An Introduction to Machine Learning with Python Programming
12 Sep 2023 - 20 Oct 2023

iHUB, DivyaSampark, IIT Roorkee

Linear Regression

and
Ritvij Bharat Private Limited (RBPL)

Presented By:
Shreyas Shukla

Fig Ref: Pierian Data

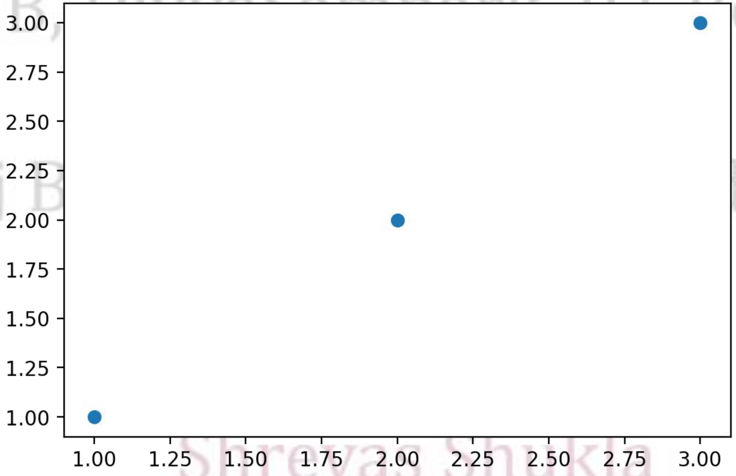
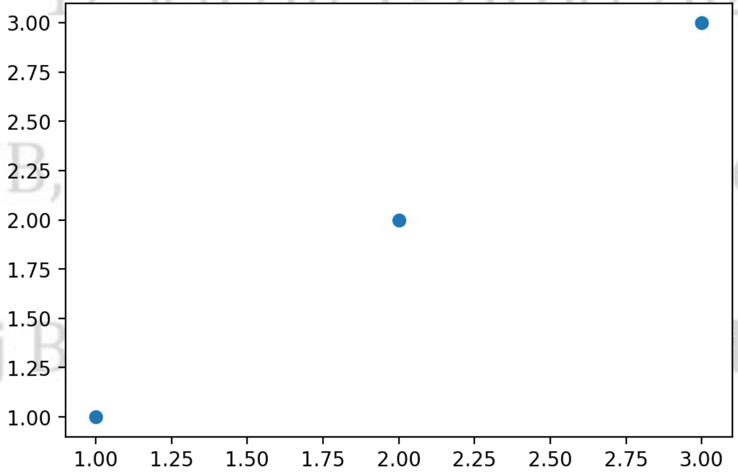


Fig Ref: Pierian Data

An Introduction to Machine Learning with Python Programming

12 Sep 2023 - 20 Oct 2023



iHUB, ... oorkee

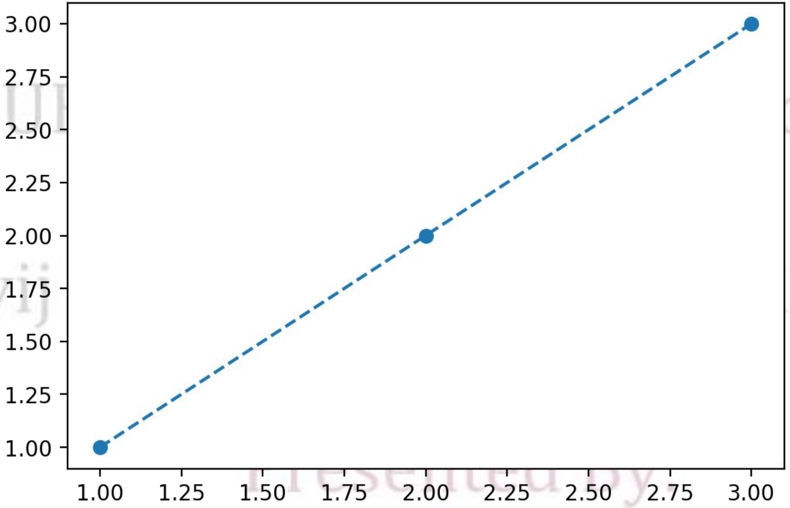
Ritvij B... (RBPL)

Presented By:

Shreyas Shukla

Fig Ref: Pierian Data

An Introduction to Machine Learning with Python Programming
12 Sep 2023 - 20 Oct 2023

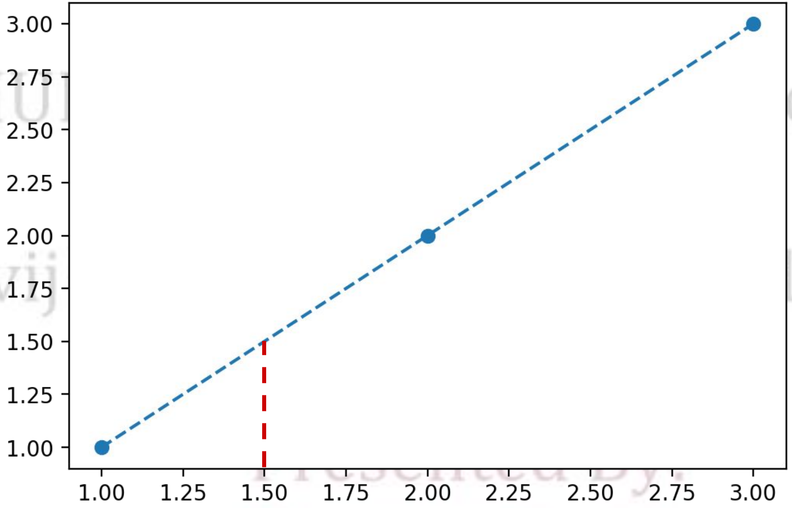


Shreyas Shukla

Fig Ref: Pierian Data

An Introduction to Machine Learning with Python Programming

12 Sep 2023 - 20 Oct 2023



Shreyas Shukla

Fig Ref: Pierian Data

An Introduction to Machine Learning with Python Programming
12 Sep 2023 - 20 Oct 2023

iHUB, DivyaSampark, IIT Roorkee
and
Ritvij (RBPL)

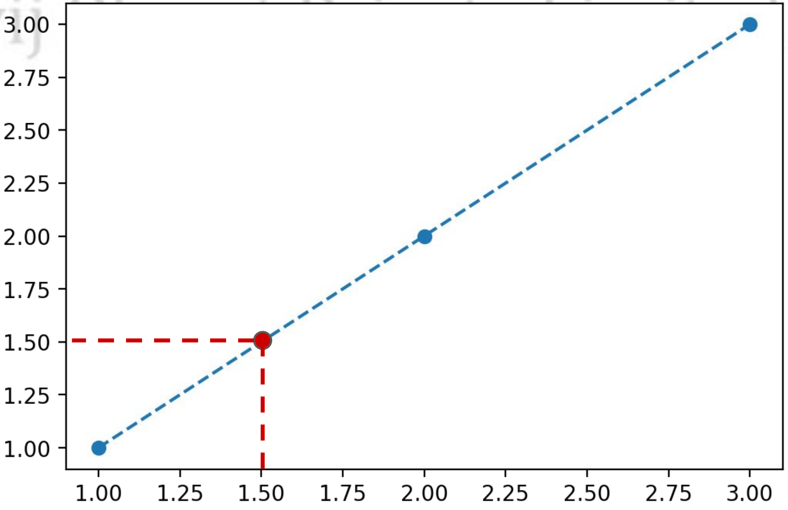
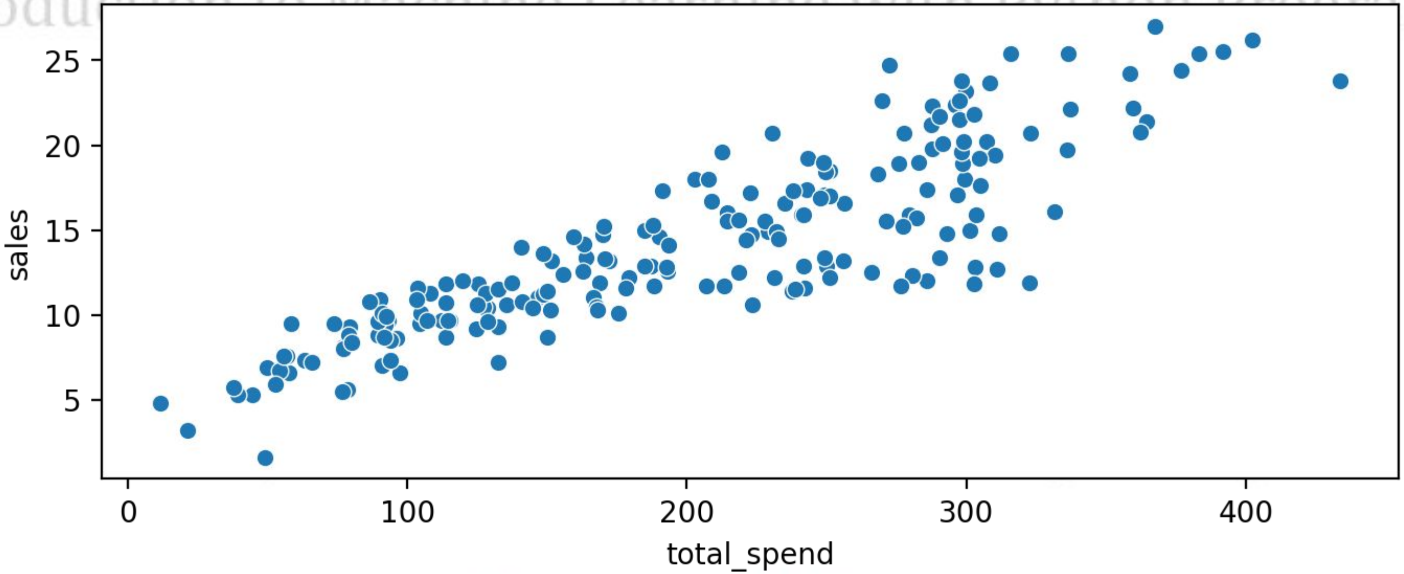
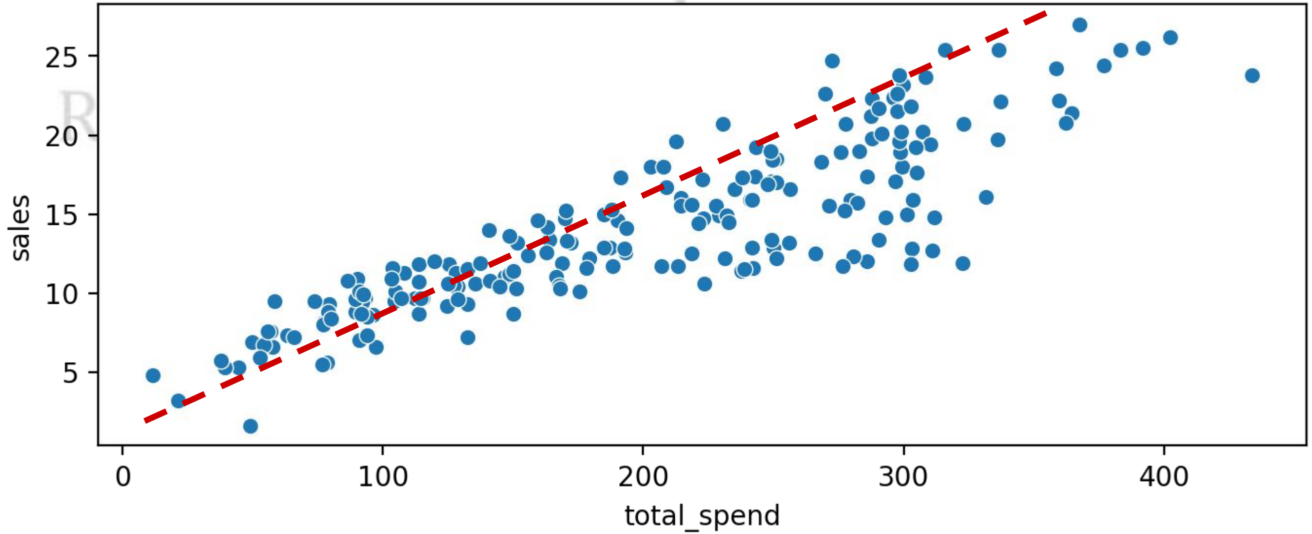


Fig Ref: Pierian Data



sales
25
20
15
10
5
0
100
200
300
400
total_spend


sales
25
20
15
10
5
0
100
200
300
400
total_spend

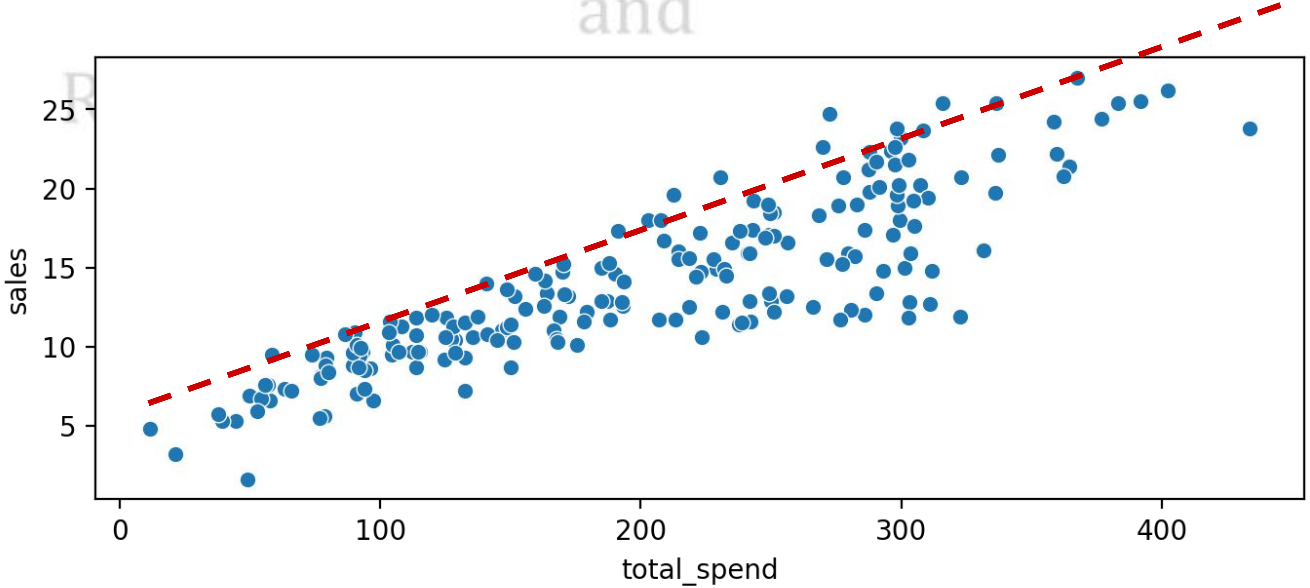


Fig Ref: Pierian Data

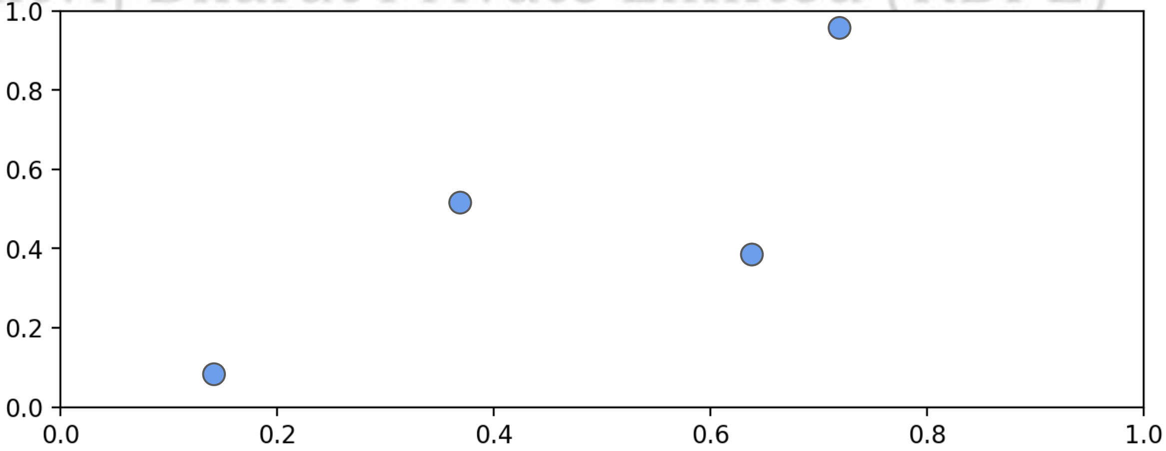


Fig Ref: Pierian Data

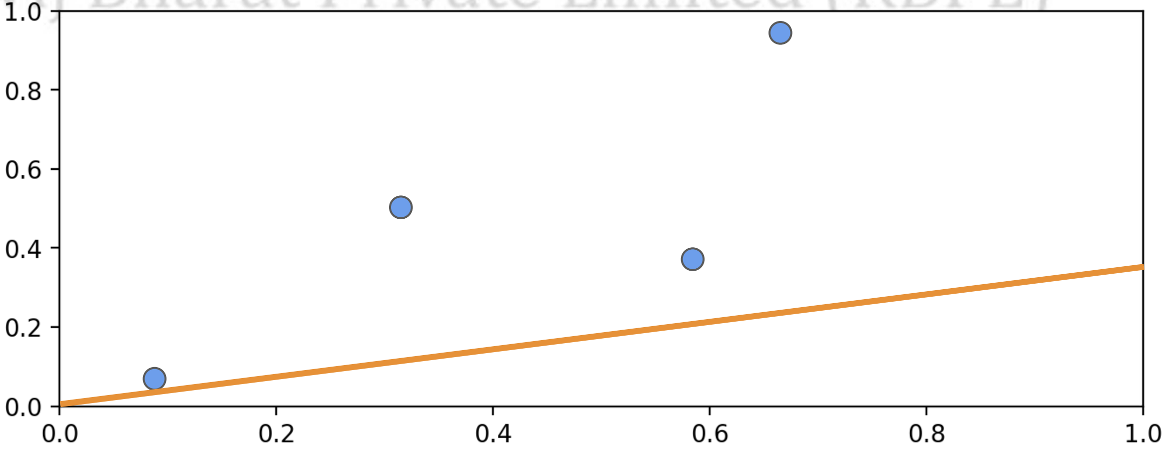


Fig Ref: Pierian Data

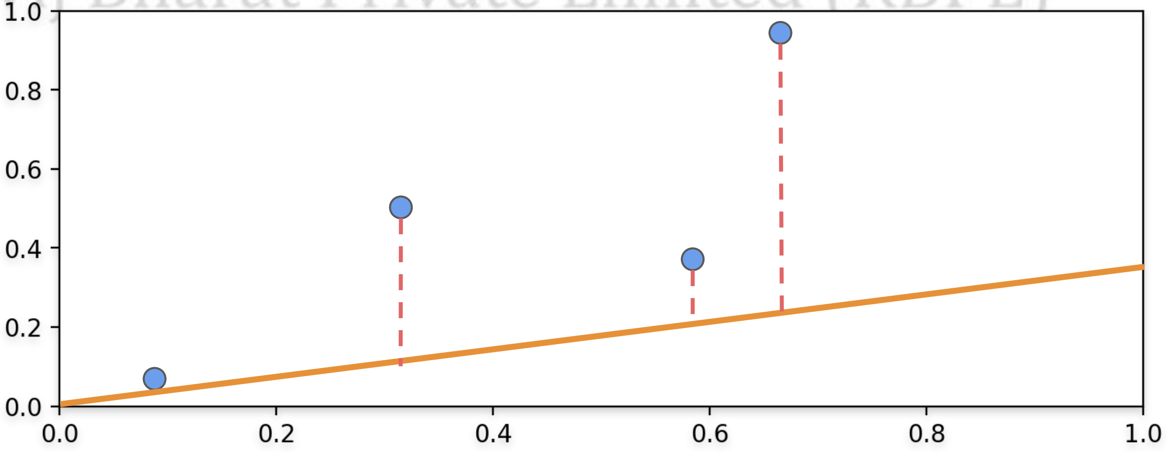


Fig Ref: Pierian Data

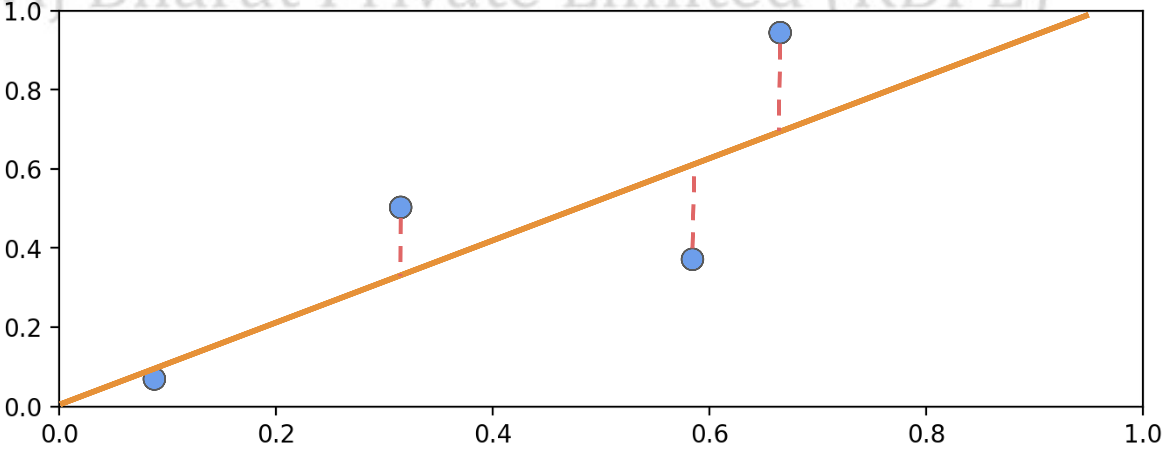


Fig Ref: Pierian Data

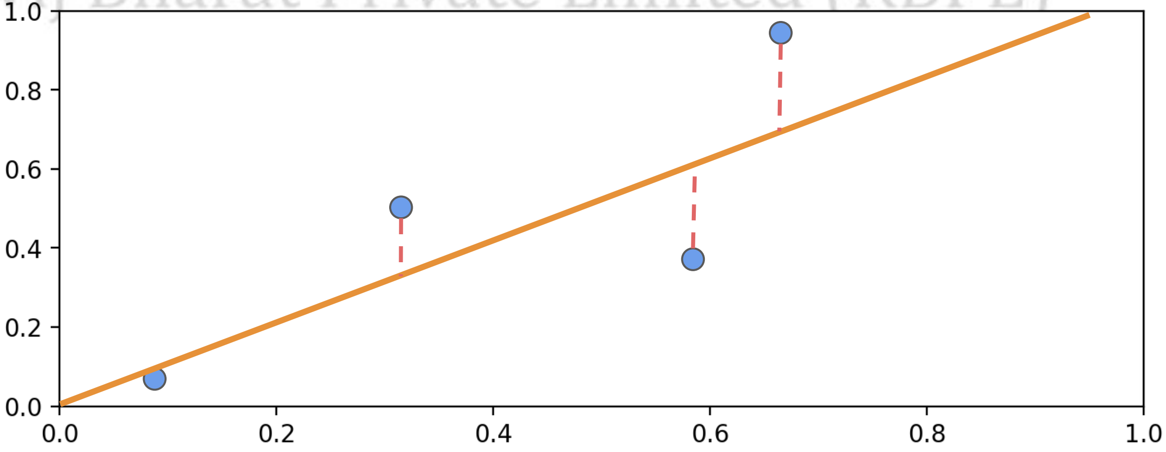


Fig Ref: Pierian Data

Ordinary Least Squares (OLS)

Ordinary Least Squares (OLS) operates by minimizing the total sum of the squared deviations between the observed values of the dependent variable in the provided dataset and the values predicted by the linear function.

Fig Ref: Pierian Data

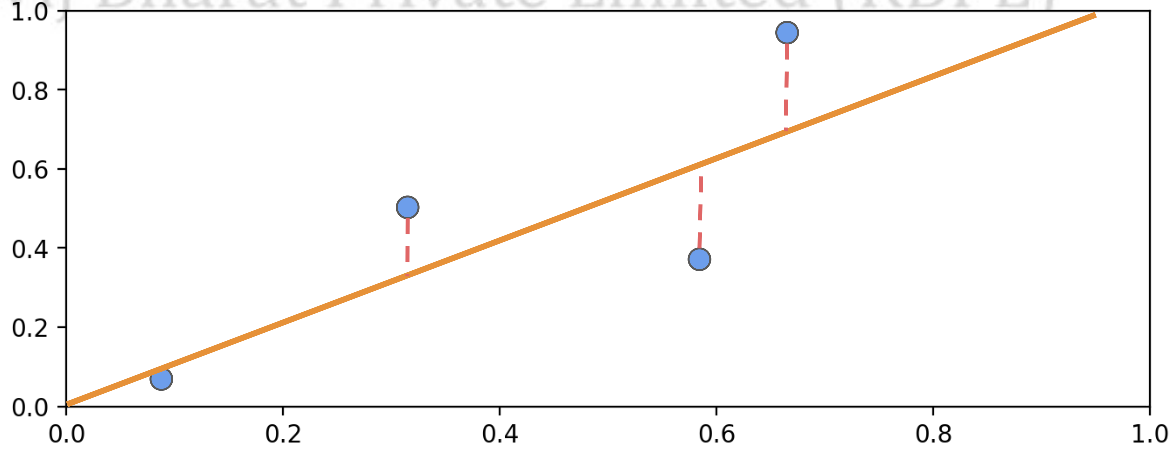


Fig Ref: Pierian Data

We can visualize squared error to minimize:

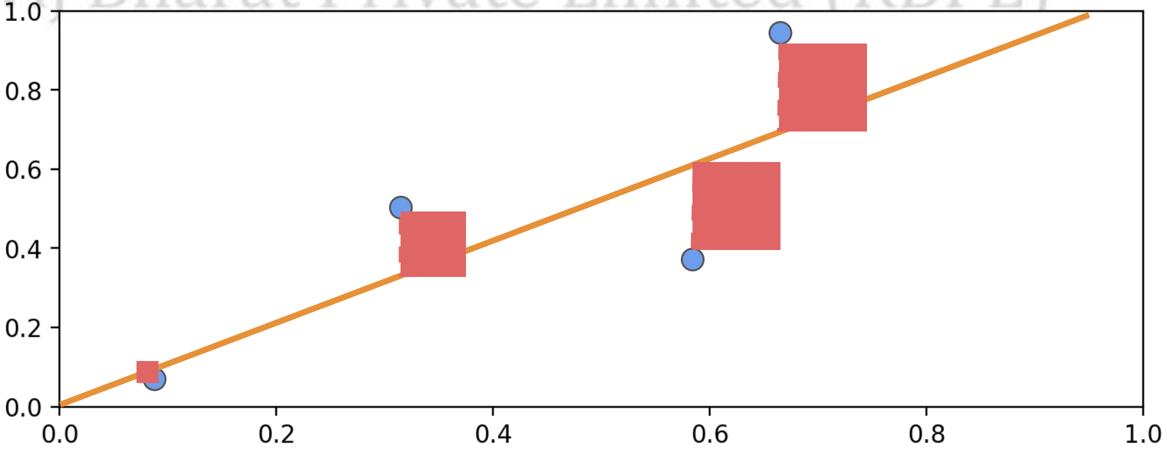


Fig Ref: Pierian Data

Let's continue exploring OLS by converting a real data set into mathematical notation, then working to solve a linear relationship between features and a variable!

Fig Ref: Pierian Data

Algorithm Theory - Part Two
OLS Equations

An Introduction to Machine Learning with Python Programming
12 Sep 2023 - 20 Oct 2023

iHUB, DivyaSampark, IIT Roorkee
and
Ritvij Bharat Private Limited (RBPL)

Presented By:
Shreyas Shukla

Fig Ref: Pierian Data

- **y = mx + b**
  - m is slope
  - b is intercept with y-axis

Fig Ref: Pierian Data

We can see for **y=mx+b**
there is only room for one possible feature x.

OLS will allow us to directly solve for **m** and **b.**

Linear Regression enables us to establish a connection between multiple features in order to estimate a desired output.

Area m^2	Bedrooms	Bathrooms	Price
200	2	2	Rs.50,00,000
190	1	1	Rs. 40, 50,000
230	3	2	Rs. 60, 50,000
180	2	1	Rs. 40, 00,000
210	3	1	Rs. 50, 50,000

Fig Ref: Pierian Data

Let’s translate this data into generalized mathematical notation.

X			y
Area m^2	**Bedrooms**	**Bathrooms**	**Price**
200	2	2	Rs.50,00,000
190	1	1	Rs. 40, 50,000
230	3	2	Rs. 60, 50,000
180	2	1	Rs. 40, 00,000
210	3	1	Rs. 50, 50,000

Fig Ref: Pierian Data

Let’s translate this data into generalized mathematical notation.

X			**y**
x_1	x_2	x_3	y
200	3	2	Rs.50,00,000
190	2	1	Rs.40,50,000
230	3	3	Rs.60,50,000
180	1	1	Rs.40,00,000
210	2	2	Rs.50,50,000

Fig Ref: Pierian Data

X			y
x_1	x_2	x_3	y

Fig Ref: Pierian Data

Reformat for **y = x** equation

y		X	
y	x_1	x_2	x_3

Fig Ref: Pierian Data

Every feature should possess an associated Beta coefficient.

y	X		
y	x_1	x_2	x_3

$$\hat{y} = \beta_0 x_0 + \cdots + \beta_n x_n$$

Fig Ref: Pierian Data

Same as the common notation for a simple line: **y=mx+b**

y	X		
y	x_1	x_2	x_3

$$\hat{y} = \beta_0 x_0 + \cdots + \beta_n x_n$$

Fig Ref: Pierian Data

Fully generalized for any number of features.

y	X		
y	x_1	x_2	x_3

$$\hat{y} = \beta_0 x_0 + \cdots + \beta_n x_n$$

$$\hat{y} = \sum_{i=0}^{n} \beta_i x_i$$

Fig Ref: Pierian Data

y hat because there is usually no set of Betas to create a perfect fit to y!

$$\hat{y} = \sum_{i=0}^{n} \beta_i x_i$$

Line equation:

$$\hat{y} = \sum_{i=0}^{n} \beta_i x_i$$

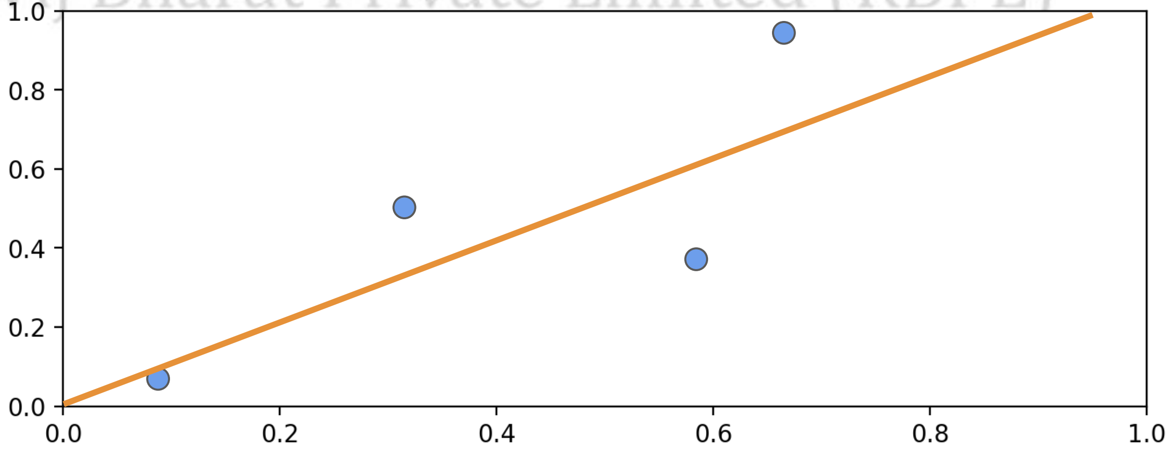


Fig Ref: Pierian Data

$$\hat{y} = \sum_{i=0}^{n} \beta_i \boxed{x_i}$$

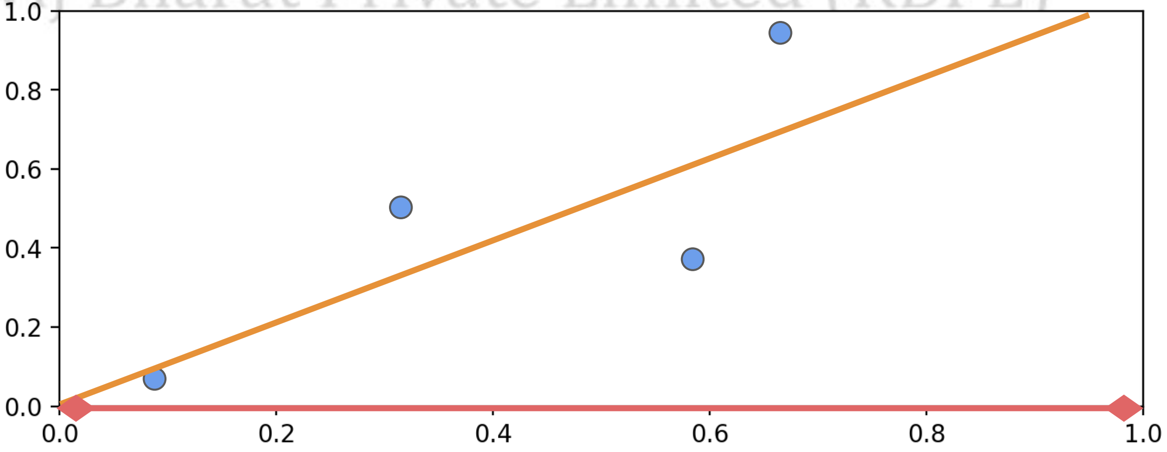


Fig Ref: Pierian Data

For simple problems with one X feature we can easily solve for Betas values with an analytical solution.

Algorithm Theory
Cost Function

An Introduction to Machine Learning with Python Programming
12 Sep 2023 - 20 Oct 2023

iHUB, DivyaSampark, IIT Roorkee
and
Ritvij Bharat Private Limited (RBPL)

Presented By:
Shreyas Shukla

Fig Ref: Pierian Data

Recall we are searching for Beta values for a best-fit line.

$$\hat{y} = \sum_{i=0}^{n} \beta_i x_i$$

Fig Ref: Pierian Data

How to choose these beta coefficients?

$$\hat{y} = \sum_{i=0}^{n} \beta_i x_i$$

Fig Ref: Pierian Data

We've decided to define a "best-fit" as **minimizing the squared error.**

$$\hat{y} = \sum_{i=0}^{n} \beta_i x_i$$

The residual error for some row ***j*** is:

$$y^j - \hat{y}^j$$

Fig Ref: Pierian Data

Squared Error for some row ***j*** is then:

$$\left(y^j - \hat{y}^j\right)^2$$

Sum of squared errors for **m** rows is then:

$$\sum_{j=1}^{m} \left(y^{j} - \hat{y}^{j} \right)^{2}$$

Average squared error for **m** rows:

$$\frac{1}{m}\sum_{j=1}^{m}\left(y^{j}-\hat{y}^{j}\right)^{2}$$

Fig Ref: Pierian Data

- Begin by defining a cost function **J.**

$$J(\boldsymbol{\beta})$$

A **cost function** is defined by some measure of error.

$$J(\boldsymbol{\beta})$$

- This means we wish to **minimize** the cost function.

$$J(\boldsymbol{\beta})$$

Our cost function can be defined by the squared error:

$$J(\boldsymbol{\beta}) = \frac{1}{2m}\sum_{j=1}^{m}\left(y^j - \hat{y}^j\right)^2$$

Error between real **y** and predicted **ŷ**

$$J(\boldsymbol{\beta}) = \frac{1}{2m}\sum_{j=1}^{m}\left(\boxed{y^j - \hat{y}^j}\right)^2$$

Fig Ref: Pierian Data

Squaring corrects for negative and positive errors.

$$J(\boldsymbol{\beta}) = \frac{1}{2m} \boxed{\sum_{j=1}^{m}} \left(y^j - \hat{y}^j \right)^{\boxed{2}}$$

Fig Ref: Pierian Data

Summing error for m rows.

$$J(\boldsymbol{\beta}) = \frac{1}{2m} \sum_{j=1}^{\boxed{m}} \left(y^j - \hat{y}^j \right)^2$$

Summing error for m rows.

$$J(\boldsymbol{\beta}) = \frac{1}{2m}\boxed{\sum_{j=1}^{m}}\left(y^j - \hat{y}^j\right)^2$$

Divide by **m** to get **mean**

$$J(\boldsymbol{\beta}) = \boxed{\frac{1}{2m}} \sum_{j=1}^{m} \left(y^j - \hat{y}^j\right)^2$$

Additional ½ is for convenience for derivative.

$$J(\boldsymbol{\beta}) = \frac{1}{\boxed{2}m} \sum_{j=1}^{m} \left(y^j - \hat{y}^j \right)^{\boxed{2}}$$

$$J(\boldsymbol{\beta}) = \frac{1}{2m}\sum_{j=1}^{m}\left(y^j - \hat{y}^j\right)^2$$

$$= \frac{1}{2m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)^2$$

From calculus we know that to minimize a function we can take its derivative and set it equal to zero.

$$\frac{\partial J}{\partial \beta_k}(\boldsymbol{\beta}) = \frac{\partial}{\partial \beta_k}\left(\frac{1}{2m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)^2\right)$$

$$= \frac{1}{m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)(-x_k^j)$$

Fig Ref: Pierian Data

$$
\boxed{\frac{\partial J}{\partial \beta_k}}(\boldsymbol{\beta}) = \boxed{\frac{\partial}{\partial \beta_k}}\left(\frac{1}{2m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)^2\right)
$$

$$
= \frac{1}{m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)(-x_k^j)
$$

Recall from calculus to minimize a function we can take its derivative and set it equal to zero.

$$\frac{\partial J}{\partial \beta_k}(\boldsymbol{\beta}) = \frac{\partial}{\partial \beta_k}\left(\frac{1}{\boxed{2}m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)^{\boxed{2}}\right)$$

$$= \frac{1}{m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)(-x_k^j)$$

- Unfortunately, it is not scalable to try to get an analytical solution to minimize this cost function.
- So we use **gradient descent** to minimize this **cost function.**

An Introduction to Machine Learning with Python Programming
12 Sep 2023 - 20 Oct 2023

iHUB, DivyaSampark, IIT Roorkee
and
Ritvij Bharat Private Limited (RBPL)

Algorithm Theory
Gradient Descent

Presented By:
Shreyas Shukla

Fig Ref: Pierian Data

we can describe this cost function through vectorized matrix notation and use **gradient descent** to have a computer figure out the set of Beta coefficient values that minimize the **cost/loss** function.

Fig Ref: Pierian Data

Derivative of the cost function:

$$\frac{\partial J}{\partial \beta_k}(\boldsymbol{\beta}) = \frac{1}{m} \sum_{j=1}^{m} \left(y^j - \sum_{i=0}^{n} \beta_i x_i^j \right) (-x_k^j)$$

We can use a **gradient** to express the derivative of the cost function with respect to each β

$$\nabla_{\boldsymbol{\beta}} J = \begin{bmatrix} \frac{\partial J}{\partial \beta_0} \\ \vdots \\ \frac{\partial J}{\partial \beta_n} \end{bmatrix}$$

Fig Ref: Pierian Data

$$\nabla_{\beta} J = \begin{bmatrix} -\frac{1}{m}\sum_{j=1}^{m}\left(y^{j} - \sum_{i=0}^{n}\beta_i x_i^{j}\right)x_0^{j} \\ \vdots \\ -\frac{1}{m}\sum_{j=1}^{m}\left(y^{j} - \sum_{i=0}^{n}\beta_i x_i^{j}\right)x_n^{j} \end{bmatrix}$$

Fig Ref: Pierian Data

We know we can vectorize our data:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^1 & x_2^1 & \dots & x_n^1 \\ 1 & x_1^2 & x_2^2 & \dots & x_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^m & x_2^m & \dots & x_n^m \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{bmatrix}$$

$$\nabla_{\boldsymbol{\beta}} J = \begin{bmatrix} -\frac{1}{m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)x_0^j \\ \vdots \\ -\frac{1}{m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)x_n^j \end{bmatrix}$$

Fig Ref: Pierian Data

$$\nabla_{\beta} J = \begin{bmatrix} -\boxed{\frac{1}{m}\sum_{j=1}^{m}}\left(y^j - \boxed{\sum_{i=0}^{n}\beta_i x_i^j\right) x_0^j} \\ \vdots \\ -\frac{1}{m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right) x_n^j \end{bmatrix}$$

$$\nabla_{\beta} J = -\frac{1}{m} \begin{bmatrix} \sum_{j=1}^{m} y^j x_0^j \\ \vdots \\ \sum_{j=1}^{m} y^j x_n^j \end{bmatrix} + \frac{1}{m} \begin{bmatrix} \sum_{j=1}^{m} \sum_{i=0}^{n} \beta_i x_i^j x_0^j \\ \vdots \\ \sum_{j=1}^{m} \sum_{i=0}^{n} \beta_i x_i^j x_n^j \end{bmatrix}$$

Fig Ref: Pierian Data

We can now calculate the gradient for any set of Beta values!

$$\nabla_{\boldsymbol{\beta}} J = -\frac{1}{m}\begin{bmatrix} \sum_{j=1}^{m} y^j x_0^j \\ \vdots \\ \sum_{j=1}^{m} y^j x_n^j \end{bmatrix} + \frac{1}{m}\begin{bmatrix} \sum_{j=1}^{m}\sum_{i=0}^{n} \beta_i x_i^j x_0^j \\ \vdots \\ \sum_{j=1}^{m}\sum_{i=0}^{n} \beta_i x_i^j x_n^j \end{bmatrix}$$

What is the best way to "guess"?

$$\nabla_{\beta} J = -\frac{1}{m}\begin{bmatrix} \sum_{j=1}^{m} y^j x_0^j \\ \vdots \\ \sum_{j=1}^{m} y^j x_n^j \end{bmatrix} + \frac{1}{m}\begin{bmatrix} \sum_{j=1}^{m} \sum_{i=0}^{n} \beta_i x_i^j x_0^j \\ \vdots \\ \sum_{j=1}^{m} \sum_{i=0}^{n} \beta_i x_i^j x_n^j \end{bmatrix}$$

Fig Ref: Pierian Data

Common mountain analogy

source:pierian_data

An Introduction to Machine Learning with Python Programming
12 Sep 2023 - 20 Oct 2023

iHUB, DivyaSampark, IIT Roorkee

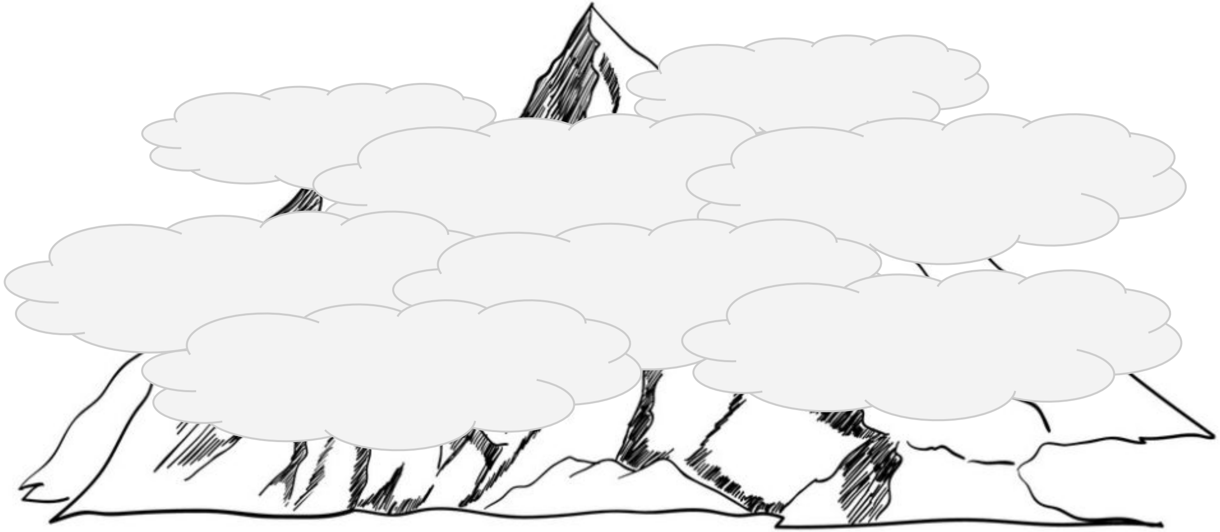


Fig Ref: Pierian Data

source:pierian_data

An Introduction to Machine Learning with Python Programming
12 Sep 2023 - 20 Oct 2023

iHUB, DivyaSampark, IIT Roorkee

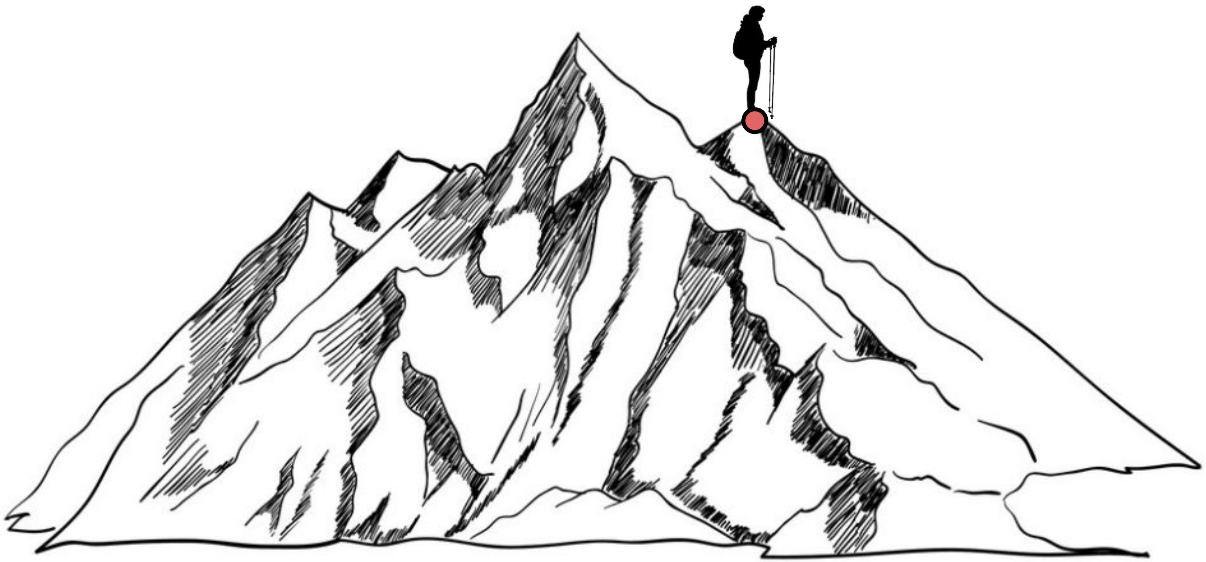


Fig Ref: Pierian Data

source:pierian_data

An Introduction to Machine Learning with Python Programming
12 Sep 2023 - 20 Oct 2023

iHUB, DivyaSampark, IIT Roorkee

Fig Ref: Pierian Data

source:pierian_data

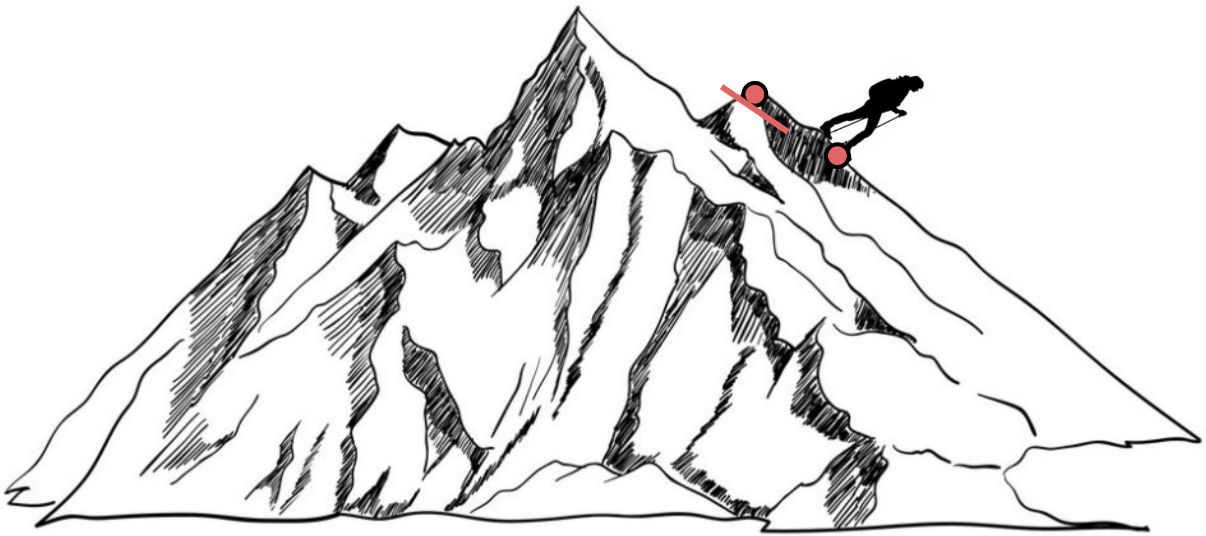


Fig Ref: Pierian Data

source:pierian_data

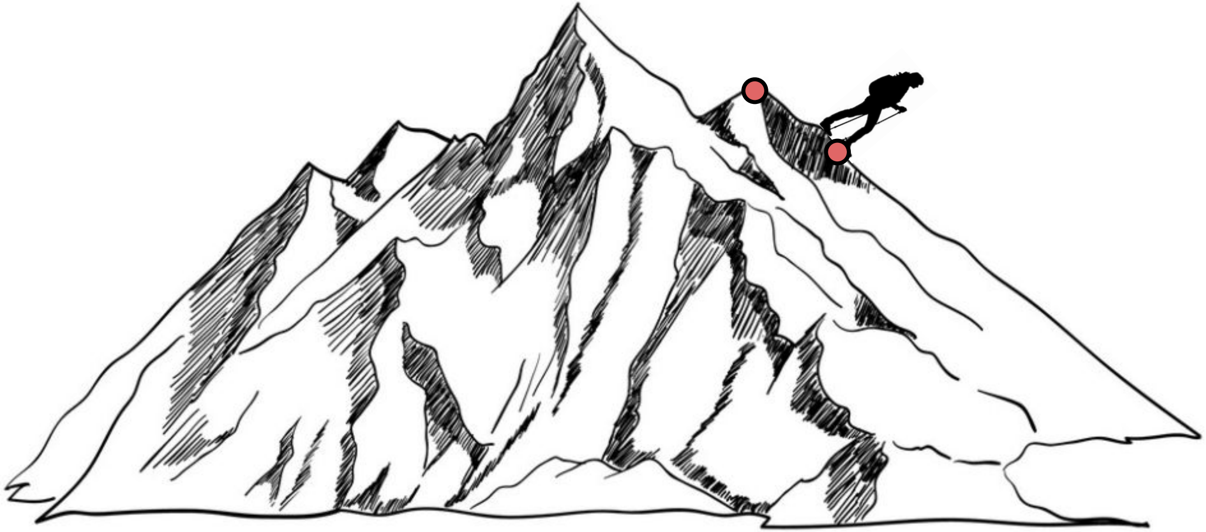


Fig Ref: Pierian Data

source:pierian_data

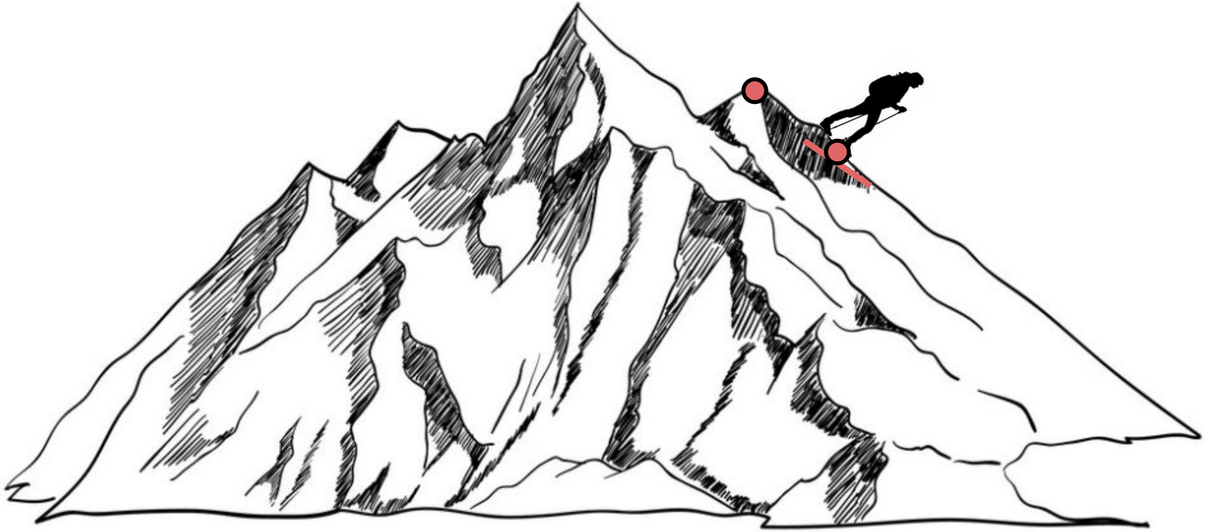


Fig Ref: Pierian Data

source:pierian_data

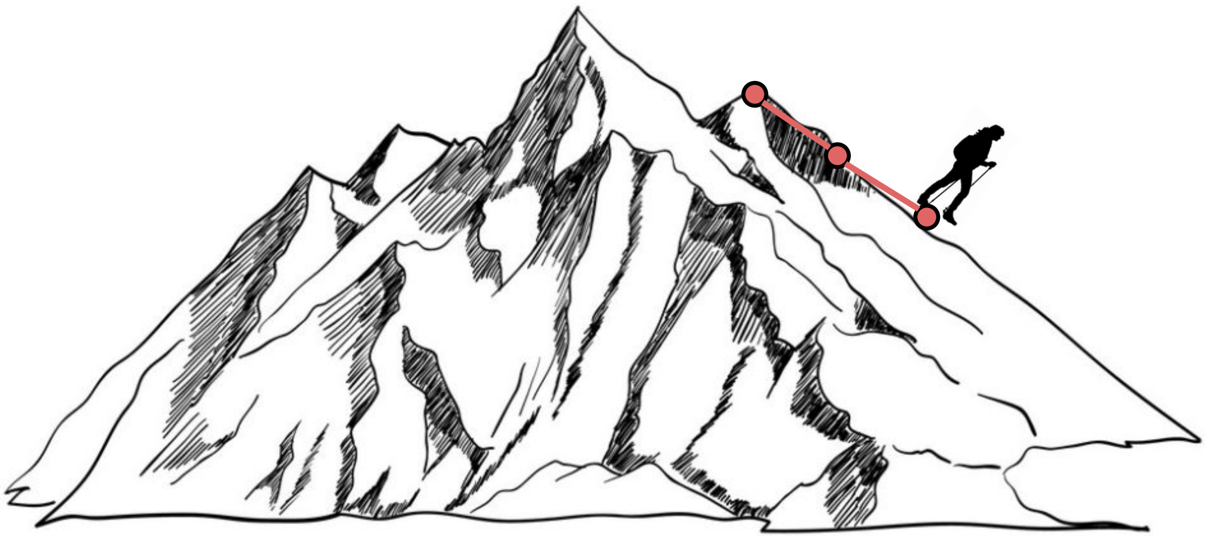


Fig Ref: Pierian Data

source:pierian_data

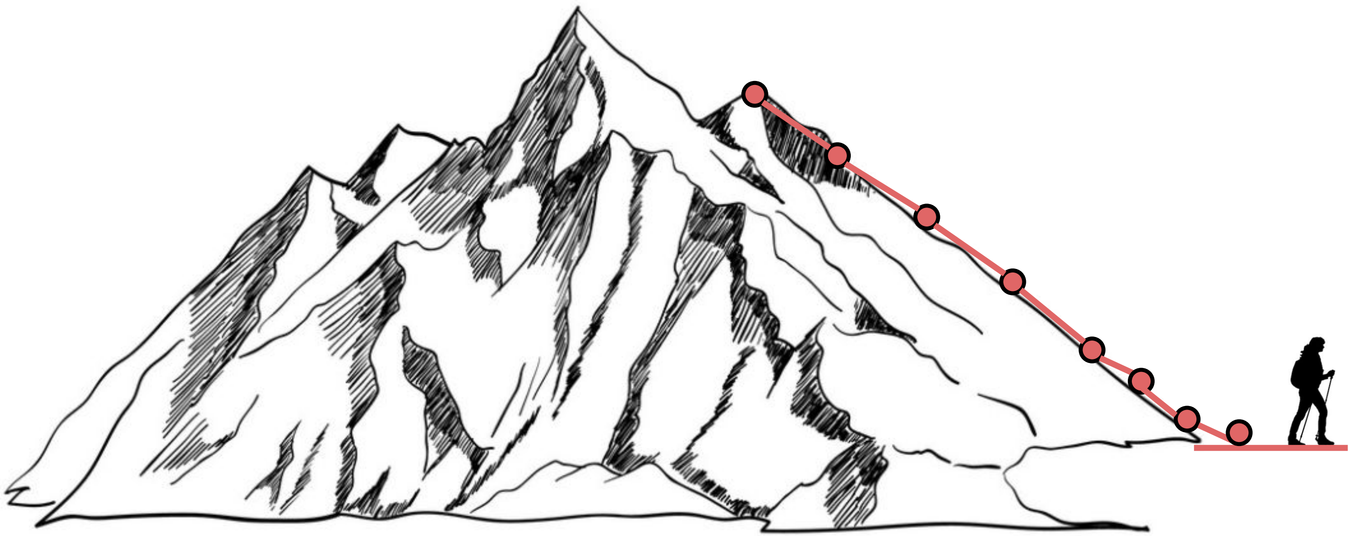


Fig Ref: Pierian Data

source:pierian_data

1 dimensional cost function (single Beta)

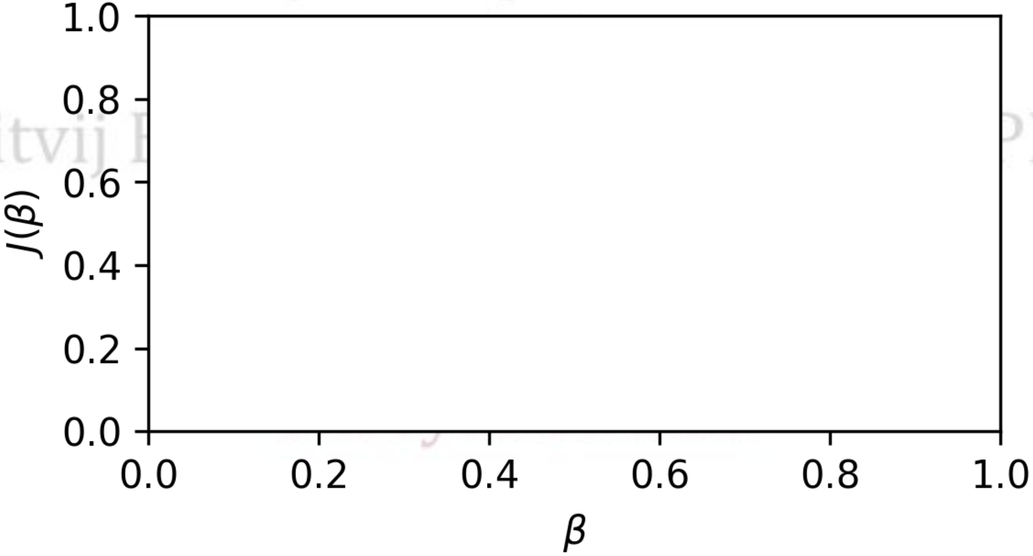
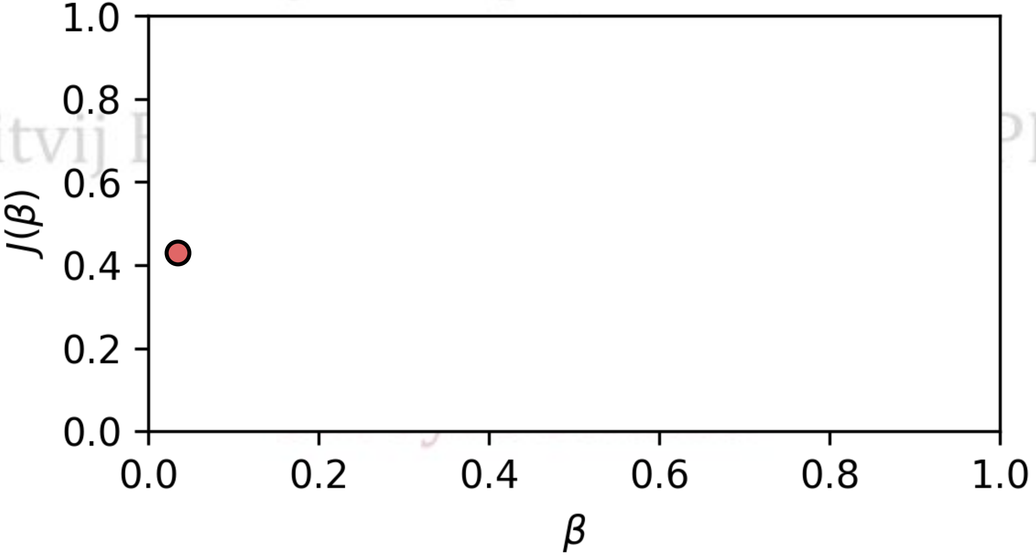
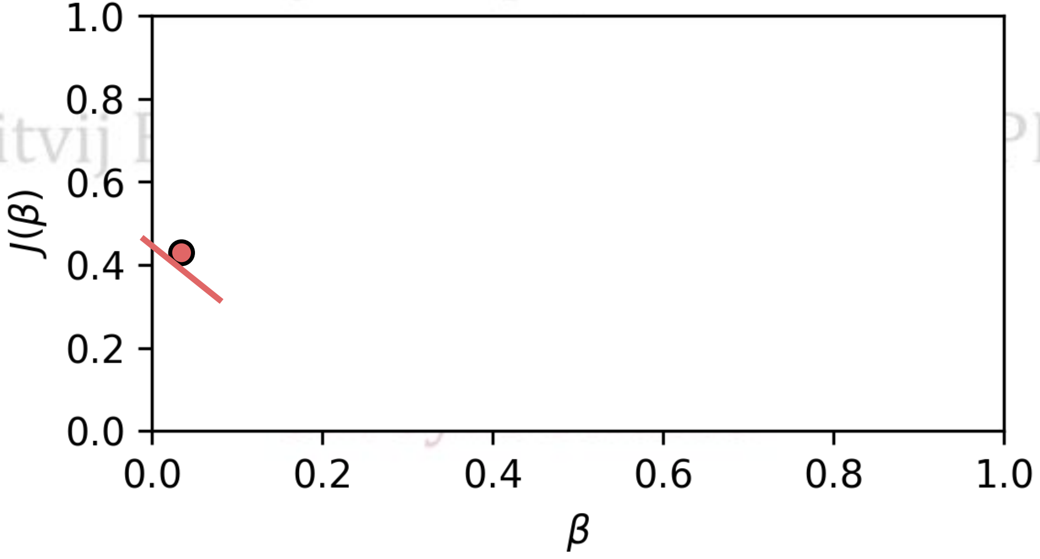


Fig Ref: Pierian Data

Choose a starting point



Calculate gradient at that point



Step forward proportional to **negative** gradient

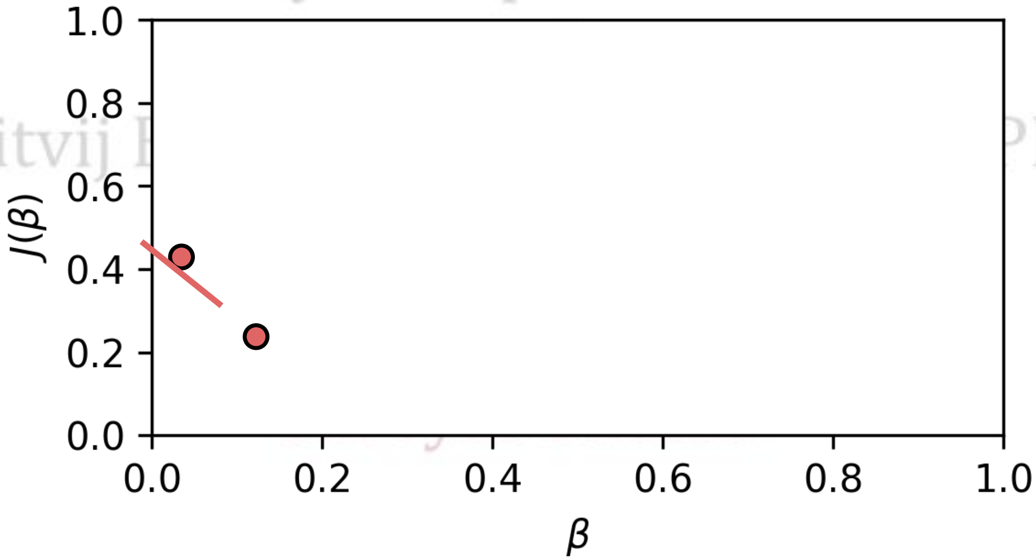
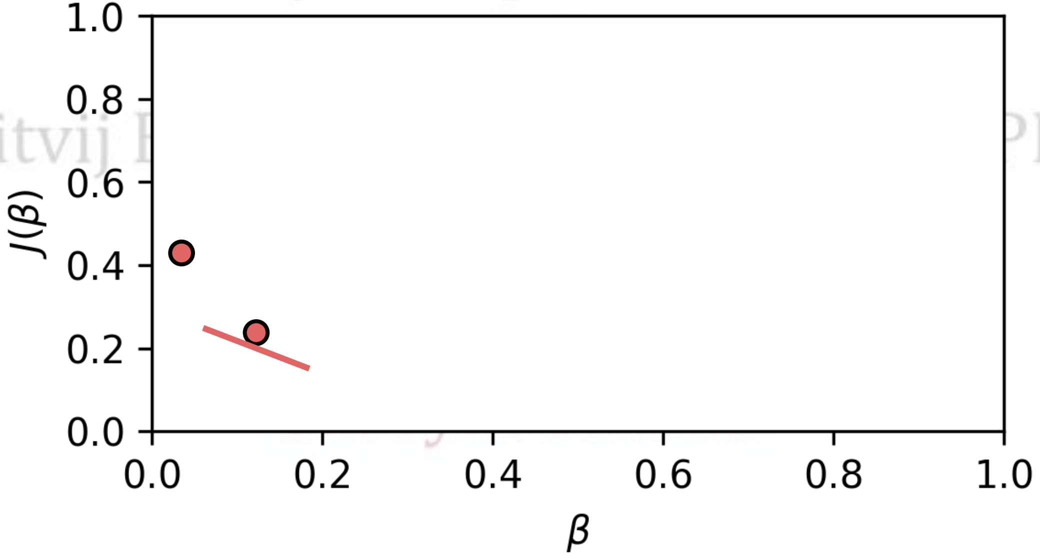
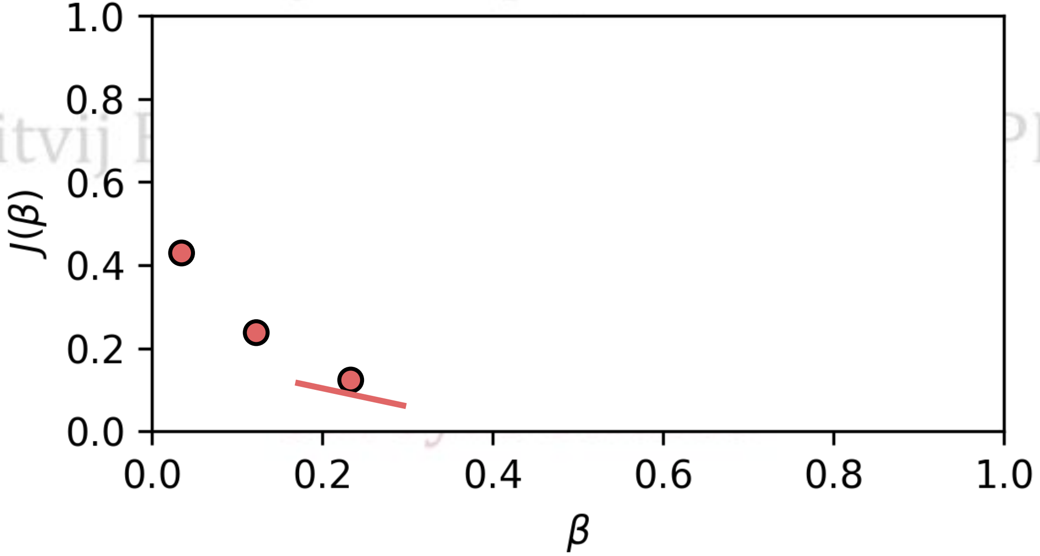


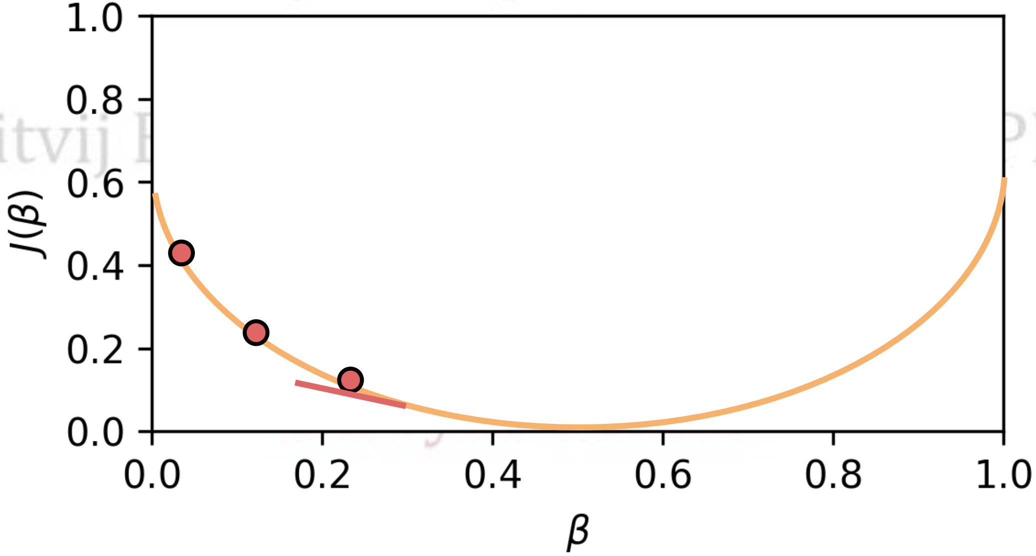
Fig Ref: Pierian Data

Repeat

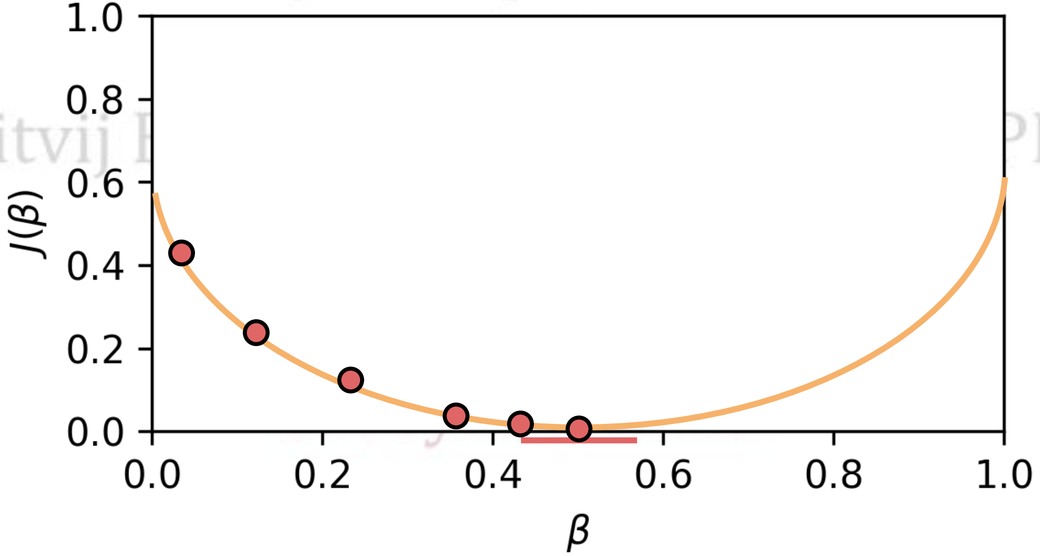


Repeat




1.0
0.8
0.6
0.4
0.2
0.0
J(β)
0.0
0.2
0.4
0.6
0.8
1.0
β

Eventually we will find the Beta that minimizes the cost function!



Steps are proportional to negative gradient!
Steeper gradient at start gives larger steps.
Smaller gradient at end gives smaller steps.

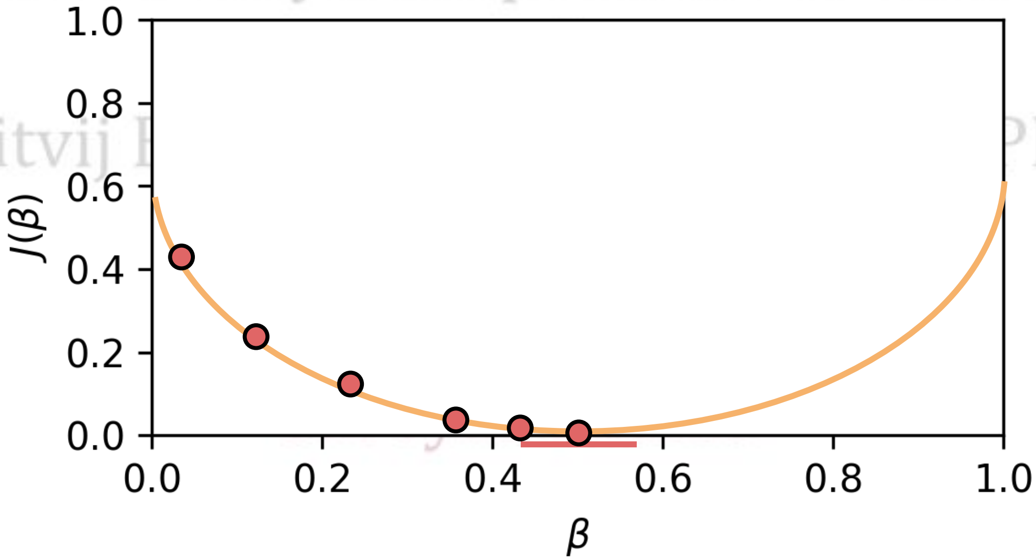


Fig Ref: Pierian Data

To further understand this, let's visualize this gradient descent search for two Beta values.

Process is still the same:

- Calculate gradient at point.
- Move in a step size proportional to negative gradient.
- Repeat until minimum is found.

An Introduction to Machine Learning with Python Programming

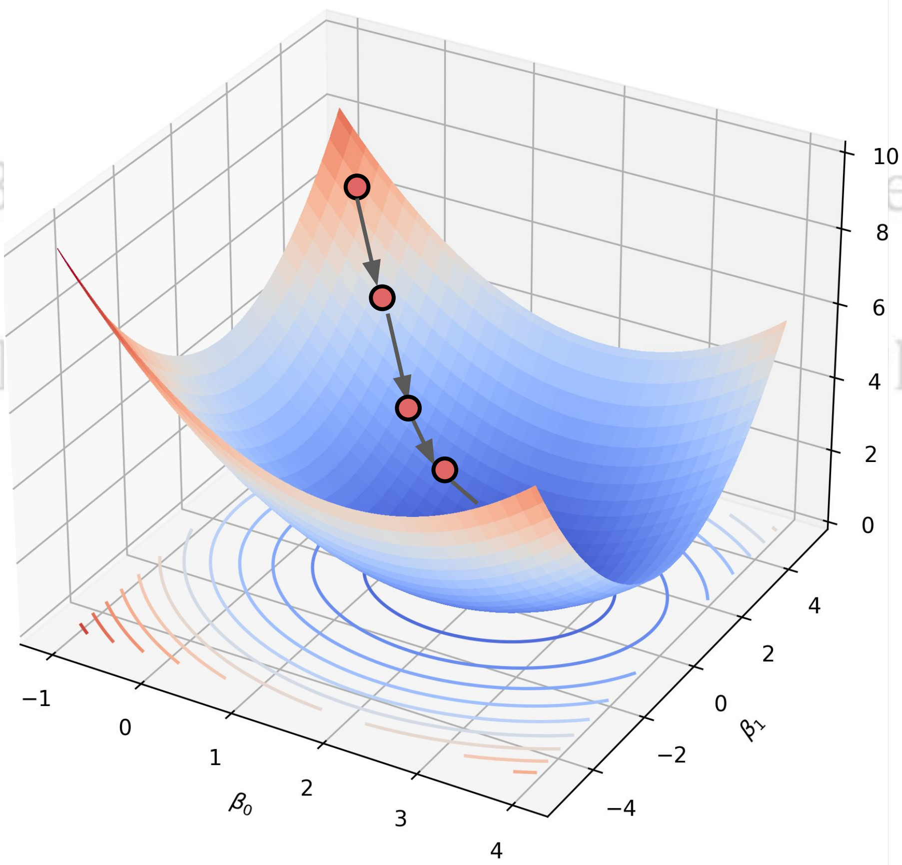


Fig Ref: Pierian Data

Cost Function/Loss Function: It basically ***determines how well a machine learning model performs for a given dataset.*** It calculates the difference between the expected value and predicted value and represents it as a single term.

Gradient Descent: "Gradient Descent is an optimization algorithm which is used for optimizing the cost function or error in the model." It enables the models to take the gradient or direction to reduce the errors by reaching to least possible error. Gradient descent is an iterative process where the model gradually converges towards a minimum value, and if the model iterates further than this point, it produces little or zero changes in the loss.

Fig Ref: Pierian Data

An Introduction to Machine Learning with Python Programming
12 Sep 2023 - 20 Oct 2023

iHUB, DivyaSampark, IIT Roorkee
and
Ritvij Bharat Private Limited (RBPL)

Simple Linear Regression

Presented By:
Shreyas Shukla

Fig Ref: Pierian Data

Now that we understand what is happening “under the hood” for linear regression, let’s begin by coding through an example of **simple linear regression.**

An Introduction to Machine Learning with Python Programming

12 Sep 2023 - 20 Oct 2023

Chapter 3 of ISLR.

iHUB, DivyaSampark, IIT Roorkee

and

Ritvij Bharat Private Limited (RBPL)

Presented By:

Shreyas Shukla

Fig Ref: Pierian Data

Simple Linear Regression

- Limited to one X feature (y=mx+b)
- We will create a best-fit line to map out a linear relationship between total advertising spend and resulting sales.

Let’s head over to the notebook!

Fig Ref: Pierian Data