EAS506

HW2

Name : Matthew Sah

UBITName : msah

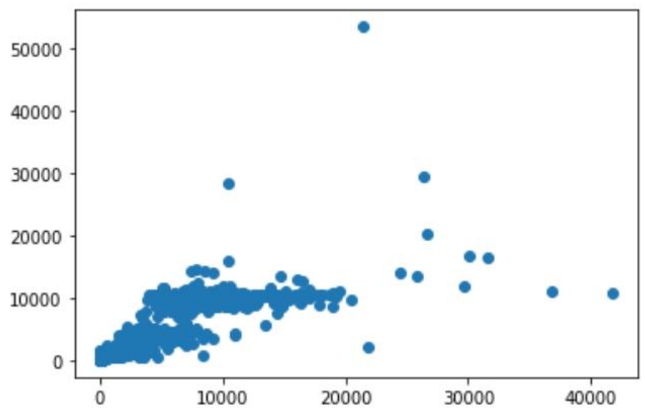
Student Number : 503203765

1)

For question 1, I used python and most of the libraries from SKlearn; mainly PCA, PLSRegression, Ridge, Lasso, cross_val_score, and LinearRegression.

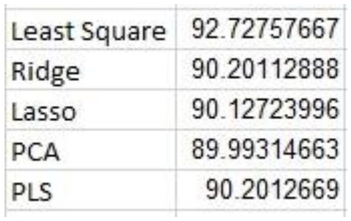
Cleaning the data I removed Apps(target value) and whether it was private from the data set. Splitting the data with an 8:2 train to test ratio. Leaving the feature as a 621*16 data form and the target value as a 156*1 (only including the target value).

With a quick plot we can see the data doesn't have too many outliers and are highly concentrated.



I then started to compare all accuracies from different methods including least squares, ridge regression, lasso, PCA + linear regression, and partial least squares.

The highest accuracy achieved was from least squared with a accuracy of 92%, however the difference between accuracies were not very different with a 3% difference at most.



Least Square	92.72757667
Ridge	90.20112888
Lasso	90.12723996
PCA	89.99314663
PLS	90.2012669

For PCR, due to the lack of library in python, I first had to use a PCA library from skLearn then follow up with a linear regression to achieve PCR.

For the PCR model, while K is at around 6, the accuracy reaches it's peak value, of 89.9% , which is extremely similar to PLS. PLS reaches 89.8% accuracy while K is around 6.

To summarize for question 1, we can predict with a high possibility that we get the amount of applications correct. There is very minor difference between these five approaches.

2)

For question 2 I also used python in jupyter notebook along with some sklearn libraries. I used python since i was more familiar with the libraries and the libraries needed to complete the required task were very detailed documented. Also, for backstep and forward selection, there is a detailed library from mlxtend that simplifies the process by a lot.

For the data I tried to go through the predictors but I didn't feel there were significant values that required discarding. I read all the data into python with pandas and removed the caravan insurance column from the training data into a sperate variable. I also named all the columns for formatting issues.

After formatting the data, i first used ordinary least squares to find its initial accuracy, which I achieved an extremely low accuracy of 8%. I then used backstep and forward selection to acknowledge the more relevant variables.

For subset selection I initially used 3 as the K values. I started with sequential forward selection, the most influential columns were the 42nd, 46th, and 81st columns, and had an accuracy of 3.9%. For backward selection I achieved also 3.9% but with the 17th, 46th, and 81st variable. With further inspection the 81th variable was relevance to the number of surfboard policies, and the 46th variable was whether there was contribution to third party insurance in agriculture. The 42nd was average income and 18th was medium level of education. The 81st and variable made most sense seeing that with a caravan would probably allow better transportation of surfboards.

However, with further inspection with higher levels of K, we could achieve higher accuracy. I tested k = 10 on forward selection and received a 2 percent of increase in accuracy up to 5.4%. With higher values of K does not guarantee higher values of accuracy, I further tested with k = 30 and k = 50 but highest accuracy was peaked at 5%.

Another thing to note is that the backward selection method took significantly the most time. With the second longest method (forward selection at 1.3 seconds), backward selection took 173 seconds.

Ridge regressor resulted with a mean of 3.01% of accuracy and the Lasso regressor finished with 1.3% of accuracy.

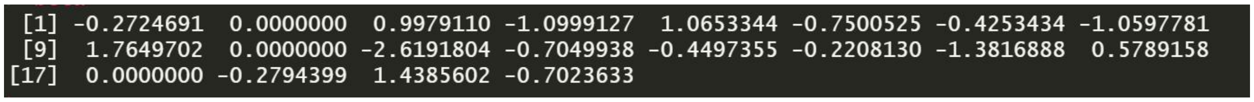
To summarize, it would be pretty hard to predict who would want to buy a caravan insurance but if the more relevant variables were selected, there would be a higher chance of obtaining the best accuracy.

3)

For question 3, I mainly used R in Rstudio. I generated a 1000 * 20 matrix with rnorm to achieve normal distribution. To pick out BETA, I randomly chose three values with the sample function. By assigning 0 to the designated values, I am able to simulate the non-relevant variables.

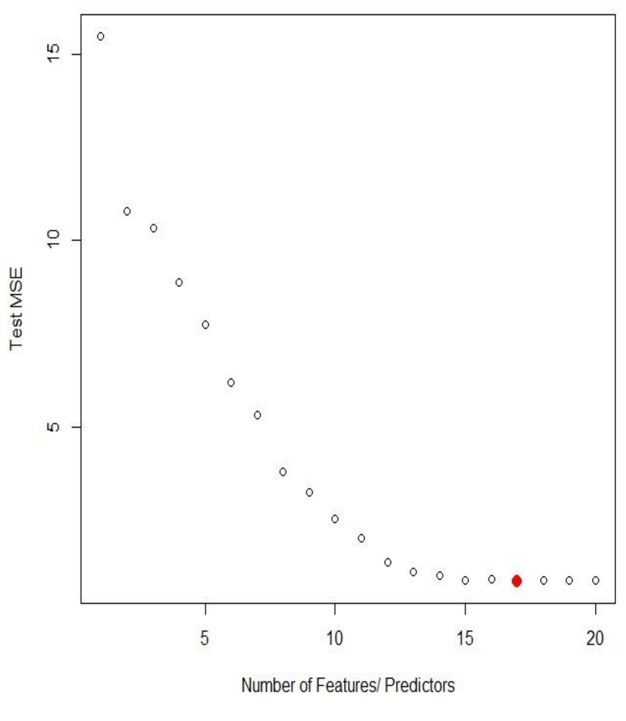
For this example my beta values were at {2, 10 ,17} which can be shown as below. The lowest test MSE appeared then there were 17 out of 20 variables selected. Since three of the variables were intentionally emptied, the best subset selection had correctly selected the non-relevant variables. See the featured labels to see that the BETA variables were left out.

Beta

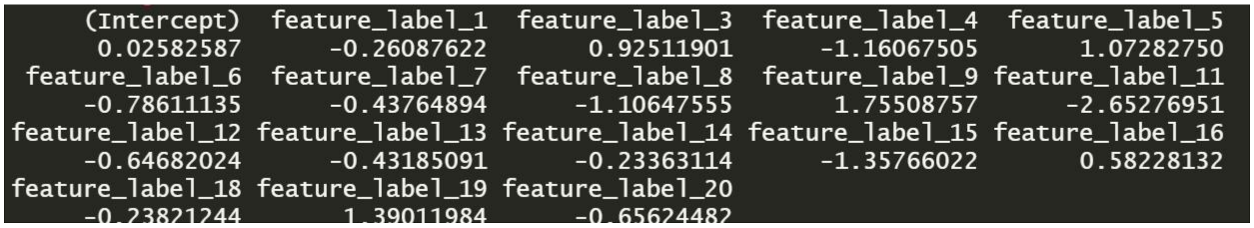


```
 [1] -0.2724691  0.0000000  0.9979110 -1.0999127  1.0653344 -0.7500525 -0.4253434 -1.0597781
 [9]  1.7649702  0.0000000 -2.6191804 -0.7049938 -0.4497355 -0.2208130 -1.3816888  0.5789158
[17]  0.0000000 -0.2794399  1.4385602 -0.7023633
```

Test Mse

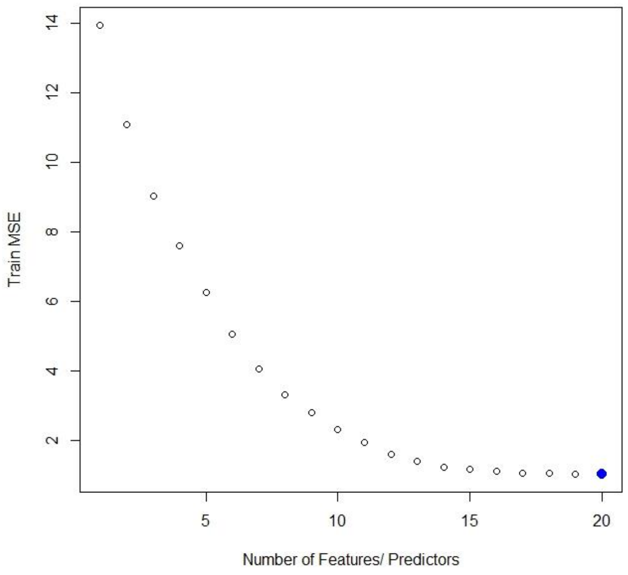


Feature Labels – the emptied variables were left unselected



```
     (Intercept)  feature_label_1  feature_label_3  feature_label_4  feature_label_5
      0.02582587      -0.26087622       0.92511901      -1.16067505       1.07282750
 feature_label_6  feature_label_7  feature_label_8  feature_label_9 feature_label_11
     -0.78611135      -0.43764894      -1.10647555       1.75508757      -2.65276951
feature_label_12 feature_label_13 feature_label_14 feature_label_15 feature_label_16
     -0.64682024      -0.43185091      -0.23363114      -1.35766022       0.58228132
feature_label_18 feature_label_19 feature_label_20
     -0.23821244       1.39011984      -0.65624482
```

Training MSE


Train MSE
Number of Features/ Predictors