MACHINE LEARNING

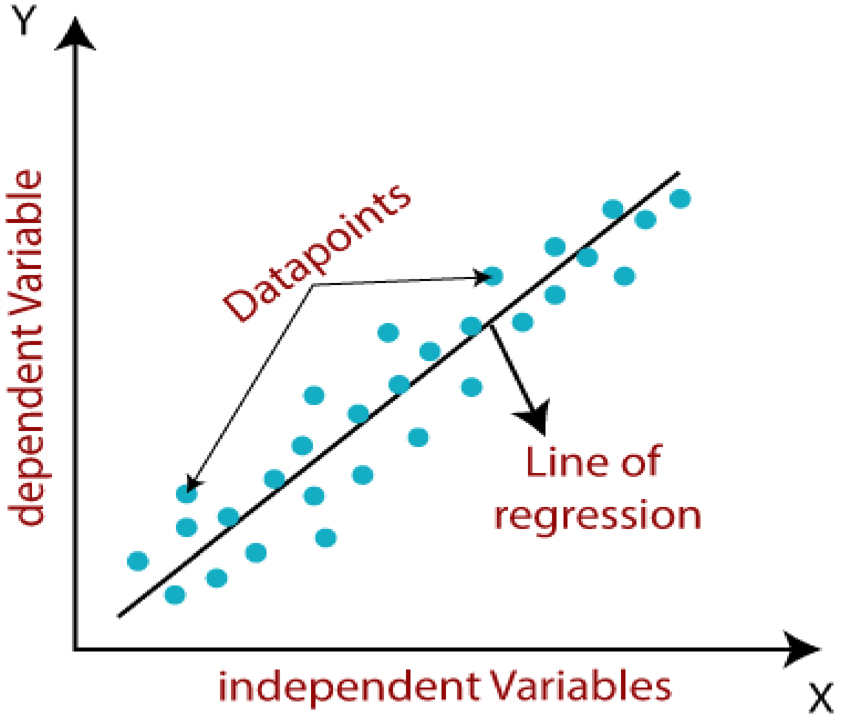
-LINEAR REGRESSION

SESHA

INDEX:

LINEAR REGRESSION

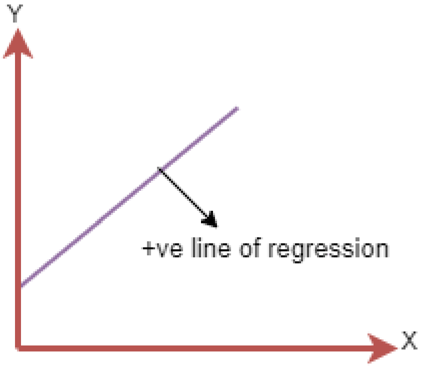
- When working with linear regression, our main goal is to find the best fit line that means the error between predicted values and actual values should be minimized. The best fit line will have the least error.
- Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.
- The linear regression model provides a sloped straight line representing the relationship between the variables. Consider the below image:



➢ LINEAR REGRESSION LINE:

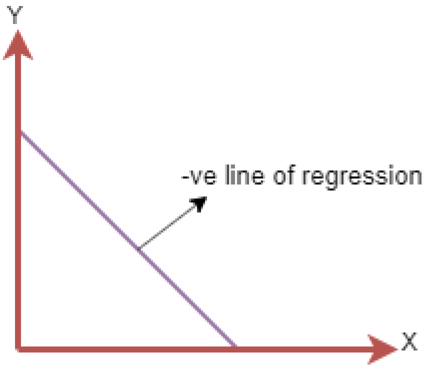
- A linear line showing the relationship between the dependent and independent variables is called a **regression line**.

  Positive Linear Relationship:
  If the dependent variable increases on the Y-axis and independent variable increases on X-axis, then such a relationship is termed as a Positive linear relationship.



The line equation will be: **$Y= a_0+a_1x$**

  Negative Linear Relationship:
  If the dependent variable decreases on the Y-axis and independent variable increases on the X-axis, then such a relationship is called a negative linear relationship.



The line of equation will be: **$Y= -a_0+a_1x$**

➢Cost function:

- The different values for weights or coefficient of lines (a_0, a_1) gives the different line of regression, and the cost function is used to estimate the values of the coefficient for the best fit line.
- For Linear Regression, we use the **Mean Squared Error (MSE)** cost function, which is the average of squared error occurred between the predicted values and actual values.

➢ **Residuals:**

- The distance between the actual value and predicted values is called residual. If the observed points are far from the regression line, then the residual will be high, and so cost function will high. If the scatter points are close to the regression line, then the residual will be small and hence the cost function.

➢Gradient Descent:

- Gradient descent is used to minimize the MSE by calculating the gradient of the cost function.
- A regression model uses gradient descent to update the coefficients of the line by reducing the cost function.
- It is done by a random selection of values of coefficient and then iteratively update the values to reach the minimum cost function.

➢ASSUMPTIONS OF LINEAR REGRESSION

Regression is a parametric approach. ‘Parametric’ means it makes assumptions about data for the purpose of analysis. Due to its parametric side, regression is restrictive in nature. It fails to deliver good results with data sets which doesn’t fulfil its assumptions. Therefore, for a successful regression analysis, it’s essential to validate these assumptions.

- **Linear relationship between the features and target:**
  Linear regression assumes the linear relationship between the dependent and independent variables.
- **Small or no multicollinearity between the features:**
  Multicollinearity means high-correlation between the independent variables. Due to multicollinearity, it may difficult to find the true relationship between the predictors and target variables. Or we can say, it is difficult to determine which predictor variable is affecting the target variable and which is not. So, the model assumes either little or no multicollinearity between the features or independent variables.
- **Homoscedasticity Assumption:**
  Homoscedasticity is a situation when the error term is the same for all the values of independent variables. With homoscedasticity, there should be no clear pattern distribution of data in the scatter plot.
- **Normal distribution of error terms:**
  Linear regression assumes that the error term should follow the normal distribution pattern. If error terms are not normally distributed, then confidence intervals will become either too wide or too narrow, which may cause difficulties in finding coefficients.
  It can be checked using the **q-q plot**. If the plot shows a straight line without any deviation, which means the error is normally distributed.
- **No autocorrelations:**
  The linear regression model assumes no autocorrelation in error terms. If there will be any correlation in the error term, then it will drastically reduce the accuracy of the model. Autocorrelation usually occurs if there is a dependency between residual errors.

➢ ADVANTAGES AND DISADVANTAGES OF LR:

ADVANTAGES	DISADVANTAGES
❖ **Linear regression performs exceptionally well for linearly separable data**	❖ The assumption of linearity between dependent and independent variables
❖ **Easier to implement, interpret and efficient to train**	❖ It is often quite prone to noise and overfitting
❖ **One more advantage is the extrapolation beyond a specific data set**	❖ It is prone to multicollinearity
❖ **It handles overfitting pretty well using dimensionally reduction techniques, regularization, and cross-validation**	❖ Linear regression is quite sensitive to outliers

➢TYPES OF LINEAR REGRESSION

Linear regression can be further divided into two types of the algorithm:

- **Simple Linear Regression:**
  If a single independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Simple Linear Regression.
- **Multiple Linear regression:**
  If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

➢ Mathematics and Intuition behind Linear Regression

❖ Table of Contents:

- Linear Regression is the supervised learning algorithm.
  It is used to predict continuous values.
  It uses the equation of the line to predict output values.
- `y = mx+c`
- consider, we have below values for x and y and want to predict values (y_pred) for some x values. so how can we do that??
  Let's see...

```
x = np.round(np.random.uniform(5, 25, 10))
y = x*2
print(y, x)

output:

[34. 14. 28. 24. 30. 30. 44. 30. 18. 14.] [17.  7. 14. 12. 15. 15. 22. 15.  9.

7.]
```

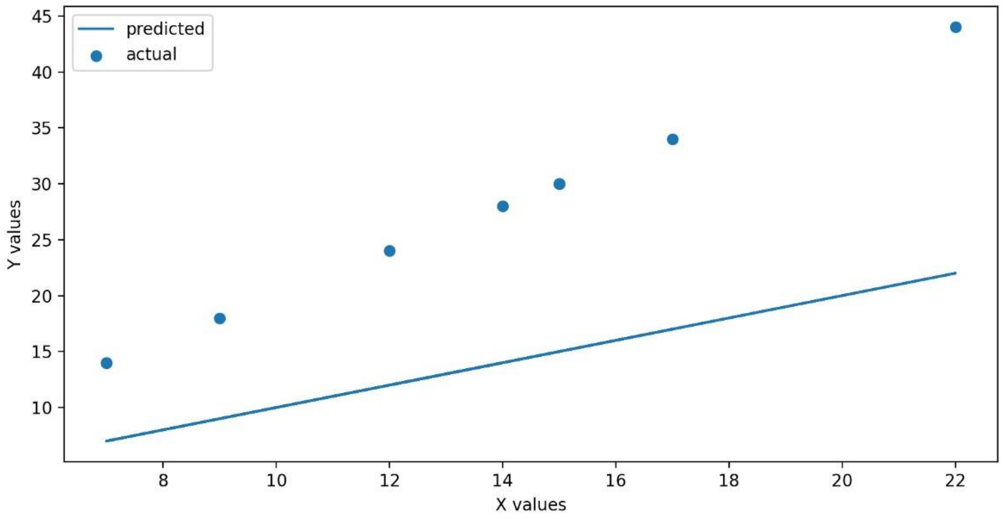
- we generated some values for x and y and we will predict values (y_pred) for x values, after that we will compare those predicted values (y_pred) with actual values of y, and also, we will find the error between actual and predicted values.
- we have the equation of a line. **y = mx+c.**
  so first of all we will predict values when **m=1** and **c =0**.

```
def predict_y(x, m=1, c=0):
    return m*x+c
y_pred = predict_y(x)
y_pred


output:
array([17.,  7., 14., 12., 15., 15., 22., 15.,  9.,  7.])
```

- we have found **y_pred**
  Now it's time to compare both **predicted** and **actual** values of **y**.
-

```
plt.figure(figsize=(10, 5), dpi=200)
plt.scatter(x, y, label="actual")
plt.plot(x, y_pred, label="predicted")
plt.xlabel('X values')
plt.ylabel('Y values')
plt.legend()
plt.show()
plt.savefig('lr.jpg')
```



- So here we can see that our predicted line is not the best-fitted line. we used values for **m=1** and **c=0**. we can use any combination of values and an infinite number of combinations of values, so how can we get those optimal values to find our best-fit line.
- here comes the **Gradient Descent Algorithm.**
- First, we will understand the **cost function**.

❖ Cost Function:

- The **cost function** is used to find the error between actual and predicted values.
  below is the formula for the cost function.

$$J(\theta) = \frac{1}{2m}\sum (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$costfunction = J(\Theta)$$
$$predictedvalue = h_\Theta(x^{(i)})$$
$$actualvalue = y^{(i)}$$

- $m = totalobservations$

- below is a python code for the cost function:

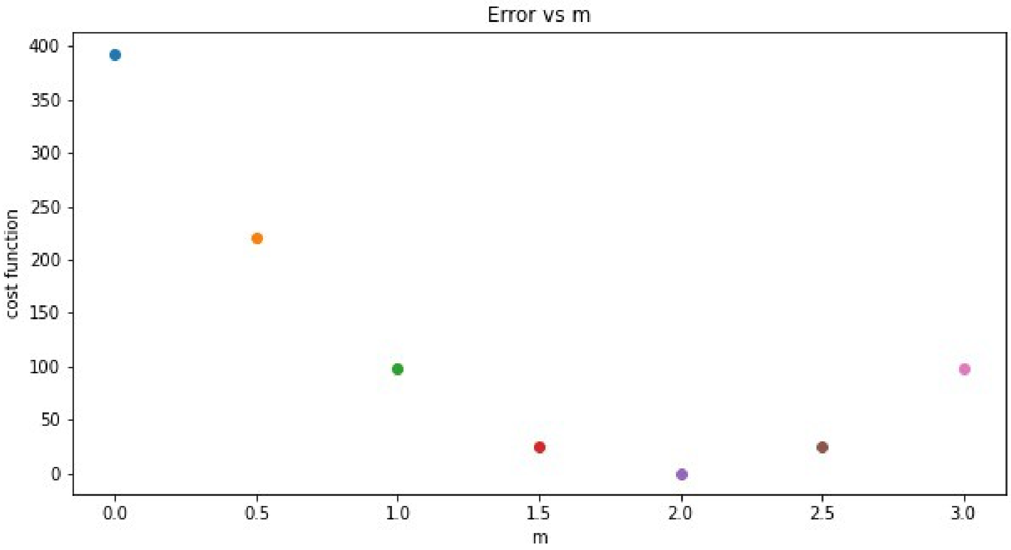
```
def cost_function(y_pred, y):
    return (np.sum(np.square(y_pred-y)))/(2*len(y))
cost_function(y_pred, y)
output:
98.35
```

- we found that our error is **98.35** when m=1 and c=0 and it's very huge**.**
- Now, we will see **cost values** for different values of '**m**' with **c** = **0** for simplicity. we can also use any value of '**c**' but for that, we need to plot a **3D diagram** and it will difficult to understand the concept.
- below python code is to find error values for different m values and we also plot the graph for different m values.

```
plt.figure(figsize=(10,5))M = [0, 0.5, 1, 1.5, 2, 2.5, 3]
for m in M:
    error = cost_function(predict_y(x, m), y)

    print(f'for m = {m} error is {error}')
    plt.scatter(m, error)

plt.ylabel("cost function")
plt.xlabel("m")
plt.title("Error vs m")
plt.savefig('costfunc.jpg')
plt.show()output:for m = 0 error is 393.4
for m = 0.5 error is 221.2875
for m = 1 error is 98.35
for m = 1.5 error is 24.5875
for m = 2 error is 0.0
for m = 2.5 error is 24.5875
for m = 3 error is 98.35
```



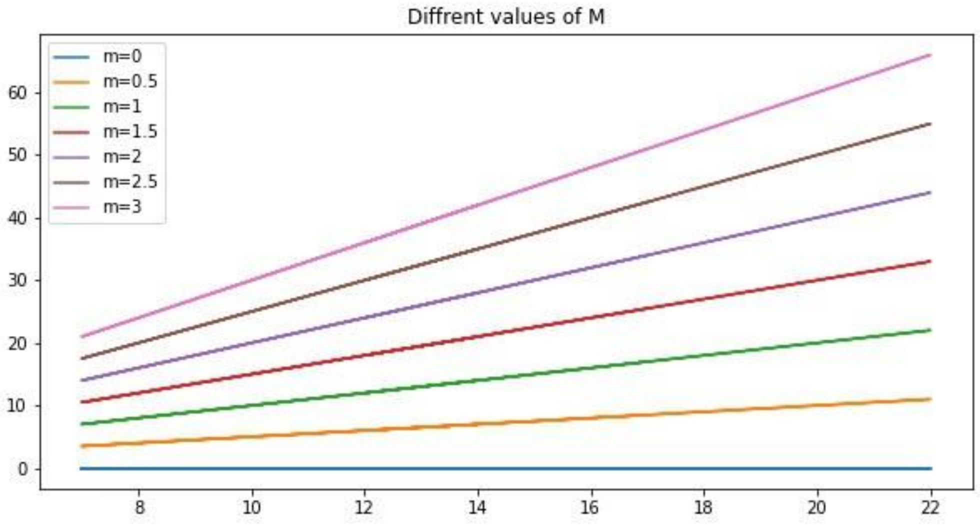
- here we can see that for **m=2 error is 0, which** is the optimal value of m.
- below figure, we are plotting different predicted lines for different values of **m**

➢

```
plt.figure(figsize=(10,5))def plot_m(m, x):
    plt.plot(x, predict_y(x, m), label=f'm={m}')for m in M:
    plot_m(m, x)

plt.title('Diffrent values of M')
plt.legend()
plt.savefig('diff_M.jpg')

plt.show()
```



❖ Gradient Descent Algorithm:

- It is a very important algorithm in machine learning.
  It is also used in deep learning.
  This algorithm is used to find the optimal attribute values.

$$\begin{aligned}&\text{repeat until convergence } \{\\ &\quad \theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)\\ &\quad \theta_1 := \theta_1 - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)\cdot x^{(i)}\\ &\}\end{aligned}$$

➢

- This is the formula used by the gradient descent algorithm.
  1st term is the old value of **θ** and 2nd term is the derivative of the **cost function**.
  α is the learning rate used for how big or small steps the algorithm has to take
  to find the optimal values.
  If the learning rate is too small it will take too many steps to find the values, and therefore it will take too much time to find values.
  If it is too large then it will overshoot the values and never reach optimal values.

This algorithm runs until it does not find the optimal value for the **θ**.

- If the learning rate is small we will observe that there will be a large number of points that are very close to each other in the graph and the learning rate is large there will be a very small number of points that are far away from each other.

- Till now we have discussed **linear regression, cost function,** and **gradient descent algorithm**.
- **Now it's time for the implementation of linear regression**.

❖ Implementation:

```
from sklearn.linear_model
import LinearRegressionlr = LinearRegression()
lr.fit(x.reshape(-1,1), y)
pred = lr.predict(x.reshape(-1,1))print(y, pred, sep='\n')



output:[34. 14. 28. 24. 30. 30. 44. 30. 18. 14.]
[34. 14. 28. 24. 30. 30. 44. 30. 18. 14.]
```

- value of **m**:

```
lr.coef_
output:
array([2.])
```

- value of **c**:

```
lr.intercept_
output:
-
7.105427357601002
```

- value of c is very small nearly equals to 0.

❖ Summary:

- we discuss **linear Regression, cost function,** and **the gradient descent algorithm.**

➢LASSO REGRESSION

- Lasso regression is a regularization technique used for more accurate prediction.
- Lasso regression is used for eliminating automated variables and the selection of features.
- Lasso regression makes coefficients to absolute zero

➢REG REGRESSION

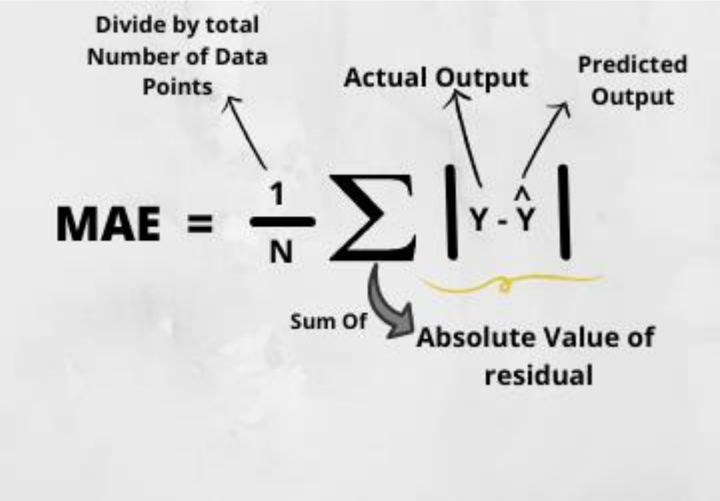
- ridge regression is a model turning method that is used for analyzing data suffering from multicollinearity
- It is used to reduce the overfitting the and helps to get the less variance, less bias.
- In ridge regression the slope values will shrink and never reaches to zero.

➢Metrics for Regression Model

- Mean Absolute Error(MAE)
- Mean Squared Error(MSE)
- RMSE
- RMSLE
- R squared
- Adjusted R Squares

1) Mean Absolute Error(MAE)

MAE is a very simple metric which calculates the absolute difference between actual and predicted values.


Divide by total Number of Data Points
Actual Output
Predicted Output
MAE = 1/N ∑ |Y - Ŷ|
Sum Of
Absolute Value of residual

Advantages of MAE

- The MAE you get is in the same unit as the output variable.
- It is most Robust to outliers.

Disadvantages of MAE

- The graph of MAE is not differentiable so we have to apply various optimizers like Gradient descent which can be differentiable.

2) Mean Squared Error(MSE)

MSE is a most used and very simple metric with a little bit of change in mean absolute error. Mean squared error states that finding the squared difference between actual and predicted value.

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \hat{y}\right)^2}_{\text{The square of the difference between actual and predicted}}$$

Advantages of MSE

The graph of MSE is differentiable, so you can easily use it as a loss function.

Disadvantages of MSE

- The value you get after calculating MSE is a squared unit of output. for example, the output variable is in meter(m) then after calculating MSE the output we get is in meter squared.
- If you have outliers in the dataset then it penalizes the outliers most and the calculated MSE is bigger. So, in short, It is not Robust to outliers which were an advantage in MAE.

- ## 3) Root Mean Squared Error(RMSE)
- As RMSE is clear by the name itself, that it is a simple square root of mean squared error.
- $$\text{RMSE} = \sqrt{\text{MSE}}$$

  $$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)}$$

Advantages of RMSE

- The output value you get is in the same unit as the required output variable which makes interpretation of loss easy.

Disadvantages of RMSE

- It is not that robust to outliers as compared to MAE.

for performing RMSE we have to NumPy NumPy square root function over MSE.

4) Root Mean Squared Log Error(RMSLE)

Taking the log of the RMSE metric slows down the scale of error. The metric is very helpful when you are developing a model without calling the inputs. In that case, the output will vary on a large scale.

To control this situation of RMSE we take the log of calculated RMSE error and resultant we get as RMSLE.

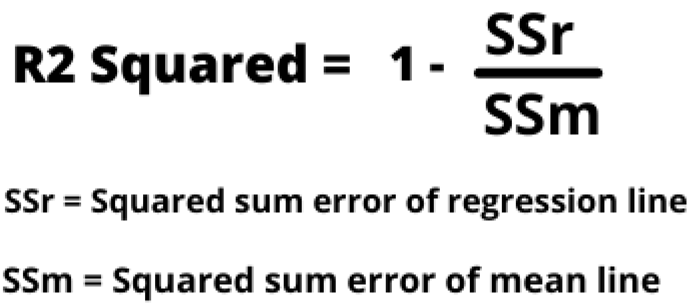
5) R Squared (R2)

R2 score is a metric that tells the performance of your model, not the loss in an absolute sense that how many wells did your model perform.

In contrast, MAE and MSE depend on the context as we have seen whereas the R2 score is independent of context.

So, with help of R squared we have a baseline model to compare a model which none of the other metrics provides. The same we have in classification problems which we call a threshold which is fixed at 0.5. So basically R2 squared calculates how must regression line is better than a mean line.

Hence, R2 squared is also known as Coefficient of Determination or sometimes also known as Goodness of fit.



$$\textbf{R2 Squared} = 1 - \frac{SSr}{SSm}$$

SSr = Squared sum error of regression line

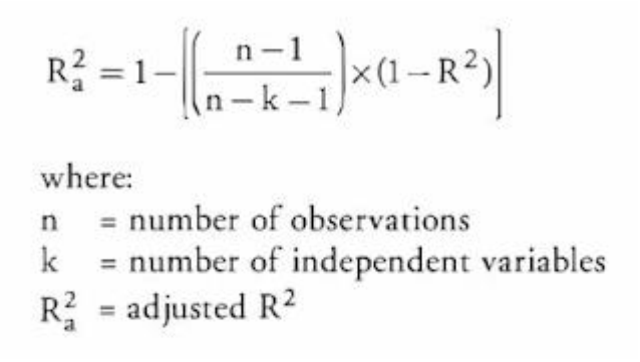
SSm = Squared sum error of mean line

6) Adjusted R Squared

The disadvantage of the R2 score is while adding new features in data the R2 score starts increasing or remains constant but it never decreases because It assumes that while adding more data variance of data increases.

But the problem is when we add an irrelevant feature in the dataset then at that time R2 sometimes starts increasing which is incorrect.

Hence, To control this situation Adjusted R Squared came into existence.



$$R_a^2 = 1 - \left[\left(\frac{n-1}{n-k-1}\right) \times (1 - R^2)\right]$$

where:

n = number of observations

k = number of independent variables

R_a^2 = adjusted R^2

-

-
- **2) Mean Squared Error(MSE)**
- MSE is a most used and very simple metric with a little bit of change in mean absolute error. Mean squared error states that finding the squared difference between actual and predicted value.