Practical: 6

Aim: Implement Naive Bayes Classification Techniques using any data analytics tools.

Source Code:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
data = load_iris()
X = data.data
y = data.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = GaussianNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

Practical: 7

Aim: Implement K-means Clustering Techniques using any data analytics tools.

Source Code:

```
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
data = load_iris()
X = data.data
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
labels = kmeans.labels_
plt.scatter(X[:, 0], X[:, 1], c=labels)
```

```
plt.title("K-Means Clustering")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```

Practical: 8

Aim: Implement DBScan clustering Techniques using any data analytics tools.

Source Code:

```
from sklearn.datasets import make_moons
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt
X, _ = make_moons(n_samples=300, noise=0.05)
dbscan = DBSCAN(eps=0.2, min_samples=5)
labels = dbscan.fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels)
plt.title("DBSCAN Clustering")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```

Practical: 9

Aim: Implement Eclat Associations Rule Mining Techniques using any data analytics tools.

Source Code:

```
from mlxtend.frequent_patterns import fpgrowth, association_rules
from mlxtend.preprocessing import TransactionEncoder
import pandas as pd
dataset = [
    ['milk', 'bread', 'nuts'],
    ['milk', 'bread'],
```

```
    ['milk', 'bread', 'nuts', 'apple'],
    ['bread', 'nuts'],
    ['milk', 'bread', 'apple']
]
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
frequent_itemsets = fpgrowth(df, min_support=0.6, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="support", min_threshold=0.6)
print(rules)
```

Practical: 10

Aim: implement Apriori Associations Rule Mining Techniques using any data analytics tools.

Source Code:

```
from mlxtend.frequent_patterns import apriori, association_rules
from mlxtend.preprocessing import TransactionEncoder
import pandas as pd
dataset = [
    ['milk', 'bread', 'nuts'],
    ['milk', 'bread'],
    ['milk', 'bread', 'nuts', 'apple'],
    ['bread', 'nuts'],
    ['milk', 'bread', 'apple']
]
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
frequent_itemsets = apriori(df, min_support=0.6, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
print(rules)
```

Practical: 11

Aim: Visualize all the statistical messures(mean, mode, median, range, inter quartile range, etc.) using Histogram, Boxplots, Scatter plots, etc.

Source Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
data = np.random.normal(loc=50, scale=15, size=100)
df = pd.DataFrame(data, columns=['Values'])
mean = df['Values'].mean()
median = df['Values'].median()
mode = df['Values'].mode()[0]
range_val = df['Values'].max() - df['Values'].min()
q1 = df['Values'].quantile(0.25)
q3 = df['Values'].quantile(0.75)
iqr = q3 - q1
plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
sns.histplot(df['Values'], bins=15, kde=True, color='skyblue')
plt.axvline(mean, color='red', linestyle='--', label='Mean')
plt.axvline(median, color='green', linestyle='--', label='Median')
plt.axvline(mode, color='orange', linestyle='--', label='Mode')
plt.legend()
plt.title('Histogram')
plt.subplot(1, 3, 2)
sns.boxplot(x=df['Values'], color='lightgreen')
plt.title('Boxplot')
plt.subplot(1, 3, 3)
```

```
sns.scatterplot(x=range(100), y=df['Values'], color='purple')
plt.axhline(mean, color='red', linestyle='--', label='Mean')
plt.axhline(median, color='green', linestyle='--', label='Median')
plt.legend()
plt.title('Scatter Plot')
plt.tight_layout()
plt.show()
```