Lecture #3 Content Modelling / Performance analysis

Statistics/Mathematics for Machine learning

1.0 Introduction

2.0 Machine Learning Workflow

3.0 Mathematics / Statistics for Machine Learning (simple version)

3.1 Modeling

3.1.1 Linear regression

3.1.2 Polynomial regression

3.1.3 Multivariable regression

3.2 Model performance analysis

3.2.1 R^2 (Regression coefficient)

3.2.2 MSE (Mean square error)

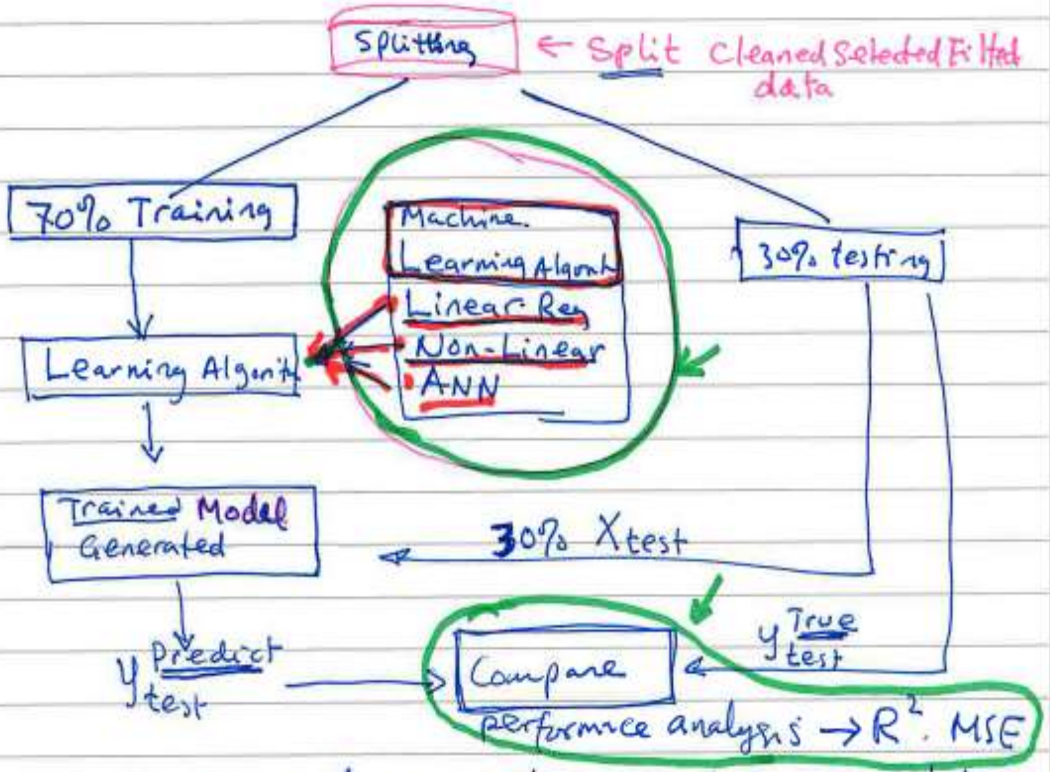
4.0 Summary

Lecture 3 How Machine Learning Modeling and model Performance analysis works??

1.0 Introduction

In the previous lecture, we processed and make our data ready for modeling.

The Second and the third Steps are to Use the processed data for MODELING and Perform MODEL ACCURACY analysis, respectively



→ As Shown above, applying 70% training data On the theree ML algoritms (Show in green circle), we build the ML-model → This is Modeling

→ Using 30% X_{test}, into the Model, we predict y_{pred}

→ Using y_{pred} and y_{test}, → we perform MODEL Accuracy analysis.

Therefor this chapter presents the Mathematics/Statistics of the Modeling (in green) and Performace analysis (in green)

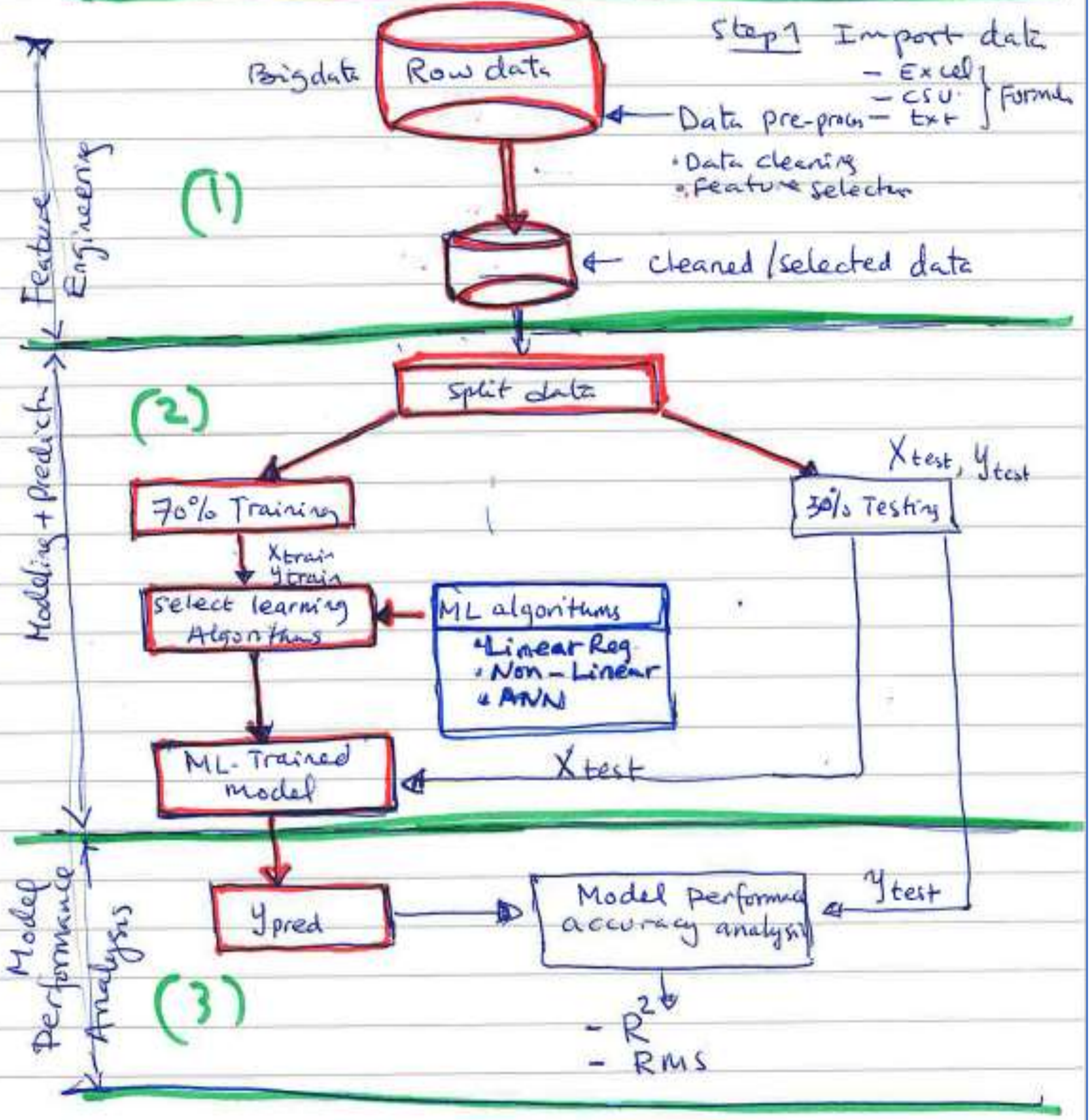
2.0 Machine Learning WORKFLOW

The workflow consists of three main parts

(1) Feature Engineering
- Data Preprocessing
- Feature Selection

(2) Modelling and Prediction

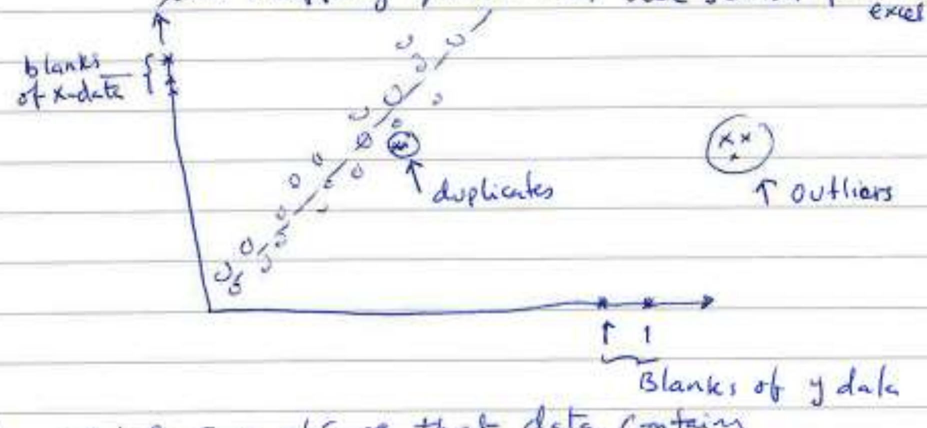
(3) Model Performance Accuracy analysis



To understand the ML- work flow,
let us concider the following dataset

			dup	dup			dup	dup			outlier		zero?		zero		
x	1	2.5	3.4	3.4	4.8	7.2	9.5	9.5	11	12	30	2	22	21	0	15	20
y	4	8	10	10	—	17	20	20	23	24	20	38	0	—	36	30	38
					Blank									Blank			

Step 1 → Let us display the data to acess the quality of the dataset and to estimate the mapping function → use scatter plot in Excel



① → we can obsere that data contains
- → outliers (4)
- → (2) unrecorded blanks, in X, y Sensors
- → duplicates (2)

→ mapping function = Linear function

② — Before we do modelling, we need to remove outliers, blanks, duplicates and apply if necessary Smoothing filter (Moving averge / exponential smoothing)

③ Modeling → add and select trend line > linear

④ Add Equation and R^2

— Result shows $y = 1.7482x + 3.4828$
$R^2 = 0.9965$

$R^2 =$ __ means that the model predicts/describe 99% of the dataset

3.0 Mathematics / Statistics For ML- (Simple Version)

In the previous example, we have seen/used Excel built in library to perform modeling (math) and performance analysis (statistics).

In this section, we will learn the concept/maths

Q1 (a) How Linear/poly/multivariate reg work?

Q2 (b) How performance analysis performed?

The first and the second questions are based on

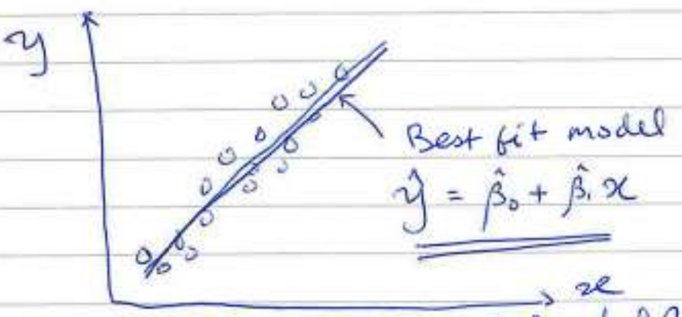
(1) Partial diff eq.
(2) Matrix Algebra
(3) Statistics

Let's see the Maths/statistics of ML implimend in the Excel/Matlab/Python Libraries

3.1 Modeling

3.1.1 Linear Regression Modeling

Figure below shows the dataset and the 'best-fit' model



$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

Q How to determine 'best-fit' model?

Since the data trend in the above figure is linear, we choose a linear mapping function

$$\boxed{y = \beta_0 + \beta x}$$

The machine learning algorithm use the input/output data and the model to find an optimized $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize the Sum Square residual Error b/w the model and the dataset

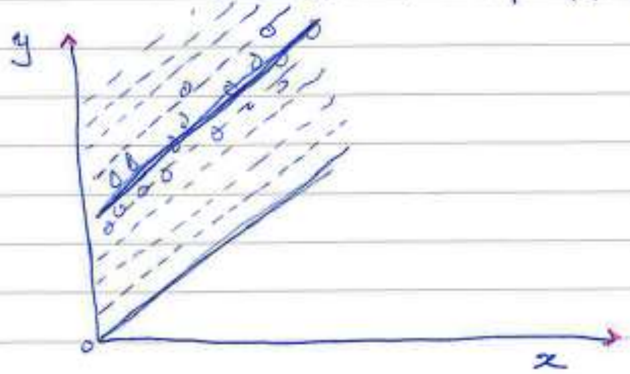
— The model is therfore called "best-fit", which is an optimized curve fit that provides a minimum Error.

→ The method of finding the optimized Parameters is called Least square Error method

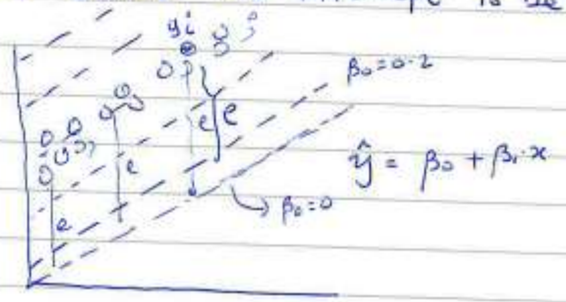
To understand how Least square Error Computation method works, Let's consider the following

→ Assume that the Slope is known (β_1)

— Task! The Task is to find the optimized intercept (β_0)



Let us start the intercept to be Zero



The Sum of error^2 is given as

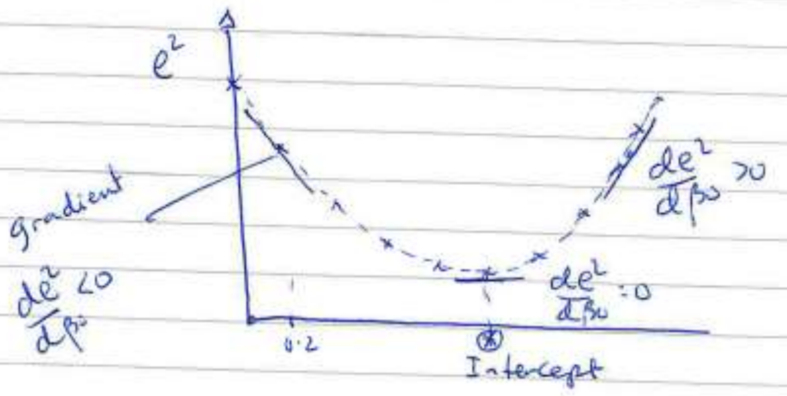
$$e^2 = \sum (y_i - \beta_0 - \beta_1 x_i)^2$$

Assume $\beta_1 = 0.5$

→ Let us increase the intercept to be 0.2
Then, compute e^2.. for $\beta_0 = 0.2$

Keep on increasing the value of the intercept and compute e^2.

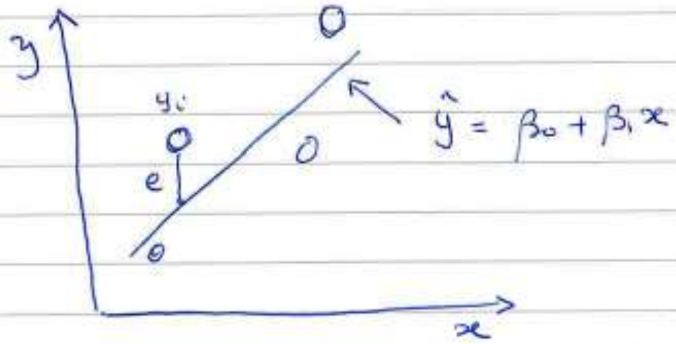
→ plotting e^2 vs intercept we get a parabola since e^2.



As we can see, gradient at ⊛ = 0 which is the point of minimum.

→ It means that, the Least Square Error is obtained when $\frac{\partial e^2}{\partial \beta_0} = 0$, Similarly $\partial e^2/\partial \beta_1 = 0$ – slope

Therefore the optimized β_0 and β_1 are estimated by the Least Square Error Sum method



Mathematically: Sum of residual square Error

$$L = e^2 = \sum (y_i - \hat{y})^2$$

$$= \sum (y_i - \beta_0 - \beta_1 x_i)^2$$

→ Minimizing the function w.r.t β_0, β_1

$$\frac{\partial e^2}{\partial \beta_0} = 0 \Rightarrow -2 \sum (y_i - \beta_0 - \beta_1 x_i) = 0 \quad \text{-- (1)}$$

$$\frac{\partial e^2}{\partial \beta_1} = 0 \Rightarrow -2 \sum x_i (y_i - \beta_0 - \beta_1 x_i) = 0 \quad \text{-- (2)}$$

From (1) and (2) we get

$$\sum \beta_0 + \beta_1 \sum x_i = \sum y_i \quad \text{-- 3}$$

$$\beta_0 \sum x_i + \beta_1 \sum x_i^2 = \sum x_i y_i \quad \text{-- 4}$$

Since $\sum \beta_0 = n\beta_0$,

$$\underbrace{\begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix}}_{X} \cdot \underbrace{\begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}}_{B} = \underbrace{\begin{bmatrix} \sum y_i \\ \sum x_i y_i \end{bmatrix}}_{Y} \quad \text{-- 5}$$

Two Methods to compute β_0, β_1

From eq 5; the Coefficient, B, Can be Calculated by Matrix inversion

Method 1

$$\boxed{B = X^{-1} \cdot Y}$$

$B = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$, the coefficients, β_0 & β_1 are the optimized parameters that provides the best-fit model, which is Least Square Error.

Method 2

For implimenting the Computation in Python, from eq 3, we can write

$$\sum_{i=1}^{n} \beta_0 + \beta_1 \sum x_i = \sum y_i$$

Replace

Since $\left\{ \begin{array}{l} \sum_{i=1}^{n} \beta_0 = n\beta_0 \\ \sum y_i = n\bar{Y} \\ \sum x_i = n\bar{x}, \end{array} \right\}$ $\bar{Y}, \bar{x}$ are mean of x_i, y_i's

Replacing above,

$$n\beta_0 + \beta_1 n\bar{x} = n\bar{Y}$$

The intercept $\boxed{\beta_0 = \bar{Y} - \beta_1 \bar{x}}$ -- (7)

To find the slope, β_1 we use equations Eq 4 and Eq(7)

$$\beta_0 \sum x_i + \beta_1 \sum x_i^2 = \sum x_i y_i$$

$$n\beta_0 \bar{x} + \beta_1 \sum x_i^2 = \sum x_i y_i$$

$$n(\bar{y} - \beta_1 \bar{x}) \cdot \bar{x} + \beta_1 \sum x_i^2 = \sum x_i y_i$$

$$n\bar{x}\bar{y} + \beta_1\left(\sum x_i^2 - n\bar{x}^2\right) = \sum x_i y_i$$

$$\beta_1\left(\sum x_i^2 - n\bar{x}^2\right) = \sum x_i y_i - n\bar{x}\bar{y}$$

slope $$\boxed{\beta_1 = \frac{\sum x_i y_i - n\bar{x}\cdot\bar{y}}{\sum x_i^2 - n\bar{x}^2}} \quad \text{--- } 8$$

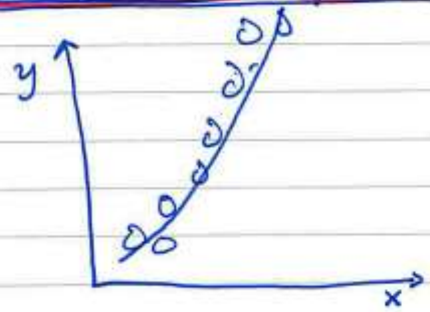
→ We will impliment eq (7) and (8) Later.

→ This is implimented in built in libraries (Eg. Excel, Matlab, Python)

Example

when we feed (x_i, y_i) data set to ML and select the learning algoritm (Linear function), ML compte slope/intercept to find the "best-fit" model

3.1.2 Polynomial Regression



Since the trend of the dataset is curved, we estimate the mapping function to be a polynomial.

$$\boxed{\hat{y}_i = \beta_0 + \beta_1 x_i + \beta_2 X^2}$$

Your Task #1

Using Least square sum Error method,

a) Show the following matrix

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum y_i x_i \\ \sum y_i x_i^2 \end{bmatrix}$$

(b) Show how to find the Coeff matrix.

$$\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \_\_\_\_\_\_\_\_$$

3.1.3 Multivariable Regression

For the Simple linear and Polynomial function the input is only one variable x, which is related to the output, y

- But for the multivariable regression, the number of inputs are more than or equal to two ($x_1, x_2 \cdots x_n$), which are related to the output, y. This regression is called Multvariable regression

The model can be written as

$$\boxed{\hat{y}_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n}$$

Your task #2

Assume that the input ~~pair~~ features are x_1, x_2 and will be related to y

The multivariable function is

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

Using the Least square error method.

(a) Show the following matrix

$$\begin{bmatrix} n & \sum x_{i1} & \sum x_{i2} \\ \sum x_{i1} & \sum x_{i1}^2 & \sum x_{i1}x_{i2} \\ \sum x_{i2} & \sum x_{i1}x_{i2} & \sum x_{i2}^2 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum y_i x_{i1} \\ \sum y_i x_{i2} \end{bmatrix}$$

(b) Show how to find the Coeff. matrix

$$\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} \ ?$$

Summary

Task #1 and Task #2 along with the example shown for the linear regression, the Python / Excel built in libraries use to find the optimized coefficients that gives the "best-fit" model.
During Lab exercise, we use the libraries, we don't need to impliment these from the scratch!

4.2 Model Performance accuracy analysis.

In Sections 4.1.1, 4.1.2 and 4.1.3, we have seen how ML algorithms works. Here, we have seen the application of Least Square Error to compute the Coefficients. The three Sections ~~are~~ belongs to the Category of Linear Regression.

For non-linear regression, ~~there are~~ we follow the Same procedure to determine Coefficients.

However, for practical application, you don't need to impliment the functions to compute the coefficients. ML built in Libraries do the job for us as we have seen examples in Excel.

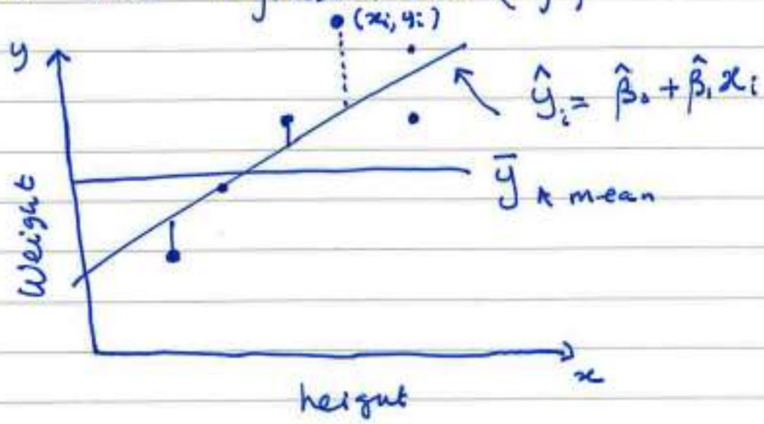
→ Once we generate the ML traind model, the third Step is to acess the model accuracy. For this, we use:

(a) Regression Coefficient, R^2

(b) MSE (mean Square Error)

In this section, we will see the concepts behind these.

Let's measure how data spread around the mean ($\bar{y}$) value and around the regression line ($\hat{y}$)



y_i = measured data

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n} \quad - \text{mean}$$

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_i \rightarrow \text{Best fitted line}$$

Both $\bar{y}$ and $\hat{y}$ are superimposed

Now imagin we must predict the weight of one person.

- If we have no knowledge of the height, we must predict his/her weight to be the average, $\bar{y}$

$\Rightarrow$ The prediction Error is $y_i - \bar{y}$

- If we predict each weights in this way, the total sum of square prediction will be $\sum_{i=1}^{n} (y_i - \bar{y})^2$

* If on the otherhand, we know the height, we can predict the weight from the fitted line ($\hat{y}$)

$$\text{Prediction Error} = y_i - \hat{y}$$

$$\text{Sum Square Error} = \sum_{i=1}^{n} (y_i - \hat{y})$$

> The strength of the Linear relationship can be measured by computing the Reduction in the Sum of square obtained by using $\hat{y}$, not $\bar{y}$

> The difference is

$$\sum (y_i - \bar{y})^2 - \sum (y_i - \hat{y})^2$$

The bigger the difference is the more tightly clustered the points around the least square line (fitted line, $\hat{y}$)

$\Rightarrow$ The Stronger relationship between x_i, y

Thus, $\sum (y_i - \bar{y})^2 - \sum (y_i - \hat{y})^2$ is the goodness of fit statistics.

Since the difference has unit, to use the goodness-fit in an absolute scale ...

we use, the correlation coeff (r^2) as ...

3.2.1 (Coefficient of determination) - Correlation Coefficient, R^2 →

$$R^2 = \frac{\sum (y_i - \bar{y})^2 - \sum (y_i - \hat{y})^2}{\sum (y_i - \bar{y})^2}$$

$$R^2 = 1 - \frac{\sum (y_i - \hat{y})^2}{\sum (y_i - \bar{y})^2}$$

where $R^2 = 1 - \frac{RSS}{TSS}$

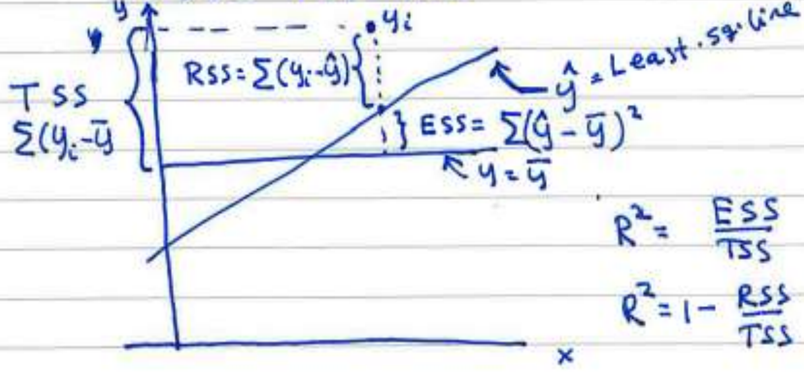
(1) $\sum (y_i - \bar{y})^2 =$ Total sum square (TSS)
- TSS measures the overall spread of points around $y = \bar{y}$

(2) $\sum (\hat{y}_i - \bar{y}) =$ Regression sum square (RSS)
⇒ RSS measures the overall spread of points around the least square line ($\hat{y}$)

(3) $\sum (y_i - \hat{y})^2 =$ Residual sum square (RSS)

Explained sum square (Regression sum sq)

ESS = TSS − RSS



$$R^2 = \frac{ESS}{TSS}$$

$$R^2 = 1 - \frac{RSS}{TSS}$$

ESS = measures the reduction of the spread of the spread of the points obtained by the least square line, $\hat{y}$, rather than the mean, $y = \bar{y}$

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

$$\bar{y} = \frac{\sum y_i}{n}$$

How to Calculate these in Python?

(1) Residual Sum Square (RSS)

$$RSS = ((y_i - \beta_0 - \beta_1 x_i)^{**2}).sum()$$

or

$$RSS = np.sum((y_i - \beta_0 - \beta_1 x_i)^{**2})$$

(2) Explained Sum Square (ESS)

$$ESS = ((\beta_0 + \beta_1 x_i - y_i.mean())^{**2}).sum()$$

or

$$np.sum((\beta_0 + \beta_1 x_i - np.mean(y_i))^{**2})$$

(3) Total Sum Square (TSS)

$$TSS = ((y_i - y_i.mean())^{**2}).sum()$$

or

$$= np.sum((y_i - np.mean(y_i))^{**2})$$

$$R^2 = 1 - \frac{RSS}{TSS}$$

$$= \frac{ESS}{TSS}$$

3.2.2 Mean sum Square Error

How to calculate mean square error (MSE)?

$$MSE: \frac{1}{N} \sum (y_i - \hat{y})^2$$

$$= ((y_i - \beta_0 - \beta_1 x_i)^{**2}).sum() / len(y_i)$$

or

$$= ((y_i - y_{pred})^{**2}).sum() / len(y)$$

or

$$(((y_i - \beta_0 - \beta_1 x_i)^{**2}).sum).mean($$

$$np.mean((y_i - \beta_0 - \beta_1 x_i)^{**2}.sum())$$

$$np.mean(np.sum((y_i - \beta_0 - \beta_1 x_1)^{**2}$$

How to use Sk-learn built in library?

Step 1. Import the function from sklearn.

Step 2. Compute

Example

```
from sklearn import r2_score, mse
import numpy as np.

print('MSE:' %.2f' % np.mean((ytest, ypred)**2))
print('R²:' %.2f % r2_score(ytest, ypred))
```

4.0 Summary

In lecture 3, we have seen the Machine learning workflow, the mathematics how linear regression finds the optimized slope and intercept.

Moreover, we have seen how model performance accuracy analysis performed with R^2 and MSE

It is important to follow the three main Machine learning workflows when we do machine learning modeling. These are

(1) Data preprocessing
- → Cleaning
- – Feature selection
- – Data filtration

(2) Machine learning modeling

(3) Model performance analysis