**NNFS LAB 4: Linear Regression**

**Linear Regression**

Group 1:

Parth Deshpande 191060022

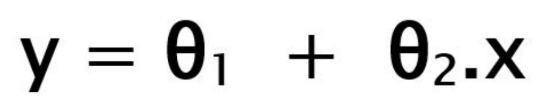
Ishan Deshpande 191060021

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.



While training the model we are given:

x: input training data (univariate – one input variable(parameter))

y: labels to data (supervised learning)

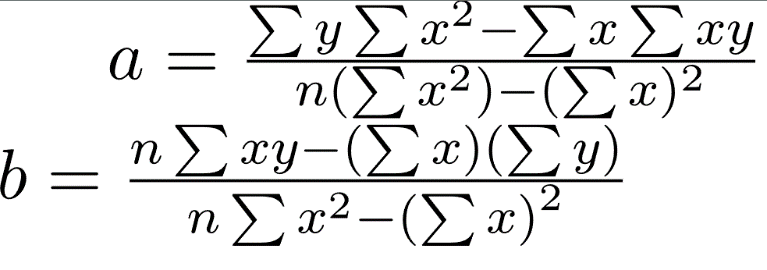
When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ1 and θ2 values.

θ1: intercept

θ2: coefficient of x

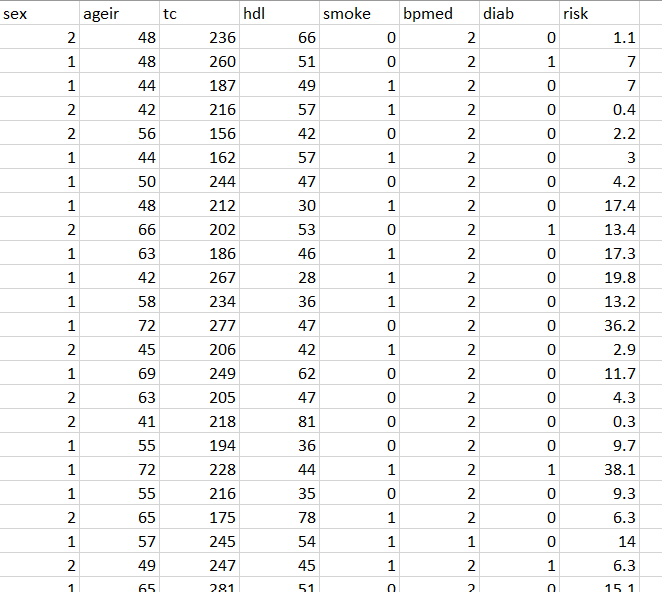
Once we find the best θ1 and θ2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

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



**Multivariable Linear Regression - Predicting Heart Risk Level**

**Implementation:**

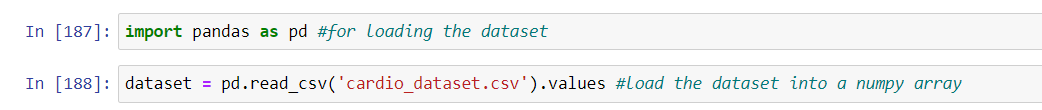
<https://github.com/heisenberg-88/nnfs_lab4_linear_regression>

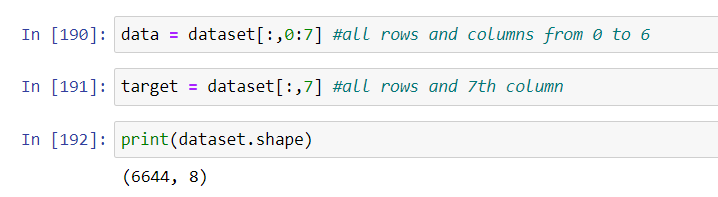
This is the **cardio\_dataset.csv** given for the regression model to predict the heart disease.

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

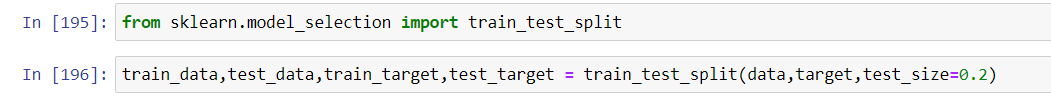
In this dataset, first 7 columns are **features** and the last column (**label**) shows the risk percentage.

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

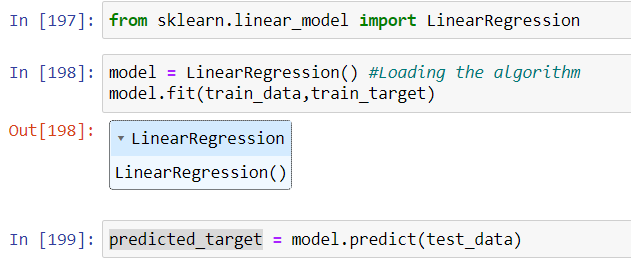
Here, we separate the features (first 7 columns) and labels (last column) into **data** and **target** respectively



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



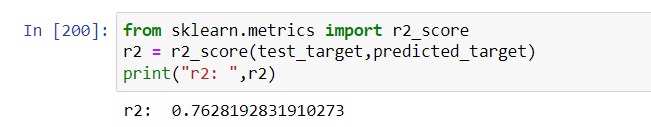
Importing LinearRegression from sklearn and fitting on the dataset.



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.

****

**When r2 score is closer to 1: More accurate**

**closer to 0: Less Accurate**

**Here we’ve got accuracy of 76.28%**

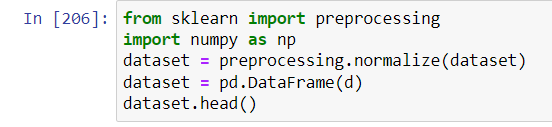
**The reason behind less accuracy is the spreading of the input features**.

**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.**

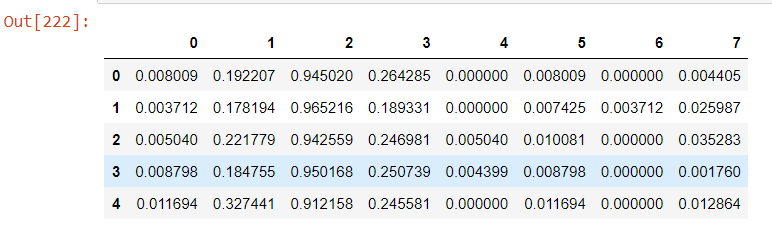
For this, we’ll use various normalization techniques in pre-processing the dataset.

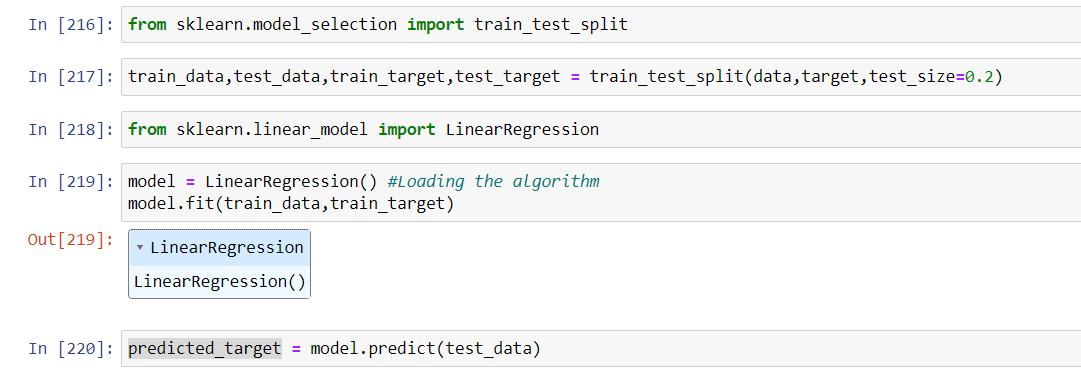
Let’s try sklearn. preprocessing.normalize

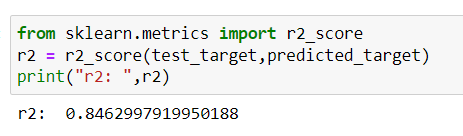
Using this, first we’ll normalize the dataset and then divide it into the features and labels.



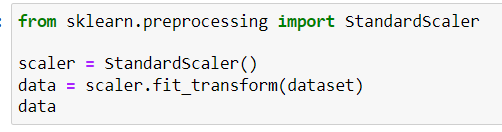
The normalized data is as shown below:



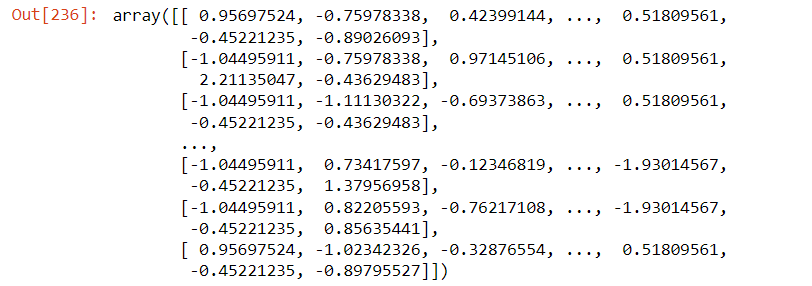
After using linearRegression on the normalized dataset, we get more accuracy.

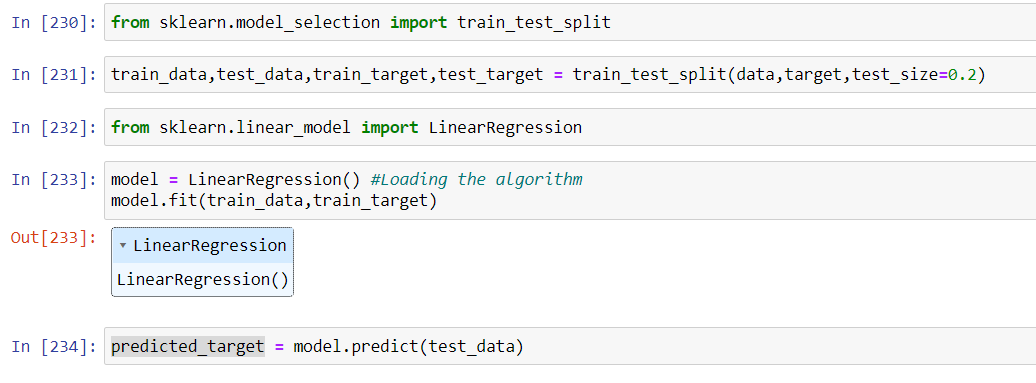
**Here, we’ve got accuracy of 84.62 %**

Let’s try sklearn. preprocessing.StandardScaler

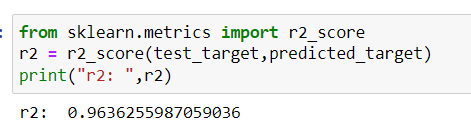
****

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

Then we apply LinearRegression() on this data.

****

**Here we’ve got accuracy of 96.36%**

****