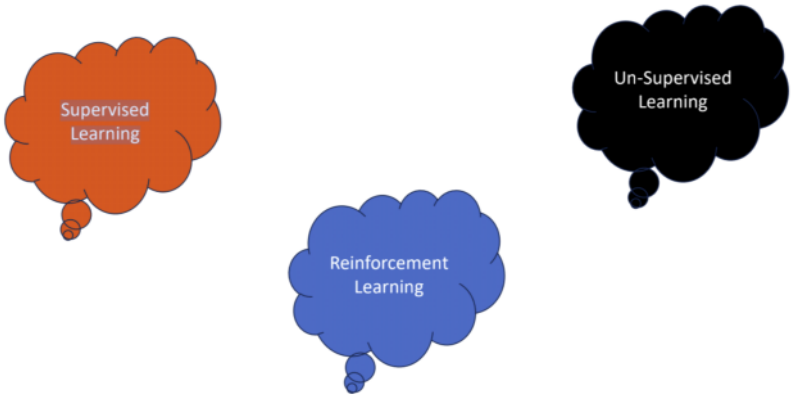
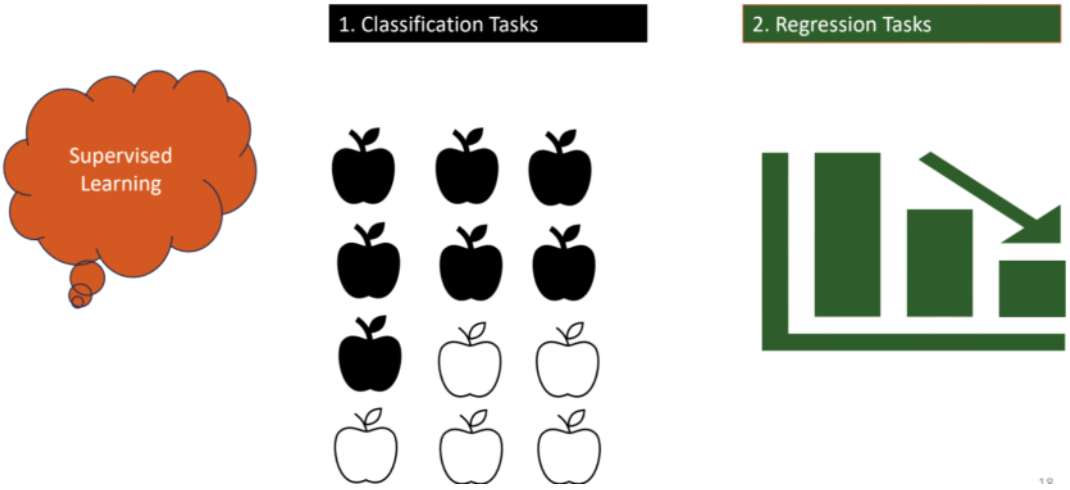
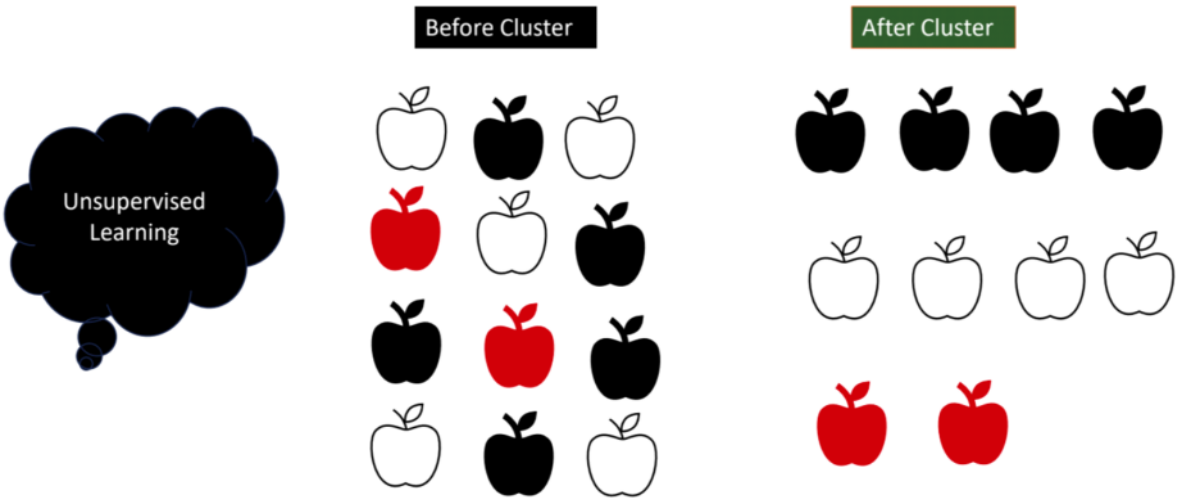
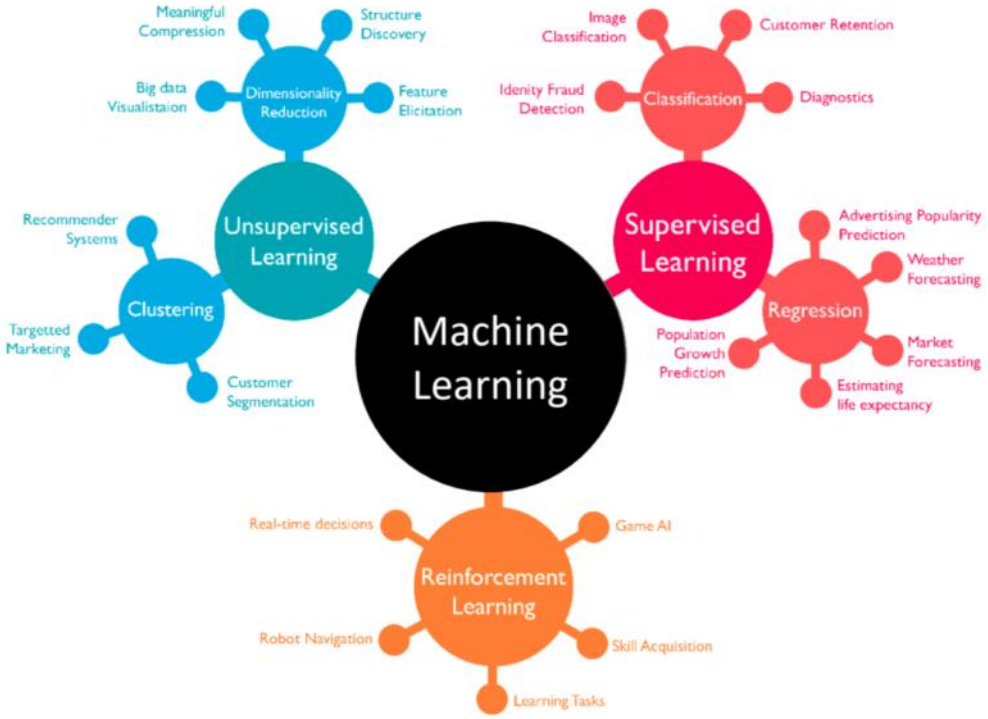
Basic ML Concepts:

February 9, 2024 10:53 AM

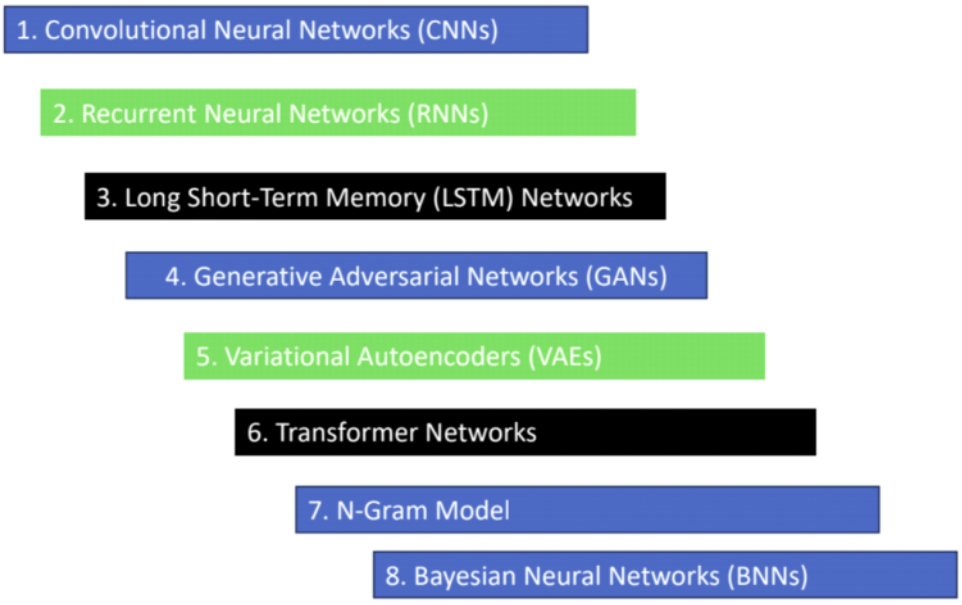
- **Types of Machine Learning:**
  - 

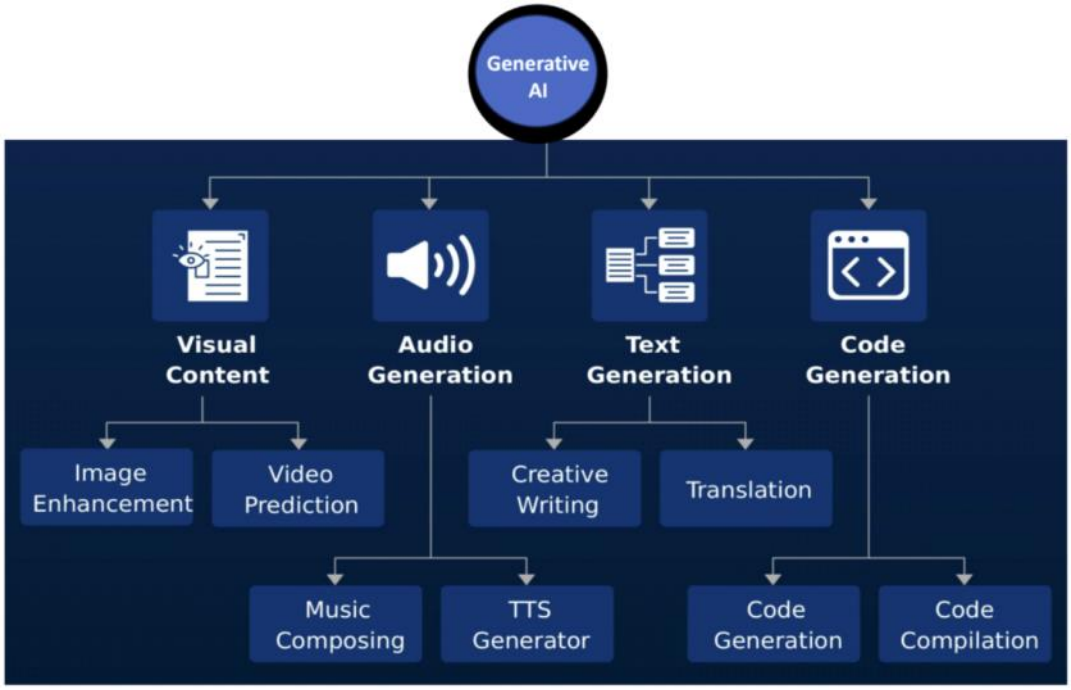
- **Statistical Machine Learning:**
  - 

  - 

- 

- **Neural Machine learning:**



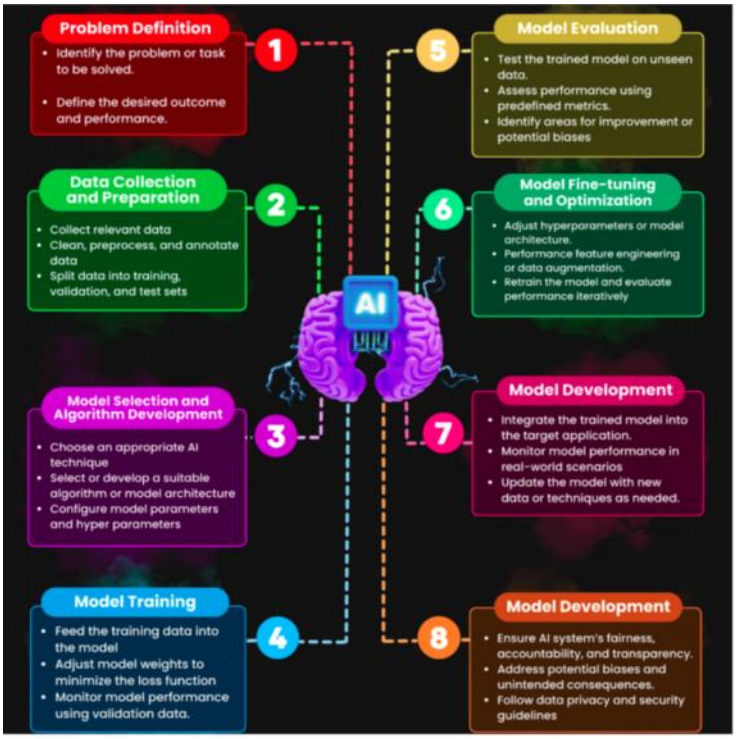
- 

- Chat-GPT DALLE / DALLE 2 Google Bard AI Midjourney AI

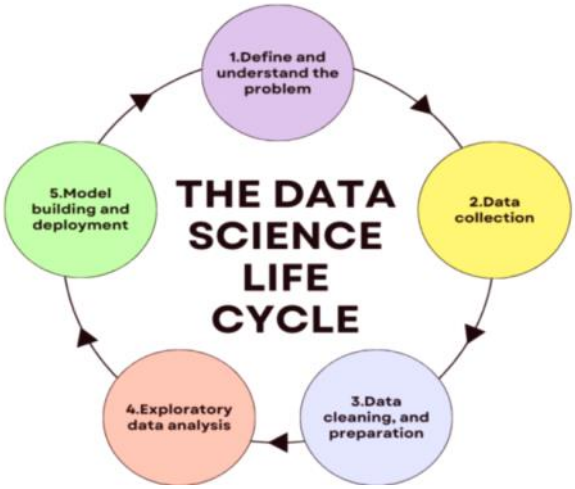
- **Generative AI:**

- Recurrent Neural Networks (RNNs)
  Long Short-Term Memory (LSTM) Networks
  Generative Adversarial Networks (GANs)
  Variational Autoencoders (VAEs)
  Transformer Networks
  N-Gram Model

- **AI Application:**



- **Data science Life cycle:**



Linear regression:

February 9, 2024 11:19 AM

- Basic Concepts:
  - **Regression in Machine Learning:**
    - Regression is a technique used to predict numerical values based on input features.
    - It models the relationship between a dependent variable (what you want to predict) and independent variables (features).
  - **Example: Predicting House Prices:**
    Imagine you're predicting house prices based on square footage. The regression model finds a line that best fits
    the data: **Price = 100 * SquareFootage + 50000.** Here, 100 is the increase in price for each square foot increase,
    and $50,000 is the starting price estimate. This model helps estimate prices for different house sizes.
  -
  - **Types of Regression:**

There are several types of regression techniques, each designed to handle different types of data and relationships between variables. Here are some common types of regression:

1. Linear Regression:
   - Simple Linear Regression: Predicting a continuous dependent variable using a single independent variable.
   - Multiple Linear Regression: Predicting a dependent variable using multiple independent variables.

2. Polynomial Regression:
   - Modeling nonlinear relationships by adding polynomial terms to the regression equation.

3. Ridge Regression:
   - Adding a penalty term to the coefficients to prevent overfitting.

4. Lasso Regression:
   - Similar to ridge regression, but with a penalty that encourages some coefficients to become exactly zero, leading to feature selection.

5. Elastic Net Regression:
   - A combination of ridge and lasso regression, providing a balance between their strengths.

6. Logistic Regression:
   - Used for binary or multinomial classification tasks, predicting the probability of an event occurring.

7. Poisson Regression:
   - Modeling count data, often used in situations where the dependent variable represents counts.

8. Time Series Regression:
   - Modeling time-dependent data, considering temporal patterns and autocorrelation.

9. Nonlinear Regression:
   - Fitting a nonlinear function to the data to capture complex relationships.

10. Quantile Regression:
    - Modeling different quantiles of the dependent variable, useful for understanding conditional distributions.

11. Support Vector Regression (SVR):
    - Utilizes support vector machines for regression tasks, particularly suited for high-dimensional spaces.

12. Bayesian Regression:
    - Incorporates Bayesian statistics to estimate parameters and uncertainties in regression models.

13. Kernel Regression:
    - Uses kernel functions to capture complex patterns in the data.

14. Generalized Linear Models (GLM):
    - Generalization of linear regression for various types of dependent variables, including binary and count data.

15. Stepwise Regression:
    - An automated method for selecting a subset of important features.

16. Piecewise Regression:
    - Fits different regression models to different segments of the data, useful for data with changing trends.

17. Principal Component Regression (PCR):
    - Combines principal component analysis (PCA) and linear regression.

○ Mathematical Concept:

- Linear regression is a fundamental supervised machine learning algorithm used for predicting a continuous numerical value (also known as the dependent variable) based on one or more input features (independent variables).
- It models the relationship between the dependent variable and the independent variables as a linear equation.
- The goal is to find the best-fitting line (or hyperplane in higher dimensions) that minimizes the difference between the observed and predicted values.
- This best-fitting line represents the linear relationship between the input features and the target variable.

General Equation for Linear Regression:

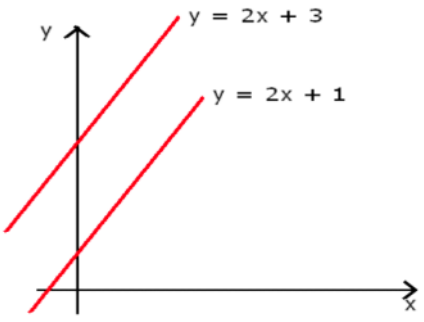
Y = M*X + C
Y = M1*X1 + M2*X2 + ...Mn*Xn + C

Here,
M = Coefficient of the input feature X
C = Intercept
X = Features
Y = Predicted Output / Label

Single $y = b_0 + b_1 * x_1$

Dependent variable (DV) Independent variables (IVs)

Multiple $y = b_0 + b_1 * x_1 + b_2 * x_2 + \dots + b_n * x_n$



X = 10, 30, 50
Y = 2*10 + 3 =23
Y = 2*30 + 3 = 63
Y = 2*50 + 3 = 103

Fig: Straight Line

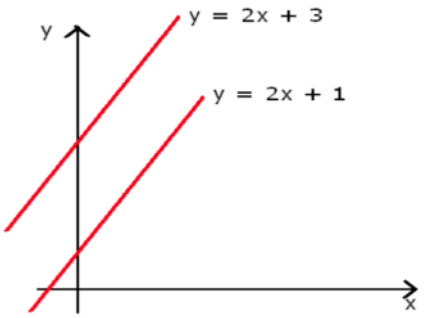
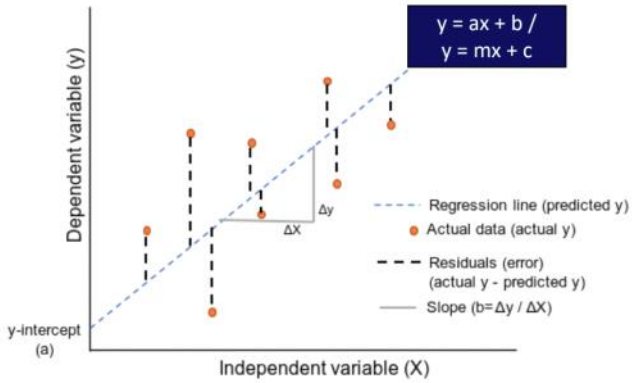
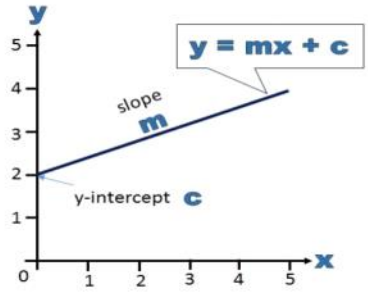


Fig: Straight Line

X = 10, 30, 50
Y = 2*10 + 3 =23
Y = 2*30 + 3 = 63
Y = 2*50 + 3 = 103

X	Actual	Predicted
10	25	23
30	60	63
50	100	103

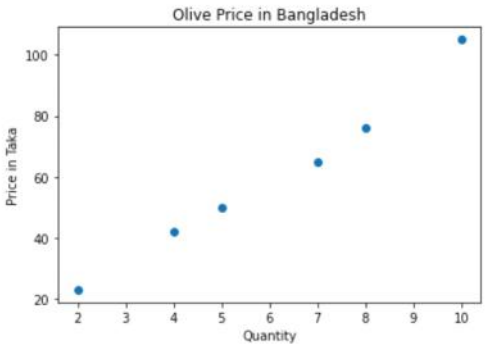




- **Residual** = Observed Value - Predicted Value

Observed	Predicted	Residual
25	23	2
60	63	3
100	103	3

	x	y
0	5	50
1	7	65
2	4	42
3	8	76
4	2	23
5	10	105



	x	y
0	5	50
1	7	65
2	4	42
3	8	76
4	2	23
5	10	105

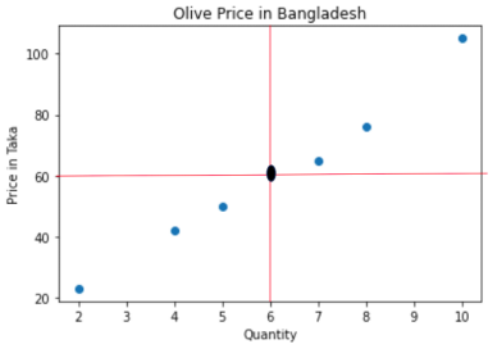
Mean Values

```
df.x.mean()
```

```
6.0
```

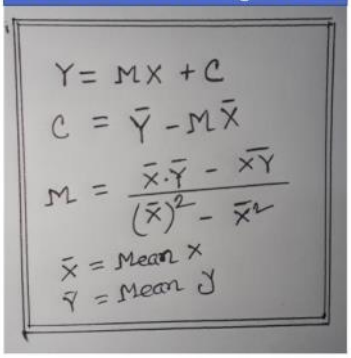
```
df.y.mean()
```

```
60.166666666666664
```



-

Formula of Linear Regression



Now Solve it

Data Set

	A	B
1	x	y
2	5	50
3	7	65
4	4	42
5	8	76
6	2	23
7	10	105
8	7	?

17

-

Calculation Table for Single Variable Linear Regression

	A	B	C	D	E	F	G	H	I
1	x	y	xy	x2	x̄	ȳ	(xy) bar	(x̄)2	(x2) bar
2	5	50	250	25					
3	7	65	455	49	Sum=36	Sum=361	Sum=2577		Sum=258
4	4	42	168	16	36/6	361/6	2577/6		258/6
5	8	76	608	64					
6	2	23	46	4	Avg=6	Avg=60.17	Avg=429.5	36	Avg=43
7	10	105	1050	100	Average	Average	Average		Average

-

	A	B	C	D	E	F	G	H	I	J
1	x	y	xy	x2	x̄	ȳ	(xy) bar	(x̄)2	(x2) bar	Final Calculations
2	5	50	250	25						M = ((6*60.17)-429.5) / (36-43)
3	7	65	455	49	Sum=36	Sum=361	Sum=2577		Sum=258	M = 9.782
4	4	42	168	16	36/6	361/6	2577/6		258/6	C = 60.17-(9.782*6)
5	8	76	608	64						C = 1.48
6	2	23	46	4	Avg=6	Avg=60.17	Avg=429.5	36	Avg=43	Y = (9.782 * X) + 1.48
7	10	105	1050	100	Average	Average	Average		Average	Predict, y = (9.782*7)+1.48
8	7	69.95		49						Ans = 69.95

- $$m = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sum(x - \bar{x})^2}$$

  $$c = \bar{y} - m\bar{x}$$

  Prediction_y = m * (input_X) + c

  Where:
  x is a data point on the independent variable (x-axis).
  y is the corresponding dependent variable (y-axis).
  $\bar{x}$ is the mean of the independent variable.
  $\bar{y}$ is the mean of the dependent variable.

Slope, m = Σ((x - x̄) * (y - ȳ)) / Σ((x - x̄)^2)
Intercept, c = ȳ - m * x̄

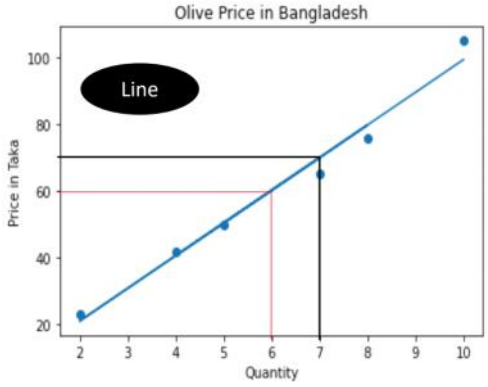
- Where:
  x is a data point on the independent variable (x-axis).
  y is the corresponding dependent variable (y-axis).
  $\bar{x}$ is the mean of the independent variable.
  $\bar{y}$ is the mean of the dependent variable.

- Data Set

	x	y
0	5	50
1	7	65
2	4	42
3	8	76
4	2	23
5	10	105

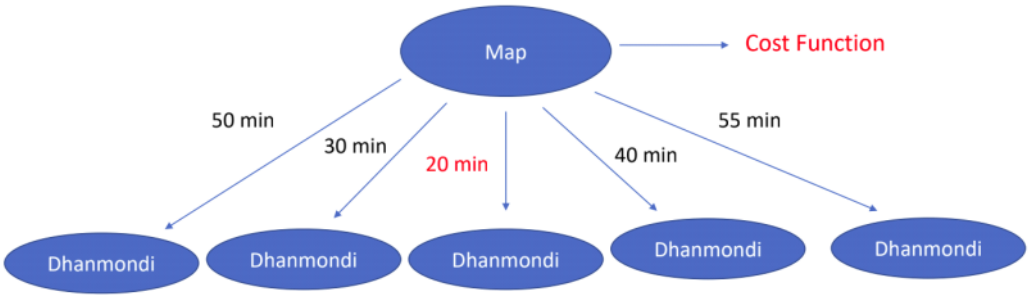
Value of M & C

```
reg.coef_
array([9.78571429])

reg.intercept_
1.4523809523809703
```



○ Cost Function:

The cost function is a function, which is associates a cost with a **decision.**



○ Residuals:

- Residuals are the differences between the observed values of the dependent variable and the predicted values generated by the regression model.
- They are calculated as (Yi - Ypred), where Yi is the observed value and Ypred is the predicted value.
- Residuals are used to assess the fit of a regression model and to diagnose potential issues like underfitting, overfitting, or the presence of outliers.

Residual = Observed Value - Predicted Value

Observed	Predicted	Residual
25	23	2
60	63	3
100	103	3

- **L1, L2 loss,** and **residuals** are related concepts, both involving differences between predicted and actual values in regression analysis.
- Loss is a measure of the differences, while residuals are the actual differences themselves.

- **However, loss specifically refers to a loss function used for optimization purposes, while residuals are used for model assessment and diagnosis.**

○ L1 Loss and L2 Loss:

L1 Loss (Absolute Loss or Mean Absolute Error):
- L1 loss is a type of loss function used to measure the difference between predicted values and actual observed values in regression problems.
- It calculates the absolute difference between the predicted value and the actual value for each data point and then averages these absolute differences.
- Mathematically, the L1 loss for the ith data point is (|Yi - Ypred|), where Yi is the observed value and Ypred is the predicted value.
- L1 loss tends to be less sensitive to outliers compared to squared loss (L2 loss).

L2 Loss (Squared Loss or Mean Squared Error):
- - L2 loss measures the squared difference between predicted values and actual observed values in regression problems.
- - It calculates the squared difference between the predicted value and the actual value for each data point and then averages these squared differences.
- - Mathematically, the L2 loss for the ith data point is (Yi - Ypred)^2, where Yi is the observed value and Ypred is the predicted value.
- - L2 loss penalizes larger errors more heavily due to the squaring operation.

Mean Absolute Error, $MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|$

Mean Squared Error, $MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$

Root Mean Squared Error, $RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$

1. Loss (or Error) for a Single Sample:
- When you calculate the difference between the actual value and the predicted value for a single data point, it's generally referred to as a "loss" or "error" for that specific data point.
- This term is used to describe the discrepancy between the prediction and the true value for a single instance.

2. Cost (or Loss) for the Entire Dataset:
- When you calculate the average or total of these losses/errors across the entire dataset, it's often referred to as the "cost" or "loss" for the dataset.
- The term "cost" or "loss" is used to describe the overall quality of the model's predictions for the entire dataset.

Gradient Descent and R Squared

February 10, 2024 4:32 AM

- **Overfitting and Underfitting:**
  - Training Performance: Low (Gorib)
    Testing Performance: Low (Gorib)

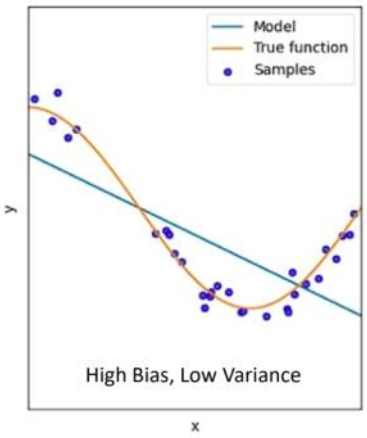


Fig 01: Underfitting

Training Performance: Good
Testing Performance: Good

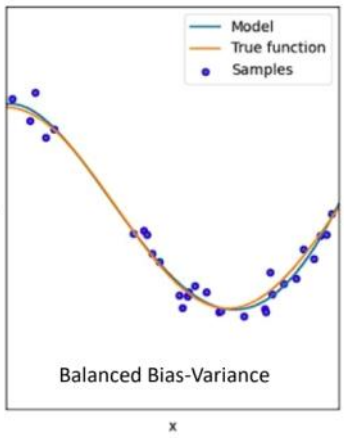


Fig 02: Best fitting

Training Performance: High (Rich)
Testing Performance: Low (Gorib)

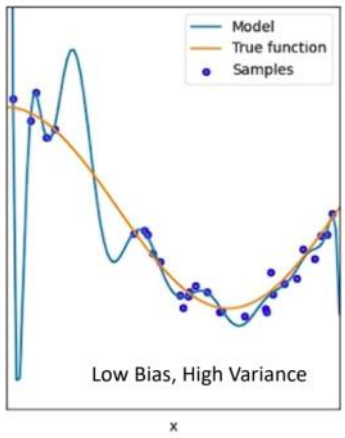


Fig 03: Overfitting

  - **Bias & Variance**

1. Bias: The difference between the average predicted value and the true value.
2. Variance: The variability of model predictions across different training datasets.

Let's denote:

- **Y**: True value being predicted.
- **f(X)**: The true relationship between features (X) and the target variable (Y).
- **ŷ**: The prediction made by a trained model.
- **E**: Expectation (average) operator.
- **D**: Data.
- **h(D)**: Model trained on dataset D.

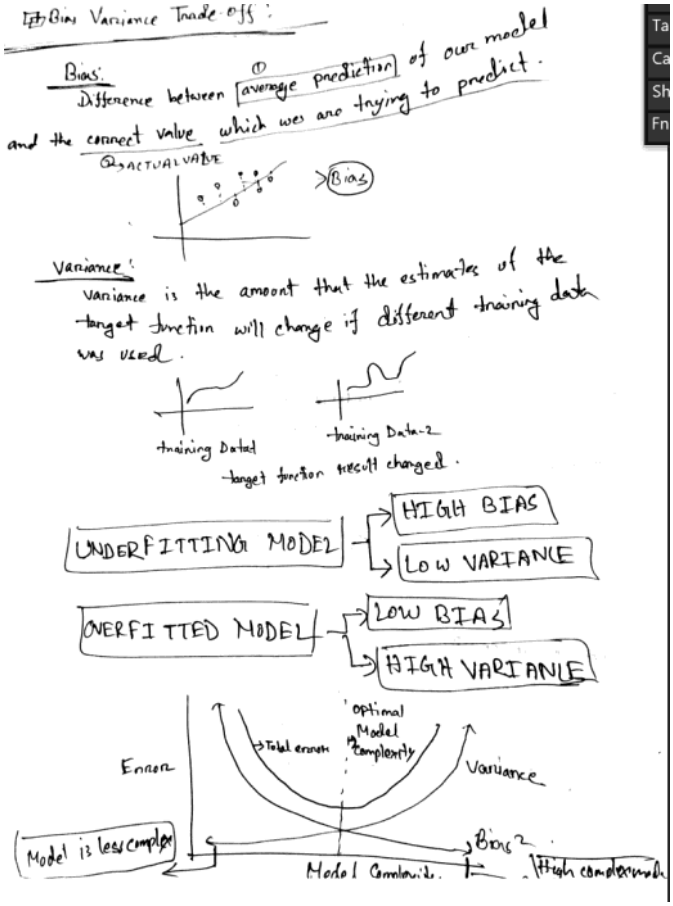
1. **Bias**:
   $Bias(\hat{f}(x)) = E[\hat{f}(x)] - f(x)$
   This equation represents the difference between the expected prediction of our model $\hat{f}(x)$ and the true value $f(x)$.
2. **Variance**:
   $Variance(\hat{f}(x)) = E[(\hat{f}(x) - E[\hat{f}(x)])^2]$
   This equation calculates the variability of predictions that our model $\hat{f}(x)$ makes across different training datasets.

Now, let's put it into a combined equation, known as the Bias-Variance Decomposition:

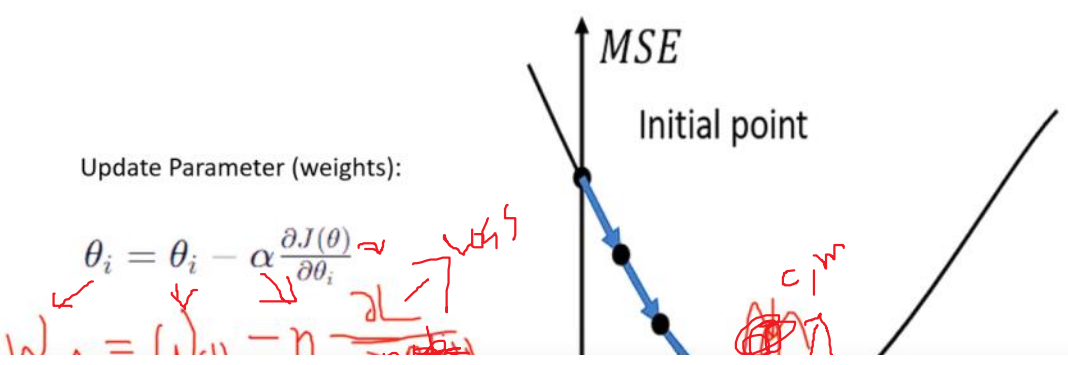
$$Error(\hat{f}(x)) = Bias(\hat{f}(x))^2 + Variance(\hat{f}(x)) + Irreducible\ Error$$

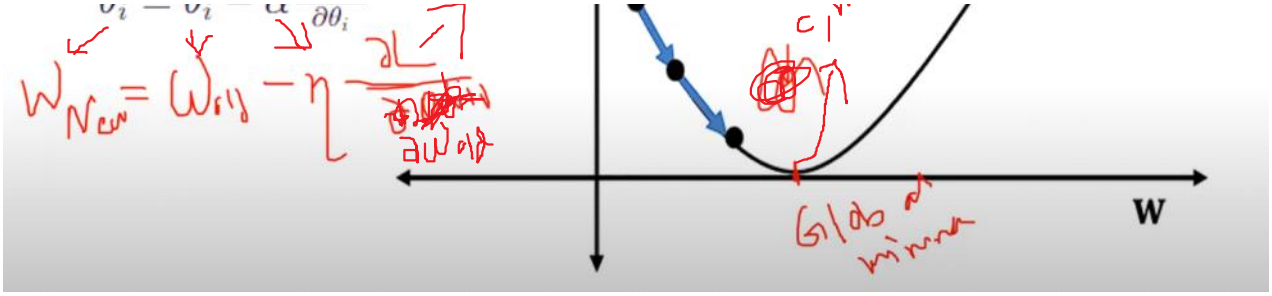
Here, the error of our model ($Error(\hat{f}(x))$) can be decomposed into the sum of squared bias, variance, and an irreducible error term that represents noise in the data that cannot be captured by the model.

```
In [1]: import numpy as np
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LinearRegression
        from mlxtend.evaluate import bias_variance_decomp

        np.random.seed(0)
        X = np.random.rand(100, 1) * 10
        y = 2 * X.squeeze() + np.random.randn(100)  # True relationship is y = 2X + noise

        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
        model = LinearRegression()
        model.fit(X_train, y_train)

        # Calculate bias and variance using the bias_variance_decomp function
        mse, bias, variance = bias_variance_decomp(model, X_train, y_train, X_test, y_test, loss='mse')

        print("MSE (Mean Squared Error):", mse)
        print("Bias^2:", bias)
        print("Variance:", variance)

MSE (Mean Squared Error): 0.9388721228182039
Bias^2: 0.9178184739745323
Variance: 0.021053648843671263
```

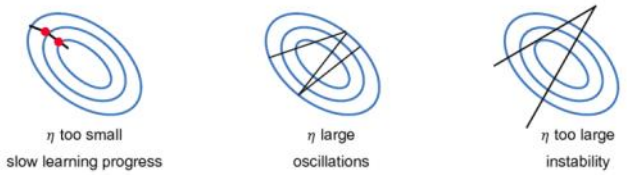


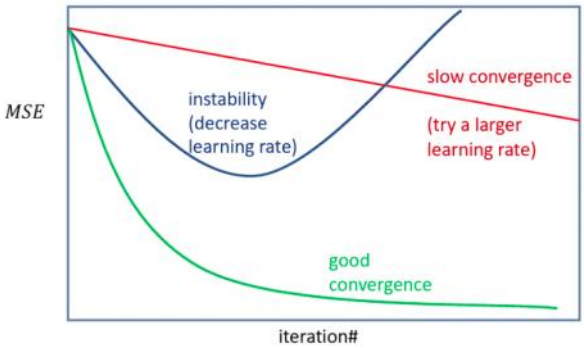
- Gradient Descent :[Mathematics]





The speed of the local/global minimum is affected from the learning rate.





The training curve could be used to monitor the effect of the learning rate value and observe the convergence. We plot the training loss against the number of training iterations.

The update rule for each parameter θ_i (where i indexes the parameters) in gradient descent can be represented as:

$$\theta_i = \theta_i - \alpha \frac{\partial J(\theta)}{\partial \theta_i}$$

Where:

- α is the learning rate, a hyperparameter that controls the size of the steps taken during optimization.
- $J(\theta)$ is the cost function.
- $\frac{\partial J(\theta)}{\partial \theta_i}$ is the partial derivative of the cost function with respect to θ_i, which gives the gradient of the cost function with respect to that parameter.

Gradient descent continues to update the parameters until convergence, where the algorithm finds parameter values that minimize the cost function.

The Mean Squared Error (MSE) is calculated as the squared difference between the actual values ($y^{(i)}$) and the predicted values ($h_\theta(x^{(i)})$):

$$\text{MSE} = \frac{1}{m}\sum_{i-1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

1. **Mean Squared Error (MSE)**:
   - MSE quantifies the average of the squared differences between predicted values and actual values across the dataset.
   - The formula is expressed as: $\text{MSE} = \frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2$, where:
     - y_i represents the actual value from the dataset.
     - $\hat{y}_i$ denotes the predicted value by the model.
     - n signifies the total number of data points.

2. **Substitution of $\hat{y}_i$ with $mx_i + c$**:
   - In linear regression, predictions ($\hat{y}_i$) can be computed using the equation of a straight line, $y = mx + c$.
   - Here, m denotes the slope of the line, x_i is the input feature value, and c represents the y-intercept.
3. **Revised MSE Formula**:
   - After replacing $\hat{y}_i$ with the linear equation $mx_i + c$, the MSE is recalculated as:
     - $\text{MSE} = \frac{1}{n}\sum_{i=0}^{n}(y_i - (mx_i + c))^2$
   - This adjusted formulation is utilized to evaluate the performance of a linear regression model, aiming to minimize the MSE by identifying optimal values for m and c through model training.

Step: 01

Gradient (m) = 0
Intercept (c) = 0
Learning Rate (L) = ~0.0001

Step: 02

Calculate the partial derivative of the Cost function with respect to m. Let the partial derivative of the Cost function with respect to m be Dm.

$$D_m = \frac{\partial(Cost\ Function)}{\partial m} = \frac{\partial}{\partial m}\left(\frac{1}{n}\sum_{i=0}^{n}(y_i - y_{i\,pred})^2\right)$$

$$= \frac{1}{n}\frac{\partial}{\partial m}\left(\sum_{i=0}^{n}(y_i - (mx_i + c))^2\right)$$

$$= \frac{1}{n}\frac{\partial}{\partial m}\left(\sum_{i=0}^{n}(y_i^2 + m^2x_i^2 + c^2 + 2mx_ic - 2y_imx_i - 2y_ic)\right)$$

$$= \frac{-2}{n}\sum_{i=0}^{n}x_i(y_i - (mx_i + c))$$

$$= \frac{-2}{n}\sum_{i=0}^{n}x_i(y_i - y_{i\,pred})$$

Step: 03

Similarly, let's find the partial derivative with respect to c. Let the partial derivative of the Cost function with respect to c be Dc.

$$D_c = \frac{\partial(Cost\ Function)}{\partial c} = \frac{\partial}{\partial c}\left(\frac{1}{n}\sum_{i=0}^{n}(y_i - y_{i\,pred})^2\right)$$

$$= \frac{1}{n}\frac{\partial}{\partial c}\left(\sum_{i=0}^{n}(y_i - (mx_i + c))^2\right)$$

$$= \frac{1}{n}\frac{\partial}{\partial c}\left(\sum_{i=0}^{n}(y_i^2 + m^2x_i^2 + c^2 + 2mx_ic - 2y_imx_i - 2y_ic)\right)$$

$$= \frac{-2}{n}\sum_{i=0}^{n}(y_i - (mx_i + c))$$

$$\frac{-2}{n}\sum_{i=0}^{n}(y_i - y_{i\,pred})$$

Step: 04 Update the value of the gradient and intercept.

$$m = m - L \times D_m$$

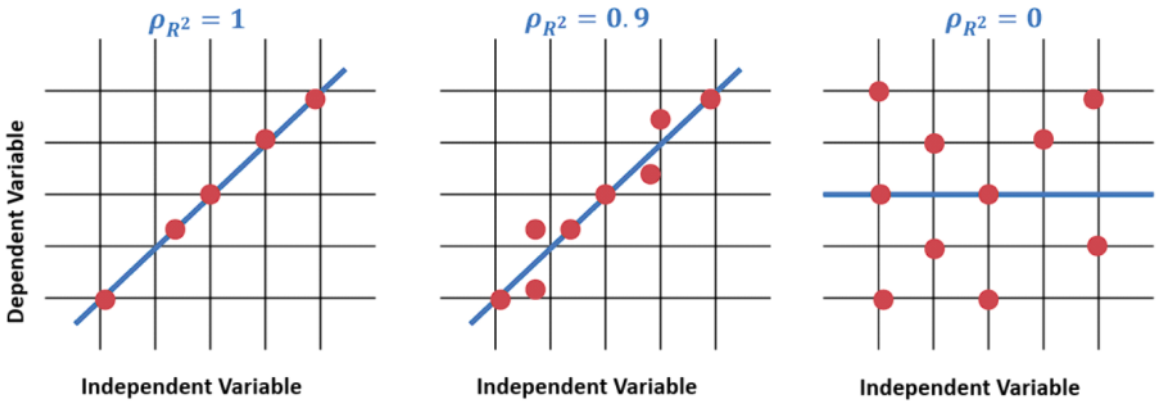
$$c = c - L \times D_c$$

Repeat the steps!
1000 times

- **R Squared Value / Model Accuracy**

- $$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

  - Residual sum of squared errors of our regression model (SSres)
  - Total sum of squared errors (SStot)

- 

- Way no: 01

```
reg.score(xtest, ytest)
```

- Way no: 02

```
y_pred = reg.predict(xtest) #Predicted y
from sklearn.metrics import r2_score
Score = r2_score(ytest, y_pred)
```

- **Measures for Classification: MCE & ACC**

1. **Misclassification Error (MCE)**:
   - Misclassification Error measures the proportion of incorrectly classified instances in a classification problem.
   - It is calculated as the total number of misclassified instances divided by the total number of instances.
   - Mathematically, MCE can be expressed as:
     $$MCE = \frac{\text{Number of misclassified instances}}{\text{Total number of instances}}$$
2. **Accuracy (ACC)**:
   - Accuracy measures the proportion of correctly classified instances in a classification problem.
   - It is calculated as the total number of correctly classified instances divided by the total number of instances.
   - Mathematically, Accuracy can be expressed as:
     $$ACC = \frac{\text{Number of correctly classified instances}}{\text{Total number of instances}}$$