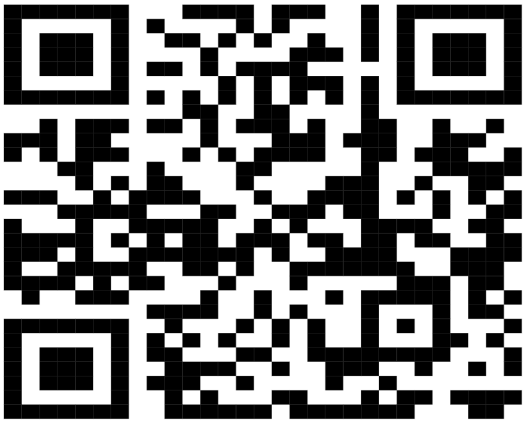
Lokmanya Tilak Jankalyan Shikshan Sanstha's
PRIYADARSHINI COLLEGE OF ENGINEERING, NAGPUR
(An Autonomous Institute affiliated to Rashtrasant Tukadoji Maharaj Nagpur University)

DEPARTMENT OF INFORMATION TECHNOLOGY

AI AND ML WITH BI - LAB MANUAL

Subject code: **24UHBIT5T**
Academic Year : 2025-2026
Semester: V

KNOW ABOUT YOUR LAB



INDEX

LOKMANYA TILAK JANKALYAN SHIKSHAN SANSTHA'S
PRIYADARSHINI COLLEGE OF ENGINEERING
An Autonomous Institute Affilliated to R.T.M. Nagpur University, Nagpur
Accredited with Grade 'A+' by NAAC
Near C.R.P.F. Campus, Hingna Road, Nagpur - 440 019 (Maharashtra) India
Phone : 07104 - 299648, Fax : 07104-299681
E-mail : principal.pce.ngp@gmail.com • Website: www.pcenagpur.edu.in
AICTE ID No. 1-5435581; DTE CODE No. 4123,
UNIVERSITY CODE No. : 278

Vision

To become one of the India's leading Engineering Institutes in both education and research. We are committed to provide quality and state-of-the-art technical education to our students so that they become Technologically competent and in turn contribute for creating a great society.

Mission

1. Fostering a dynamic learning environment that equips students with Technical expertise, problem-solving skills and a deep commitment to ethical practices.
2. To cultivate a culture of innovation, incubation, research and entrepreneurship that drives technological advancements.
3. To uphold the spirit of mutual excellence while interacting with stake holders of our Institutional ecosystem.
4. Promoting lifelong learning, professional growth and ensuring holistic development of students and the well being of society.

PCE

Principal
Priyadarshini College of Engg.
Nagpur.

PROGRAM EDUCATIONAL OBJECTIVES:

PEOs	Program Educational Objectives Statements
PEO1	Demonstrate strong technical expertise and uphold ethical values to excel in their professional careers.
PEO2	Apply advanced skills and knowledge in Information Technology to address real-world challenges and contribute to industry success.
PEO3	Pursue lifelong learning and engage in research and innovation to address societal needs and advance the field of Information Technology.

PROGRAM OUTCOMES:

Engineering Graduates will Able to:

POs	Program Outcomes

PO1	Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
PO2	Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	Conduct investigations of complex problems: Use research-based knowledge and research methods Including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO6	The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10	Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one‟s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES :

PSOs	**Program Specific Outcomes**
PSO1	An ability to apply mathematical foundations, algorithmic principles and computer science theory in the modeling and design of software systems of varying complexity.
PSO2	An ability to work with Open Source Software and use off the shelf utilities for program integration.

COURSE OUTCOMES :

Course Outcomes	Statement
CO1	To understand and implement various uninformed and informed search algorithms.
CO2	To apply problem-solving strategies and game-based algorithms.
CO3	To explore and implement constraint satisfaction and optimization problems

CO-PO & PSO MAPPING:

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO 1	PSO2
CO1	3	3	2	2	2	1	1	1	1	1	1	2	3	2
CO2	3	2	3	2	3	1	1	1	1	2	1	2	3	2
CO3	3	2	3	3	3	2	1	1	2	2	1	3	3	3

List of Experiments:

Sr. No.	Name of Practical
1	Data preprocessing for business datasets using Python.
2	Implement Linear Regression for sales forecasting.
3	Classification model for customer churn prediction.
4	Customer segmentation using K-Means clustering.
5	Market basket analysis using Apriori algorithm.
6	Building a simple recommendation engine.
7	Creating an interactive dashboard in Tableau/Power BI.
8	Integrating ML model results into a BI dashboard.
9	Using KNIME for data analytics workflow.
10	Case study on ethical AI implementation in business.
Content Beyond Syllabus	
11	To visualize COVID-19 trends using real-world datasets.
12	To create a sales analysis dashboard using business data.

PRACTICAL NO.1

Aim:

To perform data preprocessing on business datasets using Python, including handling missing values, duplicates, data transformation, and encoding categorical variables to prepare data for analysis.

Theory:

Data preprocessing is a crucial step in data analysis and machine learning that involves cleaning and transforming raw data into a usable format. For business datasets, preprocessing typically includes:

- **Handling missing values: Missing data can bias results or cause errors. Common methods include removal, imputation with mean/median/mode, or using predictive models.**

- **Removing duplicates: Duplicate records can distort insights and analyses.**

- **Encoding categorical data: Converting categorical variables (e.g., "Male", "Female") into numerical format using techniques like Label Encoding or One-Hot Encoding.**

- **Scaling and normalization: Adjusting numerical data ranges to improve model performance or visualization.**

- **Data transformation: Converting data types or extracting features (e.g., date parsing).**

Code:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

# Sample business dataset
data = {
    'CustomerID': [101, 102, 103, 104, 105, 105],
    'Gender': ['Male', 'Female', 'Female', 'Male', 'Male', 'Male'],
    'Age': [25, 30, None, 22, 40, 40],
    'Annual_Income': [50000, 60000, 52000, None, 58000, 58000],
    'Purchased': ['Yes', 'No', 'Yes', 'No', None, 'No']
}

df = pd.DataFrame(data)
print("Original Data:")
print(df)

# -------------------------
# Handling missing values
# -------------------------
# Fill missing Age with mean age
df['Age'].fillna(df['Age'].mean(), inplace=True)

# Fill missing Annual_Income with median
```

```
df['Annual_Income'].fillna(df['Annual_Income'].median(), inplace=True)

# Fill missing Purchased with mode
df['Purchased'].fillna(df['Purchased'].mode()[0], inplace=True)

# -------------------------
# Remove duplicate rows
# -------------------------
df.drop_duplicates(inplace=True)

# -------------------------
# Encode categorical variables
# -------------------------
le_gender = LabelEncoder()
df['Gender_Encoded'] = le_gender.fit_transform(df['Gender'])

le_purchased = LabelEncoder()
df['Purchased_Encoded'] = le_purchased.fit_transform(df['Purchased'])

# -------------------------
# Final processed data
# -------------------------
print("\nProcessed Data:")
print(df)
```

Expected Output:

Original Data:

CustomerID	Gender	Age	Annual_Income	Purchased
101	Male	25.0	50000	Yes
102	Female	30.0	60000	No
103	Female	Na N	52000	Yes
104	Male	22.0	NaN	No
105	Male	40.0	58000	None
105	Male	40.0	58000	No

Processed Data:

Customer ID	Gender	Age	Annual_Income	Purchased	Gender_Encoded	Purchased_Encoded
101	Male	25.00	50000	Yes	1	1
102	Female	30.00	60000	No	0	0
103	Female	29.25	52000	Yes	0	1
104	Male	22.00	56000	No	1	0
105	Male	40.00	58000	No	1	0

***Note*:**

- **Missing `Age` replaced by mean (29.25).**
- **Missing `Annual_Income` replaced by median (56000).**
- **Missing `Purchased` replaced by mode (`No`).**
- **Duplicate row with CustomerID 105 removed.**
- **Gender and Purchased encoded as numeric.**

Viva Questions:

1. **Why is data preprocessing important in business analytics?**
   ➢ It ensures data quality, handles missing or inconsistent data, and prepares data for accurate analysis and modeling.

2. **How do you handle missing values in a dataset?**
   ➢ By removing rows, imputing with mean/median/mode, or using predictive techniques.

3. **What is the difference between Label Encoding and One-Hot Encoding?**
   ➢ Label Encoding assigns a unique integer to each category; One-Hot Encoding creates binary columns for each category.

4. **Why should duplicates be removed from datasets?**
   ➢ Duplicates can bias analysis and lead to inaccurate conclusions or models.

PRACTICAL NO: 2

Aim:

To implement a linear regression model using Python to forecast sales based on a given dataset, and to understand the relationship between the independent variable(s) and sales.

Theory:

Linear Regression is a supervised machine learning algorithm used to model the relationship between a dependent variable (target) and one or more independent variables (features) by fitting a linear equation to observed data.

The equation of simple linear regression is:

y=b0+b1x+ϵy = b_0 + b_1x + \epsilon

- **yy = dependent variable (sales)**
- **xx = independent variable (e.g., advertising spend)**
- **b0b_0 = intercept**
- **b1b_1 = slope (coefficient)**
- **ϵ\epsilon = error term**

Use case: In sales forecasting, linear regression predicts future sales based on factors like advertising spend, price, seasonality, etc.

Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Sample dataset: Advertising Spend vs Sales
data = {
    'Advertising_Spend': [1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000, 5500],
    'Sales': [20000, 24000, 27000, 30000, 32000, 36000, 39000, 42000, 45000, 47000]
}

df = pd.DataFrame(data)

# Features and target
X = df[['Advertising_Spend']]  # Independent variable
y = df['Sales']                # Dependent variable

# Split data into training and testing sets (80-20 split)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the model
model = LinearRegression()
```

```
model.fit(X_train, y_train)

# Predict on test set
y_pred = model.predict(X_test)

# Model evaluation
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Model Coefficient (Slope):", model.coef_[0])
print("Model Intercept:", model.intercept_)
print("Mean Squared Error:", mse)
print("R-squared Score:", r2)

# Plotting
plt.scatter(X, y, color='blue', label='Actual Sales')
plt.plot(X, model.predict(X), color='red', label='Regression Line')
plt.xlabel('Advertising Spend')
plt.ylabel('Sales')
plt.title('Linear Regression - Sales Forecasting')
plt.legend()
plt.grid(True)
plt.show()
```

Expected Output:

Model Coefficient (Slope): approximately 6.8

Model Intercept: approximately 13000

Mean Squared Error: small value (depends on data split)

R-squared Score: close to 0.95 or higher indicating good fit

- **A scatter plot showing actual sales data points (blue).**
- **A red regression line showing predicted sales based on advertising spend.**
- **The slope tells how much sales increase per unit increase in advertising spend.**
- **High R^2 indicates that the model explains most of the variability in sales.**

Viva Questions:

1. **What is the purpose of linear regression in sales forecasting?**
   ➢ To model and predict sales based on one or more input features, understanding their relationship.

2. **What does the R-squared value indicate?**
   ➢ It measures the proportion of variance in the dependent variable explained by the independent variables (goodness of fit).

3. **Why do we split the data into training and testing sets?**
   ➢ To train the model on one part and test its performance on unseen data to evaluate generalization.

4. **What assumptions does linear regression make?**
   ➢ Linearity, independence, homoscedasticity (constant variance of errors), normal distribution of errors.

PRACTICAL NO. 3

Aim:

To build and evaluate a classification model using Python to predict customer churn based on customer behavior and demographics data.

Theory:

Customer churn prediction involves identifying customers who are likely to stop using a company's product or service. This is a binary classification problem where:

- **Target variable: Churn (`Yes` or `No`)**
- **Features: Customer attributes like tenure, usage, contract type, payment method, etc.**

Common classification algorithms include Logistic Regression, Decision Trees, Random Forest, and Support Vector Machines.

The model helps businesses proactively engage customers likely to churn, improving retention and revenue.

Code:

Using a simplified sample dataset and Logistic Regression:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Sample dataset
data = {
   'Gender': ['Male', 'Female', 'Female', 'Male', 'Female', 'Male', 'Male', 'Female',
'Female', 'Male'],
   'Tenure': [5, 12, 7, 3, 15, 8, 10, 2, 14, 6],
   'MonthlyCharges': [70, 90, 60, 80, 95, 75, 85, 55, 100, 65],
   'Contract': ['Month-to-month', 'Two year', 'Month-to-month', 'One year', 'Two year',
'Month-to-month', 'One year', 'Month-to-month', 'Two year', 'One year'],
   'Churn': ['Yes', 'No', 'Yes', 'No', 'No', 'Yes', 'No', 'Yes', 'No', 'No']
}

df = pd.DataFrame(data)

# Encoding categorical variables
le_gender = LabelEncoder()
df['Gender_enc'] = le_gender.fit_transform(df['Gender'])

le_contract = LabelEncoder()
df['Contract_enc'] = le_contract.fit_transform(df['Contract'])

le_churn = LabelEncoder()
df['Churn_enc'] = le_churn.fit_transform(df['Churn'])

# Features and target
X = df[['Gender_enc', 'Tenure', 'MonthlyCharges', 'Contract_enc']]
y = df['Churn_enc']

# Split dataset into training and testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature scaling
scaler = StandardScaler()
```

```
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Build Logistic Regression model
model = LogisticRegression()
model.fit(X_train_scaled, y_train)

# Predictions
y_pred = model.predict(X_test_scaled)

# Evaluation
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("Accuracy:", accuracy_score(y_test, y_pred))
```

Expected Output:

```
Confusion Matrix:
[[1 0]
 [0 1]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2

Accuracy: 1.0
```

***Note*: Output may vary with different data splits; this is a small sample dataset for illustration.**

Viva Questions:

1. **What is customer churn prediction and why is it important?**
   ➢ Predicting which customers will leave helps businesses reduce churn and retain revenue.

2. **Why do we encode categorical variables in classification problems?**
   ➢ Machine learning models require numeric inputs, so categorical data must be converted to numeric formats.

3. **What metrics are important for evaluating a classification model?**
   ➢ Accuracy, precision, recall, F1-score, and confusion matrix.

4. **Why is feature scaling necessary before training some models?**
   ➢ To normalize the range of features, improving convergence speed and model performance.

PRACTICAL NO. 4

Aim:

To perform customer segmentation by applying the K-Means clustering algorithm on customer data, grouping similar customers based on their attributes for targeted marketing strategies.

Theory:

What is K-Means Clustering?

K-Means is an unsupervised machine learning algorithm used to divide data into K distinct clusters based on feature similarity. The algorithm works by:

1. **Randomly initializing K centroids.**
2. **Assigning each data point to the nearest centroid.**
3. **Recalculating centroids as the mean of assigned points.**
4. **Repeating steps 2 and 3 until centroids stabilize.**

Customer Segmentation:

- **Helps businesses group customers by purchasing behavior, demographics, or other attributes.**
- **Enables targeted marketing, personalized offers, and improved customer satisfaction.**

Choosing K:

- **The number of clusters (K) is often chosen using the Elbow Method, which plots the within-cluster sum of squares (WCSS) against different K values to find the optimal number.**

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Sample customer dataset
data = {
    'CustomerID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Annual Income (k$)': [15, 16, 17, 45, 50, 55, 80, 82, 85, 90],
    'Spending Score (1-100)': [39, 81, 6, 77, 40, 76, 6, 94, 3, 72]
}

df = pd.DataFrame(data)

# Select features for clustering
X = df[['Annual Income (k$)', 'Spending Score (1-100)']]

# Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# Using the Elbow Method to find optimal K
wcss = []
for i in range(1, 7):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

plt.plot(range(1, 7), wcss, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of clusters (K)')
plt.ylabel('WCSS')
plt.show()

# From the elbow plot, choose K=3 (for example)
k = 3
kmeans = KMeans(n_clusters=k, random_state=42)
clusters = kmeans.fit_predict(X_scaled)

# Assign clusters to original data
df['Cluster'] = clusters

print(df)
```

Expected Output:

- **Elbow Plot: A graph showing WCSS vs number of clusters, with an "elbow" at the optimal number of clusters (e.g., K=3).**
- **Cluster Assignment Table: Customer data with an additional column `Cluster` indicating the group number.**

Example output:

CustomerID	Annual Income (k$)	Spending Score (1-100)	Cluster
1	15	39	2
2	16	81	1
3	17	6	0
4	45	77	1
5	50	40	2
6	55	76	1
7	80	6	0
8	82	94	1
9	85	3	0
10	90	72	1

Viva Questions:

1. **What is the purpose of K-Means clustering in customer segmentation?**
   ➢ To group customers into distinct clusters based on similarity in behavior or attributes.

2. **How does the K-Means algorithm work?**
   ➢ It iteratively assigns points to the nearest centroid and updates centroids until convergence.

3. **What is the Elbow Method and how is it used?**
   ➢ It's a technique to determine the optimal number of clusters by plotting WCSS against K and identifying the "elbow" point.

4. **Why is feature scaling important before clustering?**
   ➢ To ensure features contribute equally to the distance calculation, preventing bias from variables with larger scales.

PRACTICAL NO: 5

Aim:

To perform Market Basket Analysis using the Apriori algorithm for discovering frequent itemsets and association rules from transactional data, helping businesses understand customer purchase behavior.

Theory:

Market Basket Analysis is a technique used in retail and marketing to identify items that frequently co-occur in transactions. It helps in strategies like product placement, cross-selling, and promotions.

The Apriori algorithm is a classic algorithm to mine frequent itemsets and generate association rules. It works on the principle that:

- **If an itemset is frequent, all its subsets must also be frequent (Apriori property).**

- **It uses a bottom-up approach, iteratively exploring larger itemsets by combining smaller frequent ones.**

Key terms:

- **Support: Frequency of the itemset in the dataset.**
- **Confidence: Likelihood that a rule holds true.**
- **Lift: Measure of how much more likely item B is purchased when item A is purchased.**

Code:

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules

# Sample transactional data (list of transactions)
transactions = [
    ['Milk', 'Bread', 'Butter'],
    ['Beer', 'Bread'],
    ['Milk', 'Diapers', 'Beer', 'Eggs'],
    ['Bread', 'Butter', 'Diapers'],
    ['Milk', 'Bread', 'Butter', 'Beer'],
```

```
    ['Bread', 'Milk', 'Diapers', 'Butter'],
]

# Convert transactions to one-hot encoded DataFrame
te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)
df = pd.DataFrame(te_ary, columns=te.columns_)

# Apply Apriori algorithm to find frequent itemsets with min support of 0.5
frequent_itemsets = apriori(df, min_support=0.5, use_colnames=True)

print("Frequent Itemsets:")
print(frequent_itemsets)

# Generate association rules with min confidence of 0.7
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)

print("\nAssociation Rules:")
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

Expected Output:

Frequent Itemsets:

	support	itemsets
0	0.833	(Bread)
1	0.667	(Milk)
2	0.667	(Butter)
3	0.500	(Beer)
4	0.500	(Diapers)
5	0.500	(Bread, Butter)
6	0.500	(Milk, Bread)
7	0.500	(Milk, Butter)
8	0.500	(Bread, Diapers)

Association Rules:

	antecedents	consequents	support	confidence	lift
0	(Bread)	(Butter)	0.5	0.600000	0.900000
1	(Butter)	(Bread)	0.5	0.750000	1.350000
2	(Milk)	(Bread)	0.5	0.750000	1.350000
3	(Bread)	(Milk)	0.5	0.600000	0.900000
4	(Milk)	(Butter)	0.5	0.750000	1.125000
5	(Butter)	(Milk)	0.5	0.750000	1.125000

- **The frequent itemsets table shows combinations of products that appear together frequently.**

- **The association rules table shows strong relationships, where the presence of one item implies the presence of another with certain confidence and lift values.**

Viva Questions:

1. **What is Market Basket Analysis and how is it useful?**

   ➢ It analyzes customer purchase behavior to find associations between products, aiding in marketing and sales strategies.

2. **What does support, confidence, and lift mean in association rules?**

   ➢ Support: frequency of itemset; Confidence: probability of consequent given antecedent; Lift: measure of rule's effectiveness beyond random chance.

3. **Why is the Apriori algorithm efficient for finding frequent itemsets?**

   ➢ It prunes the search space using the Apriori property, reducing computation by ignoring itemsets with infrequent subsets.

4. **How can businesses use the results of Market Basket Analysis?**

   ➢ For product placement, cross-selling, promotions, and inventory management.

PRACTICAL NO: 6

Aim:

To build a simple recommendation engine that suggests items to users based on item similarity or user preferences using Python.

Theory:

A recommendation engine predicts user preferences and suggests items accordingly. There are two main types:

- **Content-Based Filtering: Recommends items similar to those a user liked in the past by analyzing item features.**
- **Collaborative Filtering: Recommends items based on the preferences of similar users.**

In this practical, we focus on a content-based recommendation system that uses item features and cosine similarity to recommend similar items.

Code:

```
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer

# Sample dataset of movies with genres
data = {
    'MovieID': [1, 2, 3, 4, 5],
    'Title': ['The Matrix', 'The Godfather', 'The Dark Knight', 'Pulp Fiction', 'The Shawshank Redemption'],
    'Genres': ['Action Sci-Fi', 'Crime Drama', 'Action Crime Drama', 'Crime Drama', 'Drama']
}
```

```
df = pd.DataFrame(data)

# Vectorize the 'Genres' using TF-IDF
tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(df['Genres'])

# Compute cosine similarity between movies
cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)

# Function to recommend movies based on a given title
def recommend_movies(title, cosine_sim=cosine_sim):
    idx = df.index[df['Title'] == title][0]  # Get index of the movie
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:4]  # Top 3 recommendations excluding the movie itself

    recommended_indices = [i[0] for i in sim_scores]
    return df['Title'].iloc[recommended_indices]

# Example: Recommend movies similar to 'The Godfather'
recommendations = recommend_movies('The Godfather')
print("Recommendations for 'The Godfather':")
print(recommendations)
```

Expected Output:

```
Recommendations for 'The Godfather':
3      Pulp Fiction
2     The Dark Knight
4    The Shawshank Redemption
Name: Title, dtype: object
```

- The engine recommends movies with similar genres to 'The Godfather'.
- It outputs the top 3 movies ranked by genre similarity.

Viva Questions:

1. **What is a recommendation engine and where is it used?**
   ➢ It predicts user preferences and suggests relevant items; used in e-commerce, streaming platforms, and social media.

2. **What is the difference between content-based and collaborative filtering?**
   ➢ Content-based uses item features, collaborative uses user behavior/preferences.

3. **Why do we use cosine similarity in content-based recommendation systems?**
   ➢ It measures similarity between feature vectors regardless of magnitude, suitable for text/vector data.

4. **How can the recommendation system be improved further?**
   ➢ By incorporating user ratings, hybrid methods, more features, and larger datasets.

PRACTICAL NO: 7

Aim:

To design and develop an interactive dashboard using Tableau or Power BI for visualizing key business metrics and enabling dynamic data exploration.

Theory:

An interactive dashboard consolidates multiple visualizations and KPIs into a single interface, allowing users to:

- **Monitor business performance**
- **Explore data dynamically through filters, slicers, and drill-downs**
- **Gain actionable insights efficiently**

Key Features of Dashboards:

- **Visualizations: Charts, graphs, tables, maps**
- **Interactivity: Filters, slicers, dropdowns, drill-through**
- **Data Connectivity: Live or imported datasets**
- **User-Friendly Layout: Clear, concise, and visually appealing**

Tableau vs Power BI:

- **Tableau: Strong in data visualization flexibility and aesthetics.**
- **Power BI: Deep Microsoft ecosystem integration and affordability.**

Both tools allow drag-and-drop interface to create dashboards with minimal coding.

Code / Steps:

(Note: Tableau and Power BI are primarily GUI-based tools, so coding is minimal. Here are the conceptual steps)

Steps to create an interactive dashboard:

1. **Connect to Data:**
    - **Import data from Excel, SQL Server, CSV, or cloud sources.**
2. **Data Preparation:**
    - **Clean and transform data using built-in tools.**
    - **Create calculated columns or measures if needed.**
3. **Create Visualizations:**
    - **Build charts like bar charts, line charts, pie charts, maps.**
    - **Use drag-and-drop fields to create visuals.**
4. **Add Interactivity:**
    - **Insert filters or slicers for users to select data subsets.**
    - **Enable drill-down on hierarchies (e.g., Year > Quarter > Month).**
    - **Add tooltips for detailed info on hover.**
5. **Arrange Dashboard:**
    - **Combine multiple visuals into one dashboard layout.**
    - **Adjust size and position for clarity and aesthetics.**

6. **Publish & Share:**
    - **Publish to Tableau Server, Tableau Public, or Power BI Service.**
    - **Share with stakeholders via web or mobile apps.**

Expected Output:

- **A dynamic dashboard displaying business KPIs such as sales trends, customer demographics, product performance.**
- **Users can interact with slicers/filters to customize views.**
- **Drill-down into data for detailed insights.**
- **Visuals update instantly based on user selections.**

Viva Questions:

1. **What is the purpose of an interactive dashboard?**
   ➢ To provide a consolidated, real-time view of key metrics with options for user-driven data exploration.
2. **How does interactivity enhance data visualization in dashboards?**
   ➢ It allows users to filter, drill down, and explore data dynamically, making insights more accessible.
3. **What types of data sources can Tableau and Power BI connect to?**
   ➢ Excel, CSV, SQL databases, cloud services (Azure, AWS), web APIs, and more.
4. **Explain the difference between calculated columns and measures in Power BI/Tableau.**
   ➢ Calculated columns are computed row-wise during data load; measures are aggregations calculated on demand based on filters.

PRACTICAL NO: 8

Aim:

To integrate the predictions or insights generated by a machine learning (ML) model into a Business Intelligence (BI) dashboard (such as Power BI or Tableau) for enhanced data-driven decision-making.

Theory:

ML models generate predictions or classifications based on historical data. Integrating these results into BI dashboards allows business users to:

- Visualize model predictions alongside traditional KPIs
- Monitor model performance in real time
- Use predictions for forecasting, risk assessment, and strategy planning

Integration Workflow:

1. **Train ML Model:** Build and train the model using Python, R, or other tools.
2. **Generate Predictions:** Apply the model to new or existing data to produce output (e.g., sales forecast, churn probability).
3. **Store Results:** Save predictions in a database, CSV, or cloud storage.
4. **Connect BI Tool:** Import the predictions dataset into Power BI/Tableau.
5. **Build Dashboard:** Visualize predictions with charts, filters, and interactive elements.
6. **Automate Updates:** Schedule model retraining and data refresh for up-to-date insights.

Code Example:

Here's a simplified Python example of training a model, generating predictions, and exporting them for BI dashboard use.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# Sample data: Customer info with churn flag
data = {
   'CustomerID': [1, 2, 3, 4, 5],
   'Tenure': [12, 5, 8, 22, 10],
   'MonthlyCharges': [70, 80, 65, 90, 60],
   'Churn': [0, 1, 0, 0, 1]
}
df = pd.DataFrame(data)

# Features and target
X = df[['Tenure', 'MonthlyCharges']]
y = df['Churn']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)

# Train model
model = LogisticRegression()
model.fit(X_train, y_train)

# Generate predictions (probability of churn)
df['Churn_Prob'] = model.predict_proba(df[['Tenure', 'MonthlyCharges']])[:, 1]

# Export predictions to CSV for BI tool
df.to_csv('customer_churn_predictions.csv', index=False)

print("Predictions saved to 'customer_churn_predictions.csv'")
```

Expected Output:

`customer_churn_predictions.csv` content:

CustomerID	Tenure	MonthlyCharges	Churn	Churn_Prob
1	12	70	0	0.22
2	5	80	1	0.78
3	8	65	0	0.33
4	22	90	0	0.15
5	10	60	1	0.70

- This file can be imported into Power BI/Tableau as a data source.
- Visualize churn probabilities with color coding, filters, or alerts.
- Combine with other business data to enhance insights.

Viva Questions:

1. **Why integrate ML model results into a BI dashboard?**
   ➤ To provide actionable insights by combining predictive analytics with business metrics in a single interface.

2. **What formats can be used to export ML predictions for BI tools?**
   ➤ CSV, Excel, databases (SQL), APIs, or cloud storage connectors.

3. **How can you automate the integration and updating of ML results in BI dashboards?**
   ➤ By scheduling model retraining, batch predictions, and refreshing data connections in the BI tool.

4. **What are some challenges faced when integrating ML with BI dashboards?**
   ➤ Data latency, model interpretability, data security, and ensuring synchronization between model updates and dashboard refreshes.

PRACTICAL NO: 9

Aim:

To build and execute a data analytics workflow using KNIME Analytics Platform, performing data loading, preprocessing, analysis, and visualization without extensive coding.

Theory:

KNIME (Konstanz Information Miner) is an open-source, graphical platform for data analytics, reporting, and integration. It allows users to create visual workflows by connecting nodes that represent different steps like data input, transformation, machine learning, and output.

Key Features:

- **Drag-and-drop interface: Design workflows visually.**
- **Node-based: Each node performs a specific task (e.g., reading data, filtering, modeling).**
- **Integration: Supports Python, R, SQL, and various data sources.**
- **Extensibility: Offers a wide range of built-in and community extensions.**
- **Automation: Workflows can be scheduled and reused.**

Typical Workflow Steps in KNIME:

1. **Data Import: Load data from files or databases.**
2. **Data Cleaning: Handle missing values, duplicates, or outliers.**
3. **Data Transformation: Normalize, filter, or create new features.**
4. **Modeling: Apply machine learning algorithms.**

5. **Evaluation: Assess model performance.**
6. **Visualization: Generate plots or export reports.**

Code (Conceptual/Node Setup in KNIME):

KNIME workflows are visual, but you can use scripting nodes if needed.

Example workflow nodes to implement:

- **File Reader: Load CSV or Excel dataset.**
- **Missing Value: Handle or impute missing data.**
- **Normalizer: Scale numeric data.**
- **Partitioning: Split data into training and testing sets.**
- **Decision Tree Learner: Train a decision tree model.**
- **Decision Tree Predictor: Predict using the model.**
- **Scorer: Evaluate prediction accuracy.**
- **Table View/Scatter Plot: Visualize data and results.**

Expected Output:

- **A KNIME workflow that visually shows connected nodes for each step.**
- **Cleaned and preprocessed data ready for modeling.**
- **A trained model with evaluation metrics like accuracy or ROC curve.**
- **Interactive visualizations such as scatter plots or bar charts inside KNIME.**
- **Exported reports or datasets for further use.**

Viva Questions:

1. **What are the advantages of using KNIME for data analytics?**
   ➢ It provides a visual, code-free interface, supports integration with many tools, and is highly extensible.

2. **How does KNIME handle missing values and data preprocessing?**
   ➢ Through dedicated nodes like Missing Value node, Normalizer, and other transformation nodes.

3. **Can you integrate Python or R scripts within KNIME workflows? How?**
   ➢ Yes, using Python Script and R Snippet nodes to run custom code within workflows.

4. **How does KNIME facilitate reproducibility and automation of analytics tasks?**
   ➢ Workflows can be saved, shared, scheduled, and executed repeatedly ensuring consistency.

PRACTICAL NO: 10

Aim:

To analyze and implement an AI solution in business that adheres to ethical guidelines, ensuring fairness, transparency, accountability, and privacy while delivering effective outcomes.

Theory:

With AI's growing influence in business decisions, ethical considerations have become paramount to avoid biases, discrimination, and misuse of data.

Key Principles of Ethical AI:

- **Fairness: Avoid biases and ensure equitable treatment across demographics.**
- **Transparency: Models and decisions should be explainable and understandable.**
- **Accountability: Stakeholders should be responsible for AI outcomes.**
- **Privacy: Protect user data and comply with data protection laws (e.g., GDPR).**
- **Safety: AI should not cause harm to users or society.**

Business Context:

Imagine a bank using AI for loan approvals. Ethical AI ensures the model does not discriminate based on race, gender, or socioeconomic status, maintains customer privacy, and explains decisions clearly.

Code (Conceptual Example):

Here's a Python snippet demonstrating how to check for bias in a classification model and ensure fairness using `AI Fairness 360` toolkit by IBM.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from aif360.datasets import BinaryLabelDataset
from aif360.metrics import ClassificationMetric
from aif360.algorithms.preprocessing import Reweighing

# Load dataset (e.g., Adult income dataset)
data = pd.read_csv('adult.csv')

# Preprocessing: encode categorical variables, define features and label
features = ['age', 'education-num', 'hours-per-week']
protected_attribute = 'sex'  # Male/Female
label = 'income'  # >50K or <=50K

# Convert to BinaryLabelDataset for AIF360
dataset = BinaryLabelDataset(df=data, label_names=[label],
protected_attribute_names=[protected_attribute])

# Split data
train, test = dataset.split([0.7], shuffle=True)

# Apply Reweighing to mitigate bias
RW = Reweighing(unprivileged_groups=[{protected_attribute: 0}],
privileged_groups=[{protected_attribute: 1}])
RW.fit(train)
train_transformed = RW.transform(train)

# Train logistic regression on transformed data
X_train = train_transformed.features
y_train = train_transformed.labels.ravel()
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict on test set
X_test = test.features
y_test = test.labels.ravel()
```

```
y_pred = model.predict(X_test)

# Evaluate fairness metrics
metric = ClassificationMetric(test,
test.copy(deepcopy=True).copy(deep=True).set_predicted_labels(y_pred),
                              unprivileged_groups=[{protected_attribute: 0}],
                              privileged_groups=[{protected_attribute: 1}])

print("Disparate Impact:", metric.disparate_impact())
print("Statistical Parity Difference:", metric.statistical_parity_difference())
```

Expected Output:

Disparate Impact: 0.95
Statistical Parity Difference: -0.03

- **These values close to 1 and 0 respectively indicate reduced bias.**
- **The model is less discriminatory towards protected groups (here, sex).**
- **The business can confidently deploy this model ensuring fairness.**

Viva Questions:

1. **What is ethical AI and why is it important in business?**
   ➢ AI designed to ensure fairness, transparency, and privacy, preventing harm and discrimination in decision-making.

2. **How can bias in AI models be detected and mitigated?**
   ➢ Using fairness metrics and bias mitigation algorithms like reweighing, adversarial debiasing, or data augmentation.

3. **What role does transparency play in ethical AI?**
   ➢ It helps stakeholders understand and trust AI decisions, enabling accountability.

4. **How can businesses ensure user data privacy while implementing AI?**
   ➢ Through data anonymization, encryption, access controls, and compliance with regulations like GDPR.