7. Assuming a set of documents that need to be classified, use the naïve Bayesian Classifier model to perform this task. Built-in Java classes/API can be used to write the program. Calculate the accuracy, precision, and recall for your data set.

CODE:

```
import pandas as pd
msg = pd.read_csv('document.csv', names=['message', 'label'])

msg['labelnum'] = msg.label.map({'pos': 1, 'neg': 0})
X = msg.message
y = msg.labelnum

from sklearn.model_selection import train_test_split
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y)
from sklearn.feature_extraction.text import CountVectorizer

count_v = CountVectorizer()
Xtrain_dm = count_v.fit_transform(Xtrain)
Xtest_dm = count_v.transform(Xtest)

from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
clf.fit(Xtrain_dm, ytrain)
pred = clf.predict(Xtest_dm)


from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score
print('Accuracy Metrics: \n')
print('Accuracy: ', accuracy_score(ytest, pred))
print('Recall: ', recall_score(ytest, pred))
print('Precision: ', precision_score(ytest, pred))
print('Confusion Matrix: \n', confusion_matrix(ytest, pred))
```

8.Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set. You can use Java/ Python ML library classes/API.

CODE:

```
import numpy as np
import csv
import pandas as pd
from pgmpy.models import BayesianModel
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.inference import VariableElimination
import warnings
heart= pd.read_csv("heartdisease.csv")
model=BayesianModel([
    ("age","trestbps"),("age","fbs"),("sex","trestbps"),("trestbps","target"),
    ("fbs","target"),("target","restecg"),("target","thalach"),("target","chol")
])
model.fit(heart,estimator=MaximumLikelihoodEstimator)
heart_infer=VariableElimination(model)
q=heart_infer.query(variables=["target"],evidence={"age":40})
print(q)
q1=heart_infer.query(variables=["target"],evidence={"age":40,"sex":1, "trestbps":140,"chol":211})
print(q1)
```

9.Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same data set for clustering using k-Means algorithm. Compare the results of these two algorithms and comment on the quality of clustering. You can add Java/ Python ML library classes/API in the program.
CODE:

```
from sklearn.cluster import KMeans
from sklearn import preprocessing
from sklearn.mixture import GaussianMixture
from sklearn.datasets import load_iris
import sklearn.metrics as sm
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

dataset=load_iris()
X=pd.DataFrame(dataset.data)
X.columns=['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']
y=pd.DataFrame(dataset.target)
y.columns=['Targets']
plt.figure(figsize=(14,7))
colormap=np.array(['red','lime','black'])

plt.subplot(1,3,1)
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y.Targets],s=40)
plt.title('Real')

plt.subplot(1,3,2)
model=KMeans(n_clusters=3)
model.fit(X)
predY=np.choose(model.labels_,[0,1,2]).astype(np.int64)
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[predY],s=40)
plt.title('KMeans')

scaler=preprocessing.StandardScaler()
scaler.fit(X)
xsa=scaler.transform(X)
xs=pd.DataFrame(xsa,columns=X.columns)
gmm=GaussianMixture(n_components=3)
gmm.fit(xs)
```

```
y_cluster_gmm=gmm.predict(xs)
plt.subplot(1,3,3)
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y_cluster_gmm],s=40)
plt.title('GMM Classification')
```

10.Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions. Java/ Python ML library classes can be used for this problem.

CODE:

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

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

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

knn = KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)

print("Correct and Wrong Predictions:\n")
for i in range(len(y_test)):
    actual = y_test[i]
    predicted = y_pred[i]
    if actual == predicted:
        print(f"Correct: Actual={iris.target_names[actual]},
Predicted={iris.target_names[predicted]}")
    else:
        print(f"Wrong: Actual={iris.target_names[actual]},
Predicted={iris.target_names[predicted]}")

accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:",accuracy)
```

11.Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.

CODE:
```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(0)
X = np.linspace(0, 10, 100)
y = np.sin(X) + np.random.normal(scale=0.2, size=X.shape)
tau = 1.0
predictions = np.zeros_like(X)

for i in range(len(X)):
   xi = X[i]
   weights = np.exp(-((X - xi) ** 2) / (2 * tau ** 2))
   W = np.diag(weights)
   X_w = np.vstack([np.ones(len(X)), X]).T
   theta = np.linalg.pinv(X_w.T @ W @ X_w) @ X_w.T @ W @ y
   predictions[i] = np.hstack([1, xi]) @ theta

plt.figure(figsize=(10, 6))
plt.scatter(X, y, color='blue', label='Data Points')
plt.plot(X, predictions, color='red', label='LWR Fit')
plt.xlabel('X')
plt.ylabel('y')
plt.title('Locally Weighted Regression')
plt.legend()
plt.show()
```

12.Create the following plots using Matplotlib, Pandas Visualization,
Seaborn on iris dataset, wine reviews datasets.
a) Scatter Plot
b) Line chart
c) Histogram

CODE:

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

file_path = 'iris.csv'
iris_df = pd.read_csv(file_path)

plt.figure(figsize=(10, 6))
sns.scatterplot(data=iris_df, x='sepal_length', y='sepal_width', hue='species', palette='viridis')
plt.title('Scatter Plot of Sepal Length vs Sepal Width')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.show()
plt.figure(figsize=(10, 6))
sns.lineplot(data=iris_df, x=iris_df.index, y='sepal_length', hue='species', palette='viridis')
plt.title('Line Plot of Sepal Length over Index')
plt.xlabel('Index')
plt.ylabel('Sepal Length (cm)')
plt.show()

plt.figure(figsize=(10, 6))
sns.histplot(data=iris_df, x='petal_length', bins=30, kde=True, hue='species', palette='viridis')
plt.title('Histogram of Petal Lengths')
plt.xlabel('Petal Length (cm)')
plt.ylabel('Frequency')
plt.show()

file_path1 = 'winemag-data_first150k.csv'
wine_reviews = pd.read_csv(file_path1)

plt.figure(figsize=(10, 6))
sns.scatterplot(data=wine_reviews, x='points', y='price')
plt.title('Scatter Plot of Wine Points vs Price')
plt.xlabel('Points')
plt.ylabel('Price')
plt.show()

wine_reviews_sorted = wine_reviews.sort_values('price')

plt.figure(figsize=(10, 6))
sns.lineplot(data=wine_reviews_sorted, x=wine_reviews_sorted.index, y='points')
plt.title('Line Plot of Wine Points Sorted by Price')
plt.xlabel('Index')
plt.ylabel('Points')
plt.show()

plt.figure(figsize=(10, 6))
sns.histplot(data=wine_reviews, x='price', bins=30, kde=True)
plt.title('Histogram of Wine Prices')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```