Abhishek Pulicherla-199501245857
Jayanthan Subbaiah-199405243917
Jyothi Reddybathula-199603281081

Estimating the Cetane Number of Diesel Fuel

Introduction:

Currently there are two types of engines, Otto and Diesel Engines. The Otto Engine runs on petrol and ignites the fuel by using spark plug.

The Diesel Engine runs on Diesel Fuel. It works by compressing the Fuel-Air mixture. This increases the air temperature inside the cylinder.

In the Diesel engine, the fuel is ignited from compression. The inclination of the diesel fuel to ignite and burn is specified by the Cetane Number. The Normal Cetane Numbers for Diesel Fuel is around 40-50. But the premium Diesel has Cetane number of about 60.

The task in our project is to estimate the Cetane Number of samples of Diesel Fuel using the near Infrared Spectrum (NIR-Spectrum). We have two sets of measured NIR-spectrum and for one of the sets we also have access to the actual Cetane Number. Here, we use Supervised Learning Regression Methods to predict Cetane numbers of the sample set for which the Cetane values are not known. We also find the Generalization Errors in order to determine the correctness of our predictions.

In this project, we have used Matlab. We have used both the Regression Learner Application and the Neural Net fitting Application. In this report, we also have the predictions of the unknown cetane numbers from the Principal Component Analysis (PCA) linear regression model, a Partial Least Squares (PLS), Linear Model and Multi-Layer Perceptron (MLP).

Methodology:

We have used all Supervised Learning Regression Methods in this Project. To predict Cetane Numbers, we use both script-based m-files and the matlab applications in the given dataset cnTestX.

First, the data is imported to the matlab workspace and then we do a Principal Component Analysis (PCA) on the dataset cnTrainX using both matlab command pca in the script-environment and in the regression learning application.

Next, we train a linear model using the regression learner app. We directly implement the linear regression model in the matlab script-environment. We use a 5-fold cross validation to estimate the Generalization errors. We also predict the cetane numbers of the cnTestX data using another linear model called Partial Least Squares(PLS). In this, we don’t have to do PCA before training the model. We implement the PLS method in the script-environment of matlab.

An important part of our implementation is choosing the number of PLS-components to use since it takes advantage of the response of the training data, i.e the dataset cnTrainY, to construct the set of linear combinations used in the linear regression model.

Therefore, we implement the Principal Component Multi-Layer Perceptron (MLP). Hence, we find the optimal values for the number of units in the hidden layer and to use PCA Components.

Abhishek Pulicherla-199501245857
Jayanthan Subbaiah-199405243917
Jyothi Reddybathula-199603281081

Data:

The data sets that we are given consists of a total of 245 observations of the NIR-spectrum where every observation has 401 channels. For 112 of these observations we have no access to the correct cetane number.

The data sets are:
cnTrainX - 133 observations of NIR-spectrum
cnTrainY - The correct cetane numbers of the 133 samples in *cnTrainX*
cnTestX - 112 observations for which we don't have the correct cetane number

The given cetane number in *cnTrainY* , we see that it has a range between 40.7 and 60. The mean value is 48.9.

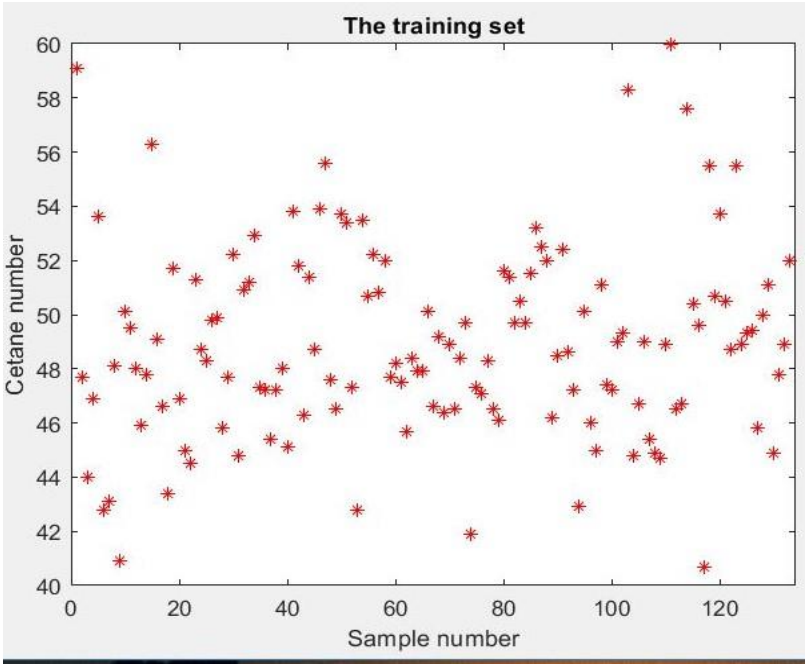

Fig 1: Plot of the cetane numbers found in *cnTrainY*

Looking at the NIR-spectrum data found in *cnTrainX* , we know that it is hard to do any intuitive analysis of the content since we have no specification of how the NIR-spectrum is measured and formed. The numerical values of the data in *cnTrainX* have a range of -0.0398 to 0.0618. The range of the data in *cnTestX* is -0.0399 to 0.0601 and hence not significantly deviating from the range of *cnTrainX*.

Abhishek Pulicherla-199501245857
Jayanthan Subbaiah-199405243917
Jyothi Reddybathula-199603281081

Results:

```
    86.2143    6.2911    4.7145    1.5258    0.4785    0.2105

PCA
PCA is keeping enough components to explain 95% variance.
After training, 3 components were kept.
Explained variance per component (in order): 86.2%, 6.3%, 4.7%, 1.5%, 0.5%, 0.2%
```

Fig 2: The Explained Variance Values of the first six components

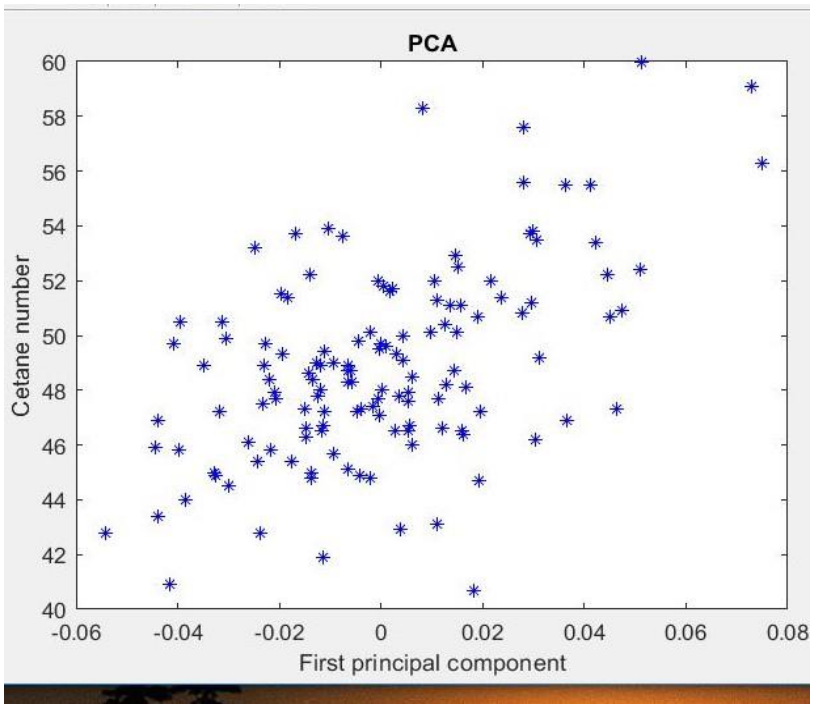

Fig 3: Plot of the cetane numbers found in *cnTrainY* and the first principal component of the NIR-spectrum data set *cnTrainX*

We use the app to train a linear regression model to predict the unknown cetane numbers. The results from training the model is shown in Fig. 4. In this figure we get information about the mean squared error (MSE) and the root mean squared error (RMSE).

Model 1: Trained

Results

RMSE	2.1539
R-Squared	0.63
MSE	4.6393
MAE	1.7479
Prediction speed	~400 obs/sec
Training time	4.5524 sec

Model Type
Preset: Linear
Terms: Linear
Robust option: Off

Feature Selection
All features used in the model, before PCA

Fig 4: The results from the linear regression model using the 3 most significant principal components.

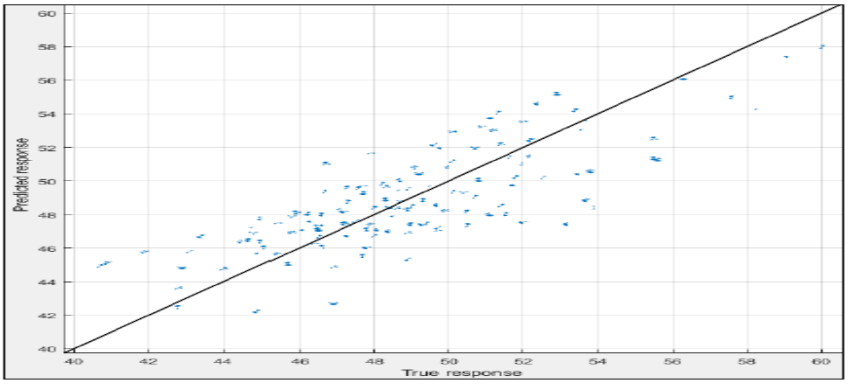

Fig 5: Predicted and true cetane values of *cnTrainY* using the regression learner app

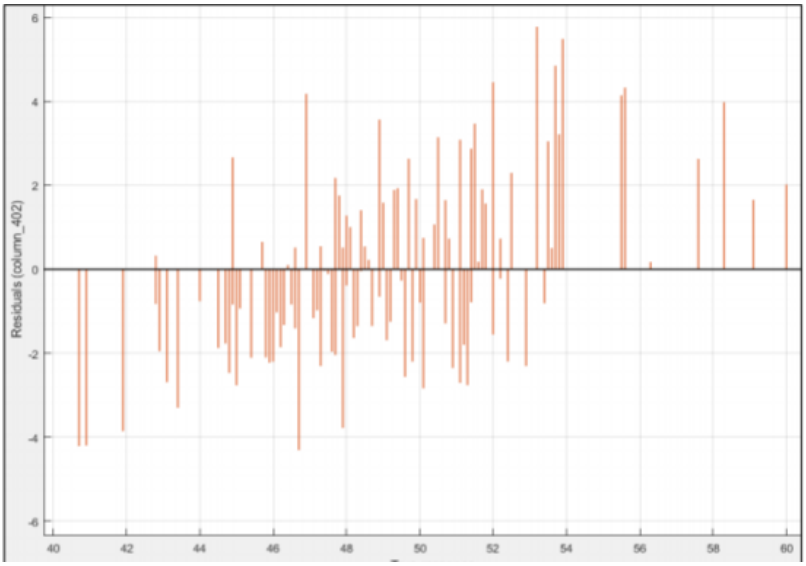

Fig 6: Residuals of the predicted cnTrainY using the regression learner app

Here, we plot the true value of *cnTrainY* and the values predicted by the trained model. As we can see, for low values the prediction is a little high and for high values the prediction is a bit low.

In Fig.7 we show the generalization error by using 5-fold cross validation. In the linear regression learner application, we use 3 PCa-components in manual implementation.

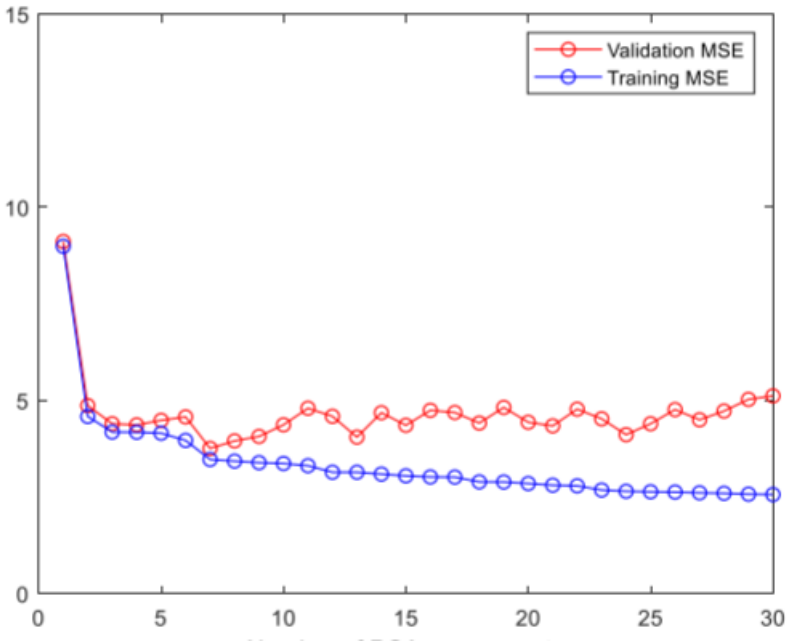

Fig 7: Cross validation and training errors using different number of PCA-components

The above applications we can compare the predicted values of the scripted based implementations.

In Fig 8 we can also plot the residuals. Predicted cetane numbers of cntrainY using the three most significant principal components implemented in the matlab sscript environment.

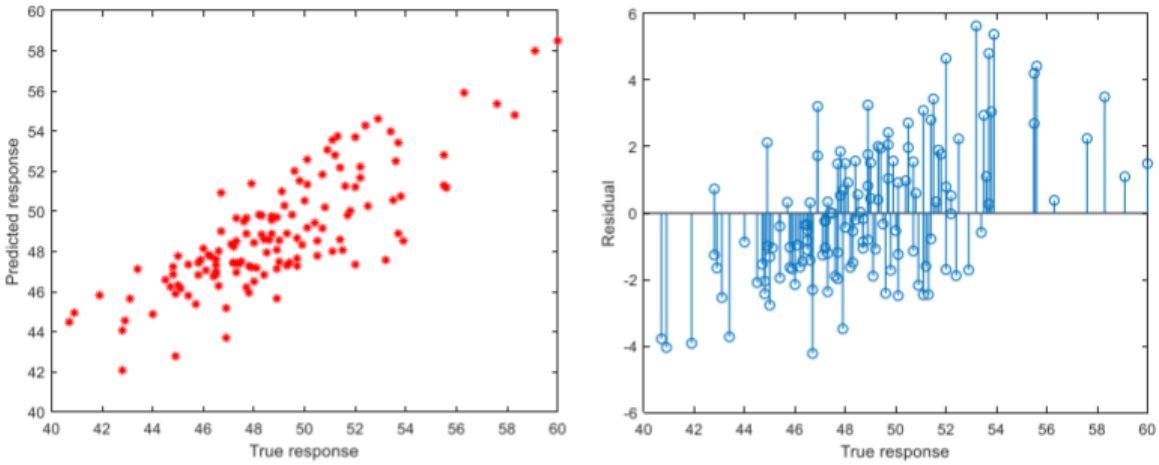

Fig 8: Predicted cetane numbers of cntrainY

In the Fig. 9, here we find the good results. It will show in the histogram representation. We can show the results of this:

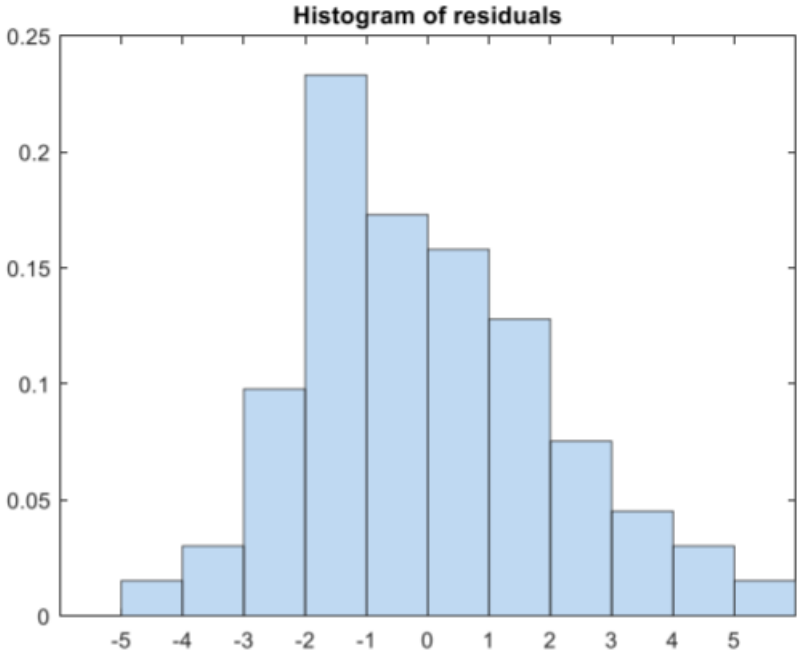

Fig 9: Histogram plot of the residuals

In the PCA linear regression model, in Fig 10 we plot the cetae number predictions for the 112 NIR-spectrum measurements of cntestX using the application. we can estimate that the generalization error of this prediction will have a MSE of 4.64 and RMSE of 2.15.

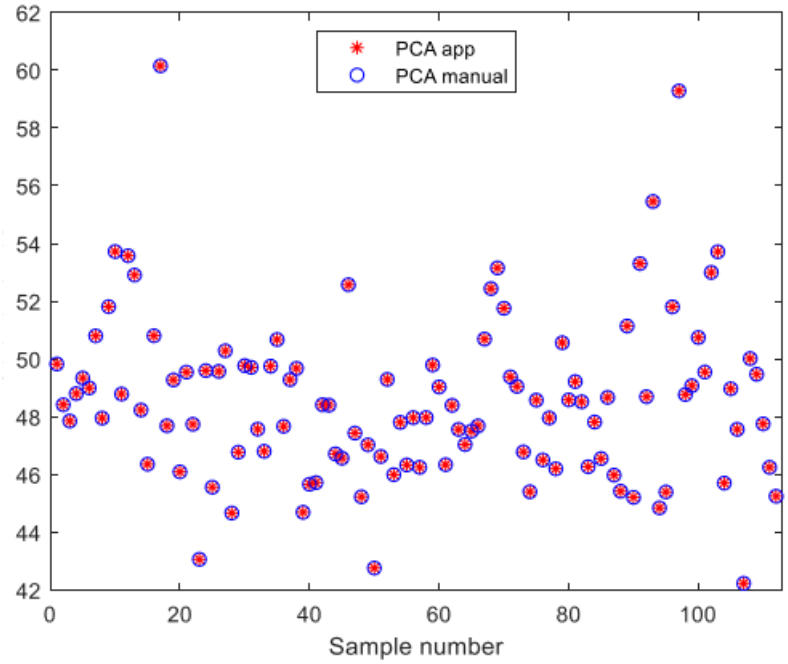

Fig 10: Predictions of the cetane number for the 112 NIR-spectrum measurements of cnTestX using both the regression learner app and a script-based implementation.

In partial least squares (PLS) implementation get the continuous errors, we firstly look at the validation and training errors.in this Fig 11 we use the 5- fold cross validation. Here we use 5 components is a good choice to avoid overfitting.

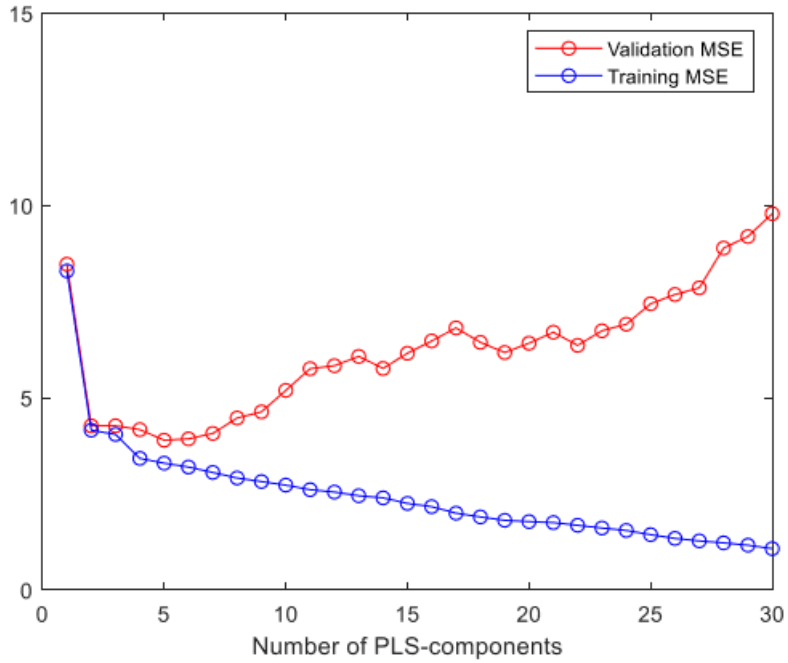

Fig 11: Cross validation and training errors using different number of PLS-components.

In linear regression model we use PCA at the true and predict cetane values of cntrainY in below Fig 12, due to our predictions we use 5 PLS-components.

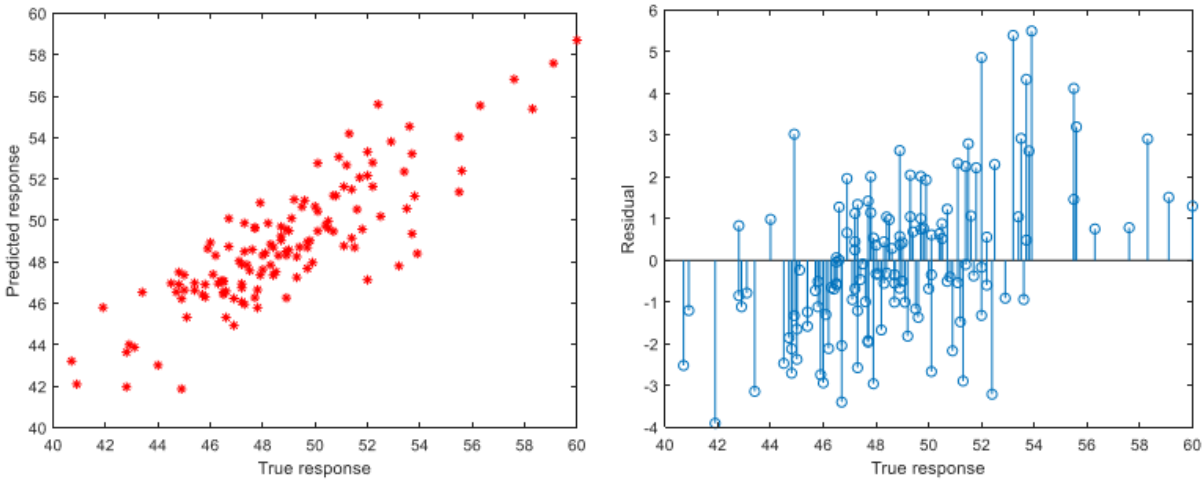

Fig 12: True and predict cetane numbers of cnTrainY using 5PLS-componets.

Here we use this model to predict the cetane numbers. This is shown in blow Fig 13 as the black crosses. The predictions is shown in the red stars and blue circles. It will not coincide for every sample, but the difference is either significantly large.

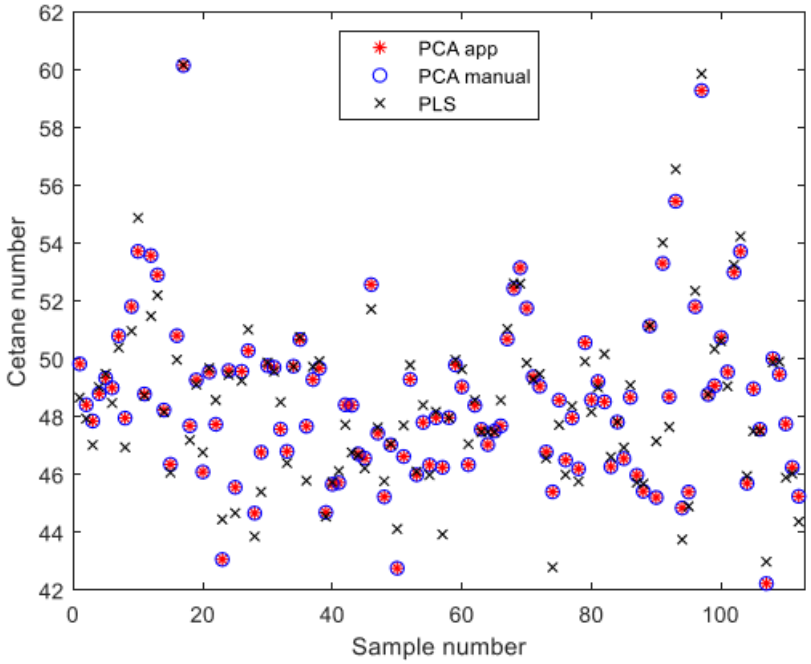

Fig 13: Predictions made using 5 PLS-components for the cetane numbers of the 112 measurements of cnTestX marked as black crosses.

Finally, to predict the cetane numbers of dataset cnTestX we can implement the principal component multi-layer perceptron (MLP). This algorithm is also available in this application called neural net toolbox. While running the script-implementation, Matlab will show the instructions, for example as shown in the below fig 14.

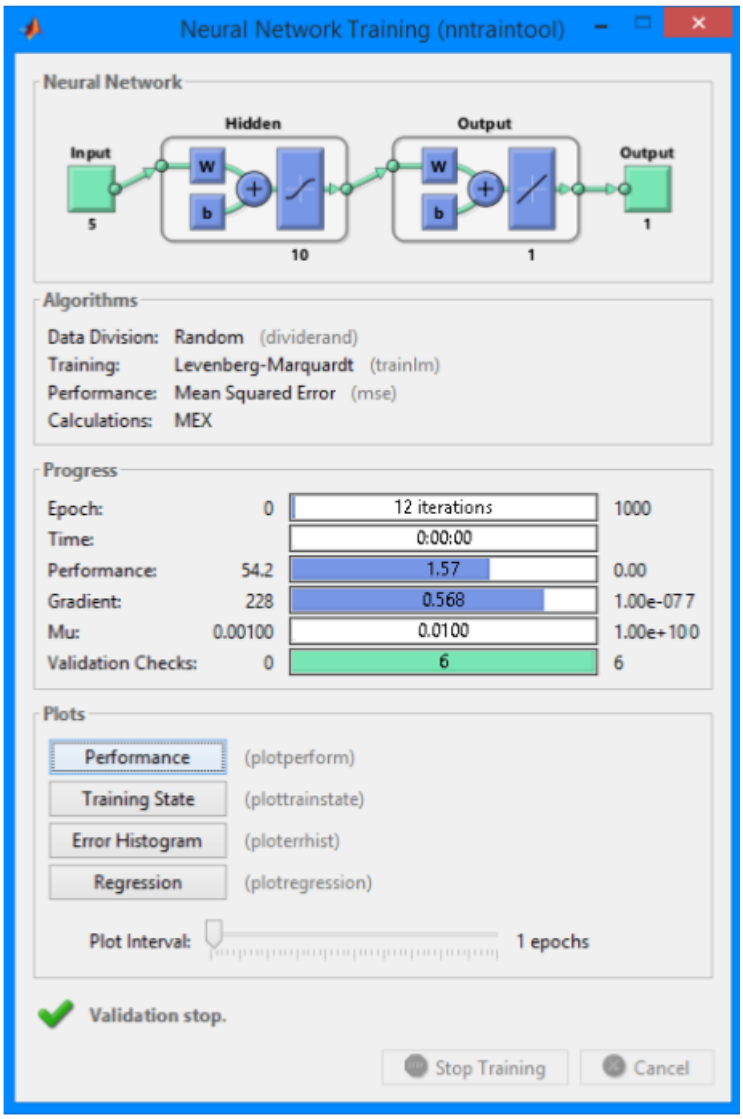

Fig 14: Training the MLP in Matlab

Here we try to reduce the units of hidden layer and to use the number of PCA-components.in this we try all the combinations of values in the hidden layer and PCA-components. Doing this one we can see the lowest validation error is not in the same time. Here we run the code many times that gives the choice of hidden layer units.

Other algorithms described in this report we look at the difference between the true and the predicted values of cnTrainY. As shown in below Fig:15

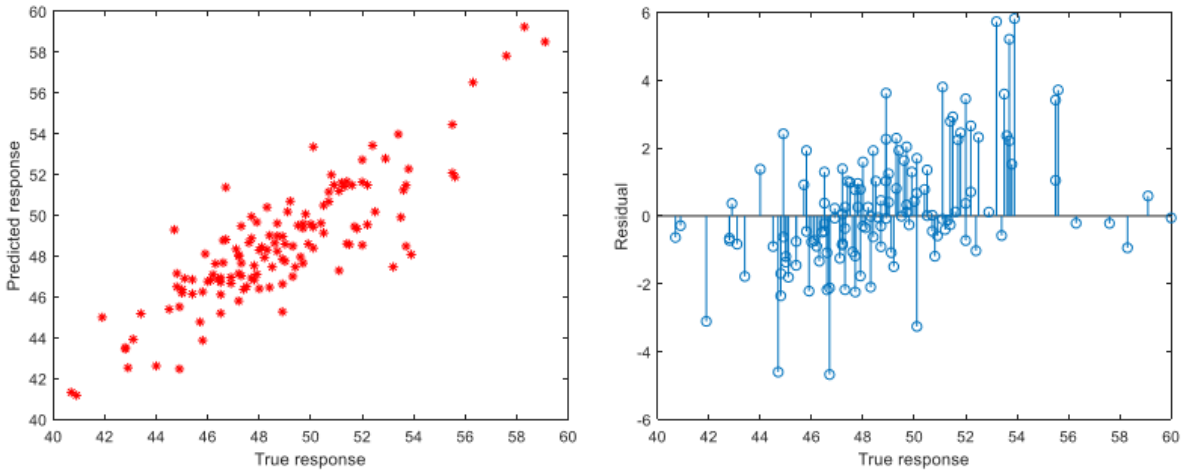

Fig 15: True and predicted cetane numbers of cnTrainY using MLP

The brief the predictions made by all the models we included as shown in below fig 16.

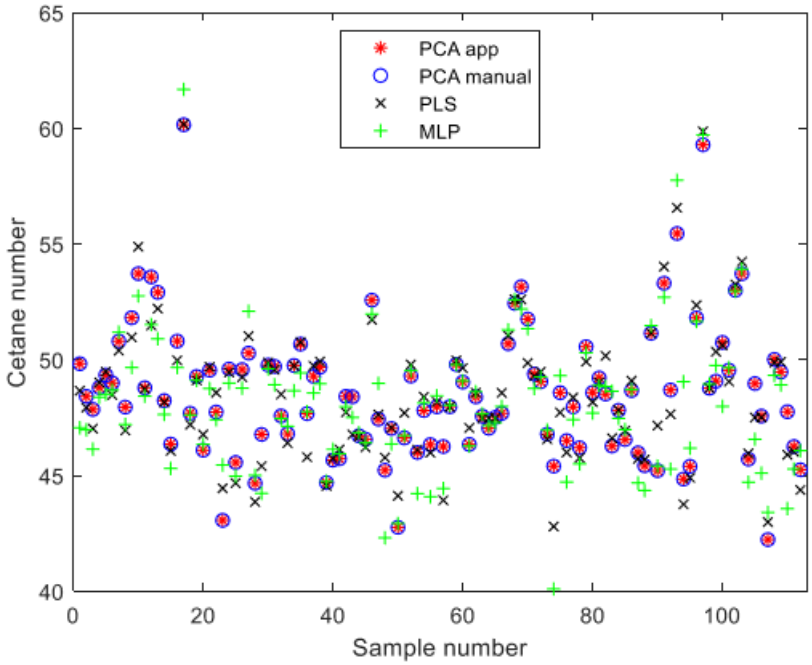

Fig 16: Predictions of the cetane numbers of cnTestX made by all the implemented algorithms.

Discussion:

By solving this task we observed that different algorithms are used to do predictions, but the results are very similar. It is not only the generalizations errors but also the actual predictions of cetane numbers of the dataset cnTestX ass show in the above fig.16.

And another one is, the difference is only behaviour of validation error curves of PCA and PLS-implementations. If you compare fig.7 and fig.11 we can clearly show that difference.

A traditionally script-based software such as Matlab nowadays also have quite a lot of easy-to-use apps making machine learning more available. It can even easily be used to compare different models just by simply choosing to implement all available models at the same time. This is shown in Fig. 17 below.

Fig 17: Comparing difference regression models using Matlab.