3.Demonstrate data exploration & data preprocessing.

```
import pandas as pd

# Sample data
data = {
        'Age': [25, 30, None, 40],
        'Salary': [50000, None, 52000, 80000]
}
df = pd.DataFrame(data)

# Explore: show missing values
print(df.isnull().sum())

# Preprocess: fill missing with mean
df.fillna(df.mean(), inplace=True)

print(df)
```

7. Implement any one classifier using Python

```
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression

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

```
# Train model with more iterations
model = LogisticRegression(max_iter=200)
model.fit(X, y)

# Predict
predictions = model.predict(X[:5])
print("Predictions:", predictions)
```

8. Implement any one clustering algorithm using python

```
from sklearn.cluster import KMeans
import numpy as np

# Sample data
X = np.array([[1, 2], [1, 4], [1, 0],
              [4, 2], [4, 4], [4, 0]])

# KMeans with 2 clusters
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)

print("Cluster centers:")
print(kmeans.cluster_centers_)

print("Labels for data points:")
print(kmeans.labels_)
```

9. Implement association mining algorithm using python

```
from mlxtend.frequent_patterns import apriori
import pandas as pd

# Simple one-hot encoded data
data = {'Milk': [1, 0, 1],
        'Bread': [1, 1, 1],
        'Butter': [0, 1, 1]}
df = pd.DataFrame(data)

# Find frequent itemsets with min support 0.5
freq_itemsets = apriori(df, min_support=0.5, use_colnames=True)

print(freq_itemsets)
```

10. Implement pagerank algorithm

```
import networkx as nx

# Create a directed graph
G = nx.DiGraph()

# Add edges representing links (from_node, to_node)
edges = [(1, 2), (1, 3), (2, 3), (3, 1)]
G.add_edges_from(edges)
```

```
# Compute PageRank
pagerank = nx.pagerank(G, alpha=0.85)

# Print PageRank values
print(pagerank)
```