Understanding the work of machine learning algorithms:

Regression models:

Regression is a process of investigating the relation between independent variables/ features and dependant variable i.e. the outcome. “Regression shows a line which passes through the data points on a graph such that the distance between the data points and line is minimum.”

Types of Regression models:

1. **Simple linear regression model:** Linear regression is a simple statistical regression which shows the relation between independent variable and dependent variable.
   A regression line can be positive linear relationship or negative linear relationship.
   The main aim of linear regression is to get the best fit line. The best fit line have the least error i.e., the error between predicted values and actual values is minimized.

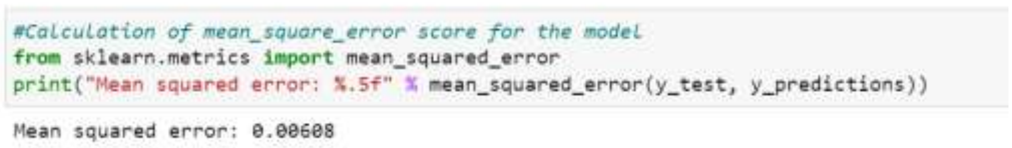
2. **Polynomial regression model:** Polynomial regression is a form of multiple linear regression which estimates the non-linear relationship between dependant and independent variable with the help of an nth degree polynomial. One of the main reasons why polynomial regression is used is when linear regression fails to represent the points in data adequately.

Neural Networks:
Combination of logistic regression is called as multi-layer perceptron i.e. neural network. Neural network has a structure similar to human brain that learn from input data and undergoes regression in each layer/node. The first layer starts the learning and is called input layer and continues its learning in next hidden layers. The last layer is output layer. Neural networks are majorly used when we cannot be able to classify data points through a straight line.

Analysing the results:

Regression Analysis:

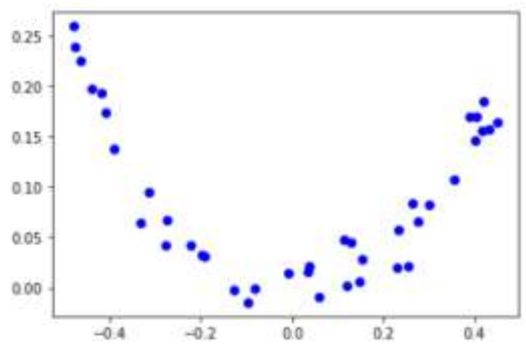
Initially, we have modelled linear regression model after data pre-processing which involves data splitting and separating the data into train and test data. Considering, **x_test** data as the dependent variable and **y_prediction** as independent variable. The analysis of the model resulted in a mean square error of **0.00608.**

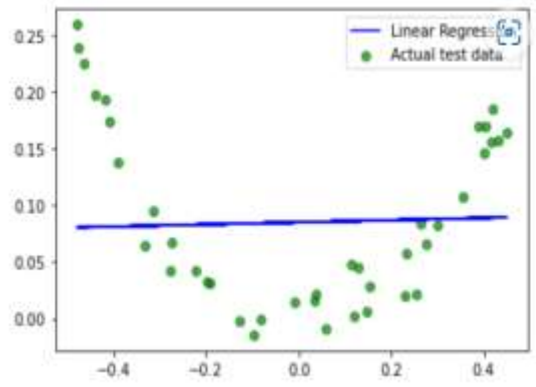


```
#Calculation of mean_square_error score for the model
from sklearn.metrics import mean_squared_error
print("Mean squared error: %.5f" % mean_squared_error(y_test, y_predictions))
```

```
Mean squared error: 0.00608
```

Plot of the data points and linear regression model:



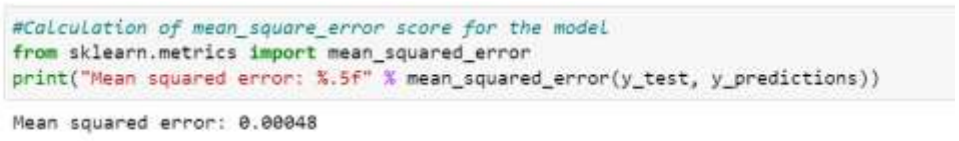


:

The plot shows the line is not a best fit for data points. The linear relationship in the graph is non consistency of predicted data with the dependant variable.

Polynomial Regression Analysis:

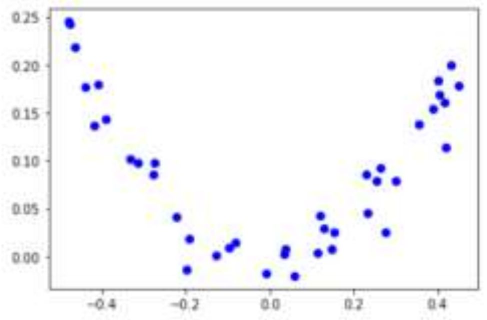
After modelling linear regression, we work with polynomial regression which involves using a polynomial equation to denote the nonlinear nature of the data points. After splitting the data into training and testing data, we predict polynomial data i.e., data for x^2 using x_train data. Then we predict y data using the polynomial function. The analysis resulted in mean square error of **0.00048.**

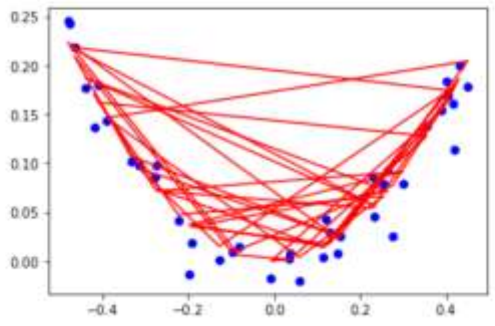


```
#Calculation of mean_square_error score for the model
from sklearn.metrics import mean_squared_error
print("Mean squared error: %.5f" % mean_squared_error(y_test, y_predictions))
```

```
Mean squared error: 0.00048
```

The plot of data points and model is as follows:

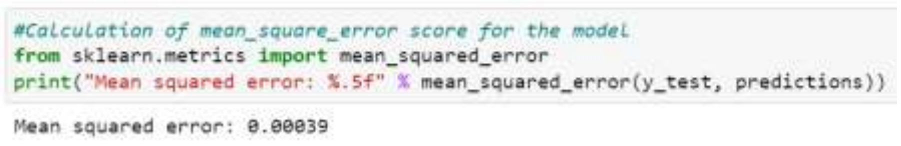




The above plot shows the non-linear relation of data points. We can say the line is best fit through the data points and denotes greater non-linear relation between dependant and independent data.

Neural network analysis:

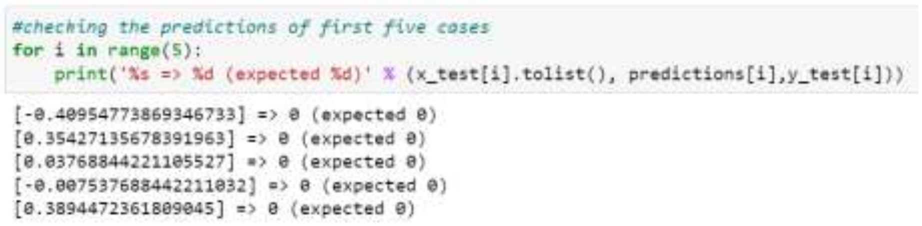
A three-layer network is considered in this analysis. The input layer contained 6 neurons, the hidden layer contained 6 nodes as mentioned in the problem and the output layer contained one neuron. The “relu” activat ion function is used at the hidden layer and “sigmoid” activation function is used at the output layer. Epochs, w hich denotes the number of times to train the model is given an input value of 500. 80% of training data and 2 0% of splitting data is used. The training data is reshaped again for better consistency. This analysis is resulted i n a root mean square error of **0.00039.**



```
#Calculation of mean_square_error score for the model
from sklearn.metrics import mean_squared_error
print("Mean squared error: %.5f" % mean_squared_error(y_test, predictions))

Mean squared error: 0.00039
```

To check the performance of model, I have considered to compared the rprediction of first five values of test data which resulted as follows:



```
#checking the predictions of first five cases
for i in range(5):
    print('%s => %d (expected %d)' % (x_test[i].tolist(), predictions[i],y_test[i]))

[-0.40954773869346733] => 0 (expected 0)
[0.35427135678391963] => 0 (expected 0)
[0.03768844221105527] => 0 (expected 0)
[-0.007537688442211032] => 0 (expected 0)
[0.3894472361809045] => 0 (expected 0)
```

This shows, almost all values are accurately predicted.

e) Model Comparison:

Linear regression	Polynomial regression	Neural networks
0.00608	0.00048	0.00039

Though various measures can be used for comparison purposes, we used mean squared error here. Since, less the mean square error, greater is the performance. As observed from the table, we can predict that the performance of neural networks is better compared to regression models.

If there is a linear relationship that exists between the dependant and independent variable, both the methods are argumentatively comparable and can be used. But, for predictive analysis which is most used in artificial intelligence, either if the data is image or binary, I feel neural networks can perform well with improved knowledge and methods. Neural networks can be used for non linear data for better prediction as shown above. Since, neural network can work with a non-linear methodology without understanding the relation between data points. Neural network can perform better than regression.