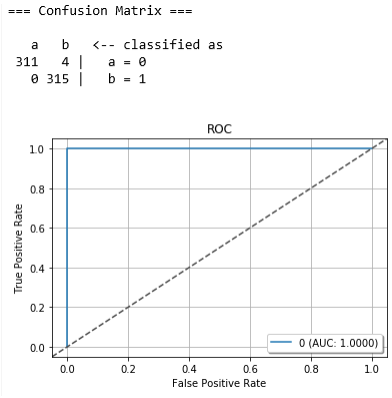


Figure 17: Confusion Matrix and ROC curve of testset

Exercise # 3

ISLR 6.8.3

We notice that the original Least Square Regression is to minimize:

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{i=1}^{n}\beta_j x_{ij})^2$$

1. Training RSS: It will decrease steadily (iv). As the value of s increase, the regression will approach to Least Square Regression. So it will fit to the train data better and better.

2. Test RSS: It will decrease initially and then increase (ii). As s increases from zero, it starts to fits on test data so RSS decrease. And when it approaches Least Squares, it begins to overfit on the train data, so RSS increase.

3. Variance: The variance will keep increase (iii). When s = 0 var = 0, and as s increase, as the model is fitting, bias decrease and variance increase for the train data.(bias-variance trade-off)

4. Bias: Keep decrease (iv). Just like the previous question, because of the trade-off, increase in variance together with decrease in bias.

5. Irreducible error: Constant (v), irreducible error is independent from models or regression methods.

Exercise # 4

ISLR 6.8.5

1. Ridge regression is to minimize:

$$\sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \sum_{j=1}^{p}\hat{\beta}_j x_{ij})^2 \text{ subject to } \sum_{j=1}^{p}\hat{\beta}_j^2 \leq s$$

And we apply n=2, p=2, $\hat{\beta}_0 = 0$, we are minimizing:

$$(y_1 - \hat{\beta}_1 x_{11} - \hat{\beta}_2 x_{12})^2 + (y_2 - \hat{\beta}_1 x_{21} - \hat{\beta}_2 x_{22})^2 \text{ subject to } \sum_{j=1}^{2} \hat{\beta}_j^2 \leq s$$

2. To find minimum, we apply all the conditions: let $x_{11} = x_{12} = A$, $x_{21} = x_{22} = -A$, $y_1 = -y_2 = B$, so we are minimizing:

$$2(B - (\hat{\beta}_1 + \hat{\beta}_2)A)^2 \text{ subject to } \hat{\beta_1^2} + \hat{\beta_2^2} \leq s$$

And we can find the minimum is 0 and the solution is $\hat{\beta}_1 + \hat{\beta}_2 = \frac{B}{A}$. It is a line that intersect with x-axis on $(\frac{B}{A}, 0)$ and with y-axis on $(0, \frac{B}{A})$. And we also know that the plot of constraint of ridge regression is a circle on the original. And by some mathematical proof and observation, we can find that the two intersections (solutions of $\hat{\beta}_1$ and $\hat{\beta}_2$)of the line and the circle hold that one's x equals to the other's y. ($\hat{\beta}_1 = \hat{\beta}_2$)

3. Lasso is to minimize:

$$\sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \sum_{j=1}^{p} \hat{\beta}_j x_{ij})^2 \text{ subject to } \sum_{j=1}^{p} |\hat{\beta}_j| \leq s$$

And we apply n=2, p=2, $\hat{\beta}_0 = 0$, we are minimizing:

$$(y_1 - \hat{\beta}_1 x_{11} - \hat{\beta}_2 x_{12})^2 + (y_2 - \hat{\beta}_1 x_{21} - \hat{\beta}_2 x_{22})^2 \text{ subject to } \sum_{j=1}^{2} |\hat{\beta}_j| \leq s$$

4. Here is Lasso constraints: $|\hat{\beta}_1| + |\hat{\beta}_2| \leq s$, a diamond centered on the origin in plot.

   And we minimize: $2(B - (\hat{\beta}_1 + \hat{\beta}_2)A)^2$(As shown in previous question) whose solution is $\hat{\beta}_1 + \hat{\beta}_2 = \frac{B}{A}$, a line. And we can find that it is parallel to two of the edges of the diamond.

   So the solution set is the entire line $\hat{\beta}_1 + \hat{\beta}_2 = s$ and $\hat{\beta}_1 + \hat{\beta}_2 = -s$. So the solution is not unique.

Exercise # 5

ISLR 8.4.5

1. Majority vote: vote for Red = 6, vote for Green = 4, so it is red.

2. Average probability:

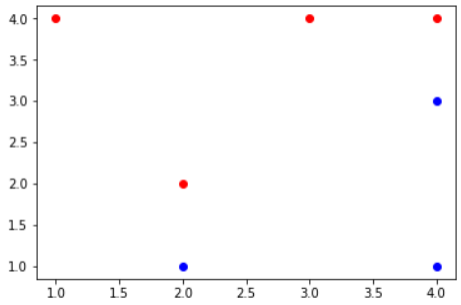
$$avg\_prob = \frac{1}{10} \sum_{i=1}^{10} p = \frac{4.5}{10} = 0.45 < 0.5$$

So it is Green.

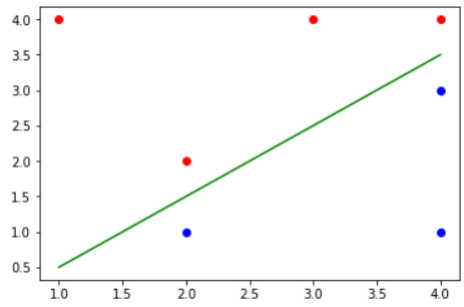
Exercise # 6

ISLR 9.7.3

1. Sketch observations:



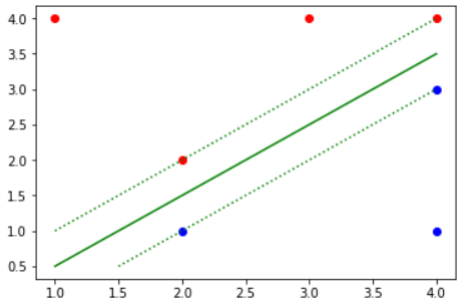
2. We define a line that crosses (2,1.5) and (4,3.5) as our hyperplane (gives minimum distance to all observed points). expression: $X_2 = X_1 - 0.5$



3. Classification rule:

$$IF\ 0.5 - X_1 + X_2 > 0,\ Y\ is\ red,\ otherwise\ it\ is\ blue$$

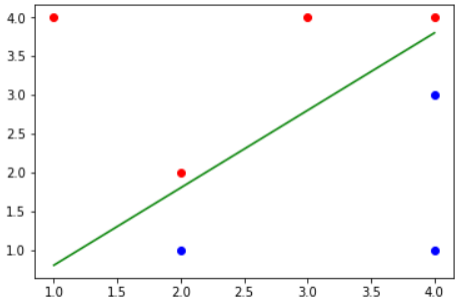
4. Indicate margin: dotted lines.



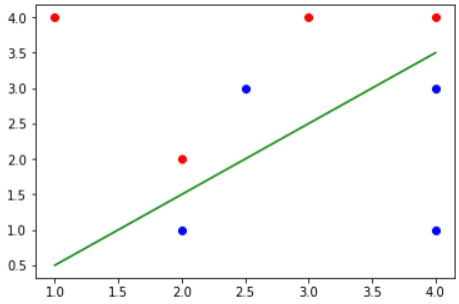
5. Svms are: (2,1,blue) (2,2,red) (4,3,blue) and (4,4,red).

6. A slight movement of (4,1,blue) will not affect the hyperplane because it is far away outside the maximum margin.

7. Not optimal:



8. No longer separated:



Exercise # 7

EXTRA ISLR 5.4.2 Bootstrap

1. $\frac{n-1}{n}$
2. $\frac{n-1}{n}$
3. for every sample it has a probability that it is jth observation is $\frac{1}{n}$, so for not in, it is $\frac{n-1}{n}$. In total, it is $(1-1/n)^n$.
4. $1-\frac{4}{5}^5 = 67.2\%$
5. $1-\frac{99}{100}^{100} = 63.4\%$
6. $1-\frac{10000-1}{10000}^{10000} = 63.2\%$
7. plot
8. pass

Exercise # 8

EXTRA ISLR 6.8.4

1. Steadily increase
2. Decrease then increase
3. Steadily decrease
4. Steadily increase
5. remain constant

Exercise # 9

EXTRA ISLR 9.7.2 Done by hand

Exercise # 10

EXTRA ISLR 8.8.4 Sketch trees, Done by hand