Long answer questions

1. A) Difference between supervised and unsupervised learning in machine learning.

A.

S.no	**Supervised learning**	**Unsupervised learning**
1	Uses labeled data (input features + corresponding outputs)	Uses unlabeled data (only input features, no outputs).
2	Predicts outcomes or classifies data based on known labels.	Discovers hidden patterns, structures, or groupings in data.
3	Less complex, as the model learns from labeled data with clear guidance.	More complex, as the model must find patterns without any guidance.
4	Types of supervised learning • **Classification** (for discrete variables) • **regression** (for continuous variables).	Types of unsupervised learning • **Clustering** • **Association**
5	Model can be tested and evaluated using labeled test data.	Cannot be tested in the traditional sense, as there are no labels.
6	**Applications:** • Fraud detection • Visual recognition • Risk assessment • Spam detection • Recommendation systems • Face recognition	**Applications:** • Market basket analysis • Semantic clustering • Delivery store optimization • Identify accident prone areas • Text recognition • Object recognition
7	**Algorithms:** • Decision tree classification • Logistic regression • Support Vector Machine	**Algorithms:** • K-means clustering • Hierarchical clustering • Apriori algorithm

1.B) Explain about feature engineering.

A. Engineering features and selecting a model

Feature engineering is critical in building effective models. It involves creating, transforming, and combining raw variables into predictors that a model uses for its predictions.

Feature Creation

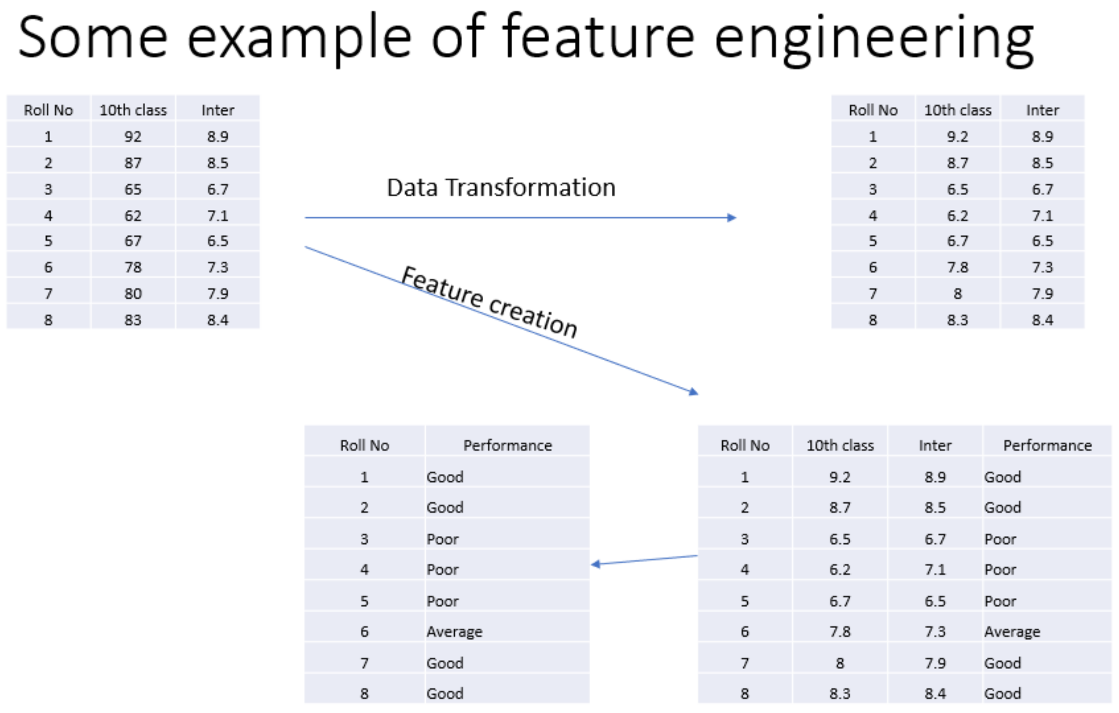
- Features may be sourced from multiple datasets and require expert consultation or literature review.
- Interaction variables (combining multiple inputs) can reveal significant effects otherwise unnoticed in single variables.

Feature Transformation

- Raw data often requires preprocessing or transformation (Standardization)

Avoiding Availability Bias

- Features should represent the entire truth, not just easily accessible data. Even though if we remove some important variables, the essence of that information should present in any of the variables.



Some example of feature engineering

Roll No	10th class	Inter
1	92	8.9
2	87	8.5
3	65	6.7
4	62	7.1
5	67	6.5
6	78	7.3
7	80	7.9
8	83	8.4

Roll No	10th class	Inter
1	9.2	8.9
2	8.7	8.5
3	6.5	6.7
4	6.2	7.1
5	6.7	6.5
6	7.8	7.3
7	8	7.9
8	8.3	8.4

Roll No	Performance
1	Good
2	Good
3	Poor
4	Poor
5	Poor
6	Average
7	Good
8	Good

Roll No	10th class	Inter	Performance
1	9.2	8.9	Good
2	8.7	8.5	Good
3	6.5	6.7	Poor
4	6.2	7.1	Poor
5	6.7	6.5	Poor
6	7.8	7.3	Average
7	8	7.9	Good
8	8.3	8.4	Good

2. A) How do we divide train data and test data? Explain different validation strategies.

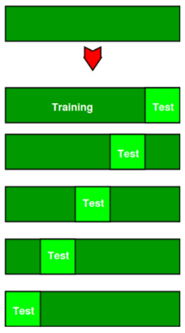
A. We split the data into 2 parts. In most of the models, we divide train and test data as 80% and 20% respectively.
Some of the validation strategies are:

i. **K- fold cross validation**

In K-Fold Cross Validation, we split the dataset into k number of subsets and we perform training on the all the subsets by leaving one subset for the evaluation of the trained model(testing). In this method, we iterate k times with a different subset reserved for testing purpose each time.

Example:

If there are 10 entries in a data set, 8 entries are considered as training data(80%) and 2 entries are considered as test data(20%)



	Training Data	**Test Data**
Iteration 1	1-8	9,10
Iteration 2	1-6,9,10	7,8
Iteration 3	1-4,7-10	5,6
Iteration 4	1,2,5-10	3,4
Iteration 5	3-10	1,2

ii. **Leave one out cross validation**

In this method, we perform training on the whole dataset but leaves only one data-point of the available dataset and then iterates for each data-point. In LOOCV, the model is trained on (n−1) samples and tested on the one omitted sample, repeating this process for each data point in the dataset. It has some advantages as well as disadvantages also.

An advantage of using this method is that we make use of all data points and hence it is low bias.

Disadvantage of using this method is takes more processing time when data set is large.

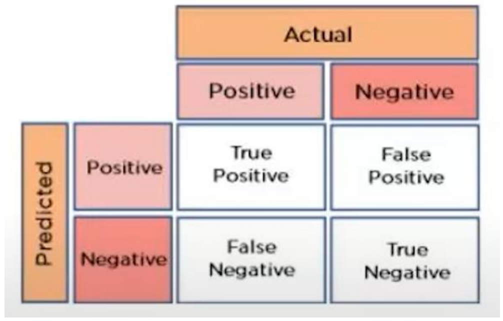
	Training Data	**Test Data**
Iteration 1	1-9	10
Iteration 2	1-8,10	9
Iteration 3	1-7,9,10	8
Iteration 4	1-6,8-10	7
Iteration 5	1-5,7-10	6
Iteration 6	1-4,6-10	5
Iteration 7	1-3,5-10	4
Iteration 8	1,2,4-10	3
Iteration 9	1,3-10	2
Iteration 10	2-10	1

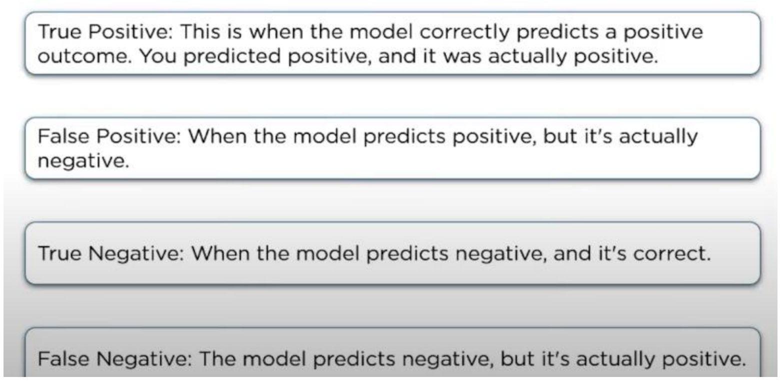
So, if there are 10 entries, it iterates 10 times.

2. B) What is confusion matrix? When do we calculate F1- score? Calculate accuracy, precision, recall and F1-Score for the below matrix.

	Actual	
predicted	**16**	**30**
	10	**144**

A. A **Confusion matrix** is a two-dimensional matrix used in classification models to evaluate the performance of a system by showing the number of correctly and wrongly classified data.




True Positive: This is when the model correctly predicts a positive outcome. You predicted positive, and it was actually positive.

False Positive: When the model predicts positive, but it's actually negative.

True Negative: When the model predicts negative, and it's correct.

False Negative: The model predicts negative, but it's actually positive.

Accuracy in machine learning is a metric that measures how well a model predicts outcomes. It's calculated by dividing the number of correct predictions by the total number of predictions. Accuracy can be calculated for classification models.

$$Accuracy = \frac{Correct\ Predictions}{Total\ Predictions}$$

We calculate accuracy from confusion matrix as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision measures the accuracy of positive predictions means out of positive predictions, how many predictions are really positive.

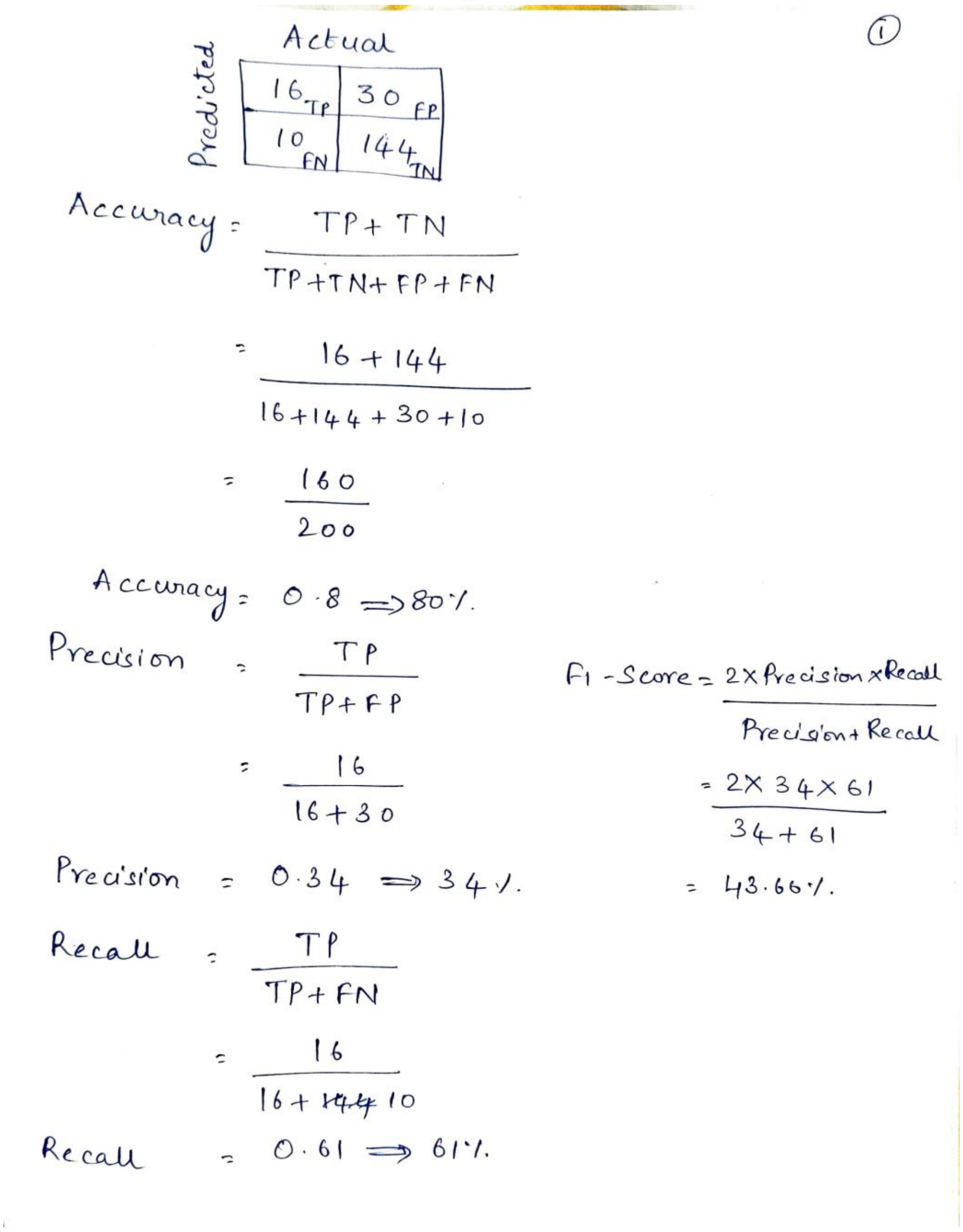
$$Precision = \frac{TP}{TP + FP}$$

Recall measures how well a model can identify all positive instances means out of real positive predictions, how many are predicted as positive.

$$Recall = \frac{TP}{TP + FN}$$

F1 score is a harmonic mean of precision and recall, which balances the importance of both metrics. When a data set with 100 entries have 90-95 similar class labels, then we calculate F1-score instead of accuracy.

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$



①

Actual

Predicted		
	16_{TP}	30_{FP}
	10_{FN}	144_{TN}

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$= \frac{16 + 144}{16 + 144 + 30 + 10}$$

$$= \frac{160}{200}$$

$$Accuracy = 0.8 \Rightarrow 80\%$$

$$Precision = \frac{TP}{TP + FP}$$

$$= \frac{16}{16 + 30}$$

$$Precision = 0.34 \Rightarrow 34\%$$

$$Recall = \frac{TP}{TP + FN}$$

$$= \frac{16}{16 + \cancel{144}\ 10}$$

$$Recall = 0.61 \Rightarrow 61\%$$

$$F_1\text{-}Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

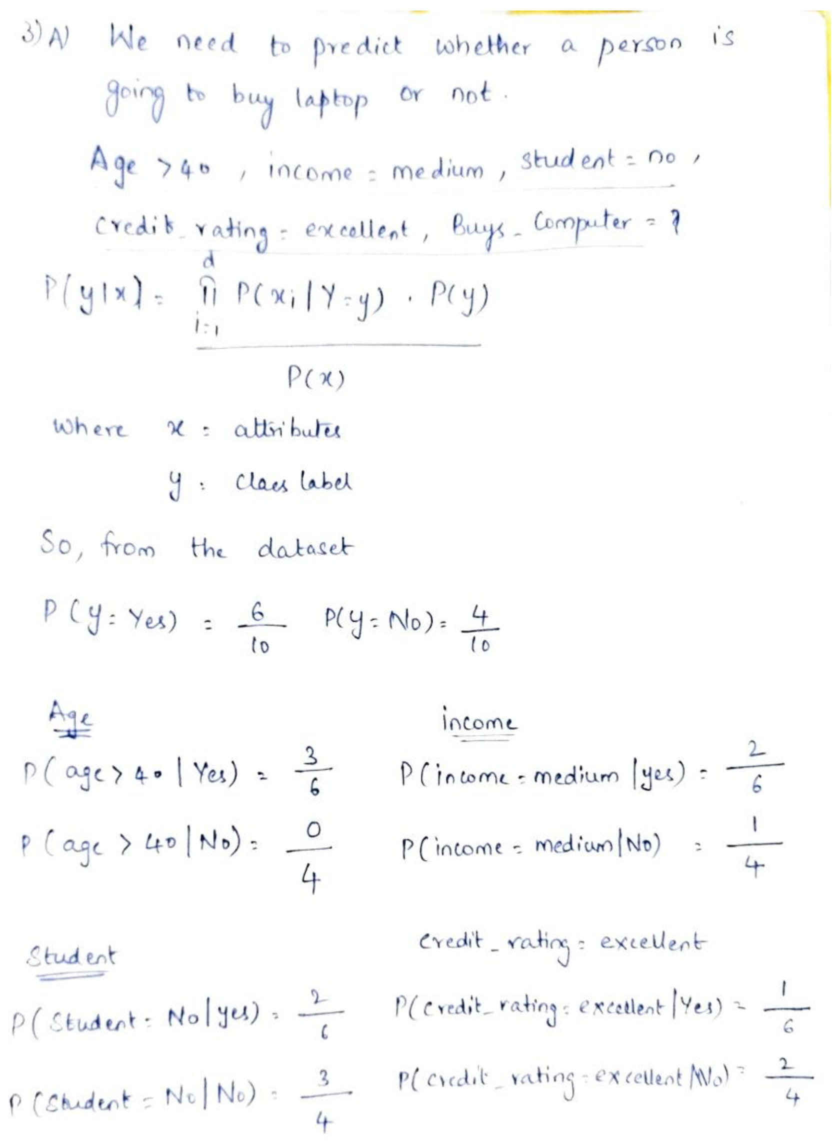
$$= \frac{2 \times 34 \times 61}{34 + 61}$$

$$= 43.66\%$$

3. **Predict whether a person is going to buy laptop or not using Naïve Bayes classification.**

Age	**Income**	**Student**	**Credit_rating**	**Buys_computer**
<=30	High	No	Fair	No
<=30	High	No	Excellent	No
31-40	High	No	Fair	Yes
>40	Medium	No	Fair	Yes
>40	Low	Yes	Fair	Yes
31-40	Low	Yes	Excellent	No
<=30	Low	Yes	Excellent	Yes
<=30	Medium	No	Fair	No
>40	Low	Yes	Fair	Yes
<=30	Medium	Yes	Fair	Yes

Predict the class label for Age >40, income = medium, student = no, credit_rating = excellent



3) A) We need to predict whether a person is going to buy laptop or not.

Age > 40 , income = medium , student = no ,

credit_rating = excellent , Buys_Computer = ?

$$P(y|x) = \frac{\prod_{i=1}^{d} P(x_i | Y = y) \cdot P(y)}{P(x)}$$

where x : attributes

y : Class label

So, from the dataset

$P(y = \text{Yes}) = \frac{6}{10}$ $P(y = \text{No}) = \frac{4}{10}$

Age

$P(\text{age} > 40 | \text{Yes}) = \frac{3}{6}$

$P(\text{age} > 40 | \text{No}) = \frac{0}{4}$

income

$P(\text{income} = \text{medium} | \text{yes}) = \frac{2}{6}$

$P(\text{income} = \text{medium} | \text{No}) = \frac{1}{4}$

Student

$P(\text{Student} = \text{No} | \text{yes}) = \frac{2}{6}$

$P(\text{Student} = \text{No} | \text{No}) = \frac{3}{4}$

credit_rating = excellent

$P(\text{credit\_rating} = \text{excellent} | \text{Yes}) = \frac{1}{6}$

$P(\text{credit\_rating} = \text{excellent} | \text{No}) = \frac{2}{4}$

A.

$$P(y = Yes \mid x) = \frac{\cancel{3}}{6} \times \frac{\cancel{2}}{\cancel{6}} \times \frac{\cancel{2}}{6} \times \frac{1}{\cancel{6}} \times \frac{\cancel{6}}{\cancel{10}\,5}$$

$$= \frac{1}{360}$$

$$= 0.002$$

$$P(y = No \mid x) = \frac{0}{4} \times \frac{1}{4} \times \frac{3}{4} \times \frac{2}{4} \times \frac{4}{10}$$

$$= 0$$

So, the person ~~w~~ is having high probability to buy a laptop.

$$P(y = \text{Yes} \mid x) = \frac{\cancel{3}}{6} \times \frac{\cancel{2}}{\cancel{6}} \times \frac{\cancel{2}}{6} \times \frac{1}{\cancel{6}} \times \frac{\cancel{6}}{\cancel{10}\,5}$$

$$= \frac{1}{360}$$

$$= 0.002$$

$$P(y = \text{No} \mid x) = \frac{0}{4} \times \frac{1}{4} \times \frac{3}{4} \times \frac{2}{4} \times \frac{4}{10}$$

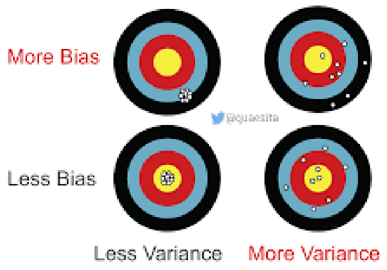
$$= 0$$

So, the person ~~is~~ is having high probability to buy a laptop.

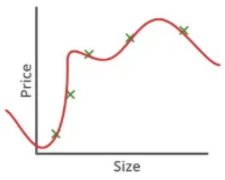
4. What is overfitting and underfitting? How is it related to bias and variance? Explain how to overcome problems faced with overfitting.

A. **Bias** is the difference between the actual value and predicted value(Training error is more)

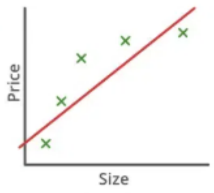
Variance is the measure of spread in data from it mean position.(Training error is less but test error is more)



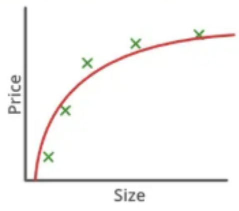
Overfitting occurs when a model learns the training data too closely, performing very well on the data it was trained on but poorly on new, unseen data. It has **low bias and high variance**.



Underfitting happens when a model is too simple and fails to capture the underlying patterns in the training data, resulting in poor performance on both training and new data sets. It has **high bias and high variance.**



Good fit refers to a model that accurately captures the underlying patterns in the training data while also performing well on unseen test data, achieving a balance between bias and variance.



Overfitting problems can be reduced by **Regularization**

- **L1 Regularization (LASSO):** Encourages models with fewer predictors, improving simplicity and robustness. In L1 regularization, we assign the values to zero for the least important attributes to overcome overfitting.

$$\text{Loss}_{L1} = \text{MSE} + \lambda \sum |w_i|$$

- **L2 Regularization (Ridge):** Reduces variance among predictor coefficients, enhancing interpretability. In this the least important values are replaced with some minimal values such as 0.001 in order to overcome overfitting.

$$\text{Loss}_{L2} = \text{MSE} + \lambda \sum w_i^2$$

In the same way underfitting can be overcome by increasing the features.

5. Write about different data structures to handle data in data science. Explain them.

A. Data structures have different storage requirements, but also influence the performance of CRUD (create, read, update, and delete) and other operations on the data set. Choosing the right data structure is very important to handle large data in data science.
The different data structures used for handling large data structures are:

- **Sparse data**
- **Tree structures**
- **Hash tables**

Sparse data

The data is represented in the form of 0's and 1's. 0 represents True i.e presence of data and 1 represents False i.e absence of data.

Example:

Let us consider Twitter Sentiment Analysis and analyze review on one movie. For this we consider tweets on the movie.

Let us review on Game Changer movie

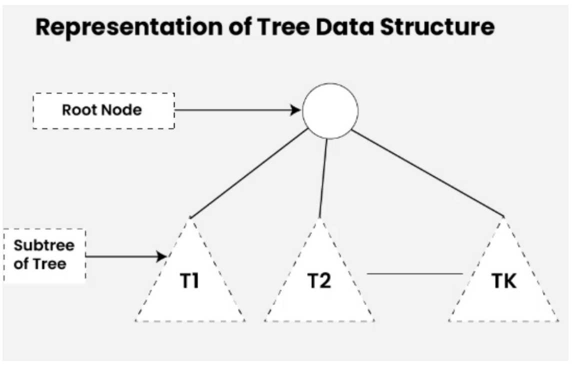
Tweet	**Ram Charan**	**Shankar**	**Kiara Advani**	**Average**	**Worst**	**good**	**Disaster**
Ramcharan acting is good	1	0	0	0	0	1	0
Shankar direction is worst	0	1	0	0	1	0	0

Story is disaster	0	0	0	0	0	0	1

In this way we store the tweets in sparse matrix. If most of the values are 0, we ignore the matrix.

Tree structures

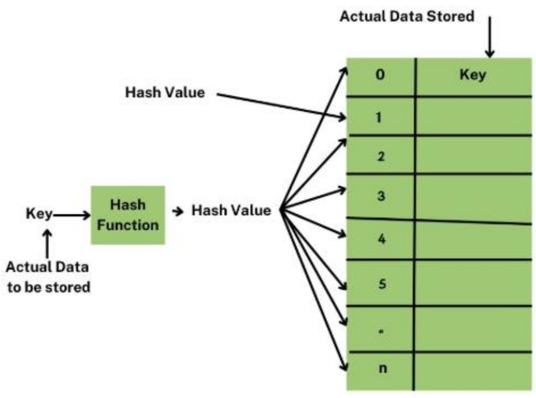
Tree data structure is a hierarchical structure that is used to represent and organize data in the form of parent child relationship.



The primary advantage of a tree structure in data structure is its ability to efficiently store and access data in a hierarchical manner, allowing for faster searching, insertion, and deletion operations compared to linear data structures like arrays, especially when dealing with data that has relationships between elements, like a file system or organizational chart; making it ideal for representing hierarchical relationships within data sets.

Hash tables

A hash table is a type of data structure in which information is stored in an easy-to-retrieve and efficient manner. In the key-value method, keys are assigned random indexes where their values are stored in an array. The index is the information of where exactly in the array the value is stored.



HASH TABLE

Collision- A **collision** happens when two different keys generate the same hash value and try to occupy the same position in the array. Since a hash table must store both keys and values, we need a way to handle these collisions efficiently.

101,A	**101%10=1**
103,B	**103%10=3**
111,C	**111%10=1**

Here collision occurs between two keys 101 and 111 as they have same hash value 1. here a collision has occurred. This can be resolved by **chaining.**

Instead of storing as single value pair, we can store as linked list. This is known as **chaining.**

6. Write python code to demonstrate Naïve Bayes classification.

A. Naïve Bayes Classification code:

Step 1. Import the packages

```
import pandas as pd
import numpy as np
from sklearn.naive_bayes import GaussianNB
```

Step 2. Get the data

```
df = pd.read_csv("/content/PLACEMENT DATASET.csv")
df
```

Step 3. Convert the non-numerical values into numerical
Note: GaussianNB works with numerical data only

```
df['gender'] = df['gender'].map({'M': 0, 'F': 1})
df['communicationskills']=df['communicationskills'].map({'GOOD':0,'AVER
AGE':1,'POOR':2})
df['codingskills']=df['codingskills'].map({'GOOD':0,'AVERAGE':1,'POOR':2})
df
```

Step 4. Divide the data into features and class tables
Here, X is features and Y is class label. (.values) is used to convert the features into numpy array. Because, GaussianNB() needs data in array form.

```
X=df[['coding skills','communication skills','gender','b.tech
percentage']].values
Y=df['placement status']
```

Step 5. Apply naive bayes algorithm on our data using fit function.
Note: Here model is an instance (Object) of Naive bayes

```
model = GaussianNB()
model = model.fit(X, Y)
```

Step 6: Apply the model on unknown sample.

# Create a new data point for prediction

```
new_data_point = np.array([[1, 1, 0, 70]]) # Example: coding skills=1,
communication skills=1, gender=0, b.tech percentage = 70
```

# Make prediction

```
predicted_y = model.predict(new_data_point)
```

# Print the predicted value

```
print(f"Predicted placement status: {predicted_y[0]}")
```

output:

```
Predicted placement status: NP
```

7. **Using k-means clustering algorithm, group the below data into three clusters thin, average and athletic.**

ID	Height (cm)	Weight (kg)
1	170	56
2	172	58
3	175	60
4	177	62
5	160	50
6	180	75
7	182	78
8	185	85
9	170	70
10	168	54

7) A) K-means clustering

Step-1 - Randomly choose any 3 centroids as we want to group into 3 clusters named thin, average, athletic.

$C_1 = [170, 56]$, $C_2 = [160, 50]$, $C_3 = [168, 54]$

Step-2 - Calculate the distance between centroid to each point using Euclidean distance.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Point	d(C1)[170,56]	d(C2)[160,50]	d(C3)[168,54]	
[170,56]	0	11.66	2.82	C1
[172,58]	2.82	14.42	5.65	C1
[175,60]	6.40	18.02	11.31	C1
[177,62]	9.21	20.80	12.04	C1
[160,50]	11.66	0	8.94	C2
[180,75]	21.47	32.01	24.18	C1
[182,78]	25.05	~~11.31~~ 35.60	27.78	C1
[185,85]	32.64	43.01	35.35	C1
[170,70]	14	22.36	16.12	C1
[168,54]	2.82	8.94	0	C3

Step-3 - Update centroids (By calculating mean)

c_1 = [170,56], [172,58], [175,60], [177,62], [180,75],
[182,78], [185,85], [170,70]

c_2 = [160,50]

$$c_k = \left(\frac{\Sigma x}{n}, \frac{\Sigma y}{n}\right)$$

c_3 = [168,54]

Step-4 - Repeat until convergence (Continue until clusters donot change).
and find out final clusters.

8. What do you mean by online learning algorithm? Explain with python code to demonstrate online learning algorithm.

A. We have 3 different options to handle large data. They are:

- **Full batch learning**(loading the complete dataset by increasing hardware features, but not a good practice)
- **Mini batch learning**(taking the dataset in batches like 100, 1000 entries depending on our hardware)
- **Online learning**(feed the algorithm one observation at a time)

Online Algorithm

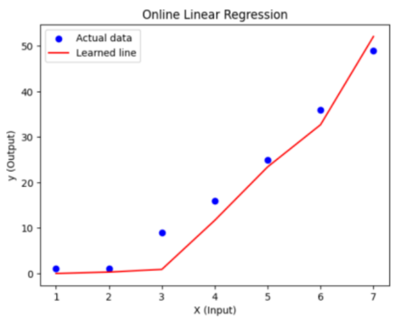
Some machine learning algorithms train on one observation at a time, updating the model and discarding the data. This "use and forget" approach solves memory issues, as single observations are too small to overwhelm modern computer memory.

Python code for online learning algorithm

```
import matplotlib.pyplot as plt

# Simple Online Linear Regression

weight = 0  # Initial weight

bias = 0  # Initial bias

learning_rate = 0.1  # How fast we learn

# Data (X = input, y = output)

X = [1, 2, 3, 4, 5,6,7]

y = [1, 1, 9, 16, 25,36,49]
```

```
# Store predictions for plotting
predictions = []
# Train the model
for i in range(len(X)):
# Make a prediction
prediction = weight * X[i] + bias
predictions.append(prediction)  # Store the prediction
# Calculate the error
error = prediction - y[i]
# Update the weight and bias
weight -= learning_rate * error * X[i]
bias -= learning_rate * error
# Plot the actual data points
plt.scatter(X, y, color='blue', label='Actual data')
# Plot the learned line (predictions)
plt.plot(X, predictions, color='red', label='Learned line')
# Add labels and title
plt.xlabel('X (Input)')
plt.ylabel('y (Output)')
plt.title('Online Linear Regression')
# Show legend
plt.legend()
# Show the plot
plt.show()
```

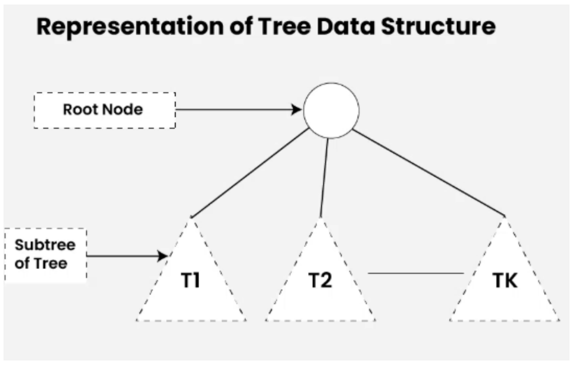
Output:



9. Compare tree structures and hash tables.

A. Tree structures

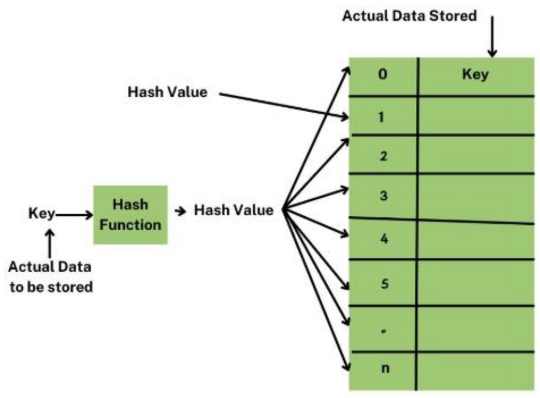
Tree data structure is a hierarchical structure that is used to represent and organize data in the form of parent child relationship.



The primary advantage of a tree structure in data structure is its ability to efficiently store and access data in a hierarchical manner, allowing for faster searching, insertion, and deletion operations compared to linear data structures like arrays, especially when dealing with data that has relationships between elements, like a file system or organizational chart; making it ideal for representing hierarchical relationships within data sets.

Hash tables

A hash table is a type of data structure in which information is stored in an easy-to-retrieve and efficient manner. In the key-value method, keys are assigned random indexes where their values are stored in an array. The index is the information of where exactly in the array the value is stored.



B. HASH TABLE

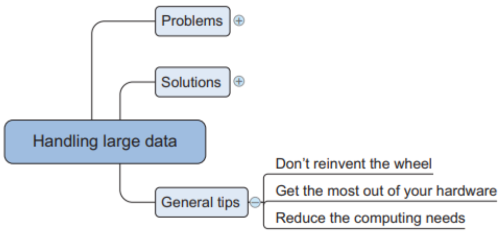
Collision- A **collision** happens when two different keys generate the same hash value and try to occupy the same position in the array. Since a hash table must store both keys and values, we need a way to handle these collisions efficiently.

101,A	**101%10=1**
103,B	**103%10=3**
111,C	**111%10=1**

Here collision occurs between two keys 101 and 111 as they have same hash value 1. here a collision has occurred. This can be resolved by **chaining.**
Instead of storing as single value pair, we can store as linked list. This is known as **chaining.**

10. What are general programming tips for dealing with large data?

A. General tips for dealing with large data sets



Don't reinvent the wheel

i. **Exploit the power of large databases**

Use SQL queries and user defined functions inside data bases to preprocess data instead doing everything in python.

ii. **Use optimized libraries**

- **Mahout** – Scalable machine learning for big data (Java-based).
- **Weka** – A collection of machine learning algorithms for data mining.
- **Scikit-learn, TensorFlow, PyTorch** – Optimized Python ML libraries

Get the Most Out of Your Hardware

i. **Feed the CPU Compressed Data:** This reduces the load on hard disks and allows the CPU to operate more efficiently.

ii. **Utilize the GPU:** For parallelizable computations, switch to GPU processing using libraries like Theano or NumbaPro, which require minimal programming effort.

iii. **Implement Multi-threading:** Use Python threads to parallelize tasks on your CPU, improving computational efficiency.

Reduce Your Computing Needs

i. **Use Compiled Code:** Leverage optimized libraries for numerical computations instead of writing everything from scratch.

ii. **Avoid Excessive Data Loading:** Process data in chunks to manage memory effectively, especially with large datasets.

iii. **Sample Your Data:** If feasible, train models on a representative sample rather than the entire dataset.

iv. **Simplify Calculations:** Use mathematical simplifications to speed up computations, such as rearranging equations for efficiency.