ASSIGNMENT NO. 5

TITLE: LINEAR REGRESSION BY USING DEEP NEURAL NETWORK.

Problem Statement:

Linear regression by using Deep Neural network: Implement Boston housing price prediction problem by Linear regression using Deep Neural network. Use Boston House price prediction dataset.

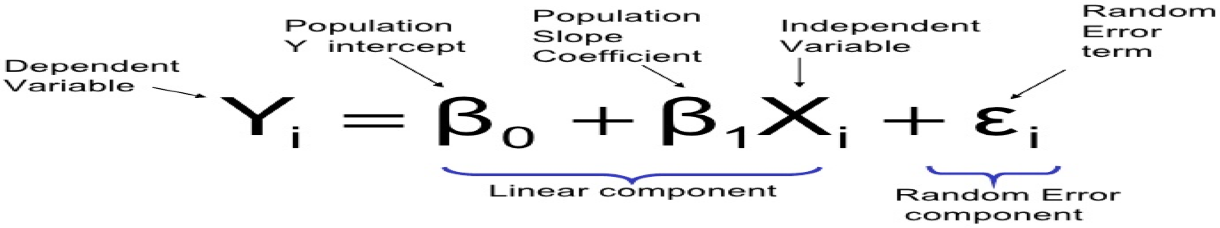
Objective:

1. Pre-process the dataset
2. Identify outliers
3. Check the correlation
4. Implement linear regression using Deep Neural network
5. Predict the price of the house given the other features.
6. Evaluate the models and compare their respective scores like R2, RMSE

Theory:

Deep Learning for Linear regression:

Linear Regression
Linear Regression is a supervised learning technique that involves learning the relationship between the features and the target. The target values are continuous, which means that the values can take any values between an interval. For example, 1.2, 2.4, and 5.6 are continuous values. Use-cases of regression include stock market price prediction, house price prediction, sales prediction, etc.



The y hat is called the hypothesis function. The objective of linear regression is to learn the parameters in the hypothesis function. The model parameters are intercept (beta 0) and the slope (beta 1). The above equation is valid for univariate data, which means there is only one column in the data as a feature.

How does linear regression learn the parameters?

$$\beta_1 = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sum (X_i - \bar{X})^2}$$

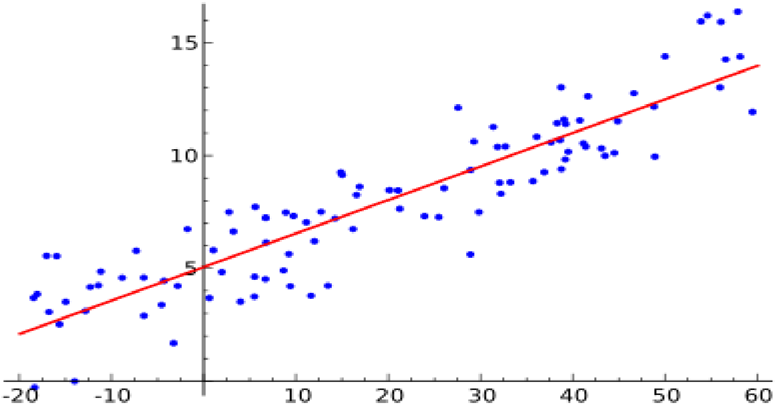
X_i = The feature X

$\bar{X}$ = Mean of X

Y_i = The target Y

$\bar{Y}$ = Mean of Y

The numerator denotes the covariance of the data and the denominator denotes the variance of the feature X. The result will be the value of beta 1 which is also called the slope. The beta 1 parameter determines the slope of the linear regression line. The intercept decides where the line should pass through in the y-axis.



In the image above the intercept-value would be 5, because it is the point where the linear regression line passes through the y-axis. In this way, the linear regression learns the relationship between the features and target.

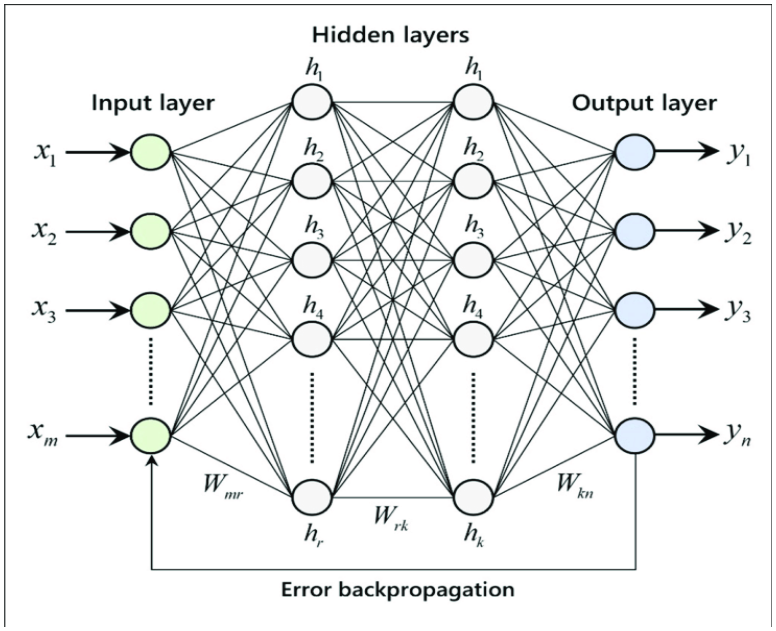
Regression using Artificial Neural Networks

Why do we need to use Artificial Neural Networks for Regression instead of simply using Linear Regression?

The purpose of using Artificial Neural Networks for Regression over Linear Regression is that the linear regression can only learn the linear relationship between the features and target and therefore cannot learn the complex non-linear relationship. To learn the complex non-linear relationship between the features and target, we need other techniques. One of those techniques is to use Artificial Neural Networks. Artificial Neural Networks can learn the complex relationship between the features and target due to the presence of activation function in each layer. Let's look at what are Artificial Neural Networks and how do they work.

Artificial Neural Networks

Artificial Neural Networks are one of the deep learning algorithms that simulate the workings of neurons in the human brain. There are many types of Artificial Neural Networks, Vanilla Neural Networks, Recurrent Neural Networks, and Convolutional Neural Networks. The Vanilla Neural Networks can handle structured data only, whereas the Recurrent Neural Networks and Convolutional Neural Networks can handle unstructured data very well. Vanilla Neural Networks to perform the Regression Analysis.



The Artificial Neural Networks consists of the Input layer, Hidden layers, Output layer. The hidden layer can be more than one in number. Each layer consists of n number of neurons. Each layer will be having an Activation Function associated with each of the neurons. The activation function is the function that is responsible for introducing non-linearity in the relationship. In our case, the output layer must contain a linear activation function. Each layer can also have regularizers associated with it. Regularizers are responsible for preventing overfitting.

Artificial Neural Networks consists of two phases,

- Forward Propagation
- Backward Propagation

Forward propagation is the process of multiplying weights with each feature and adding them. The bias is also added to the result. Backward propagation is the process of updating the weights in the model. Backward propagation requires an optimization function and a loss function.

Steps to create an ANN model

Step 1 - Import the library

```
import pandas as pd
 import numpy as np
from keras.datasets import mnist
from sklearn.model_selection import train_test_split
from keras.models import Sequential
 from keras. layers import Dense
 from keras. layers import Dropout
```

Step 2 - Loading the Dataset

Here we have used the inbuilt mnist dataset and stored the train data in X_train and y_train. We have used X_test and y_test to store the test data.

```
(X_train, y_train), (X_test, y_test) = mnist. load_data ()
```

Step 3 - Creating model and adding layers

We have created an object model for sequential model. We can use two args i.e layers and name.

```
model = Sequential ()
```

Now, We are adding the layers by using 'add'. We can specify the type of layer, activation function to be used and many other things while adding the layer.

Here we have added four layers which will be connected one after other.

```
model.add (Dense(512))
model.add (Dropout (0.3))
  model.add (Dense (256, activation='relu'))
model.add (Dropout (0.2))
```

Step 4 - Compiling the model

We can compile a model by using compile attribute. Let us first look at its parameters before using it.

- optimizer: In this we can pass the optimizer we want to use. There are various optimizer like SGD, Adam etc.
- loss: In this we can pass a loss function which we want for the model
- metrics : In this we can pass the metric on which we want the model to be scored

```
model. compile (optimizer='Adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Step 5 - Fitting the model

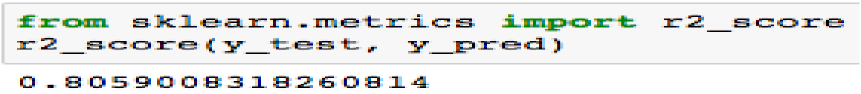
We can fit a model on the data we have and can use the model after that. Here we are using the data which we have split i.e the training data for fitting the model.
While fitting we can pass various parameters like batch_size, epochs, verbose, validation_data and so on.

model.fit (X_train, y_train, batch_size=128, epochs=2, verbose=1, validation_data=(X_test, y_test)

Step 6 - Evaluating the model

After fitting a model, we want to evaluate the model. Here we are using model.evaluate to evaluate the model and it will give us the loss and the accuracy. Here we have also printed the score.

```
score = model. evaluate (X_test, y_test, verbose=0)
print ('Test loss:', score [0])
 print ('Test accuracy:', score [1])
```



```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
0.8059008318260814
```

R² score tells us how well our model is fitted to the data by comparing it to the average line of the dependent variable. If the score is closer to 1, then it indicates that our model performs well versus if the score is farther from 1, then it indicates that our model does not perform so well.

Conclusion: We studied how to use the Artificial Neural Network for implementing linear regression model and Predict the price using Boston housing dataset . Also Evaluated the models and compare their respective scores like R2, MAE

ASSIGNMENT NO.6

TITLE: CONVOLUTIONAL NEURAL NETWORK (CNN)

Problem Statement:

Problem Statement:

Use MNIST Fashion Dataset and create a classifier to classify fashion clothing into categories.

Objective:

To understand and implement:

1. CNN
2. Create a classifier to classify fashion clothing into categories.

Theory:

What is CNN?