Data Analysis

Instructor: prof. Panagiotidou Sofia

2nd assignment

Aristotle University of Thessaloniki
Department of Mechanical Engineering

Dimitris Nentidis

Contents

1 Introduction

This assignment essentially constitutes a continuation of the previous one. While in the first assignment, models were compared based on parameters such as the R^2 or the p-value in order to draw conclusions regarding the fit or the statistical significance respectively, here, in the first part, a comparison of methods is carried out with the criterion being the mean prediction error of the testing data (test MSE) which results from the different runs of the cross-validation method. The methods being compared are the multiple regression that was examined in the second sub-question of the first assignment and the forward selection that was examined in the third. Then, an additional comparison is carried out between k-fold cross-validation and Leave-One-Out Cross-Validation for the better of the two methods examined with the first.

In the second part, the data is divided into two subsets, the training and the testing set, and the response is converted into a binary variable: one for every value that is above the average and 0 when it is below. Then, some analyses are run with different parameters, examining each time the estimation of accuracy on the testing as well as the training data, drawing conclusions about the bias and variance of the models. The methods that are examined here are the method of Logistic Regression, the Linear Discriminant Analysis, as well as the K-nearest-neighbors method.

2 Part A

In the first part, it is important to observe that we are not comparing models but methods. There is no specific model being examined; more than one model results for each method, and the average performance of these is taken into account, which is considered representative for each method.

2.1 Cross-Validation Method

In the case of the cross-validation method, with $k = 5$, a total of 4 models result. Here, the full dataset is divided into 5 subsets, 4 of which form the training data set and the 5th takes on the role of the

testing set. This process is repeated until all subsets have played the role of the testing set.

In each run, the training data is used to find the optimal model with the best fit, while the training MSE is measured along the way. Then, the remaining data of the 5th set is used to evaluate the model that resulted, thereby producing the test MSE.

In this sub-question, the objective is to evaluate different methods for finding the best model, the one with the minimum training MSE. The two methods examined are **Multiple Linear Regression**, where all the predictive variables are taken into account, and **Forward Selection**, where variables that pass a significance threshold and improve the model are selected. In this case, by defining the p-value, the level of significance, at different values, it changes which of the two methods has the upper hand in terms of performance with respect to test MSE. The results are summarized in the table below:

p-value	Forward Selection MSE
0.05	13.3707
0.01	13.3925
0.001	13.5223

Table 1: Mean test MSE for different p-values.

Multiple Linear Regression, regardless of the 3 possible p-values examined, has the same mean estimated error on the test data, MSE = 13.3794. It is evident that as the p-value decreases, the Multiple Linear Regression method gains the advantage. However, the minimum mean test MSE overall appears for p-value equal to 0.05 and the Forward Selection method. Therefore, with these parameters, the analysis in the next question will proceed.

2.2 Leave-One-Out Cross-Validation

This method is essentially a sub-case of the cross-validation method. Instead of dividing the data into k subsets, each observation constitutes a set by itself. As described in the name of the method, each time the training subset consists of all observations except one, which serves as the testing set. As before, the process continues until all observations

have played the role of the testing set. It is easily understood that k here is equal to the number of observations.

In this sub-question, a comparison is made between the prevailing method that resulted from the previous question using k-fold-cross-validation and its performance when using LOOCV. In this case, a p-value equal to 0.05 is selected and the resulting method is Forward Selection with test MSE equal to 13.3707. Running the algorithm, it turns out that using Leave-One-Out and a significance level of 0.05 for the variable selection with forward selection, the test MSE drops to 13.2232, lower than the value obtained with only 5 subsets.

The LOOCV method uses a larger portion of the data each time, having all data except one observation, thereby significantly reducing the bias. However, the same fact can lead to increased variance for specific models that are affected by outliers. On the other hand, having only 5 sets in the 5-fold can create patterns that affect the performance and lower the overall effectiveness of the method. In this case, both the variance of 5-fold is greater than that of LOOCV and the bias is greater as well, not having access to the same data. Therefore, the test MSE of LOOCV is considered more reliable.

Finally, it is noteworthy that the total time required to implement this method is orders of magnitude more than for k-fold-cross-validation. This is entirely logical, judging by how many times the process had to be repeated with different sets. Perhaps a good middle ground would be to increase the number of folds from 5 to 10, the time would not increase so drastically, and there would be a more reliable estimation of the mean error.

3 Part B

As mentioned in the introduction, in this part the response, Life Expectancy, is converted into a binary variable, taking the value 0 when the response is below its mean value and 1 when it is above. Then, the data is split into training data, 80% of the total observations, and testing data, 20% of the observations. Further information about the models and the coefficients of the predictive variables can be found in the corresponding section and is not included in this part.

The general framework within which the comparison of the models

is carried out is the relationship between bias and variance. Bias is a measure of how well the model fits the training data, while variance essentially describes how well the model generalizes to data it has not seen before.

To compare the models, the following parameters are used. First of all, the testing and training errors. These are defined by first calculating the accuracy of each model and then subtracting this accuracy from one. The smaller the error, the higher the accuracy of the model in fitting the respective data, whether training or testing. For example, for logistic regression, the following code snippet is used:

```
# Calculate accuracies and errors
train_accuracy_lr = accuracy_score(y_train, y_train_pred_lr)
test_accuracy_lr = accuracy_score(y_test, y_test_pred_lr)
train_error_lr = 1 - train_accuracy_lr
test_error_lr = 1 - test_accuracy_lr
```

Then, it is important to consider the confusion matrices. Essentially, these are tables that show how many observations were correctly predicted (true positives, 1, and true negatives, 4) and how many were incorrect (false positives, 3, and false negatives, 2). These matrices are a particularly useful tool as, beyond understanding the nature of the statistical error, they allow us to have a very clear picture of the classification of the observations.

Completing the picture is the classification report. In the first column, it gives the percentage distribution of successful classifications from the confusion matrix, recall which is defined as the number of true positives divided by the sum of true positives and false negatives—that is, essentially the total number of positives—and the f1-score which is the average of precision and recall.

Before proceeding to the methods, it is critical to give an explanation for the nature of the threshold. Essentially, it is the value between 0 and 1 above which a response is considered high.

3.1 Logistic Regression

Logistic Regression is a method for classifying responses of observations into two value levels, high and low. In reality, Logistic Regression as a procedure closely resembles Multiple Linear Regression, with the

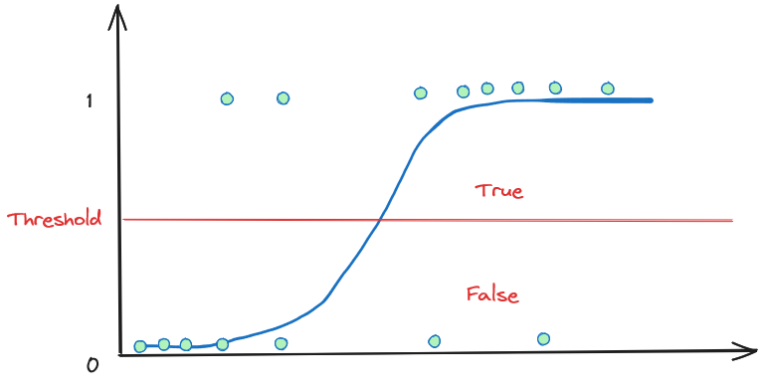

Figure 1: Schematic of a threshold.

difference that here the coefficients resulting for the variables associate the observations with the natural logarithm of the probability of having a high in an observation, performing a transformation on the data.

The models that resulted are the following:

3.1.1 Threshold = 0.4

Training Data:

```
Confusion Matrix:
[[453, 134],
 [ 51, 681]]
Classification Report:
              precision    recall  f1-score   support
           0       0.90      0.77      0.83       587
           1       0.84      0.93      0.88       732
```

Testing Data:

```
Confusion Matrix:
[[ 97,  30],
 [ 14, 189]]
Classification Report:
              precision    recall  f1-score   support
           0       0.87      0.76      0.82       127
```

```
           1       0.86      0.93      0.90       203
```

3.1.2 Threshold = 0.5

Training Data:

```
Confusion Matrix:
[[490,  97],
 [ 84, 648]]
Classification Report:
              precision    recall  f1-score   support
           0       0.85      0.83      0.84       587
           1       0.87      0.89      0.88       732
```

Testing Data:

```
Confusion Matrix:
[[103,  24],
 [ 25, 178]]
Classification Report:
              precision    recall  f1-score   support
           0       0.80      0.81      0.81       127
           1       0.88      0.88      0.88       203
```

3.1.3 Threshold = 0.6

Training Data:

```
Confusion Matrix:
[[512,  75],
 [129, 603]]
Classification Report:
              precision    recall  f1-score   support
           0       0.80      0.87      0.83       587
           1       0.89      0.82      0.86       732
```

Testing Data:

```
Confusion Matrix:
[[109,  18],
 [ 40, 163]]
Classification Report:
              precision    recall  f1-score   support
           0       0.73      0.86      0.79       127
           1       0.90      0.80      0.85       203
```

Looking at the above data, it is evident that the highest accuracy appears for threshold = 0.5 with regard to training error and for 0.4 with regard to testing error. A summary follows in the table below.

Threshold	**Training Error**	**Testing Error**
0.4	0.1403	0.1333
0.5	0.1372	0.1485
0.6	0.1547	0.1758

Table 2: Logistic Regression errors.

Overall, all the models appear to have relatively low bias and similar error on both data sets, which also indicates low variance. For threshold 0.6 both of these quantities increase. Changing the threshold appears to affect the specificity (true negative rate) and the sensitivity (true positive rate) of the models. Lowering the threshold leads to an increase in specificity, while increasing it leads to an increase in sensitivity. The fact that the testing error is lower than the training error for 0.4, although unusual, may be due to specific values that create a better fit for that particular model.

It is worth noting that, with the use of the sci-kit learning package, for this method the default number of iterations run by the algorithm was not sufficient to optimize the model. The number of iterations was increased to 5000, without resolving the issue, despite the increase in computational time.

3.2 Linear Discriminant Analysis

LDA attempts to classify the data by finding linear associations between the predictive variables, aiming to reduce the model's dimensionality.

3.2.1 Threshold = 0.4

Training Data

```
Confusion Matrix:
 [[458, 129],
  [ 22, 710]]
Classification Report:
              precision    recall  f1-score   support
           0       0.95      0.78      0.86       587
           1       0.85      0.97      0.90       732
```

Testing Data

```
Confusion Matrix:
 [[101, 26],
  [  3, 200]]
Classification Report:
              precision    recall  f1-score   support
           0       0.97      0.80      0.87       127
           1       0.88      0.99      0.93       203
```

3.2.2 Threshold = 0.5

Training Data

```
Confusion Matrix:
 [[481, 106],
  [ 40, 692]]
Classification Report:
              precision    recall  f1-score   support
           0       0.92      0.82      0.87       587
           1       0.87      0.95      0.90       732
```

Testing Data

```
Confusion Matrix:
 [[105, 22],
  [  8, 195]]
Classification Report:
              precision    recall  f1-score   support
           0       0.93      0.83      0.88       127
           1       0.90      0.96      0.93       203
```

3.2.3 Threshold = 0.6

Training Data

```
Confusion Matrix:
 [[496, 91],
  [ 62, 670]]
Classification Report:
              precision    recall  f1-score   support
           0       0.89      0.84      0.87       587
           1       0.88      0.92      0.90       732
```

Testing Data

```
Confusion Matrix:
 [[107, 20],
  [ 13, 190]]
Classification Report:
              precision    recall  f1-score   support
           0       0.89      0.84      0.87       127
           1       0.90      0.94      0.92       203
```

In general, both bias and variance in the LDA models are low. The models have a fairly satisfactory fit on the training data, and it is clear that they generalize very well, as the two errors are very close. Changes in the threshold do not drastically affect the performance of the model and allow for fine tuning to improve sensitivity and specificity without one greatly affecting the other, as was the case in logistic regression.

Threshold	Training Error	Testing Error
0.4	0.1145	0.0879
0.5	0.1107	0.0909
0.6	0.1160	0.1000

Table 3: LDA errors.

Specificity seems to be increased almost every time, especially with 0.4, without reducing sensitivity.

It is interesting that despite the significantly smaller errors that result from using this method, the test error is smaller than the training error for every threshold value. One possible explanation is that the training set observations have some outliers which increase the training error. This is not considered likely since all methods are evaluated using the same data. Increasing the size of the training set from 80% of the total data to 90%, and for threshold equal to 0.5, it is observed that the training data error drops below the testing data error, with values 0.1037 and 0.1151 respectively, however this is not part of the present report and therefore there will be no detailed reference to the model.

3.3 K-nearest-neighbors

The K-nearest-neighbors method is based on the idea that in a diagram of some parameters in N dimensions, neighboring observations will have a common response. Therefore, the response can be predicted by considering the location of an observation in the space of all observations. A subspace of N dimensions is created that contains within it the K nearest observations. Then, the response is classified based on the response of the majority of the K nearest neighbors.

3.3.1 Neighbors k = 1

Training Data

```
Confusion Matrix:
 [[587   0]
 [  0 732]]
```

```
Classification Report:
              precision    recall  f1-score   support
           0       1.00      1.00      1.00       587
           1       1.00      1.00      1.00       732
```

Testing Data

```
Confusion Matrix:
 [[ 77  50]
 [ 70 133]]
Classification Report:
              precision    recall  f1-score   support
           0       0.52      0.61      0.56       127
           1       0.73      0.66      0.69       203
```

3.3.2 Neighbors k = 3

Training Data

```
Confusion Matrix:
 [[477 110]
 [140 592]]
Classification Report:
              precision    recall  f1-score   support
           0       0.77      0.81      0.79       587
           1       0.84      0.81      0.83       732
```

Testing Data

```
Confusion Matrix:
 [[ 81  46]
 [ 69 134]]
Classification Report:
              precision    recall  f1-score   support
           0       0.54      0.64      0.58       127
           1       0.74      0.66      0.70       203
```

3.3.3 Neighbors k = 5

Training Data

```
Confusion Matrix:
 [[432 155]
 [155 577]]
Classification Report:
              precision    recall  f1-score   support
           0       0.74      0.74      0.74       587
           1       0.79      0.79      0.79       732
```

Testing Data

```
Confusion Matrix:
 [[ 86  41]
 [ 75 128]]
Classification Report:
              precision    recall  f1-score   support
           0       0.53      0.68      0.60       127
           1       0.76      0.63      0.69       203
```

3.3.4 Neighbors k = 7

Training Data

```
Confusion Matrix:
 [[421 166]
 [188 544]]
Classification Report:
              precision    recall  f1-score   support
           0       0.69      0.72      0.70       587
           1       0.77      0.74      0.75       732
```

Testing Data

```
Confusion Matrix:
 [[ 84  43]
 [ 75 128]]
```

```
Classification Report:
              precision    recall  f1-score   support
           0       0.53      0.66      0.59       127
           1       0.75      0.63      0.68       203
```

3.3.5 Neighbors k = 9

Training Data

```
Confusion Matrix:
 [[389 198]
 [187 545]]
Classification Report:
              precision    recall  f1-score   support
           0       0.68      0.66      0.67       587
           1       0.73      0.74      0.74       732
```

Testing Data

```
Confusion Matrix:
 [[ 81  46]
 [ 71 132]]
Classification Report:
              precision    recall  f1-score   support
           0       0.53      0.64      0.58       127
           1       0.74      0.65      0.69       203
```

Number of Neighbors	Training Error	Testing Error
1	0.0000	0.3636
3	0.1895	0.3485
5	0.2350	0.3515
7	0.2684	0.3576
9	0.2919	0.3545

Table 4: KNN errors.

As expected, with only one neighbor, the training error is necessarily zero. Essentially, this just tells us that the neighbor is always its own neighbor. As the number of neighbors increases, the training error also increases, approaching the testing error, which does not change drastically. For one neighbor, it is the largest, however the difference is slightly more than 4% from the best performance, which is at two neighbors. What is striking is that regardless of the number of neighbors, the accuracy is much better in predicting the positives compared to the negatives. So, in short, this method has low bias and high variance for the given set of observations.

This method differs from methods like Logistic Regression. Here there are no coefficients for the predictive variables or intercepts for the slope.

3.4 Comparison

Summarizing all of the above, it can be said that LDA consistently has training and testing errors lower than Logistic Regression, having overall better fit and better accuracy. At the same time, it has lower variance, being better able to generalize its models to data it hasn't seen before. This happens because the data do not have strong non-linear relationships, and the assumptions for normality and equal co-variances hold.

As for KNN, it generally makes no assumptions about the data, and therefore for sets where there are complex non-linear relationships it may be equally capable in prediction. However, here, the performance of the models is clearly worse.

4 Models created

4.1 Logistic Regression

Table 5: Logistic Refression coefficients and intercept for different thresholds.

Variable	**Threshold 0.4**	**Threshold 0.5**	**Threshold 0.6**
Intercept	2.6900×10^{-6}	1.8416×10^{-6}	2.6900×10^{-6}
x_1	-1.4814×10^{-3}	-1.4819×10^{-3}	-1.4814×10^{-3}
x_2	-1.3370×10^{-2}	-1.3414×10^{-2}	-1.3370×10^{-2}
x_3	8.0607×10^{-3}	6.2724×10^{-3}	8.0607×10^{-3}
x_4	3.1486×10^{-3}	3.7386×10^{-3}	3.1486×10^{-3}
x_5	6.5633×10^{-3}	8.7562×10^{-3}	6.5633×10^{-3}
x_6	3.1761×10^{-5}	3.4897×10^{-5}	3.1761×10^{-5}
x_7	3.9940×10^{-2}	3.4190×10^{-2}	3.9940×10^{-2}
x_8	-4.3707×10^{-3}	-5.1363×10^{-3}	-4.3707×10^{-3}
x_9	1.2228×10^{-2}	1.2879×10^{-2}	1.2228×10^{-2}
x_{10}	1.8919×10^{-3}	1.5298×10^{-3}	1.8919×10^{-3}
x_{11}	1.6494×10^{-2}	1.6056×10^{-2}	1.6494×10^{-2}
x_{12}	-1.3451×10^{-2}	-1.0350×10^{-2}	-1.3451×10^{-2}
x_{13}	7.1077×10^{-5}	4.2714×10^{-5}	7.1077×10^{-5}
x_{14}	-4.4294×10^{-10}	1.4012×10^{-11}	-4.4294×10^{-10}
x_{15}	-7.4266×10^{-3}	-6.1265×10^{-3}	-7.4266×10^{-3}
x_{16}	7.1989×10^{-4}	5.5155×10^{-4}	7.1989×10^{-4}
x_{17}	9.6267×10^{-3}	7.4146×10^{-3}	9.6267×10^{-3}

4.2 Linear Discriminant Analysis

Table 6: Linear Discriminant Analysis coefficients and intercepts for different thresholds.

Variable	**Threshold 0.4**	**Threshold 0.5**	**Threshold 0.6**
Intercept	220.8543	220.8543	220.8543
x_1	-1.1485×10^{-1}	-1.1485×10^{-1}	-1.1485×10^{-1}
x_2	-9.1135×10^{-3}	-9.1135×10^{-3}	-9.1135×10^{-3}
x_3	-3.4769×10^{-2}	-3.4769×10^{-2}	-3.4769×10^{-2}
x_4	-1.2630×10^{-4}	-1.2630×10^{-4}	-1.2630×10^{-4}
x_5	7.1530×10^{-3}	7.1530×10^{-3}	7.1530×10^{-3}
x_6	1.7716×10^{-5}	1.7716×10^{-5}	1.7716×10^{-5}
x_7	2.2966×10^{-2}	2.2966×10^{-2}	2.2966×10^{-2}
x_8	-1.9087×10^{-3}	-1.9087×10^{-3}	-1.9087×10^{-3}
x_9	2.3331×10^{-3}	2.3331×10^{-3}	2.3331×10^{-3}
x_{10}	1.2093×10^{-1}	1.2093×10^{-1}	1.2093×10^{-1}
x_{11}	-1.3324×10^{-4}	-1.3324×10^{-4}	-1.3324×10^{-4}
x_{12}	-5.5143×10^{-2}	-5.5143×10^{-2}	-5.5143×10^{-2}
x_{13}	1.3157×10^{-5}	1.3157×10^{-5}	1.3157×10^{-5}
x_{14}	-8.2513×10^{-10}	-8.2513×10^{-10}	-8.2513×10^{-10}
x_{15}	-3.1914×10^{-2}	-3.1914×10^{-2}	-3.1914×10^{-2}
x_{16}	6737.66198	6737.66198	6737.66198
x_{17}	477.30827	477.30827	477.30827