VASANTDADA PATIL PRATISHTHAN'S COLLEGE OF ENGINEERING AND VISUAL ARTS

ISO 9001:2015 Certified Institute

Department of Information Technology

NBA Accredited Course (Dated 01/07/2024 to 30/06/2027)

LAB MANUAL OF

Business Intelligence Lab

Lab Code: ITL601

Class: TE Information Technology

Semester: VI (Rev-2019 'C' Scheme)

Lab Incharge	H.O.D
Mrs. Darshana Gajbhiye	Dr. Pradip Mane

College Vision

To provide an environment to educate, encourage and explore students by facilitating innovative research, entrepreneurship, opportunities and employability to achieve social and professional goals.

College Mission

To foster entrepreneurship & strengthen industry institute interaction to enhance career opportunities for the employability of students.

To encourage collaborations with industries and academic institutes in terms of projects & internships by creating area for Research and Development.

To build up appropriate moral and ethical skills and to promote holistic development of students through various academic, social and cultural activities

Department Vision

To impart quality education in the field of Information Technology to meet the challenging needs of the society and industry.

Department Mission

To provide quality education to students by including Problem Solving, Teamwork and Leadership Skills to achieve their goals in the field of Information Technology.

To develop skilled IT professionals with moral principles and empower them in lifelong learning.

To educate students for global development including entrepreneurship, employability and the ability to apply technology to real life problems.

Program Educational Objectives (PEO)

Graduates will be successful with sound foundation in engineering fundamentals, trending technologies and entrepreneurship.

Graduates will be able to identify and solve real world problems.

Graduates will become ingenious and responsible citizens by demonstrating ethics with nurtured professional attitude

Program Outcomes (POs)

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identity, formulate complex engineering problems reaching substantiated conclusions using principles of Computer Engineering.
3. **Design / development of solutions:** Design / develop solutions for complex engineering problems and design system components or processes that meet the specified needs with

appropriate consideration for the society.

4. **Conduct investigations of complex problems:** Use knowledge for the design of experiments, analysis, interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select and apply appropriate techniques and modern engineering tools, including predictions and modeling to complex engineering activitieswith an understanding of the limitations.
6. **The engineer and society:** Apply the knowledge to assess social issues and the responsibilities relevant to engineering practices.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in social and environmental contexts, and demonstrate the knowledge for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilitiesand norms of the engineering practice.
9. **Individual and teamwork:** Function effectively as an individual, and as a member or leaderin diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively such as being able to comprehend and write effective reports and design documentation, make effective presentations.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management skills and apply the skills to manage projects effectively.
12. **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

Program Specific Outcomes (PSO)

Develop efficient IT based solutions by applying and integrating various domains like Artificial Intelligence, IoT, Computer Networks and Security to solve real time problems.
Apply technical knowledge in the field of Information Technology to achieve successful career and to pursue higher studies for future endeavors.
change.

Lab Objectives

The Lab experiments aims:

1. To introduce the concept of data Mining as an important tool for enterprise data management and as a cutting-edge technology for building competitive advantage
2. To enable students to effectively identify sources of data and process it for data mining
3. To make students well versed in all data mining algorithms, methods, and tools.
4. To learn how to gather and analyze large sets of data to gain useful business understanding.
5. To impart skills that can enable students to approach business problems analytically by identifying opportunities to derive business value from data.
6. To identify and compare the performance of business.

Lab Outcomes

On successful completion, of course, learner/student will be able to:

1. Identify sources of Data for mining and perform data exploration
2. Organize and prepare the data needed for data mining algorithms in terms of attributes and class inputs, training, validating, and testing files
3. Implement the appropriate data mining methods like classification, clustering or association mining on large data sets using open-source tools like WEKA
4. Implement various data mining algorithms from scratch using languages like Python/ Java etc.
5. Evaluate and compare performance of some available BI packages .Apply BI to solve practical problems: Analyze the problem domain, use the data collected in enterprise apply the appropriate data mining technique, interpret and visualize the results and provide decision support

Prerequisite: Object oriented Concept, Java programming language, Python

Hardware & Software Requirements:

Hardware Requirements	**Software Requirements**
PC i3 processor and above	Open source data mining and BI tools like WEKA, Rapid Miner, Pentaho

Lab /syllabus:

Sr. No.	**Module**	**Detailed Content**	**Hours**	**LO Mapping**
1	I	Tutorial on a) Design Star and Snowflake Schema	02	LO1
2	II	Implement using tools or languages like JAVA/ python/R a) Data Exploration b) Data preprocessing	04	LO2
3	III	Implement and evaluate using languages like JAVA/ python/R a) Classification Algorithms b) Clustering Algorithms c) Frequent Pattern Mining Algorithms	06	LO4
4	IV	Perform and evaluate using any open-source tools a) Classification Algorithms b) Clustering Algorithms c) Frequent Pattern Mining Algorithms	04	LO3
5	V	Detailed case study of any one BI tool such as Pentaho, Tableau and QlikView	04	LO5
6	VI	Business Intelligence Mini Project: Each group assigned one new case study for this A BI report must be prepared outlining the following steps: a) Problem definition, identifying which data mining task is needed b) Identify and use a standard data mining dataset available for the problem. Some links for data mining datasets are: WEKA, Kaggle, KDD cup, Data Mining Cup, UCI Machine Learning Repository etc. c) Implement appropriate data mining algorithm d) Interpret and visualize the results	06	LO6

Textbooks:

1. Han, Kamber, "Data Mining Concepts and Techniques", Morgan Kaufmann 3nd Edition.
2. G. Shmueli, N.R. Patel, P.C. Bruce, “Data Mining for Business Intelligence: Concepts, Techniques, and Applications in Microsoft Office Excel with XLMiner”, 1st Edition, Wiley India.
3. Paulraj Ponniah “Data Warehousing Fundamentals: A Comprehensive Guide for IT Professionals” Wiley Publications.

References:

1. P. N. Tan, M. Steinbach, Vipin Kumar, “Introduction to Data Mining”, Pearson Education
2. WEKA, RapidMiner Pentaho resources from the Web.
3. https://www.kaggle.com/learn/overview
4. Python for Data Science https://onlinecourses.nptel.ac.in/noc21_cs33/preview

List of Experiments

Sr. No	Experiment Title	Lab Outcome
1	Tutorial on design Star schema and Snowflake schema	LO1
2	Implement Data Exploration using tools or languages like JAVA/ python/R	LO2
3	Implement Data preprocessing using tools or languages like JAVA/ python/R	LO2
4	Perform and evaluate Classification Algorithms using any open-source tools	LO3
5	Implement and evaluate Classification Algorithms using languages like JAVA/ python/R	LO4
6	Perform and evaluate Clustering Algorithms using any open-source tools	LO3
7	Implement and evaluate Clustering Algorithms using languages like JAVA/ python/R	LO4
8	Perform and evaluate Frequent Pattern Mining Algorithms using any open-source tools	LO3
9	Implement and evaluate Frequent Pattern Mining Algorithms using languages like JAVA/ python/R	LO4
10	Detailed case study of any one BI tool such as Pentaho, Tableau and QlikView	LO5
11	Business Intelligence Mini Project: Each group assigned one new case study for this A BI report must be prepared outlining the following steps: a) Problem definition, identifying which data mining task is needed b) Identify and use a standard data mining dataset available for the problem. Some links for data mining datasets are: WEKA, Kaggle, KDD cup, Data Mining Cup, UCI Machine Learning Repository etc. c) Implement appropriate data mining algorithm d) Interpret and visualize the results e) Provide clearly the BI decision that is to be taken as a result of mining	LO6

Experiment No. 1

Aim: Tutorial on design Star schema and Snowflake schema

Theory:

Schema is a logical description of the entire database. It includes the name and description of records of all record types including all associated data-items and aggregates. Much like a database, a data warehouse also requires to maintain a schema. A database uses relational model, while a data warehouse uses Star, Snowflake, and Fact Constellation schema. In this chapter, we will discuss the schemas used in a data warehouse.

Star Schema

- Each dimension in a star schema is represented with only one-dimension table.
- This dimension table contains the set of attributes.
- The following diagram shows the sales data of a company with respect to the four dimensions, namely time, item, branch, and location.

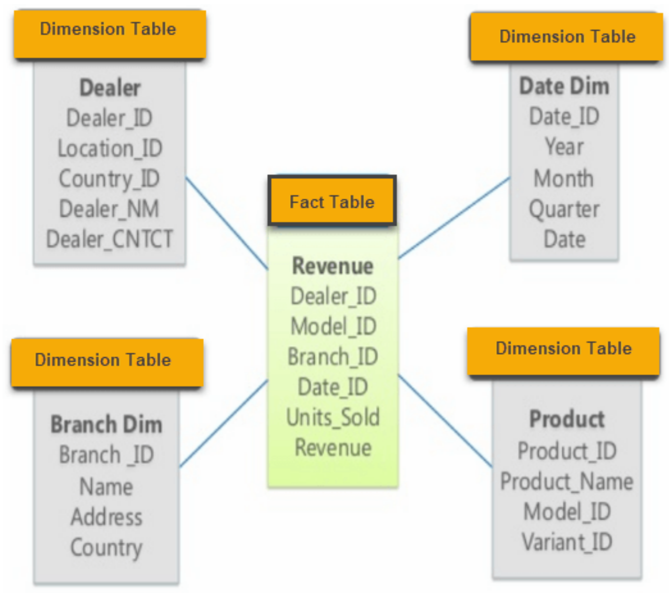


Fig. 1: Example of Star Schema Diagram

- There is a fact table at the center. It contains the keys to each of four dimensions.
- The fact table also contains the attributes, namely dollars sold and units sold.

Snowflake Schema

- Some dimension tables in the Snowflake schema are normalized.
- The normalization splits up the data into additional tables.
- Unlike Star schema, the dimensions table in a snowflake schema is normalized.
- For example, the item dimension table in star schema is normalized and split into two dimension tables, namely item and supplier table.

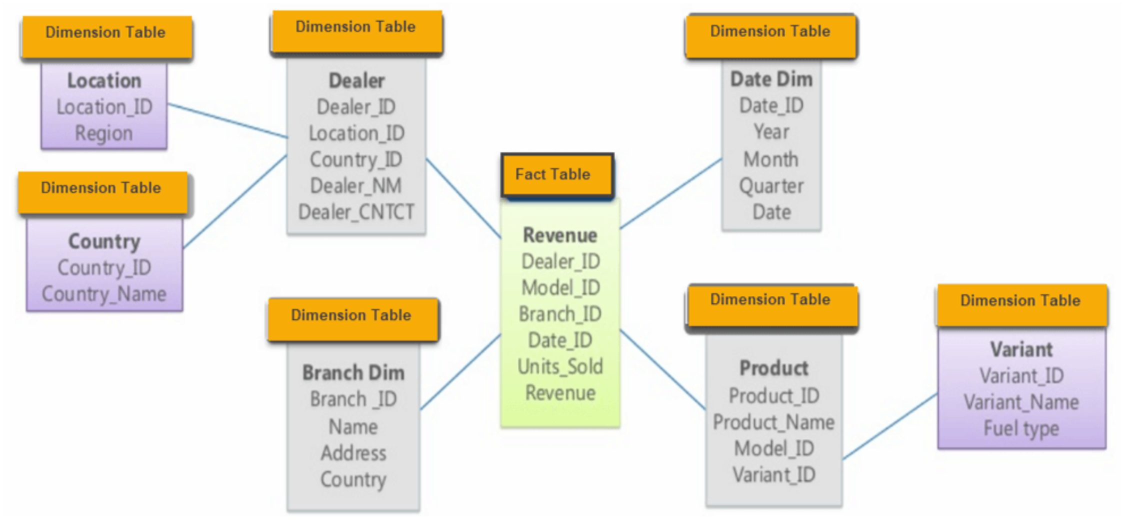


Fig. 2: Example of Snowflake Schema

- Now the item dimension table contains the attributes item_key, item_name, type, brand, and supplier-key.
- The supplier key is linked to the supplier dimension table. The supplier dimension table contains the attributes supplier_key and sup

Conclusion: We successfully studied star schema and snowflake schema and also design the schema

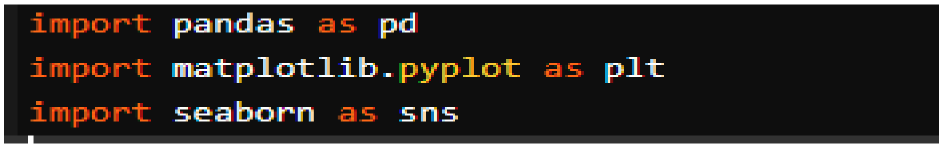
Experiment No. 2

Aim: Implement Data Exploration tools or languages like JAVA/ python/R

Solution: To perform data exploration using Python, we can use various libraries and tools available in the Python ecosystem. For this example, we will use the pandas library for data manipulation and exploration.

Data Exploration:

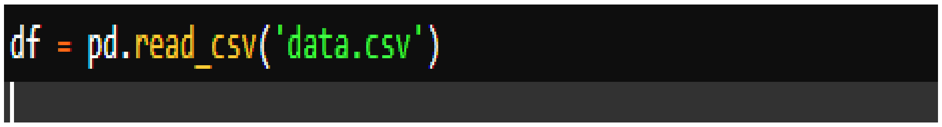
Step 1: Import the required libraries:



```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Step 2: Load the dataset:

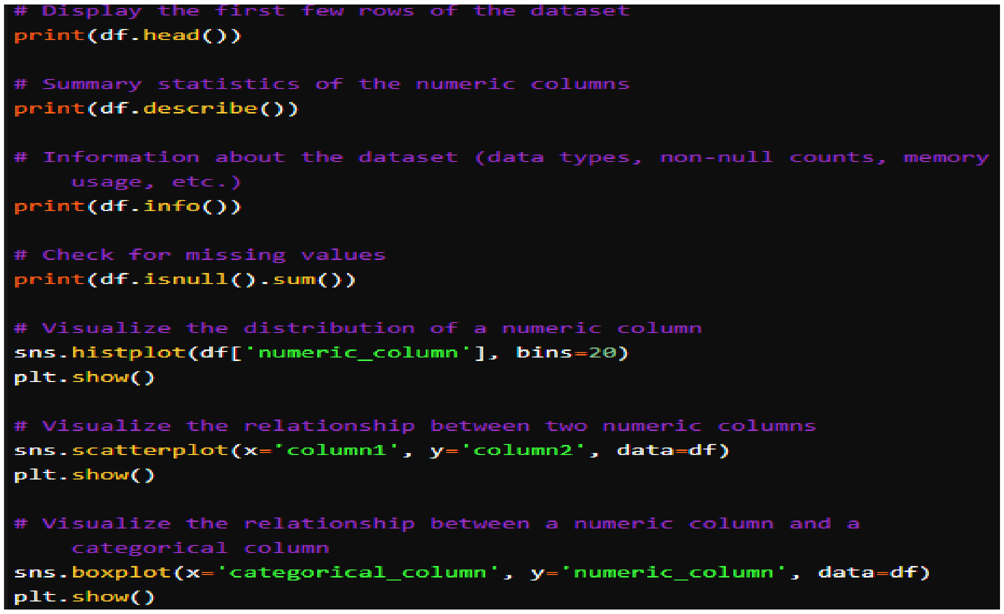
Assuming you have a dataset in a CSV file named "data.csv," load it using pandas.



```python
df = pd.read_csv('data.csv')
```

Step 3: Explore the dataset:

Use various pandas functions to gain insights into the data.



```python
# Display the first few rows of the dataset
print(df.head())

# Summary statistics of the numeric columns
print(df.describe())

# Information about the dataset (data types, non-null counts, memory
    usage, etc.)
print(df.info())

# Check for missing values
print(df.isnull().sum())

# Visualize the distribution of a numeric column
sns.histplot(df['numeric_column'], bins=20)
plt.show()

# Visualize the relationship between two numeric columns
sns.scatterplot(x='column1', y='column2', data=df)
plt.show()

# Visualize the relationship between a numeric column and a
    categorical column
sns.boxplot(x='categorical_column', y='numeric_column', data=df)
plt.show()
```

Conclusion: Hence we implemented Data Exploration using python.

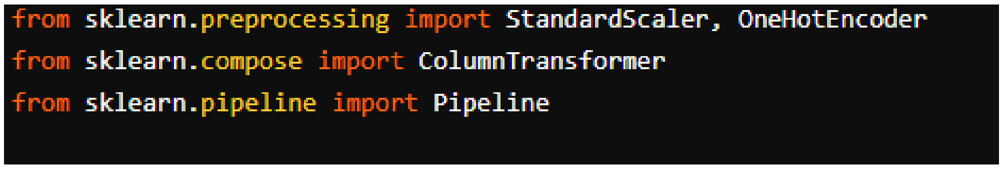
Experiment No. 3

Aim: Implement Data preprocessing using tools or languages like JAVA/ python/R

Solution: To perform data preprocessing using Python, we can use various libraries and tools available in the Python ecosystem. For this example, we will use the scikit-learn library for data preprocessing.

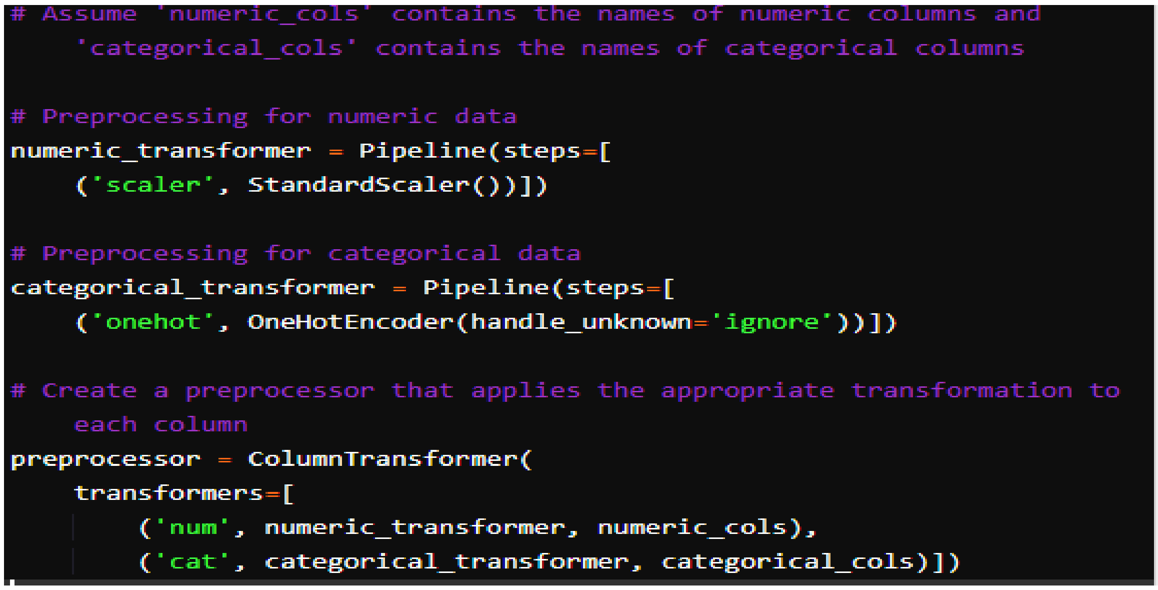
Data Preprocessing:

Step 1: Import the required libraries:



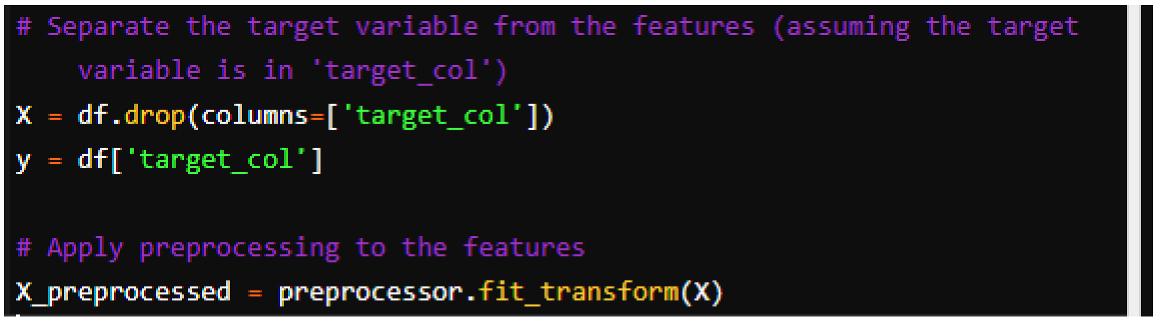
```
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
```

Step 2: Define the preprocessing steps:



```
# Assume 'numeric_cols' contains the names of numeric columns and
    'categorical_cols' contains the names of categorical columns

# Preprocessing for numeric data
numeric_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())])

# Preprocessing for categorical data
categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))])

# Create a preprocessor that applies the appropriate transformation to
    each column
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_cols),
        ('cat', categorical_transformer, categorical_cols)])
```

Step 3: Apply preprocessing to the dataset:



```
# Separate the target variable from the features (assuming the target
    variable is in 'target_col')
X = df.drop(columns=['target_col'])
y = df['target_col']

# Apply preprocessing to the features
X_preprocessed = preprocessor.fit_transform(X)
```

Now you have a preprocessed version of the dataset, X_preprocessed, which can be used for machine learning tasks.

Conclusion: Hence we implemented Data Exploration and Data preprocessing using python.

Experiment No. 4

Aim: Perform and evaluate Classification Algorithms using any open-source tools

Solution: To perform and evaluate classification algorithms using an open-source tool, we'll use the "Weka" (Waikato Environment for Knowledge Analysis) software. Weka is a popular and powerful tool for machine learning and data mining tasks, including classification.

In this example, we'll use Weka to classify the famous Iris dataset using the J48 decision tree algorithm.

Step 1: Download and Install Weka
Visit the Weka website (https://www.cs.waikato.ac.nz/ml/weka/) and download the latest version of the software. Install it following the provided instructions for your operating system.

Step 2: Launch Weka Explorer
Open Weka and click on the "Explorer" button to launch the GUI interface.

Step 3: Load the Dataset
In the Weka Explorer, click on the "Open file" button and navigate to the "iris.arff" dataset. The Iris dataset is a sample dataset that comes with Weka, and you can find it in the "data" folder within the Weka installation directory.

Step 4: Choose Classification Algorithm (J48)
Click on the "Classify" tab in the Weka Explorer.
Under the "Choose" section, select "trees" and then "J48."

Step 5: Configure Options
You can configure options for the J48 algorithm, such as pruning and confidence factor. For this example, use the default values.

Step 6: Evaluate the Model
Click on the "Start" button to run the J48 algorithm on the Iris dataset.
Weka will perform cross-validation on the dataset and display evaluation results, including accuracy, precision, recall, F1-score, and the confusion matrix.

Step 7: Analyze the Results
Examine the evaluation metrics to understand the performance of the J48 decision tree model on the Iris dataset. These metrics will help you assess the model's ability to classify the iris flowers correctly.

You can experiment with other classification algorithms available in Weka, such as k-Nearest Neighbors (kNN), Naive Bayes, Random Forest, Support Vector Machines (SVM), etc. Weka provides a user-friendly interface for trying different algorithms, configuring options, and analyzing their results. Additionally, you can perform hyper parameter tuning to optimize the models and improve their performance further.

Conclusion: Hence we perform and evaluated Classification Algorithms using WEKA open-source tool.

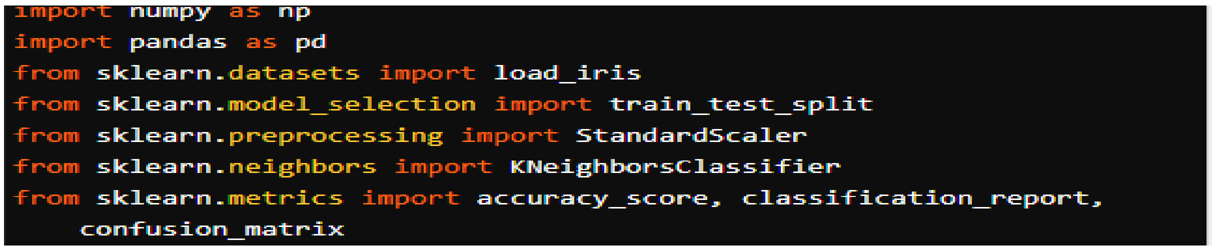
Experiment No. 5

Aim: Implement and evaluate Classification Algorithms using languages like JAVA/ python/R

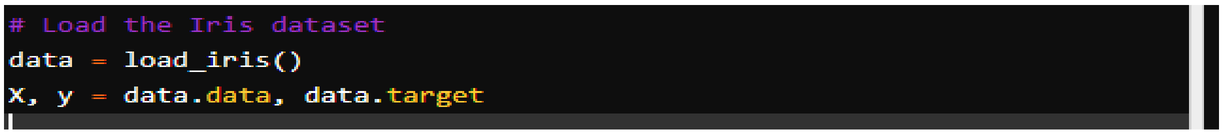
Solution: To implement and evaluate classification algorithms in Python, we'll use the scikit-learn library, which provides a wide range of classification algorithms and evaluation metrics. We'll follow these steps:

1. Import necessary libraries and load the dataset.
2. Pre-process the data (if needed).
3. Split the data into training and testing sets.
4. Choose a classification algorithm and train the model using the training data.
5. Evaluate the model using the testing data.
6. Analyze the results using evaluation metrics.

Step 1: Import libraries and load the dataset



```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report,
    confusion_matrix
```

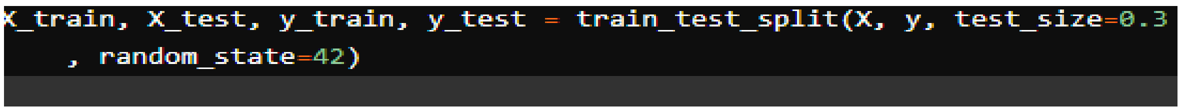


```
# Load the Iris dataset
data = load_iris()
X, y = data.data, data.target
```

Step 2: Pre-process the data (if needed)

In this case, we'll skip data pre-processing as the Iris dataset is well-prepared and does not contain missing values.

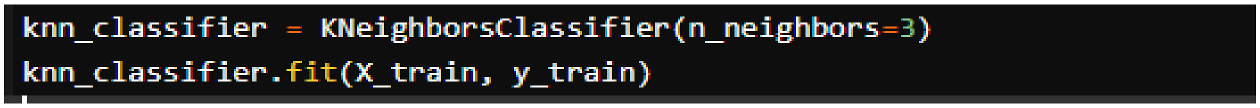
Step 3: Split the data into training and testing sets



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3
    , random_state=42)
```

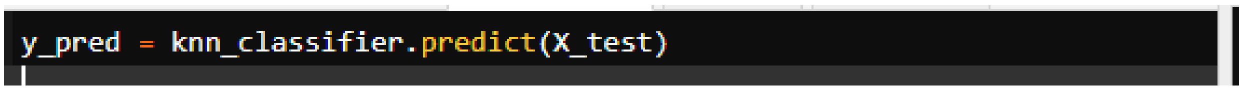
Step 4: Choose a classification algorithm and train the model

For this example, we'll use the K-Nearest Neighbors (KNN) algorithm with k=3.



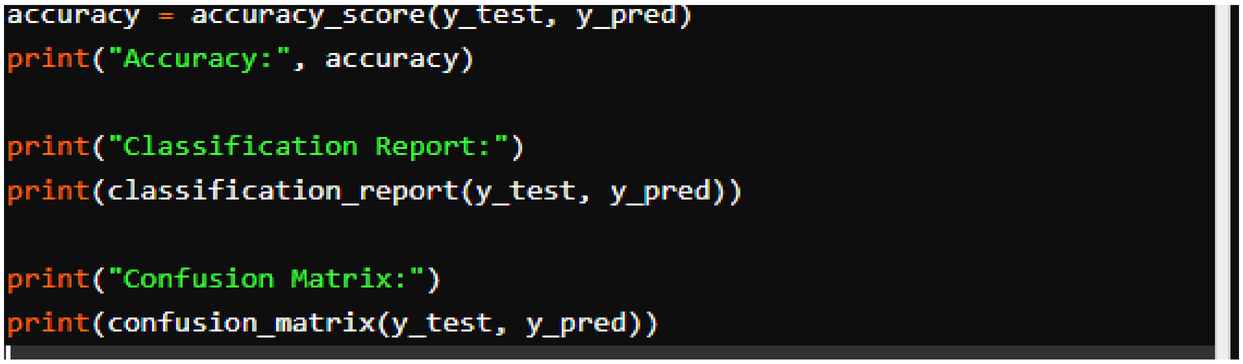
```
knn_classifier = KNeighborsClassifier(n_neighbors=3)
knn_classifier.fit(X_train, y_train)
```

Step 5: Evaluate the model using the testing data



```
y_pred = knn_classifier.predict(X_test)
```

Step 6: Analyze the results using evaluation metrics



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

print("Classification Report:")
print(classification_report(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

The above code will output the accuracy, classification report (which includes precision, recall, and F1-score for each class), and the confusion matrix (which shows the count of true positive, true negative, false positive, and false negative predictions).

You can repeat the process with other classification algorithms available in scikit-learn, such as Logistic Regression, Decision Trees, Random Forests, Support Vector Machines, etc. By comparing the results of different algorithms, you can choose the one that best fits your dataset and problem. Additionally, you can perform hyper parameter tuning using techniques like Grid Search or Randomized Search to optimize the model's performance further.

Conclusion: Hence we implemented Classification Algorithms using python.

Experiment No. 6

Aim: Perform and evaluate Clustering Algorithms using any open-source tools

Solution: To perform clustering using open-source tools like Weka, you can use the "SimpleKMeans" algorithm, which is a popular clustering algorithm available in Weka. Here's a step-by-step guide on how to perform and evaluate clustering using Weka:

1. Install Weka: If you haven't already installed Weka, you can download it from the official website: https://www.cs.waikato.ac.nz/ml/weka/

2. Launch Weka: Open Weka and click on the "Explorer" tab, which is used for data preprocessing, modeling, and evaluation.

3. Load your dataset: Click on the "Open file..." button in the "Preprocess" panel to load your dataset into Weka. Make sure your dataset is in a compatible format, such as ARFF.

4. Select the SimpleKMeans algorithm: In the "Cluster" panel, click on the "Choose" button next to the "Clusterer" dropdown menu. Then, navigate to "weka.clusterers" and choose "SimpleKMeans" from the list.

5. Configure SimpleKMeans settings: Click on the "SimpleKMeans" option in the "Choose a clusterer" window to access the settings. You can configure various parameters like the number of clusters (k) and distance metric.

6. Set the number of clusters (k): Adjust the "Number of clusters" parameter to set the desired number of clusters in your dataset. This is an essential hyperparameter that determines the number of clusters the algorithm will generate.

7. Choose a distance metric (optional): You can also choose a distance metric to measure the similarity between instances in your dataset. The default is Euclidean distance, but Weka supports other distance metrics like Manhattan distance, etc.

8. Run the SimpleKMeans algorithm: Once you have configured the SimpleKMeans settings, click the "OK" button to close the settings window. Then, click on the "Start" button in the "Cluster" panel to run the SimpleKMeans algorithm on your dataset.

9. Evaluate the clustering results: After the SimpleKMeans algorithm finishes running, you can evaluate the clustering results. Weka provides various options for visualizing and interpreting the results.

10. Visualize the clusters: In the "Cluster" panel, click on the "Visualize cluster assignments" button to see a visualization of the clustered data. This will plot the instances with different colors representing different clusters.

Evaluate clustering performance (optional): You can evaluate the clustering performance using internal evaluation metrics like Silhouette index or external evaluation metrics like Adjusted Rand Index (ARI) if you have access to the ground-truth labels. Weka provides the "Clusterer performance Evaluator" in the "Cluster" panel for this purpose.

Conclusion: Hence we performed Clustering Algorithms using open-source tool WEKA.

Experiment No. 7

Aim: Implement and evaluate Clustering Algorithms using languages like JAVA/ python/R

Solution: To implement and evaluate clustering algorithms in Python, we will use the scikit-learn library, which offers a wide range of clustering algorithms and evaluation metrics. In this example, we'll use the K-Means clustering algorithm and evaluate its performance using the Silhouette Score.

Step 1: Import necessary libraries and load the dataset.

Step 2: Preprocess the data (if needed).

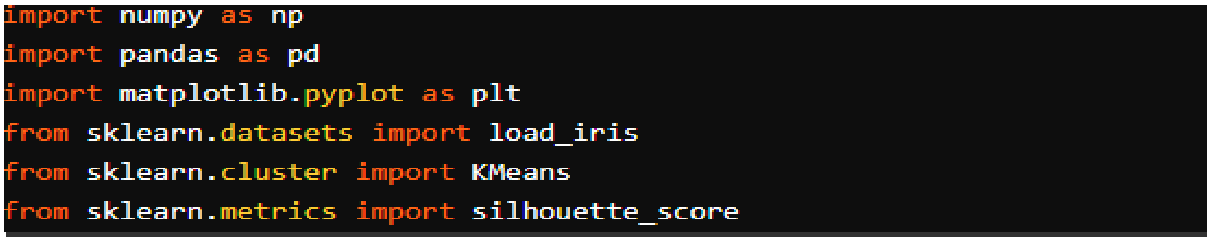
Step 3: Choose the clustering algorithm and fit the model to the data.

Step 4: Evaluate the clustering using the Silhouette Score.

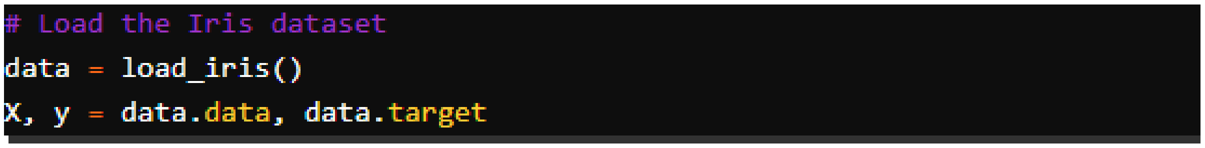
Step 5: Visualize the clustering results (optional).

For this example, we'll use the Iris dataset, a well-known dataset for clustering and classification tasks.

Step 1: Import libraries and load the dataset



```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```



```python
# Load the Iris dataset
data = load_iris()
X, y = data.data, data.target
```

Step 2: Preprocess the data (if needed)

In this case, we'll skip data preprocessing as the Iris dataset is well-prepared and does not contain missing values.

Step 3: Choose the clustering algorithm and fit the model to the data

```
# Create a KMeans clustering model with 3 clusters (since there are 3
    classes in the Iris dataset)
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)
```

Step 4: Evaluate the clustering using the Silhouette Score

```
# Predict the cluster labels for each data point
cluster_labels = kmeans.predict(X)

# Calculate the Silhouette Score
silhouette_avg = silhouette_score(X, cluster_labels)
print("Silhouette Score:", silhouette_avg)
```

Step 5: Visualize the clustering results (optional)

For visualization, we'll use only the first two features of the Iris dataset (sepal length and sepal width) and color the data points based on their cluster assignments.

```
# Plot the data points with their assigned cluster colors
plt.scatter(X[:, 0], X[:, 1], c=cluster_labels, cmap='viridis')
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.title('K-Means Clustering')
plt.show()
```

This will display a scatter plot with data points colored based on their assigned clusters. Note that we used only two features for visualization; you can choose different features or use dimensionality reduction techniques like PCA to visualize the clustering results in higher dimensions.

You can repeat the process with other clustering algorithms available in scikit-learn, such as Agglomerative Clustering, DBSCAN, Mean Shift, etc. By comparing the results of different algorithms and evaluation metrics, you can choose the one that best fits your dataset and problem. Additionally, you can experiment with different numbers of clusters to find the optimal value for your specific dataset.

Conclusion: Hence we implemented Clustering Algorithms using python.

Experiment No. 8

Aim: Perform and evaluate Frequent Pattern Mining Algorithms using any open-source tools

Solution: To perform frequent pattern mining using open-source tools like Weka, you can use the "Apriori" algorithm, which is also available in Weka. Here's a step-by-step guide on how to perform and evaluate frequent pattern mining using Weka:

1. Install Weka: If you haven't already installed Weka, you can download it from the official website: https://www.cs.waikato.ac.nz/ml/weka/
2. Launch Weka: Open Weka and click on the "Explorer" tab, which is used for data preprocessing, modeling, and evaluation.
3. Load your dataset: Click on the "Open file..." button in the "Preprocess" panel to load your dataset into Weka. Make sure your dataset is in a compatible format, such as ARFF.
4. Select the Apriori algorithm: In the "Classify" panel, click on the "Choose" button next to the "Classifier" dropdown menu. Then, navigate to "weka.associations" and choose "Apriori" from the list.
5. Configure Apriori settings: Click on the "Apriori" option in the "Choose a filter" window to access the settings. You can configure various parameters like the minimum support and minimum confidence thresholds for generating frequent itemsets and association rules.
6. Set minimum support and confidence: Adjust the "Min Support" and "Min Metric" parameters to set the minimum support and minimum confidence thresholds, respectively. These values define the level of support and confidence required for an itemset or association rule to be considered frequent.
7. Run the Apriori algorithm: Once you have configured the Apriori settings, click the "OK" button to close the settings window. Then, click on the "Start" button in the "Classify" panel to run the Apriori algorithm on your dataset.
8. Evaluate the results: After the Apriori algorithm finishes running, you can analyze the generated frequent itemsets and association rules. Weka provides various options for visualizing and interpreting the results, such as the "Association rules" tab in the "Classify" panel.
9. Interpretation: Examine the support and confidence values of the frequent itemsets and association rules. You can also assess the interestingness of the discovered patterns based on your domain knowledge and objectives.

Please note that the exact steps and options may vary slightly depending on the version of Weka you are using. Additionally, make sure to preprocess your data appropriately before applying the Apriori algorithm for accurate results.

Conclusion: Hence we performed Frequent Pattern Mining Algorithms using open source tool WEKA

Experiment No. 9

Aim: Implement and evaluate Frequent Pattern Mining Algorithms using languages like JAVA/ python/R

Solution: Frequent Pattern Mining is an essential task in data mining and finding frequent item sets in a transactional dataset. In this example, I'll show you how to implement and evaluate the Apriori algorithm, one of the most popular algorithms for frequent pattern mining, using Python.

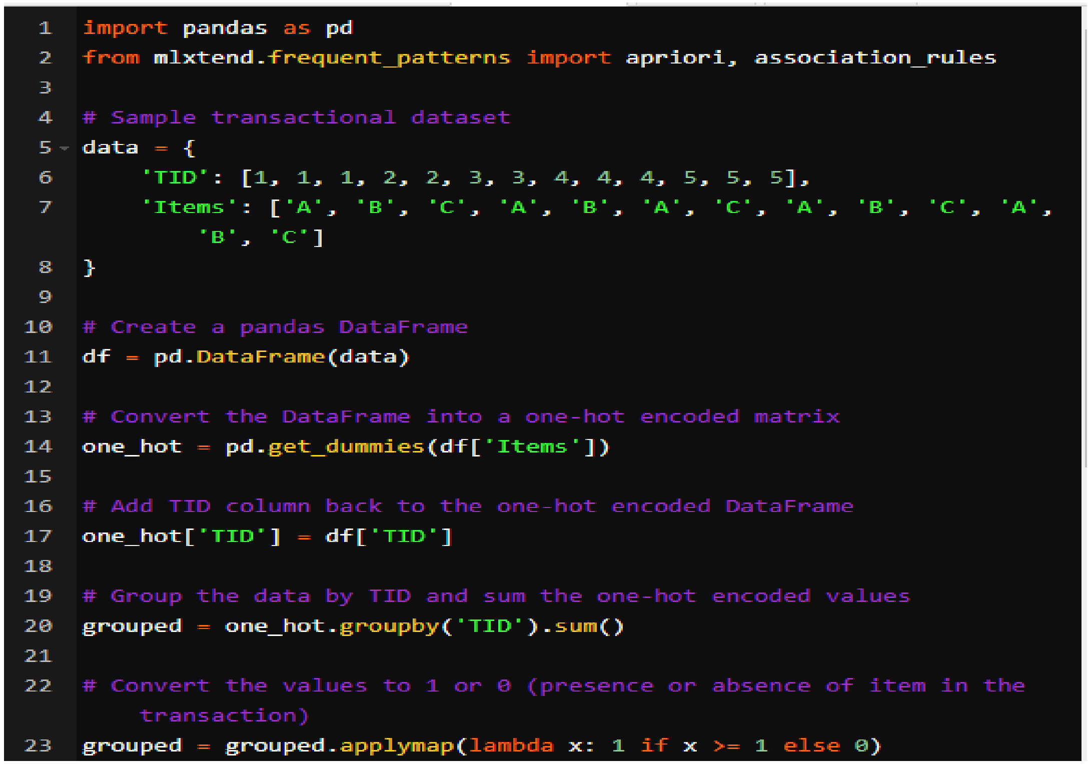
First, make sure you have Python and the required libraries installed:

1. Python 3.x
2. pandas (for handling data)
3. mlxtend (for Apriori algorithm implementation)

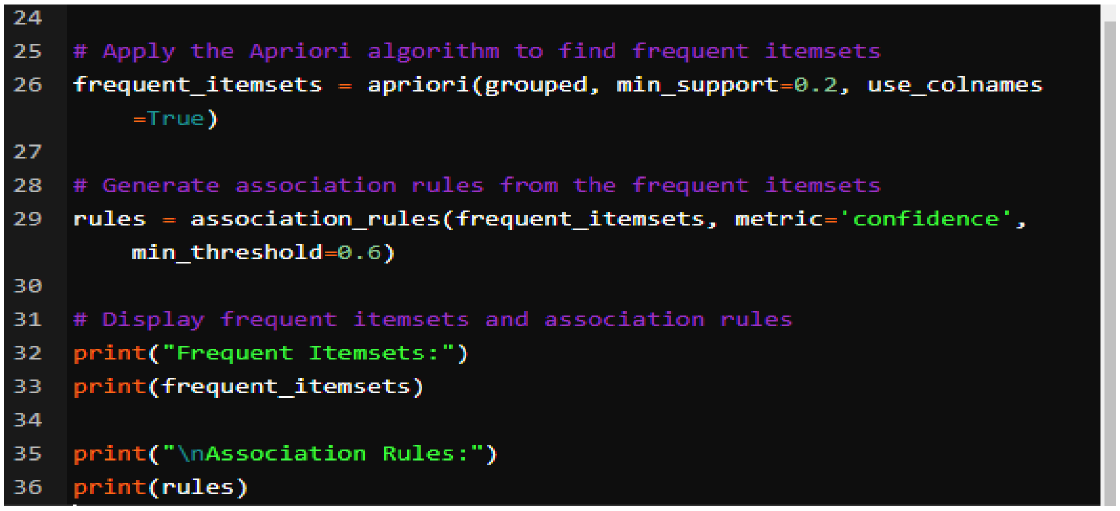
You can install the required libraries using pip:

pip install pandas mlxtend

Now, let's implement the Apriori algorithm and evaluate it using a sample dataset:



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

# Sample transactional dataset
data = {
    'TID': [1, 1, 1, 2, 2, 3, 3, 4, 4, 4, 5, 5, 5],
    'Items': ['A', 'B', 'C', 'A', 'B', 'A', 'C', 'A', 'B', 'C', 'A',
        'B', 'C']
}

# Create a pandas DataFrame
df = pd.DataFrame(data)

# Convert the DataFrame into a one-hot encoded matrix
one_hot = pd.get_dummies(df['Items'])

# Add TID column back to the one-hot encoded DataFrame
one_hot['TID'] = df['TID']

# Group the data by TID and sum the one-hot encoded values
grouped = one_hot.groupby('TID').sum()

# Convert the values to 1 or 0 (presence or absence of item in the
    transaction)
grouped = grouped.applymap(lambda x: 1 if x >= 1 else 0)
```


```python

# Apply the Apriori algorithm to find frequent itemsets
frequent_itemsets = apriori(grouped, min_support=0.2, use_colnames
    =True)

# Generate association rules from the frequent itemsets
rules = association_rules(frequent_itemsets, metric='confidence',
    min_threshold=0.6)

# Display frequent itemsets and association rules
print("Frequent Itemsets:")
print(frequent_itemsets)

print("\nAssociation Rules:")
print(rules)
```

In this example, we use a sample transactional dataset with five transactions and three different items (A, B, and C). The Apriori algorithm is applied to find frequent itemsets with a minimum support of 0.2 (20%) and association rules with a minimum confidence of 0.6 (60%).

Feel free to use your own dataset or modify the support and confidence thresholds according to your requirements. The output will show the frequent itemsets and association rules along with various metrics like support, confidence, and lift for the association rules.

Conclusion: Hence we Implemented Frequent Pattern Mining Algorithms using python

Experiment No. 10

Aim: Detailed case study of any one BI tool such as Pentaho, Tableau and QlikView

Solution:
Let's take a detailed case study of Tableau, one of the leading business intelligence (BI) tools in the market. Tableau is known for its user-friendly interface, powerful visualization capabilities, and ease of data exploration.

Case Study: Using Tableau for Sales Analysis

Business Problem:
A retail company wants to analyze its sales data to gain insights into product performance, customer behavior, and regional sales trends. The company has transactional data containing information about sales orders, products, customers, and regions.

Solution with Tableau:

Step 1: Data Preparation
The first step is to prepare the data for analysis. The data might be stored in a relational database or a spreadsheet. Tableau supports various data sources like Excel, CSV, SQL databases, cloud-based storage, etc. The data needs to be cleaned, transformed, and structured properly for analysis.

Step 2: Connect to Data Source
Open Tableau Desktop and connect to the data source containing the sales data. Tableau will guide you through the process of connecting to the data and visualizing it.

Step 3: Create Visualizations
Once the data is connected, you can start creating visualizations. Drag and drop the required fields from the data pane to the workspace. Tableau will automatically generate visualizations based on the selected fields.

a) Sales Overview Dashboard:

Create a bar chart to visualize total sales by product category.
Add a line chart to show the trend of sales over time.
Create a map to display regional sales using geospatial data.
b) Customer Analysis Dashboard:

Create a scatter plot to analyze the relationship between customer age and total spending.
Build a pie chart to show the distribution of customers by gender.
Add a heat map to visualize the concentration of customers by region.
Step 4: Create Interactive Dashboards
Combine the individual visualizations into interactive dashboards. Dashboards allow users to explore

the data dynamically and gain insights in real-time. Use filters, parameters, and actions to make the dashboard interactive and user-friendly.

Step 5: Analyze and Gain Insights
Analyze the visualizations to gain insights into various aspects of the business, such as:

Which product categories contribute the most to overall sales?
Are there any seasonal trends in sales?
Which regions are performing well and which ones need improvement?
What is the average spending of different customer segments?
Step 6: Share and Collaborate
Publish the dashboards to Tableau Server or Tableau Online, allowing stakeholders to access the analysis and collaborate on the findings. This way, decision-makers can access the insights anytime, anywhere, and make data-driven decisions.

Step 7: Schedule Data Refresh (if required)
If the data is regularly updated, schedule a data refresh to keep the dashboards up to date with the latest information.

Conclusion: Hence we completed detailed case study of any one BI tool such as Pentaho, Tableau and QlikView

Experiment No. 11

Aim: Business Intelligence Mini Project

Solution: Each group assigned one new case study.
For this a BI report must be prepared outlining the following steps:

a) Problem definition, identifying which data mining task is needed
b) Identify and use a standard data mining dataset available for the problem. Some links for data mining datasets are: WEKA, Kaggle, KDD cup, Data Mining Cup, UCI Machine Learning Repository etc.
c) Implement appropriate data mining algorithm
d) Interpret and visualize the results
e) Provide clearly the BI decision that is to be taken as a result of mining

Conclusion: Hence mini project were successfully completed.