CMR INSTITUTE OF TECHNOLOGY

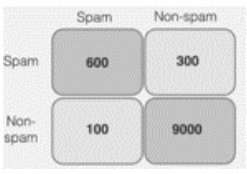
USN | | | | | | | | | | |

Internal Assessment Test I –June 2025

Sub:	Machine learning and Data analytics using Python						Sub Code:	MMC201
Date:	30/06/25	Duration:	90 min's	Max Marks:	50	Sem:	II	Branch:	MCA

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

	PART I	MARKS	OBE	
			CO	RBT
1	What is Machine Learning? Explain Reinforcement Learning with example.**OR**	[10]	CO1	L2
2	Discuss MatPlotLib Library with examples.	[10]	CO1	L2
	PART II			
3	What is the purpose of scaling the dataset? Explain MinMax Scaler and Standard Scaler.**OR**	[10]	CO2	L3
4	Apply appropriate evaluation metrics (such as MAE, MSE, RMSE) to assess the performance of a given regression model, and interpret the results.	[10]	CO3	L3

	PART III			
5	Discuss the working mechanism of the KNN algorithm and analyze how the choice of 'K' and distance metric influences its performance with suitable examples. **OR**	[10]	CO1	L4
6	Analyze how the Random Forest Classifier improves classification accuracy by combining multiple decision trees. Explain the role of bagging and feature randomness using an example.	[10]	CO1	L4
	PART IV			
7	Analyze the structure and functioning of decision trees by explaining the ID3 algorithm. Illustrate how information gain is used to build the tree with an example. **OR**	[10]	CO1	L4
8	Explain Confusion Matrix with example. What is the significance of ROC-AUC? 	[10]	CO3	L3

9	**PARTV** Explain different types of Cross Validation techniques. Apply them on a dataset and interpret the results. **OR**	[10]	CO1	L3
10	Explain underfitting and overfitting with respect to Bias and Variance.	[10]	CO1	L2

Internal Assessment Test I –June 2025

Machine learning and Data analytics using Python						**Sub Code:**		**MMC201**
Date: 30/06/25	**Duration:**	**90 mins**	**Max Marks:**	**50**	**Sem:**	**II**	**Branch:**	**MCA**

1. **What is Machine Learning? Explain Reinforcement Learning with example.**
   Machine learning is a subset of Artificial Intelligence (AI). It is focused on teaching computers to learn from data and to improve with experience, instead of being explicitly programmed to do so. In machine learning, algorithms are trained to find patterns and correlations in large data sets and to make the best decisions and predictions based on that analysis.

Reinforcement Learning

In some applications, the output of the system is a sequence of actions. In such a case, a single action is not important; what is important is the policy that is the sequence of correct actions to reach the goal. There is no such thing as the best action in any intermediate state; an action is good if it is part of a good policy. In such a case, the machine learning program should be able to assess the goodness of policies and learn from past good action sequences to be able to generate a policy. Such learning methods are called reinforcement learning algorithms. A good example is *game playing* where a single move by itself is not that important; it is the sequence of right moves that is good. A move is good if it is part of a good game playing policy. Game playing is an important research area in both artificial intelligence and machine learning. This is because games are easy to describe and at the same time, they are quite difficult to play well. A game like chess has a small number of rules but it is very complex because of the large number of possible moves at each state and the large number of moves that a game contains.

Reinforcement learning (RL) is a type of machine learning where an agent learns to make decisions by interacting with an environment and receiving rewards or penalties for its actions.

Agent: A program or system that interacts with the environment and makes decisions.

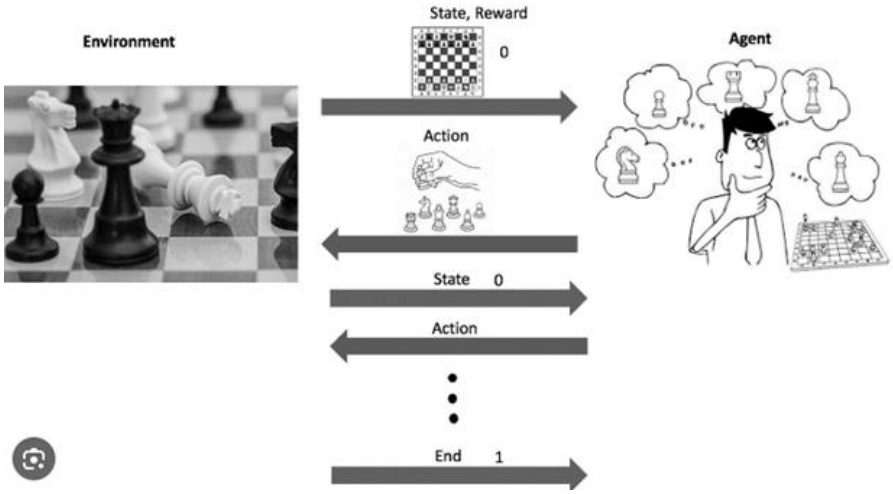
Environment: The world in which the agent operates, and which provides feedback to the agent.

Actions: The decisions the agent can make in the environment.

State: The current situation or condition of the environment that the agent is in.

Reward: A feedback provided by the environment to indicate whether an action was good or bad.

Policy: The strategy the agent uses to choose actions based on the current state.



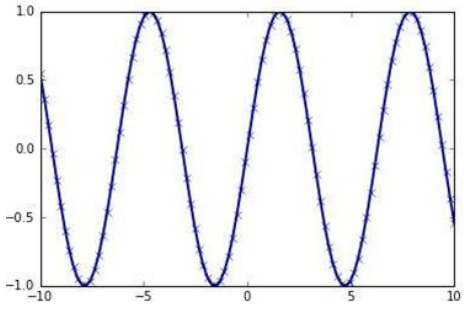
2. **Discuss MatPlotLib Library with examples.**

Matplotlib

Matplotlib is a powerful and versatile open-source plotting library for Python, designed to help users visualize data in a variety of formats. Matplotlib is the primary scientific plotting library in Python. It provides functions for making publication-quality visualizations such as line charts, histograms, scatter plots, and so on. Visualizing your data and different aspects of your analysis can give you important insights, and we will be using matplotlib for all our visualizations.

For example, this code produces the plot in Figure 1-1:

```
import matplotlib.pyplot as plt
# Generate a sequence of numbers from -10 to 10 with 100 steps in between
x = np.linspace(-10, 10, 100)
# Create a second array using sine
y = np.sin(x)
# The plot function makes a line chart of one array against another
plt.plot(x, y, marker="x")
```

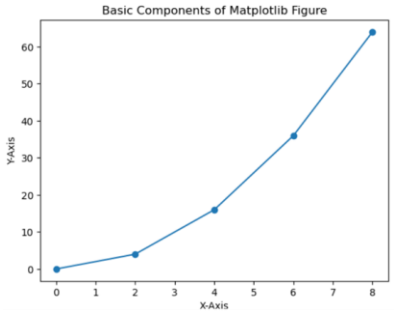


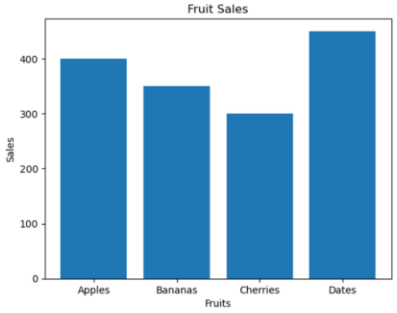
```python
import matplotlib.pyplot as plt
x = [0, 2, 4, 6, 8]
y = [0, 4, 16, 36, 64]
fig, ax = plt.subplots()
ax.plot(x, y,marker='o')
ax.set_title("Basic Components of Matplotlib Figure")
ax.set_xlabel("X-Axis")
ax.set_ylabel("Y-Axis")
plt.show()
```

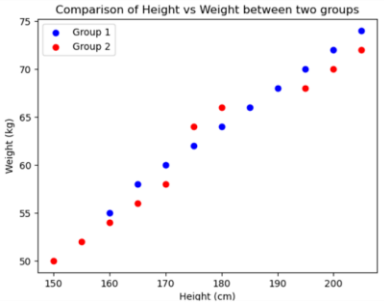
```python
import matplotlib.pyplot as plt
x = [0, 1, 2, 3, 4]
y = [0, 1, 4, 9, 16]
plt.plot(x, y)
plt.show()
```

```python
import matplotlib.pyplot as plt
import numpy as np
fruits = ['Apples', 'Bananas', 'Cherries', 'Dates']
sales = [400, 350, 300, 450]
plt.bar(fruits, sales)
plt.title('Fruit Sales')
plt.xlabel('Fruits')
plt.ylabel('Sales')
plt.show()
```

```python
import matplotlib.pyplot as plt
import numpy as np
x1 = np.array([160, 165, 170, 175, 180, 185, 190, 195, 200, 205])
y1 = np.array([55, 58, 60, 62, 64, 66, 68, 70, 72, 74])

x2 = np.array([150, 155, 160, 165, 170, 175, 180, 195, 200, 205])
y2 = np.array([50, 52, 54, 56, 58, 64, 66, 68, 70, 72])

plt.scatter(x1, y1, color='blue', label='Group 1')
plt.scatter(x2, y2, color='red', label='Group 2')

plt.xlabel('Height (cm)')
plt.ylabel('Weight (kg)')
plt.title('Comparison of Height vs Weight between two groups')

plt.legend()
plt.show()
```







3. What is the purpose of scaling the dataset? Explain MinMax Scaler and Standard Scaler.

Feature scaling marks the end of the data preprocessing in Machine Learning. It is a method to standardize the independent variables of a dataset within a specific range. In other words, feature scaling limits the range of variables so that you can compare them on common grounds.

Another reason why feature scaling is applied is that few algorithms like gradient descent converge much faster with feature scaling than without it.

MinMax Scaler

MinMax Scaler shrinks the data within the given range, usually of 0 to 1. It transforms data by scaling features to a given range. It scales the values to a specific value range without changing the shape of the original distribution.

```python
#6.a Standarzation of the data using MinMax Scaler
from sklearn.preprocessing import MinMaxScaler
mm = MinMaxScaler()
X[:,3:4] = mm.fit_transform(X[:, 3:4])
```

Normalization

$$x = \frac{x - min(x)}{max(x) - min(x)}$$

Example:

```
from sklearn.preprocessing import MinMaxScaler

mm = MinMaxScaler()
X_train[:, 3:] = mm.fit_transform(X_train[:, 3:])
X_test[:, 3:] = mm.transform(X_test[:, 3:])
print(X_train[:, 3:])
```

```
[[0.5120772946859904 0.11428571428571432]
[0.5652173913043479 0.45079365079365075]
[0.7391304347826089 0.6857142857142855]
[0.4782608695652175 0.37142857142857144]
[0.0 0.0]
[0.9130434782608696 0.8857142857142857]
[1.0 1.0]
[0.34782608695652173 0.2857142857142856]]
```

In [31]:

```
print(X_test[:, 3:])
```

```
[[0.1304347826086958 0.17142857142857149]
[0.43478260869565233 0.5428571428571427]]
```

Standardization:

Standard Scaler

StandardScaler follows Standard Normal Distribution (SND). Therefore, it makes mean = 0 and scales the data to unit variance.

Standardization

$$x = \frac{x - mean(x)}{standard\ deviation(x)}$$

```
#6.b Standarzation of the data using Standard Scaler
from sklearn.preprocessing import StandardScaler
sta = StandardScaler()
X[:,4:] = sta.fit_transform(X[:,4:])
```

Example:

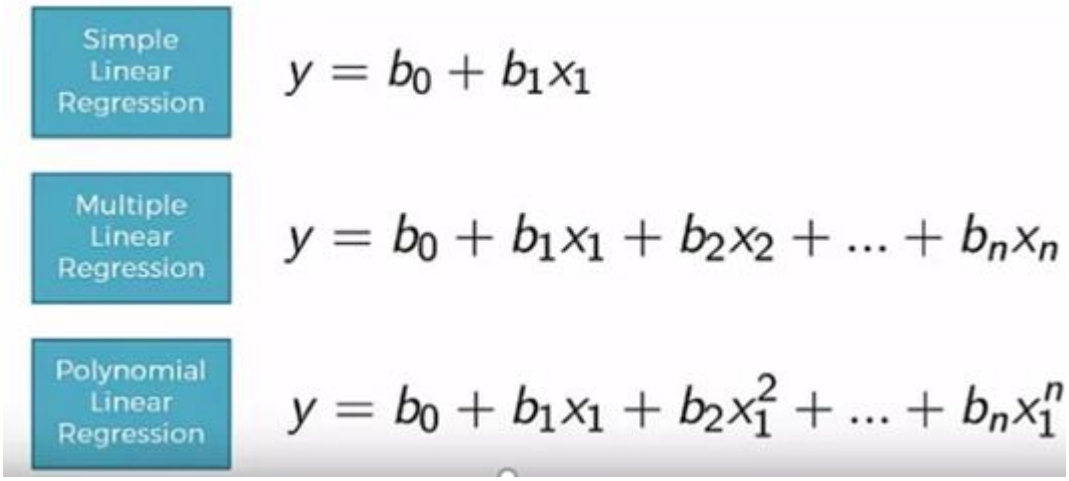
In [32]:

```
from sklearn.preprocessing import StandardScaler
sta = StandardScaler()
X_train[:, 3:] = sta.fit_transform(X_train[:, 3:])
X_test[:, 3:] = sta.transform(X_test[:, 3:])
```

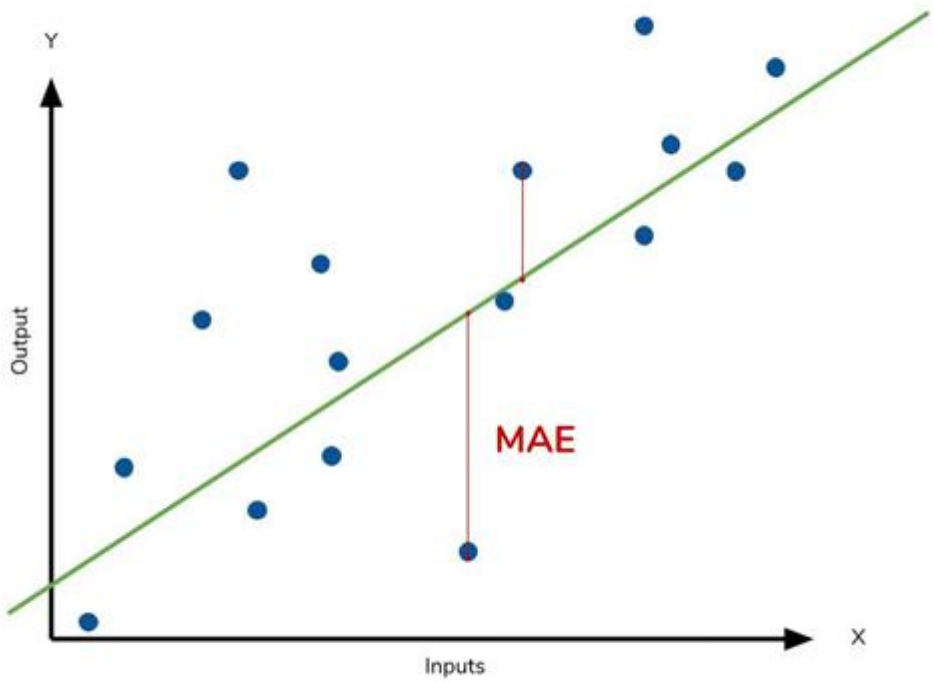
In [33]:

```
print(X_train[:, 3:])
[[-0.19159184384578537 -1.0781259408412425]
[-0.014117293757057581 -0.07013167641635436]
[0.5667085065333245 0.6335624327104541]
[-0.3045301939022482 -0.3078661727429788]
[-1.9018011447007983 -1.4204636155515822]
[1.1475343068237058 1.2326533634535486]
[1.4379472069688963 1.5749910381638883]
[-0.740149544120035 -0.5646194287757338]]
print(X_test[:, 3:])
[[-1.4661817944830116 -0.9069571034860727]
[-0.4497366439748436 0.20564033932252992]]
```

4. **Apply appropriate evaluation metrics (such as MAE, MSE, RMSE) to assess the performance of a given regression model, and interpret the results.**
   Regression is a method of estimating a relationship from given data to depict nature of data set. This relationship can then be used for the forecasting future values or for computing if there exists a relationship among the various variables.



Model Evaluation Metrics:



Mean Absolute Error(MAE):

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

To better understand, let's take an example you have input data and output data and use Linear Regression, which draws a best-fit line.

Now you have to find the MAE of your model which is basically a mistake made by the model known as an error. Now find the difference between the actual value and predicted value that is an absolute error but we have to find the mean absolute of the complete dataset.

so, sum all the errors and divide them by a total number of observations And this is MAE. And we aim to get a minimum MAE because this is a loss.

Example:

Craft Item	Actual Price ($)	Predicted Price ($)
Necklace	25	28
Bracelet	15	14
Earrings	20	22
Ring	30	29
Brooch	40	38

Imagine you are a data scientist working for a startup that sells handmade crafts online. The company recently implemented a new pricing algorithm to predict the price of crafts based on various features like size, material, and complexity. You want to evaluate the performance of this new algorithm.

Here's a set of actual prices of crafts and the prices predicted by the algorithm:

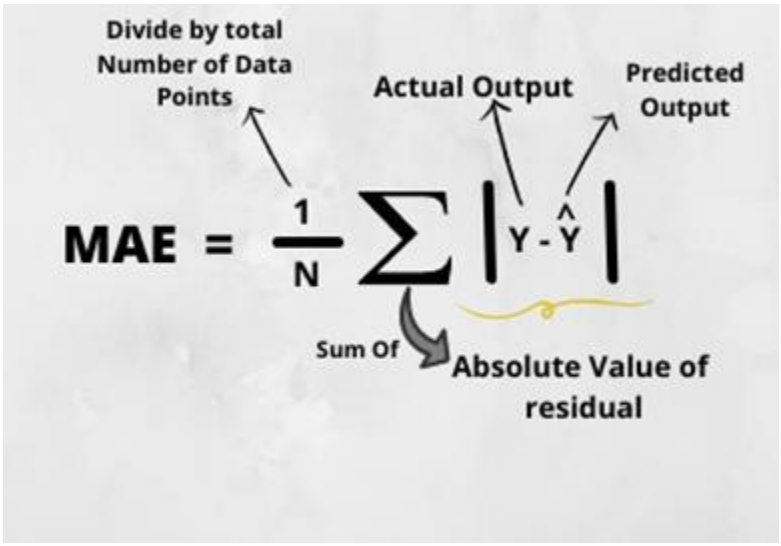
Solution:

Here, n = 5,

MAE = 1/5 * (|25-28| + |15-14| + |20-22| + |30-29| + |40-38|) = 1/5 * (3+1+2+1+2) = 1/5 * (9) = 1.8

Using Scikit Learn

Scikit-learn is one of the most common and popular libraries in machine learning. It provides a built-in function to calculate the MAE.



Output

1.8

Advantages of MAE

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

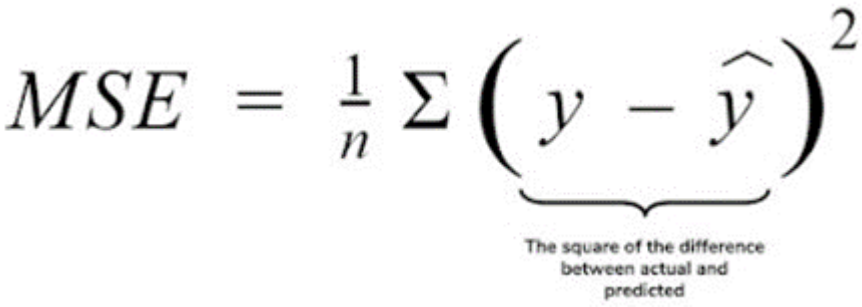
Disadvantages of MAE

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

```
from sklearn.metrics import mean_absolute_error
actual = [25,15,20,30,40]
predicted = [28,14,22,29,38]
mae=mean_absolute_error(actual,predicted)
print(mae)
```

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. So, above we are finding the absolute difference and here we are finding the squared difference.



What actually the MSE represents? It represents the squared distance between actual and predicted values. we perform squared to avoid the cancellation of negative terms and it is the benefit of MSE.

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.

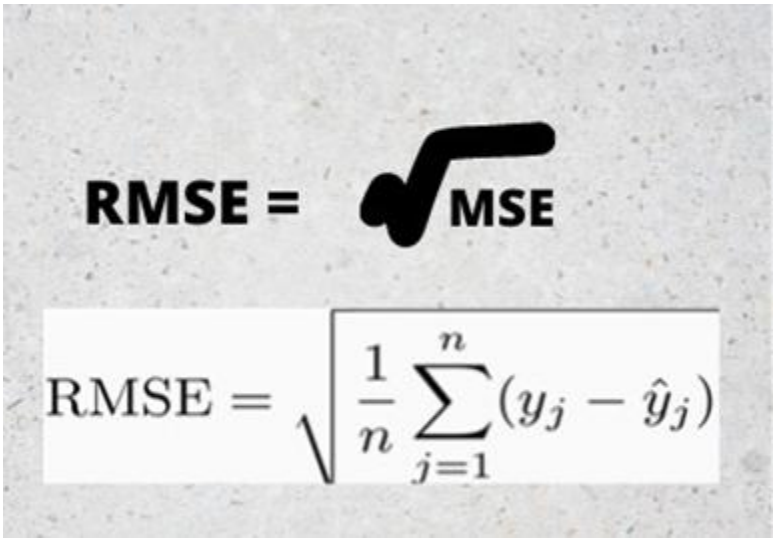
If you have outliers in the dataset then it penalizes the outliers most and the calculated MSE is bigger. So, in short, It is not Robust to outliers which were an advantage in MAE.

Output

3.8

```
from sklearn.metrics import mean_squared_error
actual = [25,15,20,30,40]
predicted = [28,14,22,29,38]
mse=mean_squared_error(actual,predicted)
print(mse)
```

Root Mean Squared Error(RMSE)



As RMSE is clear by the name itself, that it is a simple square root of mean squared error.

```
from sklearn.metrics import root_mean_squared_error
actual = [25,15,20,30,40]
predicted = [28,14,22,29,38]
rmse=root_mean_squared_error(actual,predicted)
print(rmse)
```

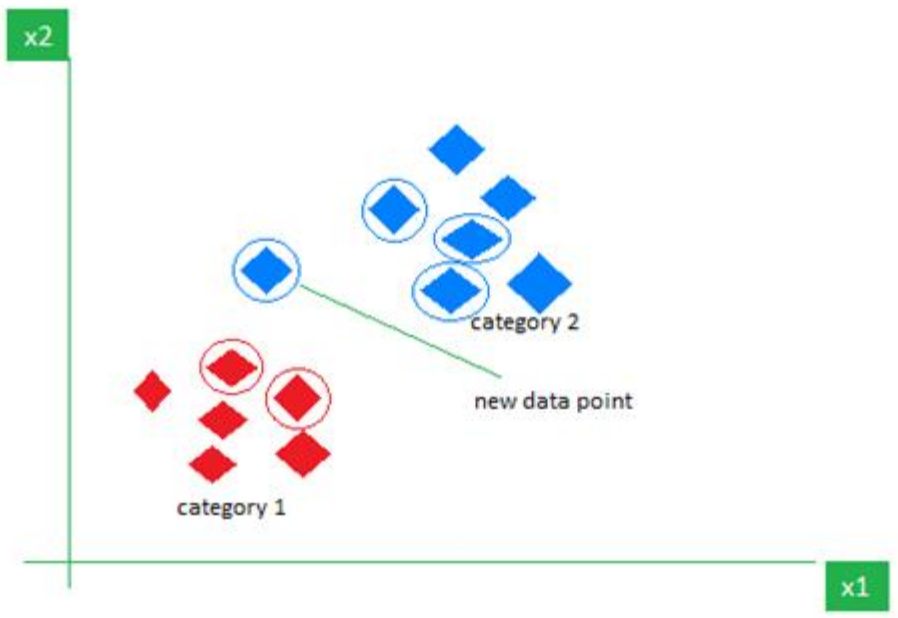
Output:
1.9493588689617927

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.
Most of the time people use RMSE as an evaluation metric and mostly when you are working with deep learning techniques the most preferred metric is RMSE.

5. **Discuss the working mechanism of the KNN algorithm and analyze how the choice of 'K' and distance metric influences its performance with suitable examples.**

K-Nearest Neighbour

The KNN algorithm is an instance-based method and is called a lazy learner. Lazy because it doesn't explicitly learn from the training data. It just memorizes the training instances which are used as "knowledge" during prediction.
As an example, consider the following table of data points containing two features:



In the **k-Nearest Neighbours (k-NN)** algorithm **k** is just a number that tells the algorithm how many nearby points (neighbours) to look at when it makes a decision.

Example:

Imagine you're deciding which fruit it is based on its shape and size. You compare it to fruits you already know.

If **k = 3**, the algorithm looks at the 3 closest fruits to the new one.

If 2 of those 3 fruits are apples and 1 is a banana, the algorithm says the new fruit is an apple because most of its neighbours are apples.

Choosing the value of k for KNN Algorithm:

The value of k is critical in KNN as it determines the number of neighbors to consider when making predictions. Selecting the optimal value of k depends on the characteristics of the input data.

If the dataset has significant outliers or noise a higher k can help smooth out the predictions and reduce the influence of noisy data. However choosing very high value can lead to underfitting where the model becomes too simplistic.

Distance Metrics Used in KNN Algorithm:

KNN uses distance metrics to identify nearest neighbour, these neighbours are used for classification and regression task. To identify nearest neighbour we use below distance metrics:

1. Euclidean Distance

Euclidean distance is defined as the straight-line distance between two points in a plane or space. You can think of it like the shortest path you would walk if you were to go directly from one point to another.

$$\text{distance}(x, X_i) = \sqrt{\sum_{j=1}^{d} (x_j - X_{i_j})^2]}$$

2. Manhattan Distance

This is the total distance you would travel if you could only move along horizontal and vertical lines (like a grid or city streets). It's also called "taxicab distance" because a taxi can only drive along the grid-like streets of a city.

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

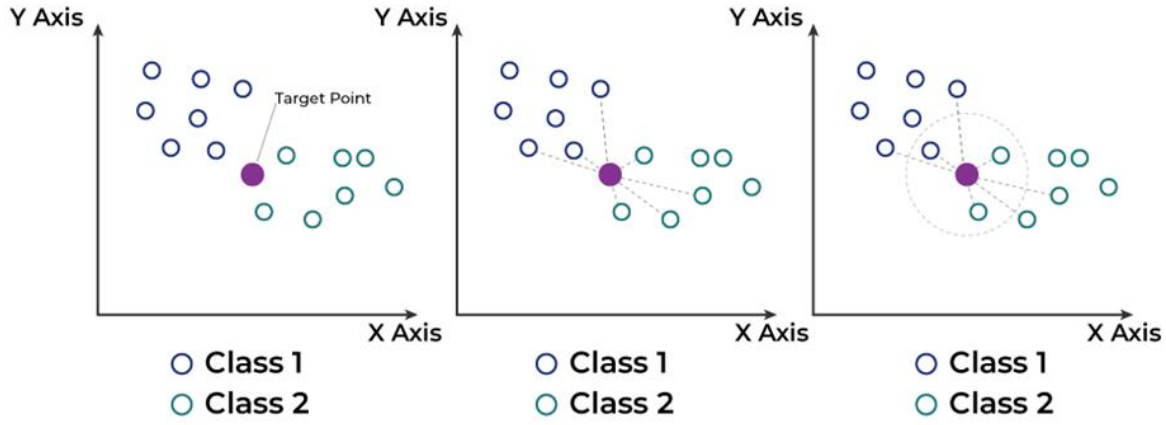
3. Minkowski Distance

Minkowski distance is like a family of distances, which includes both **Euclidean** and **Manhattan distances** as special cases.

$$d(x, y) = \left(\sum_{i=1}^{n} (x_i - y_i)^p\right)^{\frac{1}{p}}$$

From the formula above we can say that when p = 2 then it is the same as the formula for the Euclidean distance and when p = 1 then we obtain the formula for the Manhattan distance.

So, you can think of Minkowski as a flexible distance formula that can look like either Manhattan or Euclidean distance depending on the value of p.



Working of KNN algorithm

The K-Nearest Neighbors (KNN) algorithm operates on the principle of similarity where it predicts the label or value of a new data point by considering the labels or values of its K nearest neighbors in the training dataset.

Step 1: Selecting the optimal value of K

K represents the number of nearest neighbors that needs to be considered while making prediction.

Step 2: Calculating distance

To measure the similarity between target and training data points Euclidean distance is used. Distance is calculated between data points in the dataset and target point.

Step 3: Finding Nearest Neighbors

The k data points with the smallest distances to the target point are nearest neighbors.

Step 4: Voting for Classification or Taking Average for Regression

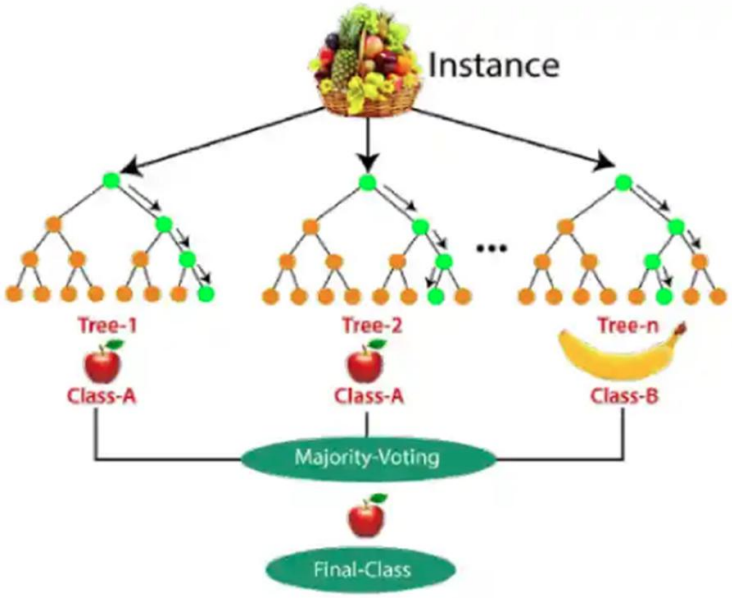
When you want to classify a data point into a category (like spam or not spam), the K-NN algorithm looks at the **K closest points** in the dataset. These closest points are called neighbors. The algorithm then looks at which category the neighbors belong to and picks the one that appears the most. This is called **majority voting**.

In regression, the algorithm still looks for the **K closest points**. But instead of voting for a class in classification, it takes the **average** of the values of those K neighbors. This average is the predicted value for the new point for the algorithm.

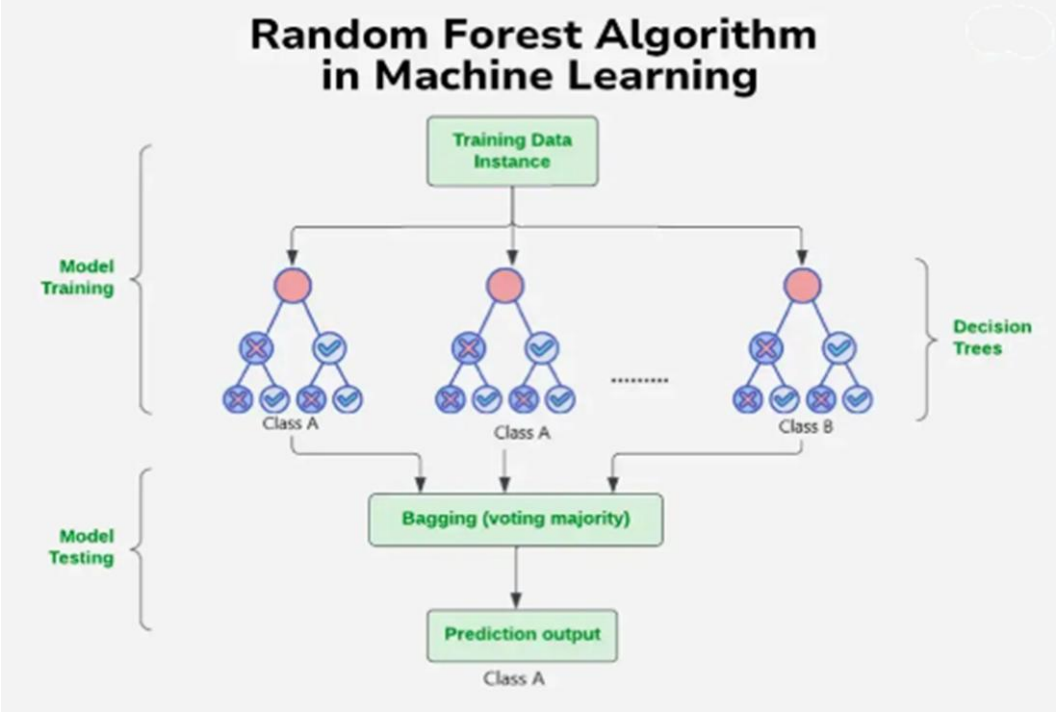
```
X_new = np.array([[5, 2.9, 1, 0.2]])
print("X_new.shape: {}".format(X_new.shape))
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
prediction = knn.predict(X_new)
print("Prediction: {}".format(prediction))
print("Predicted target name: {}".format(iris_dataset['target_names'][prediction]))
```

6. **Analyze how the Random Forest Classifier improves classification accuracy by combining multiple decision trees. Explain the role of bagging and feature randomness using an example.**

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.
It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.



Eg. From the first decision tree algorithm, a fruit is classified as apple, same fruit is classified by the second decision tree also as apple, but he third decision tree is classifying the same fruit as banana. Then the majority wins-fruit is classified as APPLE.



Steps:

· Process starts with a dataset with rows and their corresponding class labels (columns).

· Then - Multiple Decision Trees are created from the training data. Each tree is trained on a random subset of the data (with replacement) and a random subset of features. This process is known as bagging or bootstrap aggregating.

· Each Decision Tree in the ensemble learns to make predictions independently.

· When presented with a new, unseen instance, each Decision Tree in the ensemble makes a prediction.

· The final prediction is made by combining the predictions of all the Decision Trees. This is typically done through a majority vote (for classification) or averaging (for regression).

Bagging:

Multiple Decision Trees are created from the training data. Each tree is trained on a random subset of the data (with replacement) and a random subset of features. This process is known as bagging or bootstrap aggregating. Each Decision Tree in the ensemble learns to make predictions independently.

When presented with a new, unseen instance, each Decision Tree in the ensemble makes a prediction.

The final prediction is made by combining the predictions of all the Decision Trees.

7. **Analyze the structure and functioning of decision trees by explaining the ID3 algorithm. Illustrate how information gain is used to build the tree with an example.**

A decision tree is a supervised learning algorithm that can be used for both classification and regression tasks. It works by recursively partitioning the data into subsets based on feature values, making decisions at each node to maximize a specific criterion (e.g., information gain or Gini index).

Key Components:

Root Node: The top node in the tree that represents the best feature to split the data.

Internal Nodes: Represent the features used for splitting the data based on specific decision rules.

Leaf Nodes: Terminal nodes that represent the predicted outcome (class label or numerical value).

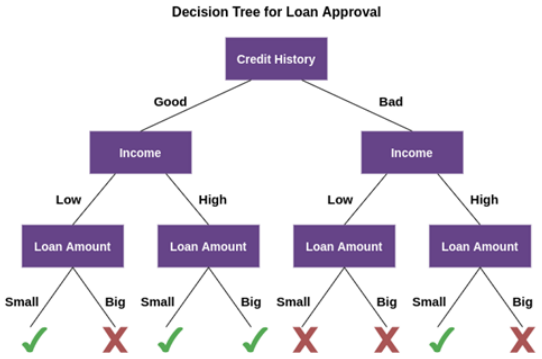
Branches: Connections between nodes representing the possible values of the features.

The process of creating a decision tree involves:

Selecting the Best Attribute: Using a metric like Gini impurity, entropy, or information gain, the best attribute to split the data is selected.

Splitting the Dataset: The dataset is split into subsets based on the selected attribute.

Repeating the Process: The process is repeated recursively for each subset, creating a new internal node or leaf node until a stopping criterion is met (e.g., all instances in a node belong to the same class or a predefined depth is reached).



A leaf node in a decision tree is the terminal node at the bottom of the tree, where no further splits are made. Leaf nodes represent the final output or prediction of the decision tree. Once a data point reaches a leaf node, a decision or prediction is made based on the majority class (for classification) or the average value (for regression) of the data points that reach that leaf.

To check mathematically if any split is pure split or not we use entropy or gini impurity. Information Gain helps us to determine which features need to be selected

ID3 Algorithm:

ID3 learns decision trees by constructing them top-down.

Entropy:

Entropy is a concept borrowed from information theory and is commonly used as a measure of uncertainty or disorder in a set of data. In the context of decision trees, entropy is often employed as a criterion to decide how to split data points at each node, aiming to create subsets that are more homogeneous with respect to the target variable.

Entropy, characterizes the (im)purity of an arbitrary collection of examples.

Given a collection S, containing positive and negative examples of some target

where p+, is the proportion of positive examples in S and p-, is the proportion of negative examples in S.

S is a collection of training examples,

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

- p+ the proportion of positive examples in S $\left(\frac{P}{P+N}\right)$
- p– the proportion of negative examples in S $\left(\frac{N}{P+N}\right)$
- Entropy is 0 if all members of S belong to the same class.
- Entropy is 1 when the collection contains equal number of +ve and –ve examples.

Example:

$$Entropy\ (S) - p+\ log2\ p+ - p-\ log2\ p-$$

Entropy ([9+, 5–])

= – 9/14 *log*2 (9/14) – 5/14 *log*2 (5/14)

= – 9/14 (0.637) - 5/14 (1.486)

= 0.4095+0.531

= 0.940

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$
\begin{aligned}
Values(Wind) &= Weak, Strong \\
S &= [9+, 5-] \\
S_{Weak} &\leftarrow [6+, 2-] \\
S_{Strong} &\leftarrow [3+, 3-] \\
Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\
&= Entropy(S) - (8/14) Entropy(S_{Weak}) \\
&\quad - (6/14) Entropy(S_{Strong}) \\
&= 0.940 - (8/14)0.811 - (6/14)1.00 \\
&= 0.048
\end{aligned}
$$

After calculating the information gain for all the attributes, the attribute with the maximum gain is chosen as the node to be split further.

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(
cancer.data, cancer.target, stratify=cancer.target, random_state=42)
tree = DecisionTreeClassifier(random_state=0)
tree.fit(X_train, y_train)
print("Accuracy on training set: {:.3f}".format(tree.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(tree.score(X_test, y_test)))
```

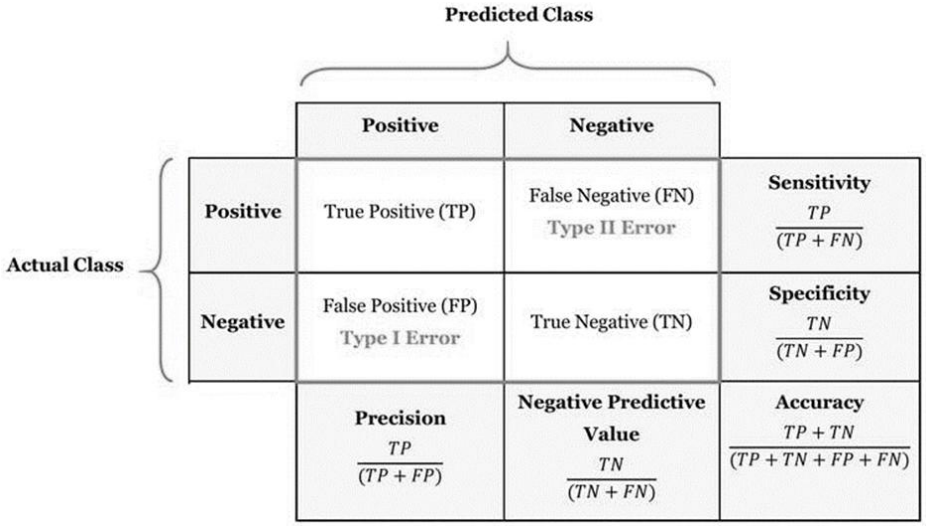
```
Accuracy on training set: 1.000
Accuracy on test set: 0.937
```

8. **Explain Confusion Matrix with example. What is the significance of ROC-AUC?**

Evaluating the performance of a Machine learning model is one of the important steps while building an effective ML model. To evaluate the performance or quality of the model, different metrics are used, and these metrics are known as performance metrics or evaluation metrics. These performance metrics help us understand how well our model has performed for the given data. In this way, we can improve the model's performance by tuning the hyper-parameters. Each ML model aims to generalize well on unseen/new data, and performance metrics help determine how well the model generalizes on the new dataset.

In a classification problem, the category or classes of data is identified based on training data. The model learns from the given dataset and then classifies the new data into classes or groups based

on the training. It predicts class labels as the output, such as Yes or No, 0 or 1, Spam or Not Spam, etc. To evaluate the performance of a classification model, different metrics are used, and some of them are as follows:



1. Accuracy

The accuracy metric is one of the simplest Classification metrics to implement, and it can be determined as the number of correct predictions to the total number of predictions.

But It is recommended not to use the Accuracy measure when the target variable majorly belongs to one class. For example, Suppose there is a model for a disease prediction in which, out of 100 people, only five people have a disease, and 95 people don't have one. In this case, if our model predicts every person with no disease (which means a bad prediction), the Accuracy measure will be 95%, which is not correct.

A confusion matrix is a tabular representation of prediction outcomes of any binary classifier, which is used to describe the performance of the classification model on a set of test data when true values are known.

2. Classification Accuracy:

It is one of the important parameters to determine the accuracy of the classification problems. It defines how often the model predicts the correct output. It can be calculated as the ratio of the number of correct predictions made by the classifier to all number of predictions made by the classifiers. The formula is given below:

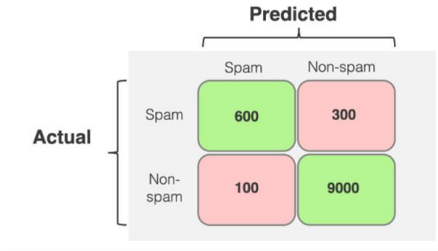
Misclassification rate: It is also termed as Error rate, and it defines how often the model gives the wrong predictions. The value of error rate can be calculated as the number of incorrect predictions to all number of the predictions made by the classifier. The formula is given below:

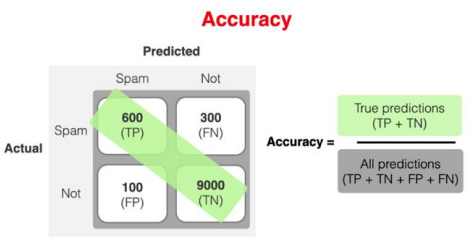
3. Precision: It can be defined as the number of correct outputs provided by the model or out of all positive classes that have predicted correctly by the model, how many of them were actually true. It can be calculated using the below formula:

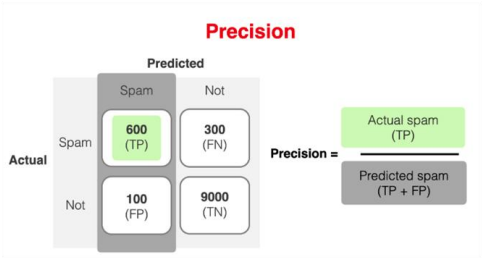
4. Recall: It is defined as the out of total positive classes, how our model predicted correctly. The recall must be as high as possible.

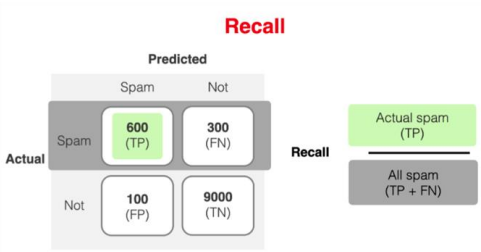
5. F-measure: If two models have low precision and high recall or vice versa, it is difficult to compare these models. So, for this purpose, we can use F-score. This score helps us to evaluate

the recall and precision at the same time. The F-score is maximum if the recall is equal to the precision. It can be calculated using the below formula:



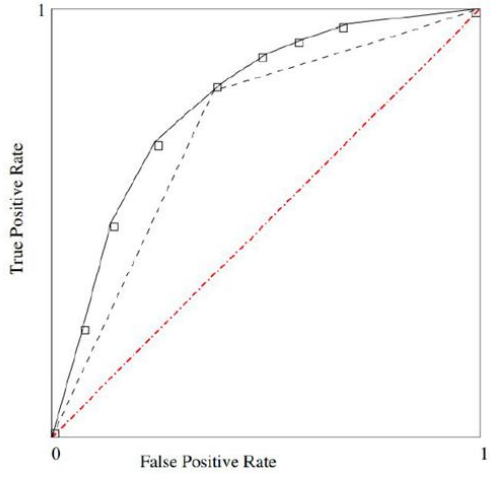






- When the cost of incorrectly identifying something as positive (a false positive) is high, - precision is crucial.
- When the cost of missing a positive case (a false negative) is high, recall is crucial. For example, in medical diagnosis, it's important to detect as many cases of a disease as possible, even if it means some false positives.

ROC-AUC:



ROC-AUC (or Receiver Operating Characteristic Area Under Curve), is a curve that maps the relationship between the True Positive Rate and False Positive Rate of the model across different cut-off thresholds.

In the ROC-AUC curve, ROC is a probability curve, and AUC represents the degree or measure of separability.

The higher the AUC, the better the model.

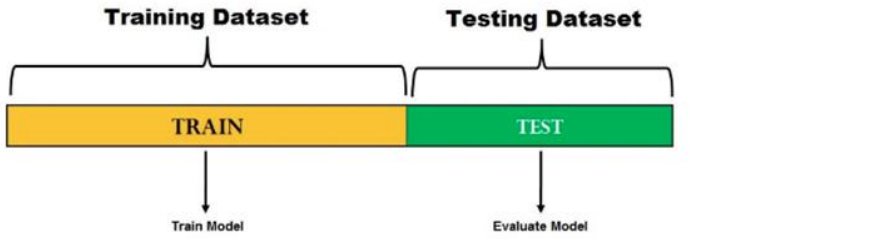
The ROC curve is generated by calculating and outlining the TPR and FPR, at various thresholds.
TPR (True Positive Rate/Sensitivity) = TP / TP +FN
FPR (False Positive Rate/Specificity) = FP / FP + TN
The ROC-AUC score ranges from 0.5 – 1, where 1 is the best

9. Explain different types of Cross Validation techniques. Apply them on a dataset and interpret the results.

Cross-Validation is a resampling technique with the fundamental idea of splitting the dataset into 2 parts- training data and test data.
Train data is used to train the model and the unseen test data is used for prediction.
Cross Validation is used to handle the Overfitting and Underfitting issues.

1. **Hold Out method**

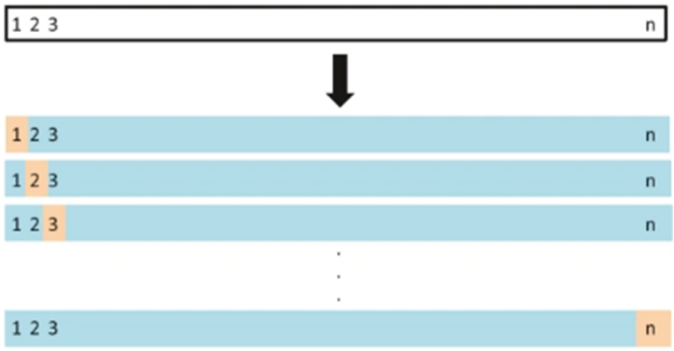
The entire dataset is divided into 2 sets – train set and test set. The data can be divided into 70-30 or 60-40, 75-25 or 80-20, or even 50-50 depending on the use case.
The data split happens randomly, and we can't be sure which data ends up in the train and test bucket during the split.
So every time, the split changes, the accuracy will also change unless we specify random_state.




```
from sklearn.model_selection import train_test_split
X_train, X_test= train_test_split(X, test_size=0.3, random_state=1)
print('Train:', X_train, 'Test:', X_test)
```

2. Leave One Out Cross-Validation

Instead of dividing the data into two subsets, we select a single observation as test data and label everything else as training data to train the model.
This process continues 'n' times, and we calculate the average of all these iterations to estimate the test set error.



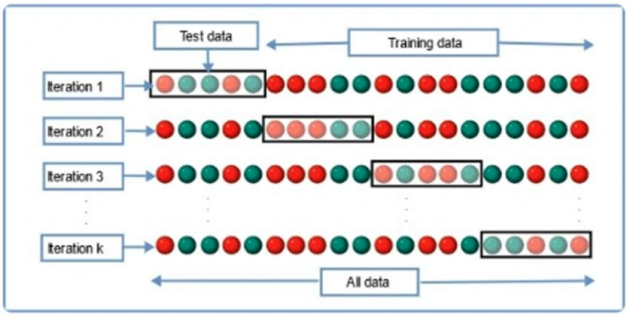
```
from sklearn.model_selection import LeaveOneOut

X = [10,20,30,40,50,60,70,80,90,100]

l = LeaveOneOut()

for train, test in l.split(X):

    print("%s %s"% (train,test))
```

3. K-Fold Cross-Validation

We divide the whole data into k sets of almost equal sizes. We select the first set as the test set and train the model on the remaining k-1 sets.

The best part about this method is that each data point appears in the test set exactly once and participates in the training set k-1 times.

As the number of folds k increases, the variance also decreases (low variance).

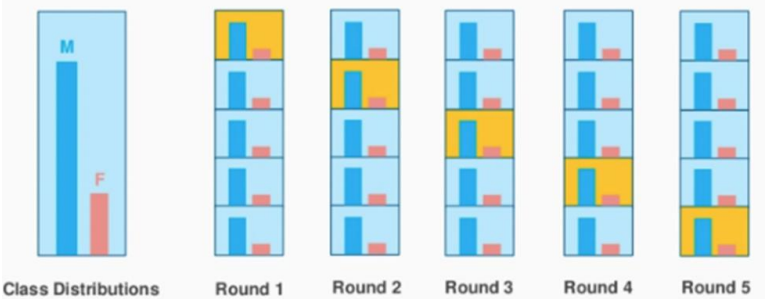



```
from sklearn.model_selection import KFold

X = ["a",'b','c','d','e','f']

kf = KFold(n_splits=3, shuffle=False, random_state=None)

for train, test in kf.split(X):

    print("Train data",train,"Test data",test)
```

4. Stratified K-Fold Cross-Validation

In the stratified K-Fold CV will create K-Folds by preserving the percentage of sample for each class.

This solves the problem of random sampling associated with Hold out and K-Fold methods.

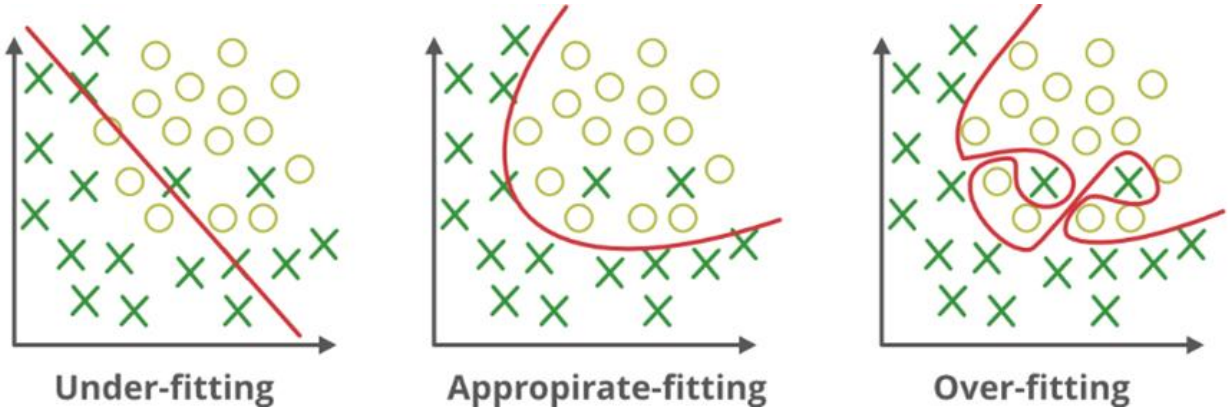


```
skf = StratifiedKFold(n_splits=3,random_state=None,shuffle=False)

for train_index,test_index in skf.split(X,y):
    print("Train:",train_index,'Test:',test_index)
    X_train,X_test = X[train_index], X[test_index]
    y_train,y_test = y[train_index], y[test_index]
```

10. Explain underfitting and overfitting with respect to Bias and Variance.

Bias and variance are two key sources of error in machine learning models that directly impact their performance and generalization ability.

The relationship between bias and variance is often referred to as the bias-variance tradeoff, which highlights the need for balance:



Increasing model complexity reduces bias but increases variance (risk of overfitting). Simplifying the model reduces variance but increases bias (risk of underfitting).

The goal is to find an optimal balance where both bias and variance are minimized, resulting in good generalization performance.

Bias: is the error that happens when a machine learning model is too simple and doesn't learn enough details from the data. It's like assuming all birds can only be small and fly, so the model fails to recognize big birds like ostriches or penguins that can't fly and get biased with predictions.
These assumptions make the model easier to train but may prevent it from capturing the underlying complexities of the data.
High bias typically leads to underfitting, where the model performs poorly on both training and testing data because it fails to learn enough from the data.

Variance: Error that happens when a machine learning model learns too much from the data, including random noise.
A high-variance model learns not only the patterns but also the noise in the training data, which leads to poor generalization on unseen data.
High variance typically leads to overfitting, where the model performs well on training data but poorly on testing data.

Overfitting:

Overfitting happens when a model learns too much from the training data, including details that don't matter (like noise or outliers).
As a result, the model works great on training data but fails when tested on new data.
Overfitting models are like students who memorize answers instead of understanding the topic. They do well in practice tests (training) but struggle in real exams (testing).

The overfitted model has **low bias** and **high variance.**

Reasons for Overfitting:
The model is too complex.
The size of the training data is small.

Underfitting:

Underfitting is the opposite of overfitting. It happens when a model is too simple to capture what's going on in the data.
In this case, the model doesn't work well on either the training or testing data.
Underfitting models are like students who don't study enough. They don't do well in practice tests or real exams.

An underfitted model has **high bias** and **low variance**.

Reasons for Underfitting:
The model is too simple, So it may be not capable to represent the complexities in the data.
The input features which is used to train the model is not the adequate representations of underlying factors influencing the target variable.
The size of the training dataset used is not enough.
Features are not scaled.