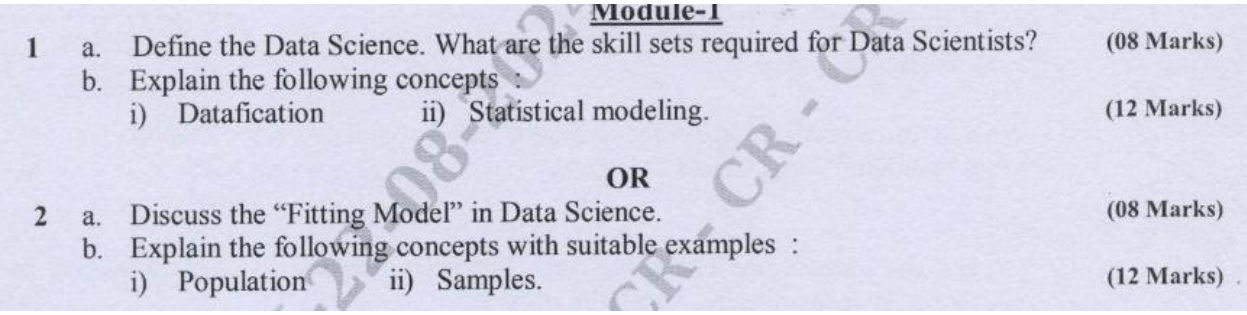
VTU Solution

21CS644 **Data Science and Visualization (Professional Elective)**



Module-1

1	a.	Define the Data Science. What are the skill sets required for Data Scientists?	(08 Marks)
	b.	Explain the following concepts : i) Datafication ii) Statistical modeling.	(12 Marks)
		OR	
2	a.	Discuss the "Fitting Model" in Data Science.	(08 Marks)
	b.	Explain the following concepts with suitable examples : i) Population ii) Samples.	(12 Marks)

Module-1

1.a) Define Data Science. What are the skill sets required for Data Scientists? (8 marks)
b) Explain the following concepts : i) Datafication ii) Statistical Modelling (12 marks)

(OR)

2. a) Discuss the "Fitting Model" in Data Science. (8 marks)
b) Explain the following concepts with suitable examples: i) Population ii) Samples – (12 marks)

Solution:

1 a) Data Science is an interdisciplinary field that utilizes scientific methods, processes, algorithms, and systems to extract insights and knowledge from structured and unstructured data. It combines various techniques from statistics, mathematics, computer science, and domain-specific knowledge to analyze data and inform decision-making. Data Science encompasses several areas, including data mining, data analysis, machine learning, and data visualization.

Skill Sets Required for Data Scientists

Data scientists typically possess a diverse skill set that includes technical, analytical, and soft skills. Here's a breakdown of the essential skills:

1. Technical Skills

- **Programming Languages**: Proficiency in languages such as Python and R, which are widely used for data analysis and machine learning.
- **Data Manipulation and Analysis**: Familiarity with libraries like Pandas, NumPy, and tools like SQL for data manipulation and analysis.
- **Machine Learning**: Understanding machine learning algorithms and frameworks (e.g., Scikit-learn, TensorFlow, Keras) for building predictive models.
- **Data Visualization**: Skills in visualization libraries (e.g., Matplotlib, Seaborn, Plotly) and tools (e.g., Tableau, Power BI) for creating insightful visualizations.

- **Statistical Analysis**: Strong foundation in statistics, including probability, hypothesis testing, regression analysis, and statistical modeling.

2. Analytical Skills

- **Problem Solving**: Ability to identify problems, formulate questions, and develop analytical solutions.
- **Critical Thinking**: Evaluating information and data critically to make informed decisions and recommendations.
- **Domain Knowledge**: Understanding the specific industry (e.g., finance, healthcare, marketing) to interpret data within context and derive actionable insights.

3. Soft Skills

- **Communication**: Ability to convey complex technical concepts to non-technical stakeholders clearly and effectively, often through storytelling and visualization.
- **Collaboration**: Working effectively in cross-functional teams with data engineers, business analysts, and domain experts.
- **Adaptability**: Willingness to learn and adapt to new tools, technologies, and methodologies as the field evolves.

4. Data Engineering Skills (optional but beneficial)

- **Database Management**: Understanding of relational and non-relational databases (e.g., MySQL, MongoDB) for efficient data storage and retrieval.
- **Big Data Technologies**: Familiarity with big data frameworks (e.g., Hadoop, Spark) for processing large datasets.

1 b) i) Datafication

Definition:
Datafication refers to the process of transforming various aspects of life and activities into quantifiable data. It involves the collection, storage, and analysis of data generated from everyday activities, behaviors, and interactions, often through digital means. This transformation allows for the extraction of insights and patterns from these data points that can inform decision-making, enhance services, and drive innovation.

Key Features:

- **Quantification**: Almost any activity, whether physical or digital, can be quantified. For example, social interactions on social media, purchasing behaviors, health metrics (like heart rate), and even environmental conditions can be turned into data.

- **Collection**: Datafication relies heavily on technology to collect data. This can involve sensors, mobile devices, social media platforms, and other digital tools.
- **Analysis**: Once data is collected, it can be analyzed to identify trends, correlations, and actionable insights. This analysis can be used in various fields, including marketing, healthcare, urban planning, and more.
- **Value Creation**: Datafication can lead to enhanced decision-making, improved customer experiences, and the creation of new business models by leveraging data insights.

Examples:

- **Social Media**: Platforms like Facebook and Twitter collect data on user interactions, preferences, and behaviors, allowing companies to target advertising more effectively.
- **Smart Devices**: IoT (Internet of Things) devices, such as fitness trackers, collect data on user health and activity levels, enabling personalized health recommendations.
- **Transportation**: Companies like Uber collect data on ride patterns, traffic conditions, and user preferences to optimize services and pricing.

ii) Statistical Modelling

Definition:
Statistical modelling is a mathematical framework used to represent and analyze relationships between variables in a dataset. It involves using statistical techniques to create models that can predict outcomes, explain relationships, and identify patterns based on empirical data.

Key Features:

- **Model Structure**: Statistical models typically have a defined structure, which includes dependent and independent variables. The dependent variable is the outcome being predicted or explained, while independent variables are the predictors or factors believed to influence the dependent variable.
- **Estimation**: Parameters of the model are estimated using statistical methods, such as least squares, maximum likelihood estimation, or Bayesian inference.
- **Assumptions**: Statistical models often rely on assumptions about the underlying data, such as normality, independence, and homoscedasticity (constant variance). Validating these assumptions is crucial for accurate model interpretation.
- **Interpretation**: The results from statistical models can provide insights into the strength and nature of **relationships between variables, as well as the overall model fit.**

2 a) In Data Science, **fitting a model** refers to the process of training a statistical or machine learning model on a dataset to learn the underlying patterns or relationships present in the data. This is a crucial step in predictive modeling and data analysis, as it enables the model to make accurate predictions or provide insights based on new, unseen data.

Key Components of Fitting a Model

1. **Training Data**:
    - The data used to fit the model is called the **training dataset**. It contains input features (independent variables) and the corresponding target variable (dependent variable) that the model aims to predict.
2. **Model Selection**:
    - Selecting an appropriate model is essential, as different algorithms may be suited for different types of problems (e.g., linear regression for continuous outcomes, logistic regression for binary outcomes, decision trees for classification, etc.).
3. **Loss Function**:
    - A loss function (or cost function) quantifies how well the model's predictions match the actual data. The goal of fitting the model is to minimize this loss. Common loss functions include:
        - **Mean Squared Error (MSE)** for regression tasks.
        - **Cross-Entropy Loss** for classification tasks.
4. **Optimization Algorithm**:
    - An optimization algorithm (like gradient descent) is used to minimize the loss function by adjusting the model parameters (coefficients). This involves iteratively updating the model's parameters based on the gradients of the loss function.
5. **Hyperparameters**:
    - These are the settings that are not learned during training (e.g., learning rate, number of trees in a random forest). Hyperparameter tuning is often performed to find the best set of hyperparameters for optimal model performance.
6. **Validation**:
    - The fitted model is usually evaluated on a separate validation dataset to assess its performance and generalization capability. Metrics like accuracy, precision, recall, F1-score, or R-squared may be used, depending on the task.

Steps in Fitting a Model

1. **Data Preprocessing**:
    - Clean the data, handle missing values, and preprocess features (e.g., normalization, encoding categorical variables).
2. **Split the Data**:
    - Divide the dataset into training, validation, and test sets (commonly in a 70/15/15 or 80/10/10 split) to prevent overfitting and evaluate model performance.
3. **Choose a Model**:
    - Select an appropriate algorithm based on the problem type (regression, classification, clustering, etc.).

4. **Fit the Model**:
    - Use the training data to train the model by fitting it using the selected optimization algorithm to minimize the loss function.
5. **Evaluate the Model**:
    - Use the validation dataset to assess the model's performance and make adjustments if necessary. This might involve tuning hyperparameters or trying different algorithms.
6. **Testing**:
    - Finally, assess the model's performance on the test set to evaluate how well it generalizes to new data.

2.b) In statistics and data science, understanding the concepts of **population** and **samples** is crucial for conducting analyses and drawing valid conclusions. Here's a detailed explanation of both concepts, along with suitable examples.

i) Population

Definition:
A population refers to the entire set of individuals or items that are of interest in a particular study. It includes all possible observations that meet a certain criterion and can be finite (e.g., all students in a university) or infinite (e.g., all possible rolls of a die).

Key Features:

- **Comprehensive**: The population encompasses all possible members of the group being studied.
- **Parameters**: Characteristics of a population are described by parameters, such as the population mean (μ\muμ), population variance (σ2\sigma^2σ2), etc.
- **Cost and Time**: Studying the entire population may be impractical due to cost, time, or accessibility constraints.

Example: Suppose you are conducting research on the average height of adult men in a city. In this case:

- **Population**: All adult men living in that city. This includes every man aged 18 and older in the city.

ii) Sample

Definition:
A sample is a subset of the population selected for analysis. Samples are used to make inferences about the population without needing to collect data from every individual within the population.

Key Features:

- **Representative**: Ideally, the sample should be representative of the population to ensure that the findings can be generalized.
- **Statistics**: Characteristics of a sample are described by statistics, such as the sample mean (x¯\bar{x}x¯), sample variance (s2s^2s2), etc.
- **Sampling Methods**: There are various methods to select samples, including random sampling, stratified sampling, systematic sampling, and convenience sampling.

Example: Continuing with the previous example:

- **Sample**: If you randomly select 100 adult men from different neighborhoods in the city to measure their heights, that group of 100 men represents your sample.

Comparison and Importance

- **Scope**: The population is the whole group, while a sample is a part of that group.
- **Data Collection**: Collecting data from a population can be time-consuming and expensive, while sampling allows for quicker and less costly data collection.
- **Inference**: By analyzing the sample, researchers can make inferences about the population. Statistical techniques help estimate population parameters based on sample statistics.

Example in Practice

Let's consider a scenario to illustrate the concepts further.

Scenario: A university wants to determine the average GPA of all its students (population) but finds it impractical to collect GPAs from every student due to time constraints.

1. **Population**: All enrolled students at the university.
    - If there are 10,000 students, the population includes each one of them.
2. **Sample**: The university decides to randomly select 500 students from its list of enrolled students to participate in the study.
    - The average GPA of these 500 students will be calculated to estimate the average GPA of all 10,000 students.

Module-2

3	a.	Briefly explain the Exploratory Data Analysis (EDA) process with an example.	(08 Marks)
	b.	Explain the working of the K – Nearest Neighbour (K – NN) algorithm, with a suitable example.	(12 Marks)
		OR	
4	a.	Briefly explain the Data Science Process with a neat diagram.	(08 Marks)
	b.	Explain the working of the K – mean algorithm with a suitable example.	(12 Marks)

Module- 2

3.a) Briefly explain the Exploratory data analysis(EDA) process with an example.(8 marks)

b) Explain the working of the K- Nearest Neighbour(K-NN) algorithm , with a suitable example (12 marks)

OR

4. a) Briefly explain the Data Science Process with a neat diagram.(8 marks)

b) Explain the working of K- Means algorithm with a suitable example (12 marks)

Solution :

3.a) Exploratory Data Analysis (EDA) is a critical step in the data analysis process, aimed at summarizing the main characteristics of a dataset, often using visual methods. EDA helps in understanding the underlying patterns, detecting anomalies, testing hypotheses, and checking assumptions, which can guide subsequent analysis and modeling.

EDA Process

The EDA process typically involves the following steps:

1. **Data Collection**:
    - Gather the data from various sources, which could be databases, CSV files, APIs, etc.
2. **Data Cleaning**:
    - Identify and handle missing values, outliers, and inconsistencies in the data.
    - Remove duplicates and ensure that data types are correct.
3. **Descriptive Statistics**:
    - Calculate summary statistics (mean, median, mode, standard deviation, quartiles) to understand the data distribution.
    - Use measures like skewness and kurtosis to analyze the shape of the data distribution.
4. **Data Visualization**:
    - Create visualizations to explore data patterns, trends, and relationships.
    - Common visualizations include:
        - Histograms for distribution
        - Box plots for identifying outliers

        - Scatter plots for relationships between variables
        - Heatmaps for correlation analysis
5. **Feature Exploration**:
    - Analyze individual features and their distributions.
    - Examine relationships between features to identify potential predictors for modeling.
6. **Hypothesis Testing** (optional):
    - Formulate and test hypotheses based on initial findings from the data.
7. **Documentation**:
    - Document the insights gained during the EDA process to inform further analysis or modeling.

Example of EDA Process

Let's consider a dataset containing information about a company's employees, including features such as age, salary, department, and years of experience. Here's a step-by-step illustration of the EDA process:

Step 1: Data Collection

- Load the dataset using pandas in Python.

python
Copy code
```
import pandas as pd

# Load dataset
data = pd.read_csv('employee_data.csv')
```

Step 2: Data Cleaning

- Check for missing values and data types.

python
Copy code
```
# Check for missing values
missing_values = data.isnull().sum()

# Data types
data_types = data.dtypes
```

- Fill or drop missing values as necessary.

Step 3: Descriptive Statistics

- Generate summary statistics.

python
Copy code
```
# Descriptive statistics
summary_statistics = data.describe()
```

Step 4: Data Visualization

- Visualize the data using various plots.

python
Copy code
```
import matplotlib.pyplot as plt
import seaborn as sns

# Histogram of employee ages
plt.figure(figsize=(10, 5))
sns.histplot(data['age'], bins=10, kde=True)
plt.title('Distribution of Employee Ages')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()

# Box plot of salaries by department
plt.figure(figsize=(10, 5))
sns.boxplot(x='department', y='salary', data=data)
plt.title('Salaries by Department')
plt.xlabel('Department')
plt.ylabel('Salary')
plt.show()
```

Step 5: Feature Exploration

- Analyze the correlation between features.

python

Copy code

```
# Correlation heatmap
plt.figure(figsize=(10, 5))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

Step 6: Hypothesis Testing (Optional)

- For example, test whether the average salary differs by department using ANOVA.

Step 7: Documentation

- Summarize insights such as:
    - The average age of employees is around 35 years.
    - There are outliers in the salary distribution.
    - There is a positive correlation between years of experience and salary.

Conclusion

The EDA process is fundamental in data analysis as it provides insights into the dataset that inform further statistical analysis and model development. By systematically exploring and visualizing the data, analysts can better understand the data's characteristics, which aids in making informed decisions based on the analysis.

3.b) The **K-Nearest Neighbors (K-NN)** algorithm is a simple, yet effective, supervised machine learning algorithm used for classification and regression tasks. It works on the principle of finding the 'k' nearest data points in the feature space and making predictions based on the majority class (for classification) or the average value (for regression) of these neighbors.

How K-NN Works

1. **Data Representation**:
    - Each data point is represented as a point in a multidimensional feature space based on its attributes. For example, if you have a dataset with features like height and weight, each data point can be represented as a point in a 2D space.
2. **Choosing the Value of K**:
    - The user must select the number of neighbors kkk. A smaller value of kkk can lead to noise affecting the prediction, while a larger value may include points from other classes, which can dilute the classification accuracy.
3. **Distance Metric**:

- K-NN uses a distance metric to determine the 'closeness' of data points. Common distance metrics include:
  - **Euclidean Distance**: Most commonly used, calculates the straight-line distance between two points.
  - **Manhattan Distance**: The sum of the absolute differences of the Cartesian coordinates.
  - **Minkowski Distance**: A generalization of both Euclidean and Manhattan distances.

4. **Finding Neighbors**:
   - For a given test point, the algorithm calculates the distance between this point and all other points in the training dataset. It then identifies the kkk points that are closest to the test point.
5. **Making Predictions**:
   - **Classification**: The algorithm assigns the class that is most frequent among the kkk nearest neighbors to the test point.
   - **Regression**: The algorithm takes the average (or weighted average) of the target values of the kkk nearest neighbors to make a prediction.

Example of K-NN Algorithm

Let's illustrate the working of the K-NN algorithm with a simple example.

Scenario:

Imagine you have a dataset of fruits, characterized by their weight and color (in terms of RGB values). You want to classify a new fruit based on its features.

Fruit	Weight (g)	Color (RGB)	Type
Apple	150	(255, 0, 0)	Apple
Orange	130	(255, 165, 0)	Orange
Lemon	120	(255, 255, 0)	Lemon
Grape	50	(128, 0, 128)	Grape
Banana	120	(255, 255, 0)	Banana

Step 1: Choose k

Let's say we choose $k = 3$.

Step 2: Calculate Distances

For a new fruit with the following characteristics:

- Weight: 140g
- Color: (255, 200, 0)

We calculate the distance to each fruit in the dataset. Using Euclidean distance, the formula is:

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

Where x_1, y_1 are the coordinates of the new fruit and x_2, y_2 are the coordinates of the other fruits.

Example Distances:

Assuming weights are in one dimension and color in another, let's calculate distances:

Where x_1, y_1 are the coordinates of the new fruit and x_2, y_2 are the coordinates of the other fruits.

Example Distances:

Assuming weights are in one dimension and color in another, let's calculate distances:

- Distance to Apple:

$$\sqrt{(140 - 150)^2 + (200 - 0)^2} = \sqrt{(-10)^2 + (200)^2} \approx 200.025$$

- Distance to Orange:

$$\sqrt{(140 - 130)^2 + (200 - 165)^2} \approx \sqrt{(10)^2 + (35)^2} \approx 36.138$$

- Distance to Lemon:

$$\sqrt{(140 - 120)^2 + (200 - 255)^2} \approx \sqrt{(20)^2 + (55)^2} \approx 58.310$$

- Distance to Grape:

$$\sqrt{(140 - 50)^2 + (200 - 128)^2} \approx \sqrt{(90)^2 + (72)^2} \approx 113.601$$

- Distance to Banana:

$$\sqrt{(140 - 120)^2 + (200 - \downarrow)^2} \approx \sqrt{(20)^2 + (55)^2} \approx 58.310$$

Step 3: Identify Neighbors

After calculating the distances, you will have:

- Apple: ~200.025
- Orange: ~36.138
- Lemon: ~58.310
- Grape: ~113.601
- Banana: ~58.310

The 3 nearest neighbors based on distance are:

1. Orange (~36.138)
2. Lemon (~58.310)
3. Banana (~58.310)

Step 4: Make Prediction

Now, we check the types of these neighbors:

- Orange: Type = Orange
- Lemon: Type = Lemon
- Banana: Type = Banana

Since "Orange" appears most frequently among the 3 nearest neighbors, the algorithm will classify the new fruit as an **Orange**.

4.a) Briefly explain the Data Science Process with a neat diagram.

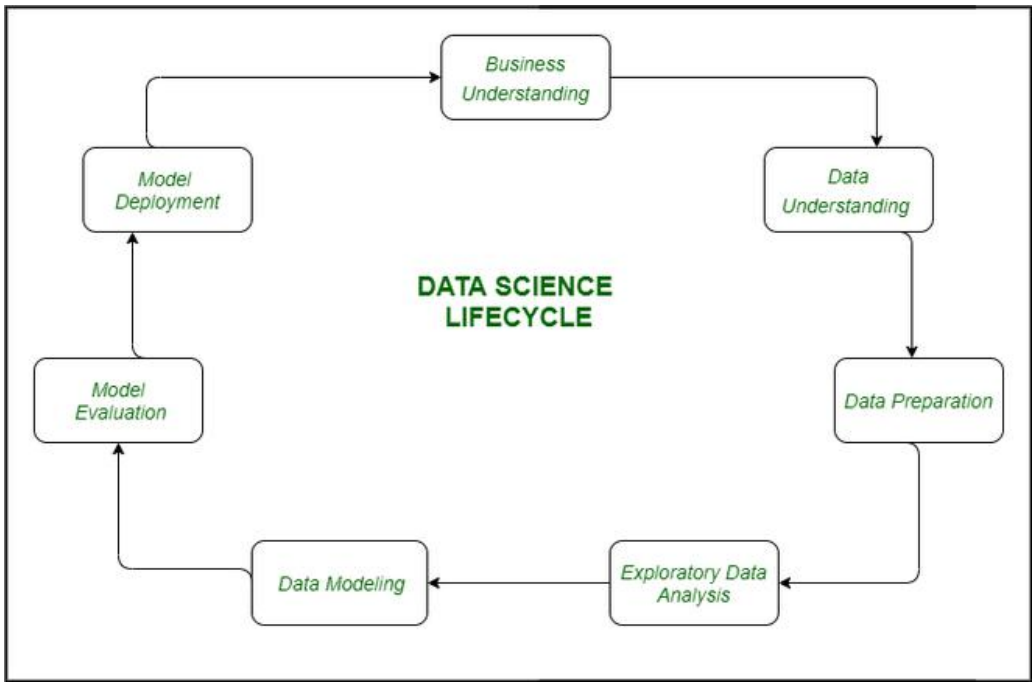
Data Science Process Life Cycle

Some steps are necessary for any of the tasks that are being done in the field of data science to derive any fruitful results from the data at hand.

- Data Collection – After formulating any problem statement the main task is to calculate data that can help us in our analysis and manipulation. Sometimes data is

collected by performing some kind of survey and there are times when it is done by performing scrapping.

- Data Cleaning – Most of the real-world data is not structured and requires cleaning and conversion into structured data before it can be used for any analysis or modeling.
- Exploratory Data Analysis – This is the step in which we try to find the hidden patterns in the data at hand. Also, we try to analyze different factors which affect the target variable and the extent to which it does so. How the independent features are related to each other and what can be done to achieve the desired results all these answers can be extracted from this process as well. This also gives us a direction in which we should work to get started with the modeling process.
- Model Building – Different types of machine learning algorithms as well as techniques have been developed which can easily identify complex patterns in the data which will be a very tedious task to be done by a human.
- Model Deployment – After a model is developed and gives better results on the holdout or the real-world dataset then we deploy it and monitor its performance. This is the main part where we use our learning from the data to be applied in real-world applications and use cases.



Data Science Process Life Cycle

4. b) Explain the working of K- Means algorithm with a suitable example (12 marks)

The K-Means algorithm is a popular unsupervised machine learning technique used for clustering. It partitions a dataset into kkk distinct, non-overlapping clusters based on feature similarity. The primary goal of K-Means is to minimize the variance within each cluster while maximizing the variance between clusters.

How K-Means Works

1. Initialization:
    - Select kkk initial centroids (cluster centers). This can be done randomly or by using techniques like K-Means++ for better initialization.
2. Assignment Step:
    - Assign each data point to the nearest centroid based on a distance metric (commonly Euclidean distance). This forms kkk clusters.
3. Update Step:
    - Calculate the new centroids for each cluster by taking the mean of all points assigned to that cluster.
4. Repeat:

- Repeat the Assignment and Update steps until the centroids do not change significantly, indicating that the algorithm has converged, or until a specified number of iterations is reached.

5. Output:
    - The final clusters and their centroids.

Example of K-Means Algorithm

Let’s consider a simple example to illustrate how the K-Means algorithm works.

Scenario:

Suppose you have a dataset representing the scores of students in two subjects: Math and Science. The goal is to cluster the students into groups based on their performance.

Step 1: Prepare the Data

Let's say the dataset is as follows:

Student	Math Score	Science Score
A	70	85
B	80	90
C	75	80
D	40	60
E	50	55
F	45	50
G	95	100
H	90	95

Step 2: Choose k

Let's choose $k = 2$ (we want to cluster the students into two groups).

Step 3: Initialize Centroids

Assuming we randomly select initial centroids:

- Centroid 1 (C1): (70, 85)
- Centroid 2 (C2): (50, 55)

Step 4: Assignment Step

Calculate the Euclidean distance from each student to both centroids and assign them to the nearest centroid.

Distance Calculation:

- For Student A:
  - Distance to C1:

$$\sqrt{(70-70)^2 + (85-85)^2} = 0$$

  - Distance to C2:

$$\sqrt{(70-50)^2 + (85-55)^2} = \sqrt{(20)^2 + (30)^2} = \sqrt{400+900} = 36.06$$

  - Assign Student A to C1.

Following the same process for all students, the assignments would look something like this:

Student	Cluster (C1 or C2)
A	C1
B	C1
C	C1
D	C2
E	C2
F	C2
G	C1
H	C1

Step 5: Update Step

Calculate the new centroids for each cluster.

New Centroids:

- For Cluster 1 (C1):
  - New C1 = Mean of points in C1 = $\frac{(70+80+75+95+90)}{5}, \frac{(85+90+80+100+95)}{5} = (82, 88)$
- For Cluster 2 (C2):
  - New C2 = Mean of points in C2 = $\frac{(40+50+45)}{3}, \frac{(60+55+50)}{3} = (45, 55)$

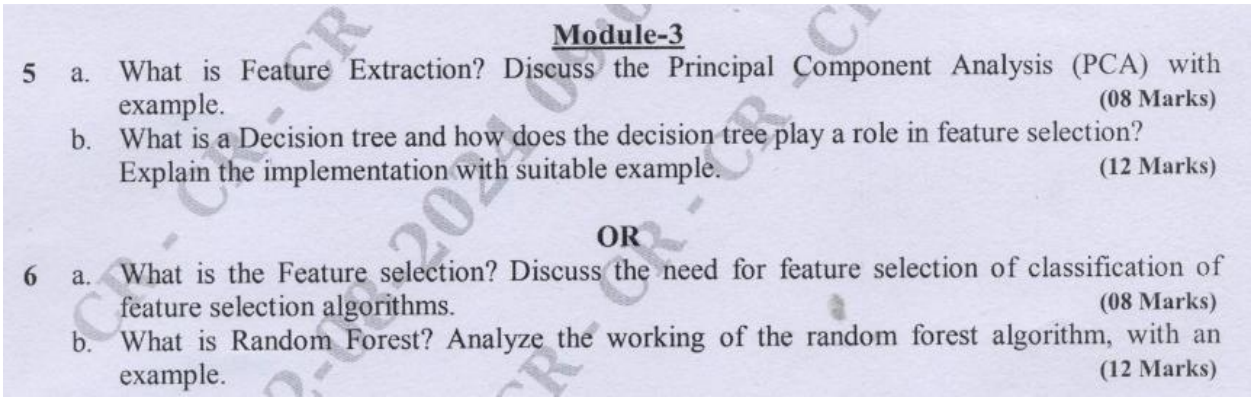
Step 6: Repeat Assignment Step

Reassign the students based on the new centroids and repeat the process until the assignments stabilize (i.e., no students switch clusters).

Final Clusters

Assuming the assignments converge after a few iterations, you may end up with two distinct clusters:

- **Cluster 1**: Students with higher scores (A, B, C, G, H)
- **Cluster 2**: Students with lower scores (D, E, F)



Module-3

5	a.	What is Feature Extraction? Discuss the Principal Component Analysis (PCA) with example.	(08 Marks)
	b.	What is a Decision tree and how does the decision tree play a role in feature selection? Explain the implementation with suitable example.	(12 Marks)
		OR	
6	a.	What is the Feature selection? Discuss the need for feature selection of classification of feature selection algorithms.	(08 Marks)
	b.	What is Random Forest? Analyze the working of the random forest algorithm, with an example.	(12 Marks)

Module-3

5. a) What is Feature Extraction? Discuss the Principal Component Analysis (PCA) with example (8 Marks)

b) What is a Decision tree and how does the decision tree play a role in feature selection? Explain the implementation with suitable example. (12 Marks)

OR

6. a) What is the Feature Selection? Discuss the need for feature selection of classification of feature selection algorithms.(8 Marks)

b) What is Random Forest? Analyze the working of the random forest algorithm , with an example. (12 Marks)

Solution:

5. a) Feature extraction is a crucial process in data preprocessing and machine learning, particularly in the context of high-dimensional data. It involves transforming raw data into a set of usable features that can effectively represent the underlying patterns in the data while reducing its dimensionality. This process aims to simplify the data without losing important information, making it easier for machine learning algorithms to learn from it.

Key Concepts of Feature Extraction

1. Dimensionality Reduction:
    - Feature extraction helps in reducing the number of input variables (features) in a dataset. This can improve model performance, reduce computation time, and mitigate overfitting.
2. Feature Selection vs. Feature Extraction:
    - Feature Selection involves selecting a subset of relevant features from the original dataset without transforming the data.
    - Feature Extraction, on the other hand, involves creating new features by transforming or combining the original features.
3. Representation:
    - The goal is to represent the data in a way that highlights the most important aspects or patterns that are relevant to the problem being solved.

Techniques for Feature Extraction

Several techniques can be used for feature extraction, depending on the type of data and the problem at hand:

1. Statistical Methods:
    - Techniques like Principal Component Analysis (PCA) reduce dimensionality by projecting data onto the directions of maximum variance.
2. Text Data:
    - For text data, methods like Term Frequency-Inverse Document Frequency (TF-IDF) and word embeddings (like Word2Vec or GloVe) are commonly used to extract meaningful features from raw text.
3. Image Data:
    - In image processing, features can be extracted using techniques like edge detection, histograms of oriented gradients (HOG), or deep learning methods using convolutional neural networks (CNNs) which automatically learn features from images.

4. Signal Processing:
    - For time-series or audio data, features like frequency components, wavelet transforms, or statistical measures (mean, variance) can be extracted.

Example of Feature Extraction

Scenario: Image Classification

Consider a scenario where you want to classify images of cats and dogs. The raw pixel data of the images can be high-dimensional (especially for large images), making it difficult for a model to learn effectively.

5 b) Principal Component Analysis (PCA) is a dimensionality reduction technique used to simplify complex datasets while preserving as much variance as possible. PCA achieves this by transforming the original features into a new set of orthogonal features called principal components, which capture the directions of maximum variance in the data.

Steps in the PCA Algorithm

Here's how PCA works, along with an example to illustrate each step:

Example Scenario

Suppose we have a dataset with two features, X1 (height) and X2 (weight), of a group of individuals. We want to reduce these two features to a single dimension while capturing as much variance as possible in the data.

Step 1: Standardize the Data

PCA is affected by scale, so we first standardize each feature to have a mean of 0 and a standard deviation of 1.

The eigenvector corresponding to Eigenvalue 1 gives us the direction of maximum variance.

Step 4: Sort Eigenvalues and Select Principal Components

We sort the eigenvalues in descending order and select the top kkk eigenvectors as our principal components. Here, we want a 1-dimensional output, so we choose the eigenvector with the highest eigenvalue (representing 90% of the variance).

Step 5: Project the Data onto the Principal Components

Finally, we project the original data onto the selected principal component(s) to obtain the reduced-dimension dataset.

If our principal component is represented by the vector P1P_1P1 , then the projected data ZZZ for each data point XXX is:

$$Z=X \cdot P1Z = X \cdot P_1Z=X \cdot P1$$

In our example, this new 1-dimensional representation captures the primary structure of the data (e.g., a combination of height and weight), summarizing the individuals along the main axis of variance.

Visualizing the Result

In a 2D plot, the points might form an elongated shape, showing a strong correlation between height and weight. The first principal component would be a line passing through this shape in the direction of maximum variance. The data points, when projected onto this line, provide a simplified 1-dimensional representation that retains most of the original variability.

5.b) A **decision tree** is a popular supervised machine learning algorithm used for both classification and regression tasks. It models decisions and their possible consequences as a tree-like structure, where each internal node represents a feature (attribute), each branch represents a decision rule, and each leaf node represents an outcome (label or value).

Structure of a Decision Tree

1. **Root Node**: The top node in the tree represents the entire dataset, which gets split into two or more subsets based on the feature that results in the most significant information gain or reduction in impurity.
2. **Internal Nodes**: These nodes represent the features used to split the data. Each internal node corresponds to a decision based on a specific feature.
3. **Branches**: These are the outcomes of the decision made at the internal node and connect the nodes.
4. **Leaf Nodes**: The end points of the tree that represent the final output or decision (class labels in classification tasks or continuous values in regression tasks).

Role of Decision Trees in Feature Selection

Decision trees inherently perform feature selection as they determine the best features to split the data based on certain criteria. Here's how decision trees contribute to feature selection:

1. **Information Gain**: Decision trees evaluate the effectiveness of a feature in classifying the training data by calculating the information gain or reduction in entropy. Features that provide the most information gain are preferred for splitting.
2. **Impurity Measures**: Metrics such as Gini impurity or entropy are used to assess the quality of splits. Features that result in lower impurity are selected.
3. **Feature Importance**: Once a decision tree is trained, it provides a measure of feature importance, which indicates how valuable each feature was in making predictions. This information can be used to select the most relevant features for a model.

Implementation of Decision Tree in Python

Let's implement a decision tree classifier using the **scikit-learn** library with a simple example.

Example: Classifying Iris Species

We'll use the well-known Iris dataset, which consists of three species of iris flowers (setosa, versicolor, and virginica) based on four features (sepal length, sepal width, petal length, and petal width).

Step 1: Import Libraries

python
Copy code

```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

Step 2: Load the Dataset

python
Copy code

```
# Load the Iris dataset
iris = load_iris()
X = iris.data  # Features
y = iris.target  # Target variable (species)
```

Step 3: Split the Dataset

python
Copy code
```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Step 4: Train the Decision Tree Model

python
Copy code
```
# Initialize the Decision Tree Classifier
clf = DecisionTreeClassifier(random_state=42)

# Train the model
clf.fit(X_train, y_train)
```

Step 5: Evaluate the Model

python
Copy code
```
# Evaluate the model
accuracy = clf.score(X_test, y_test)
print(f'Accuracy: {accuracy:.2f}')
```

Step 6: Visualize the Decision Tree

python
Copy code
```
# Plot the Decision Tree
plt.figure(figsize=(12, 8))
plot_tree(clf, filled=True, feature_names=iris.feature_names, class_names=iris.target_names)
plt.title('Decision Tree for Iris Classification')
plt.show()
```

Example Output

1. **Model Accuracy**: You should see an accuracy score printed out, indicating how well the model performed on the test set.

2. **Decision Tree Visualization**: The plot will display the structure of the decision tree, showing how the dataset was split based on the features.

Feature Importance

After training, you can also check the feature importance provided by the decision tree:

python
Copy code
```
# Get feature importances
importances = clf.feature_importances_
features = iris.feature_names
importance_df = pd.DataFrame({'Feature': features, 'Importance': importances})
print(importance_df.sort_values(by='Importance', ascending=False))
```

6. a) What is the Feature Selection? Discuss the need for feature selection of classification of feature selection algorithms.(8 Marks)
b) What is Random Forest? Analyze the working of the random forest algorithm , with an example. (12 Marks)

Solution:

Feature selection is the process of selecting a subset of relevant features (variables, predictors) from a larger set of features to use in model construction. It is a critical step in the data preprocessing pipeline, especially in machine learning, as it can significantly impact the performance and interpretability of models.

Need for Feature Selection

1. **Dimensionality Reduction**:
    - High-dimensional datasets can lead to the curse of dimensionality, where the performance of machine learning algorithms deteriorates as the number of features increases. Feature selection helps reduce the dimensionality, making the data more manageable.
2. **Improved Model Performance**:
    - By removing irrelevant or redundant features, feature selection can improve the performance of the model by reducing overfitting, which occurs when a model learns noise from the training data instead of the underlying pattern.
3. **Reduced Training Time**:
    - Fewer features mean that the model will require less computational power and time for training. This is particularly important in large datasets or when using complex algorithms.

4. **Enhanced Interpretability**:
    - A model with fewer features is generally easier to interpret. Understanding which features are most influential in making predictions can provide valuable insights into the data and the problem being addressed.
5. **Better Data Visualization**:
    - Reducing the number of features can make it easier to visualize the data, helping to identify patterns and trends.

Classification of Feature Selection Algorithms

Feature selection algorithms can be broadly classified into three categories: **Filter Methods**, **Wrapper Methods**, and **Embedded Methods**.

1. Filter Methods

- **Description**: Filter methods evaluate the relevance of features based on their intrinsic properties and statistical measures, independent of the machine learning algorithm used. They typically use metrics such as correlation, mutual information, or statistical tests.
- **Advantages**: Fast and computationally efficient; they can handle large datasets.
- **Disadvantages**: May ignore feature interactions, which can lead to suboptimal feature selection.
- **Examples**:
    - **Correlation Coefficient**: Measures the linear relationship between features and the target variable.
    - **Chi-Squared Test**: Assesses the independence between categorical features and the target variable.
    - **Mutual Information**: Measures the amount of information gained about one variable through another.

2. Wrapper Methods

- **Description**: Wrapper methods evaluate the performance of a model using different subsets of features. They use a specific machine learning algorithm to assess the accuracy of the selected features.
- **Advantages**: Consider feature interactions and provide better performance.
- **Disadvantages**: Computationally expensive, especially with a large number of features; risk of overfitting.
- **Examples**:
    - **Recursive Feature Elimination (RFE)**: Recursively removes the least important features based on the model performance until the desired number of features is reached.

    - **Forward Selection**: Starts with no features and adds features one by one, evaluating performance at each step.
    - **Backward Elimination**: Starts with all features and removes the least significant ones iteratively.

3. Embedded Methods

- **Description**: Embedded methods perform feature selection as part of the model training process. They incorporate feature selection directly into the algorithm, balancing the trade-off between performance and simplicity.
- **Advantages**: Efficient and often lead to better performance compared to filter and wrapper methods; feature selection is integrated with model training.
- **Disadvantages**: Can be less flexible as they are tied to a specific model.
- **Examples**:
    - **Lasso Regression (L1 Regularization)**: Encourages sparsity in the coefficients, effectively selecting a subset of features.
    - **Decision Trees and Random Forests**: Provide feature importance scores based on how well features split the data, allowing for automatic selection.

6.b) **Random Forest** is an ensemble learning method primarily used for classification and regression tasks. It builds multiple decision trees during training and merges their outputs to improve the accuracy and robustness of the predictions. The underlying idea is to combine the predictions of several models (in this case, decision trees) to achieve better performance than any individual model.

Key Characteristics of Random Forest

1. **Ensemble Learning**: Random Forest is based on the concept of ensemble learning, where multiple models (decision trees) are trained to make predictions.
2. **Bootstrap Aggregation (Bagging)**: It uses a technique called bagging, which involves training each decision tree on a random subset of the training data. This subset is created by sampling with replacement (bootstrap sampling).
3. **Random Feature Selection**: At each node of a decision tree, a random subset of features is selected for splitting, rather than considering all features. This adds diversity among the trees and helps to reduce overfitting.
4. **Voting Mechanism**: For classification tasks, the final prediction is made by majority voting among all the trees. For regression tasks, the average of all tree predictions is taken.

Working of the Random Forest Algorithm

1. **Data Preparation**:

    - Start with a training dataset.
2. **Bootstrap Sampling**:
    - Create multiple bootstrap samples (subsets) from the original dataset. Each subset will be used to train a different decision tree.
3. **Tree Building**:
    - For each bootstrap sample, build a decision tree:
        - At each internal node, randomly select a subset of features.
        - Choose the best feature among the selected subset to split the data at that node.
        - Repeat this process recursively until a stopping criterion is met (e.g., maximum tree depth or minimum samples per leaf).
4. **Aggregation**:
    - For classification tasks, predict the class label based on majority voting from all the trees. For regression tasks, average the predictions from all trees.
5. **Final Prediction**:
    - The aggregated output from all the trees is the final prediction of the random forest model.

Example of Random Forest Algorithm

Let's implement a Random Forest classifier using the **scikit-learn** library with the **Iris dataset**.

Step 1: Import Libraries

python
Copy code

```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

Step 2: Load the Dataset

python
Copy code

```
# Load the Iris dataset
iris = load_iris()
X = iris.data  # Features (sepal length, sepal width, petal length, petal width)
y = iris.target  # Target variable (species)
```

Step 3: Split the Dataset

python
Copy code

```
# Split the dataset into training and testing sets (70% train, 30% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Step 4: Initialize and Train the Random Forest Classifier

python
Copy code

```
# Initialize the Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
rf_classifier.fit(X_train, y_train)
```

Step 5: Make Predictions

python
Copy code

```
# Make predictions on the test set
y_pred = rf_classifier.predict(X_test)
```

Step 6: Evaluate the Model

python
Copy code

```
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Display classification report
print(classification_report(y_test, y_pred, target_names=iris.target_names))
```

Example Output

1. **Model Accuracy**: You will see an accuracy score printed, indicating how well the Random Forest model performed on the test set.
2. **Classification Report**: This will include precision, recall, and F1-score for each class, giving a detailed view of the model's performance.

Feature Importance

You can also obtain the feature importance scores from the Random Forest model:

python
Copy code

```
# Get feature importances
importances = rf_classifier.feature_importances_
features = iris.feature_names

# Create a DataFrame for better visualization
importance_df = pd.DataFrame({'Feature': features, 'Importance': importances})
print(importance_df.sort_values(by='Importance', ascending=False))
```

Module-4

7	a.	What is Data Visualization and the importance of Data Visualization? What makes a good visualization?	(08 Marks)
	b.	Explain the following concepts : i) Scatter Plot ii) Histogram iii) Box Plot.	(12 Marks)
		OR	
8	a.	Define "Data Wrangling" and Identify the steps in the flow of the data wrangling process with a neat diagram to measure employee engagement.	(08 Marks)
	b.	Explain the following concepts : i) Choropleth Map ii) Line chart iii) Bubble chart.	(12 Marks)

Solution:

7 a) What is Data Visualization and the importance of Data Visualization? What makes a good visualization?

7 b) Explain the following concepts: i) Scatter Plot ii) Histogram iii)Box Plot

8 a) Define Data Wrangling and identify the steps in the flow of the data wrangling process with a neat diagram to measure employee engagement.
8 b) Explain the following concepts: i) Choropleth Map ii) Line Chart iii) Bubble Chart

Solution:

7a) Data Visualization is the graphical representation of information and data. It uses visual elements like charts, graphs, and maps to present data trends, patterns, and outliers, making complex data easier to understand and interpret. Effective data visualization is essential for simplifying large datasets, aiding in data-driven decision-making, and uncovering insights.

Importance of Data Visualization

1. Simplifies Complex Data: Large volumes of data are distilled into visual formats that are easier to understand, allowing for quicker insight into data points and trends.
2. Enhances Decision-Making: Visual representations of data make it easier for stakeholders to make informed decisions by highlighting key metrics and relationships.
3. Identifies Trends and Patterns: Visualization helps reveal patterns, trends, and correlations that may go unnoticed in raw data, making it possible to identify issues or opportunities earlier.
4. Increases Engagement: A well-designed visualization draws attention, making it easier for audiences to engage with data, especially for non-technical users.
5. Enables Faster Analysis: By converting numbers into visuals, users can quickly process data insights, which is crucial in fast-paced business environments.

What Makes a Good Visualization?

1. Clarity: The visualization should be easy to read and interpret without clutter or excessive details. Each element should serve a clear purpose in conveying information.
2. Accuracy: Data should be represented accurately, avoiding distortions or exaggerations that could mislead the viewer.
3. Relevance: The visualization should be relevant to the audience and aligned with the purpose. Choosing the right type of chart (e.g., bar chart, line chart, or scatter plot) is essential to accurately represent the data.
4. Visual Appeal: Aesthetic elements (like color, font, and layout) should enhance readability and not distract from the data. Good use of color contrast, alignment, and spacing can make data more digestible.
5. Interactivity (if applicable): For more complex datasets, interactivity (such as filters or zoom functions) allows users to explore different aspects of the data and gain more detailed insights.
6. Context and Labeling: Clear labeling, legends, and titles are essential to ensure that viewers understand what they are looking at, including the units of measurement, axes, and data sources.
7. Focus on Key Insights: A good visualization emphasizes the most important data points or trends, directing the viewer's attention to the insights that matter most.

7 b) i) Scatter Plot

- Definition: A scatter plot is a type of data visualization that shows the relationship between two numerical variables. It displays individual data points as dots on a two-dimensional graph, where each axis represents one of the variables.

- Purpose: Scatter plots are useful for identifying relationships, correlations, or trends between two variables. For example, they can reveal if there's a positive, negative, or no correlation.
- Interpretation:
    - A positive correlation means that as one variable increases, the other also increases (dots slope upward).
    - A negative correlation means that as one variable increases, the other decreases (dots slope downward).
    - No correlation suggests no pattern, with dots scattered randomly.

ii) Histogram

- Definition: A histogram is a graphical representation that organizes data into bins (intervals) and shows the frequency of data points within each bin. Unlike a bar chart, which represents categorical data, a histogram is used for numerical data.
- Purpose: Histograms are ideal for displaying the distribution of a single continuous variable, showing how frequently different values occur within specific ranges. They help identify the shape of the data distribution (e.g., normal, skewed, bimodal).
- Interpretation:
    - A normal distribution will have a bell-shaped curve, where most data points cluster around the center.
    - A right-skewed histogram has a longer tail on the right, indicating more lower values with fewer high values.
    - A left-skewed histogram has a longer tail on the left, indicating more higher values with fewer low values.

iii) Box Plot

- Definition: A box plot (also known as a whisker plot) is a standardized way of displaying the distribution of data based on a five-number summary: minimum, first quartile (Q1), median (Q2), third quartile (Q3), and maximum. It visually shows the spread and skewness of data, including potential outliers.
- Purpose: Box plots are useful for comparing distributions between multiple groups or datasets, showing the central tendency, variability, and presence of outliers.
- Interpretation:
    - Box: Represents the interquartile range (IQR), which contains the middle 50% of the data.
    - Line inside the box: Indicates the median.
    - Whiskers: Extend to the minimum and maximum values within 1.5 times the IQR from the quartiles.
    - Outliers: Points outside the whiskers, indicating unusually high or low values.

These visualizations are invaluable tools for understanding data patterns, distributions, and relationships, each serving a distinct purpose based on the nature of the data.

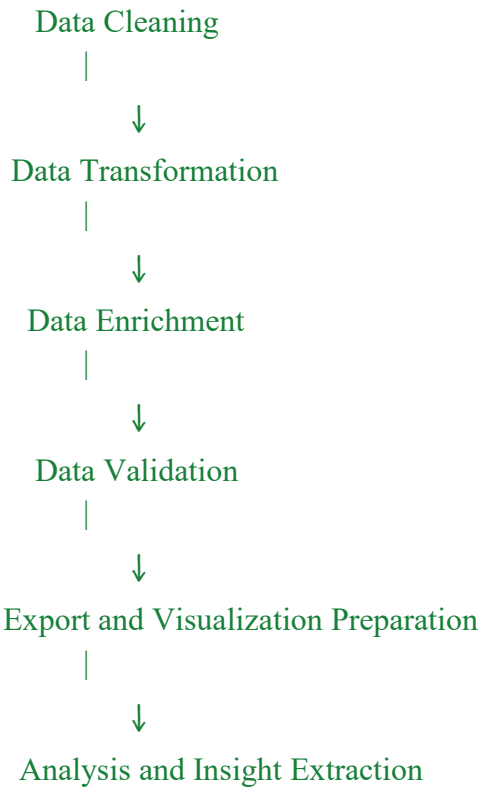
8 a) Data Wrangling (or Data Cleaning) is the process of transforming and mapping raw data from its original format into a more structured and useful format. This is especially important for tasks like measuring employee engagement, where data may come from multiple sources and require organization before analysis. Data wrangling is essential to ensure data quality, consistency, and relevance for accurate insights.

Steps in the Data Wrangling Process for Employee Engagement

Here's a step-by-step breakdown, followed by a conceptual diagram:

1. Data Collection: Gather raw data from various sources, such as employee surveys, HR records, performance reviews, and productivity metrics.
2. Data Discovery and Profiling: Analyze data to understand its structure, key attributes, and potential issues (e.g., missing values, inconsistencies). For employee engagement, this could involve profiling data on job satisfaction, feedback, attendance, etc.
3. Data Cleaning: Address any issues identified during profiling, such as handling missing values, removing duplicates, correcting data types, and standardizing formats. For engagement data, this might mean aligning survey scales or correcting data entry errors.
4. Data Transformation: Convert and structure data for analysis. This includes normalization, scaling, and formatting values as needed. For employee engagement, it may involve normalizing scores from different surveys to a common scale.
5. Data Enrichment: Integrate external or additional datasets to provide a fuller picture. In employee engagement, this might include adding demographic data or industry benchmarks.
6. Data Validation: Ensure that data is accurate, complete, and reliable. Validation steps check for anomalies and ensure the wrangled data meets predefined quality standards.
7. Data Exporting and Visualization Preparation: Prepare the cleaned and structured data for analysis or visualization. This step may involve exporting to visualization tools to analyze engagement trends and insights.

Data Wrangling Flow Diagram for Employee Engagement



Each step in this process refines the data, allowing for a more accurate and insightful analysis of employee engagement metrics. By following this structured approach, organizations can effectively measure and understand employee engagement, ultimately supporting informed decision-making.

8 b) i) Choropleth Map

- Definition: A choropleth map is a type of map that uses color shading to represent data values across different geographic areas, such as countries, states, or regions. Each area is shaded according to the data value it represents, allowing viewers to see how data varies by location.
- Purpose: Choropleth maps are useful for visualizing data related to geography, such as population density, average income, or election results, where data values differ by location.
- Interpretation:
    - Darker or more intense colors often indicate higher values, while lighter shades represent lower values.
    - The map allows easy comparison of regions, helping identify trends, hotspots, and patterns within the geographical context.

ii) Line Chart

- Definition: A line chart is a type of chart that displays data points connected by a continuous line. It's commonly used to show trends over time, where one axis (typically the x-axis) represents time intervals, and the other (y-axis) represents the variable being measured.
- Purpose: Line charts are ideal for showing data trends and progressions over time, making them a popular choice for visualizing time series data like stock prices, temperature changes, and sales growth.
- Interpretation:
    - The slope of the line indicates the rate of change. An upward slope shows an increase, while a downward slope shows a decrease.
    - Multiple lines can be used to compare different datasets on the same chart, revealing relationships or patterns across groups.

iii) Bubble Chart

- Definition: A bubble chart is a type of chart that displays three dimensions of data. It uses a scatter plot format where each point (or bubble) represents a data point with three variables: x-axis, y-axis, and bubble size, which represents the third variable.
- Purpose: Bubble charts are useful for visualizing relationships between three variables and identifying patterns or clusters within a dataset. They are often used in business and financial analysis.
- Interpretation:
    - The x-axis and y-axis show two quantitative variables, while the size of the bubble shows the third variable.
    - Larger bubbles indicate higher values of the third variable, while smaller bubbles show lower values, allowing users to compare not only the position but also the magnitude of the data points.

Each of these visualizations serves a unique purpose and can be highly effective for exploring data in geographic, time-series, and multidimensional contexts.

Module-5

Module-5

9 a. Apply a pie chart for water usage. To understand the reason behind it, generate a visual representation of water usage using a pie chart in matplotlib library.

Usage	Clothes washer	Leak	Other	Toilet	Shower	Faucet
Percentage (%)	17	12	8	24	20	19

(10 Marks)

b. What is the Visualization layout in matplotlib? Explain to create subplots and grid space using matplotlib with suitable examples. (10 Marks)

OR

10 a. Explain the Vertical and Horizontal bar chart with parameters. Create a vertical bar chart with suitable example using matplotlib library. (10 Marks)

b. What is the "Legend" function and Basic text direction? How to implement legend and basic text in the "Stacked bar chart"? (10 Marks)

Solution:

9 a) To create a pie chart in matplotlib, we'll use the plt.pie() function. Here's how to create a pie chart for the given water usage data:

```
import matplotlib.pyplot as plt

usage = ['Clothes Washer', 'Leak', 'Other', 'Toilet', 'Shower', 'Faucet']
percentage = [17, 12, 8, 24, 20, 19]

plt.pie(percentage, labels=usage, autopct='%1.1f%%')
plt.title('Water Usage Breakdown')
plt.show()
```

9 b) Matplotlib allows you to create complex visualizations by arranging multiple plots in a grid-like structure. This is achieved using subplots.

- **Subplots**:
    - plt.subplots(): This function creates a figure and a grid of subplots. You can specify the number of rows and columns in the grid.
    - plt.subplot(): This function creates a single subplot within a figure. You need to specify the row, column, and plot number.
- **Grid Space**:
    - plt.subplots_adjust(): This function adjusts the spacing between subplots, including margins, padding, and spacing between subplots.

Example:

Python

```
import matplotlib.pyplot as plt

# Create a figure with 2 rows and 2 columns of subplots
fig, axs = plt.subplots(2, 2, figsize=(10,
6))

# Plot a line chart in the first subplot
axs[0, 0].plot([1, 2, 3, 4], [1, 4, 9, 16])
axs[0, 0].set_title('Line Chart')

# Plot a bar chart in the second subplot
axs[0, 1].bar(['A', 'B', 'C'], [10, 20, 30])
axs[0, 1].set_title('Bar Chart')

# Plot a pie chart in the third subplot
axs[1, 0].pie([25, 25, 50], labels=['A', 'B', 'C'], autopct='%1.1f%%')
axs[1, 0].set_title('Pie Chart')

# Plot a scatter plot in the fourth subplot
axs[1, 1].scatter([1, 2, 3, 4], [1, 4, 9, 16])
axs[1, 1].set_title('Scatter Plot')

# Adjust spacing between subplots
plt.subplots_adjust(hspace=0.4, wspace=0.4)

plt.show()
```

10 a) **Vertical Bar Chart:**

- plt.bar(): This function creates a vertical bar chart. You need to specify the x-axis values, the height of the bars, and optional parameters like color, width, etc.

- **Horizontal Bar Chart:**
  - plt.barh(): This function creates a horizontal bar chart. It works similarly to plt.bar(), but the x and y axes are reversed.

Example (Vertical Bar Chart):

```
import matplotlib.pyplot as plt
```

```
x = ['A', 'B', 'C', 'D']
y = [10, 20, 30, 40]

plt.bar(x, y, color='blue')
plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('Vertical Bar Chart')
plt.show()
```

Use code with caution.

10. b. Legend and Basic Text in Stacked Bar Charts

- **Legend**:
    - plt.legend(): This function adds a legend to the plot. You can specify the labels for each data series.
- **Basic Text**:
    - plt.text(): This function adds text to the plot at a specific location. You can specify the x and y coordinates, the text string, and optional parameters like font size, color, etc.

```
import matplotlib.pyplot as plt

x = ['A', 'B', 'C']
y1 = [10, 20, 30]
y2 = [15, 25, 35]

plt.bar(x, y1, color='blue', label='Category 1')
plt.bar(x, y2, color='green', label='Category 2', bottom=y1)

plt.legend()
plt.text(0.5, 40, 'Total Value', ha='center')
plt.title('Stacked Bar Chart')
plt.show()
```