NNFS LAB 4: Linear Regression

Linear Regression

Group 1:

Parth Deshpande 191060022

Ishan Deshpande 191060021

Dev Sharma 191060023

Kedar Daithankar 191060019

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables.

Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.

$$y = \theta_1 + \theta_2.x$$

While training the model we are given:
x: input training data (univariate – one input variable(parameter))
y: labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ1 and θ2 values.
θ1: intercept
θ2: coefficient of x

Once we find the best θ1 and θ2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

θ1 and θ2 are also called **a** and **b.**

$$a = \frac{\sum y \sum x^2 - \sum x \sum xy}{n(\sum x^2) - (\sum x)^2}$$

$$b = \frac{n \sum xy - (\sum x)(\sum y)}{n \sum x^2 - (\sum x)^2}$$

Multivariable Linear Regression - Predicting Heart Risk Level Implementation:

https://github.com/heisenberg-88/nnfs_lab4_linear_regression

sex	ageir	tc	hdl	smoke	bpmed	diab	risk
2	48	236	66	0	2	0	1.1
1	48	260	51	0	2	1	7
1	44	187	49	1	2	0	7
2	42	216	57	1	2	0	0.4
2	56	156	42	0	2	0	2.2
1	44	162	57	1	2	0	3
1	50	244	47	0	2	0	4.2
1	48	212	30	1	2	0	17.4
2	66	202	53	0	2	1	13.4
1	63	186	46	1	2	0	17.3
1	42	267	28	1	2	0	19.8
1	58	234	36	1	2	0	13.2
1	72	277	47	0	2	0	36.2
2	45	206	42	1	2	0	2.9
1	69	249	62	0	2	0	11.7
2	63	205	47	0	2	0	4.3
2	41	218	81	0	2	0	0.3
1	55	194	36	0	2	0	9.7
1	72	228	44	1	2	1	38.1
1	55	216	35	0	2	0	9.3
2	65	175	78	1	2	0	6.3
1	57	245	54	1	1	0	14
2	49	247	45	1	2	1	6.3
1	65	281	51	0	2	0	15.1

This is the **cardio_dataset.csv** given for the regression model to predict the heart disease.

We'll be using scikit-learn api for LinearRegression implementation.

In this dataset, first 7 columns are **features** and the last column (**label**) shows the risk percentage.

We will first try directly using the LinearRegression() on this raw dataset.

```
In [187]: import pandas as pd #for loading the dataset
```

```
In [188]: dataset = pd.read_csv('cardio_dataset.csv').values #load the dataset into a numpy array
```

Here, we separate the features (first 7 columns) and labels (last column) into **data** and **target** respectively

```
In [190]: data = dataset[:,0:7] #all rows and columns from 0 to 6
```

```
In [191]: target = dataset[:,7] #all rows and 7th column
```

```
In [192]: print(dataset.shape)
          (6644, 8)
```

Now, we split the data into training and testing dataset in the amount of 80% and 20%.

```
In [195]: from sklearn.model_selection import train_test_split
```

```
In [196]: train_data,test_data,train_target,test_target = train_test_split(data,target,test_size=0.2)
```

Importing LinearRegression from sklearn and fitting on the dataset.

```
In [197]: from sklearn.linear_model import LinearRegression

In [198]: model = LinearRegression() #Loading the algorithm
          model.fit(train_data,train_target)

Out[198]: LinearRegression
          LinearRegression()

In [199]: predicted_target = model.predict(test_data)
```

Now we have predicted_y(targets) values stored in **predicted_target**.

As, for regression problem we don't measure accuracy.

We use R2 Score & Coefficiant of determination.

```
In [200]: from sklearn.metrics import r2_score
          r2 = r2_score(test_target,predicted_target)
          print("r2: ",r2)

          r2:  0.7628192831910273
```

When r2 score is closer to 1: More accurate
closer to 0: Less Accurate

Here we've got accuracy of 76.28%

The reason behind less accuracy is the spreading of the input features.

As the mean, variance and standard deviation of dataset is not normalized to a fixed range, the linear regression model is not able to find the perfect fit line.

For this, we'll use various normalization techniques in pre-processing the dataset.

Let's try sklearn. preprocessing.normalize

Using this, first we'll normalize the dataset and then divide it into the features and labels.

In [206]:
```
from sklearn import preprocessing
import numpy as np
dataset = preprocessing.normalize(dataset)
dataset = pd.DataFrame(d)
dataset.head()
```

The normalized data is as shown below:

Out[222]:

	0	1	2	3	4	5	6	7
0	0.008009	0.192207	0.945020	0.264285	0.000000	0.008009	0.000000	0.004405
1	0.003712	0.178194	0.965216	0.189331	0.000000	0.007425	0.003712	0.025987
2	0.005040	0.221779	0.942559	0.246981	0.005040	0.010081	0.000000	0.035283
3	0.008798	0.184755	0.950168	0.250739	0.004399	0.008798	0.000000	0.001760
4	0.011694	0.327441	0.912158	0.245581	0.000000	0.011694	0.000000	0.012864

```
In [216]: from sklearn.model_selection import train_test_split

In [217]: train_data,test_data,train_target,test_target = train_test_split(data,target,test_size=0.2)

In [218]: from sklearn.linear_model import LinearRegression

In [219]: model = LinearRegression() #Loading the algorithm
          model.fit(train_data,train_target)

Out[219]: ▾ LinearRegression
          LinearRegression()

In [220]: predicted_target = model.predict(test_data)
```

After using linearRegression on the normalized dataset, we get more accuracy.

```
from sklearn.metrics import r2_score
r2 = r2_score(test_target,predicted_target)
print("r2: ",r2)
```

```
r2:  0.8462997919950188
```

Here, we've got accuracy of 84.62 %

Let's try sklearn. preprocessing.StandardScaler

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
data = scaler.fit_transform(dataset)
data
```

After Normalizing using StandardScaler() we get these values in dataset:

```
Out[236]: array([[ 0.95697524, -0.75978338,  0.42399144, ...,  0.51809561,
                  -0.45221235, -0.89026093],
                 [-1.04495911, -0.75978338,  0.97145106, ...,  0.51809561,
                   2.21135047, -0.43629483],
                 [-1.04495911, -1.11130322, -0.69373863, ...,  0.51809561,
                  -0.45221235, -0.43629483],
                 ...,
                 [-1.04495911,  0.73417597, -0.12346819, ..., -1.93014567,
                  -0.45221235,  1.37956958],
                 [-1.04495911,  0.82205593, -0.76217108, ..., -1.93014567,
                  -0.45221235,  0.85635441],
                 [ 0.95697524, -1.02342326, -0.32876554, ...,  0.51809561,
                  -0.45221235, -0.89795527]])
```

Then we apply LinearRegression() on this data.

```
In [230]: from sklearn.model_selection import train_test_split

In [231]: train_data,test_data,train_target,test_target = train_test_split(data,target,test_size=0.2)

In [232]: from sklearn.linear_model import LinearRegression

In [233]: model = LinearRegression() #Loading the algorithm
          model.fit(train_data,train_target)

Out[233]: LinearRegression
          LinearRegression()

In [234]: predicted_target = model.predict(test_data)
```

Here we've got accuracy of 96.36%

```
:  from sklearn.metrics import r2_score
   r2 = r2_score(test_target,predicted_target)
   print("r2: ",r2)

   r2:  0.9636255987059036
```