1. SVM

```
import numpy as np

from sklearn.datasets import make_classification

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score

from sklearn.svm import SVC

import matplotlib.pyplot as plt


X,y=make_classification(n_samples=100,n_features=2,n_informative=2,n_redundant=0,random_state=42)

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

model=SVC(kernel='linear')

model.fit(X_train,y_train)

y_pred=model.predict(X_test)

accuracy=accuracy_score(y_test,y_pred)

print(f"Accuracy:{accuracy}")

plt.scatter(X[:,0],X[:,1],c=y,edgecolors='k',marker='o')

plt.xlabel('Feature1')

plt.ylabel('Feature2')

plt.title('SVM Decision Boundary')

plt.show()
```

2. KNN

```
from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import train_test_split

from sklearn import metrics

import pandas as pd


names=['sepal-length','sepal-width','petal-length','petal-width','CLass']

datasets=pd.read_csv("C:\\Users\\aadar\\OneDrive\\Desktop\\4th Sem\\ML\\iris-dataset.csv")


X=datasets.iloc[:,:-1]

y=datasets.iloc[:,-1]


X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)


model=KNeighborsClassifier(n_neighbors=3).fit(X_train,y_train)

ypred=model.predict(X_test)


i=0

print("%-25s %-25s %-25s" %('Original Label','Predicted Label','Correct/Incorrect'))

print('-'*75)

for label in y_test:

    print('%-25s %-25s'%(label,ypred[i]),end="")

    if(label==ypred[i]):

        print('%-25s ' %('Correct'))

    else:

        print('%-25s'%('Incorrect'))

    i=i+1


print("------------------------------------------------------------------")

print("Confusion Matrix",metrics.confusion_matrix(y_test,ypred))

print("------------------------------------------------------------------")

print("Classification Report",metrics.classification_report(y_test,ypred))

print("------------------------------------------------------------------")

print("Accuracy Score",metrics.accuracy_score(y_test,ypred))
```

3. Implement Naïve Bayesian Classifier

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB

dataset=pd.read_csv("C:\\Users\\aadar\\OneDrive\\Desktop\\4th Sem\\ML\\tennisdata.csv")
X=dataset.iloc[:,:-1]
y=dataset.iloc[:,-1]

X.head()

le_outlook=LabelEncoder()
X.Outlook=le_outlook.fit_transform(X.Outlook)

le_temperature=LabelEncoder()
X.Temperature=le_temperature.fit_transform(X.Temperature)

le_humidity=LabelEncoder()
X.Humidity=le_humidity.fit_transform(X.Humidity)

le_windy=LabelEncoder()
X.Windy=le_windy.fit_transform(X.Windy)

le_play=LabelEncoder()
y=le_play.fit_transform(y)
print(X.head())
print(y)

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
model=GaussianNB().fit(X_train,y_train)

ypred=model.predict(X_test)
print("Accuracy",accuracy_score(y_test,ypred))
```

4. Bayesian Network

```
import pandas as pd

data=pd.read_csv('C:\\Users\\aadar\\OneDrive\\Desktop\\4th Sem\\ML\\ds4 (1).csv')

prob_age={0:0.1, 1:0.2,2:0.3,3:0.2,3:0.2}

prob_gender={0:0.4,1:0.6}

prob_family={0:0.7,1:0.3}

prob_diet={0:0.5,1:0.5}

prob_lifestyle={0:0.1,1:0.3,2:0.4,3:0.2}

prob_cholestrol={0:0.4,1:0.3,2:0.3}

prob_heart_disease={

  (0,0,0,0,0,0):0.1,

  (1,1,1,1,1,1):0.9

}

def calculate_probability(age,gender,gamily,diet,lifestyle,cholestrol):

  key=(age,gender,gamily,diet,lifestyle,cholestrol)

  if key in prob_heart_disease:

    return prob_heart_disease[key]

  else:

    return 0.5


print('For Age enter SuperSeniorCitizen: 0, SeniorCitizen:1, MiddleAged: 2, Youth:3, Teen:4')

print('For Gender enter Male:0, Female:1')

print('For Family History enter Yes:1, No:0')

print('For Diet enter High:0, Medium:1')

print('For LifeStyle enter Athlete:0, Active: 1, Moderate: 2, Sedentary:3')

print('For Cholesterol enter High:0, BorderLine:1, Normal:2')


age=int(input('Enter Age: '))

gender=int(input('Enter gender: '))

family=int(input('Enter family history: '))

diet=int(input('Enter diet: '))

lifestyle=int(input('Enter lifesstyle: '))

cholestrol=int(input('Enter Cholestrol: '))


probability=calculate_probability(age,gender,family,diet,lifestyle,cholestrol)

print(f"Probability of heart disease is: {probability:.2f}")
```

5. EM Algorithm

```
import pandas as pd

from sklearn.cluster import KMeans

from sklearn.mixture import GaussianMixture

import matplotlib.pyplot as plt

data=pd.read_csv("C:\\Users\\aadar\\OneDrive\\Desktop\\4th Sem\\ML\\ds4 (1).csv")

X=data.values

Kmean=KMeans(n_clusters=3,random_state=42)

k_label=Kmean.fit_predict(X)


gmm=GaussianMixture(n_components=3,random_state=42)

gmm_label=gmm.fit_predict(X)


fig,(ax1,ax2)=plt.subplots(1,2,figsize=(12,6))


ax1.scatter(X[:,0],X[:,1],c=k_label,cmap='viridis',marker='o')

ax1.set_title('Kmean CLustering')


ax2.scatter(X[:,0],X[:,1],c=gmm_label,cmap='viridis',marker='o')

ax2.set_title('Gaussian Mixture')

plt.show()
```

6. Ada Boost

```
import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.ensemble import AdaBoostClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.datasets import load_iris

from sklearn.metrics import accuracy_score,classification_report

from sklearn.preprocessing import StandardScaler


df=load_iris()

X=df.data

y=df.target

target_names=df.target_names


scaler=StandardScaler()

X_scaled=scaler.fit_transform(X)


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

base_estimators=DecisionTreeClassifier(max_depth=1)

model=AdaBoostClassifier(estimator=base_estimators,n_estimators=50,algorithm='SAMME',random_state=42).fit(X_
train,y_train)

ypred=model.predict(X_test)

accuracy=accuracy_score(y_test,ypred)

classification=classification_report(y_test,ypred,target_names=target_names)


print("Accuracy: {:.2f}%".format(accuracy))

print("Classification report: ",classification)
```

7. Random Forest

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,precision_score,f1_score,recall_score
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.datasets import load_iris

df=load_iris()
X=df.data
y=df.target

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=42)
model=RandomForestClassifier(n_estimators=100,random_state=42).fit(X_train,y_train)
ypred=model.predict(X_test)

accuracy=accuracy_score(y_test,ypred)
f1=f1_score(y_test,ypred,average='weighted')
precision=precision_score(y_test,ypred,average='weighted')
recall=recall_score(y_test,ypred,average='weighted')
confusion=confusion_matrix(y_test,ypred)
classification=classification_report(y_test,ypred)

print("Accuracy Score: {:.2f}%".format(accuracy*100))
print("Recall Score: {:.2f}%".format(recall))
print("Precision Score: {:.2f}%".format(precision))
print("F1 Score: {:.2f}%".format(f1))
print("Confusion Matrix:\n",confusion)
print("Classification Report:\n",classification)
```