**Naïve Bays:**

Code:  
#train

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, recall\_score, precision\_score, f1\_score

from sklearn import preprocessing

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.metrics import confusion\_matrix

# Load the training data

df = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/Machine Project 2023/heart\_disease\_health\_indicators\_BRFSS2015.csv')

from google.colab import drive

drive.mount('/content/drive')

X1 = df.drop(['HeartDiseaseorAttack'] , axis = 1)

y1 = np.array(df['HeartDiseaseorAttack'])

X\_train1, X\_test1, y\_train1, y\_test1 = train\_test\_split (X1, y1, test\_size=0.25)

clf = GaussianNB()

clf.fit(X\_train1, y\_train1)

y\_pred1 = clf.predict(X\_test1)

from sklearn import preprocessing

label\_encoder = preprocessing.LabelEncoder()

y1 = label\_encoder.fit\_transform(y1)

y\_test1 = label\_encoder.fit\_transform(y\_test1)

y\_pred1 = label\_encoder.fit\_transform(y\_pred1)

accuracy1 = accuracy\_score(y\_test1, y\_pred1)

print('Accuracy:', accuracy1)

recall1 = recall\_score(y\_test1, y\_pred1)

print('Recall:', recall1)

precision1 = precision\_score(y\_test1, y\_pred1)

print('Precision:', precision1)

f11 = f1\_score(y\_test1,y\_pred1)

print('F1\_score:', f11)

Results:

Accuracy: 0.8183538315988647

Recall: 0.5501509560550151

Precision: 0.2706717280079221

F1\_score: 0.36283185840707965

Brief:

Naïve bayes is used in text classification, spam filtering, sentiment analysis, recommended systems, medical diagnosis.

Naïve bayes is supervised machine learning Algorithm.

Naïve bayes has discrete and continues features. The continues we need to estimate the mean and variance for each class then use it in the following equation:

while in the discrete we only use the following equation to calculate the probability of each class then do the same for the testing phase: P(H|X) =P(X|H)P(H)/P(X)

**SVM:**

Code:

#train

import pandas as pd

import numpy as np

from sklearn import datasets

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, recall\_score, precision\_score, f1\_score

from sklearn import preprocessing

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.metrics import confusion\_matrix

df = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/Machine Project 2023/heart\_disease\_health\_indicators\_BRFSS2015.csv')

X2 = df.drop(['HeartDiseaseorAttack'] , axis = 1)

y2 = np.array(df['HeartDiseaseorAttack'])

X\_train2, X\_test2, y\_train2, y\_test2 = train\_test\_split(X2, y2, test\_size=0.25, random\_state=42)

print(len(X\_train2))

print(len(X\_test2))

svm = SVC(C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache\_size=200, class\_weight=None, verbose=False, max\_iter=-1, decision\_function\_shape='ovr', break\_ties=False, random\_state=None)

svm.fit(X\_train2, y\_train2)

y\_pred2 = svm.predict(X\_test2)

accuracy2 = accuracy\_score(y\_test2, y\_pred2)

print('Accuracy:', accuracy2)

recall2 = recall\_score(y\_test2, y\_pred2,pos\_label=1)

print('Recall:', recall2)

precision2 = precision\_score(y\_test2, y\_pred2,pos\_label=1)

print('Precision:', precision2)

f12 = f1\_score(y\_test2,y\_pred2,pos\_label=1)

print('F1\_score:', f12)

Results:

Accuracy: 0.906401766004415

Recall: 0.0

Precision: 0.0

F1\_score: 0.0

Brief:

SVM is a binary classification. Also, it is a Supervised machine learning Algorithm.

SVM separates data from each other so it categorize it into two classes then we get maximum margin classifier then in each class we need to get support vectors.

**KNN:**

Code:

import pandas as pd

import numpy as np

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, recall\_score, precision\_score, f1\_score

from sklearn import preprocessing

# Load the training data

df = pd.read\_csv('/content/heart\_disease\_health\_indicators\_BRFSS2015.csv')

# Create feature and target arrays

X = df.drop(['HeartDiseaseorAttack'], axis=1)

y = np.array(df['HeartDiseaseorAttack'])

# Preprocess the data to handle missing values

X = X.fillna(X.mean())  # Replace missing values with the mean

# Split into training and test set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize and train the KNN classifier

knn = KNeighborsClassifier(n\_neighbors=21)

knn.fit(X\_train, y\_train)

# Predict on the test set

pred\_y = knn.predict(X\_test)

# Encode the target values

label\_encoder = preprocessing.LabelEncoder()

y\_test\_encoded = label\_encoder.fit\_transform(y\_test)

pred\_y\_encoded = label\_encoder.transform(pred\_y)

# Calculate evaluation metrics

accuracy = accuracy\_score(y\_test\_encoded, pred\_y\_encoded)

recall = recall\_score(y\_test\_encoded, pred\_y\_encoded)

precision = precision\_score(y\_test\_encoded, pred\_y\_encoded)

f1 = f1\_score(y\_test\_encoded, pred\_y\_encoded)

# Print the evaluation metrics

print('Accuracy:', accuracy)

print('Recall:', recall)

print('Precision:', precision)

print('F1\_score:', f1)

Results:

Accuracy: 0.9060036266162094

Recall: 0.02915268456375839

Precision: 0.4982078853046595

F1\_score: 0.055082227065583515

Brief:

* K-Nearest Neighbors (KNN) is a supervised machine learning algorithm
* Used for both classification and regression tasks.
* It calculates the distance between a new data point and existing points, selects the k closest neighbors, and predicts the class or value based on the neighbors. KNN is popular in recommendation systems, pattern recognition, and data mining.

**Logistic regression:**

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, recall\_score, precision\_score, f1\_score

from sklearn.impute import SimpleImputer

from sklearn import preprocessing

# Load the data

data = pd.read\_csv('/content/heart\_disease\_health\_indicators\_BRFSS2015.csv')

# Create feature and target arrays

X = data.drop(['HeartDiseaseorAttack'], axis=1)

y = np.array(data['HeartDiseaseorAttack'])

# Handle missing values using mean imputation

imputer = SimpleImputer(strategy='mean')

X\_imputed = imputer.fit\_transform(X)

# Split into training and test set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_imputed, y, test\_size=0.2, random\_state=42)

# Initialize and train the logistic regression classifier

classifier = LogisticRegression(solver='lbfgs', random\_state=0)

classifier.fit(X\_train, y\_train)

# Predict on the test set

predicted\_y = classifier.predict(X\_test)

# Encode the target values

label\_encoder = preprocessing.LabelEncoder()

y\_test\_encoded = label\_encoder.fit\_transform(y\_test)

pred\_y\_encoded = label\_encoder.transform(predicted\_y)

# Calculate evaluation metrics

accuracy = accuracy\_score(y\_test\_encoded, pred\_y\_encoded)

recall = recall\_score(y\_test\_encoded, pred\_y\_encoded)

precision = precision\_score(y\_test\_encoded, pred\_y\_encoded)

f1 = f1\_score(y\_test\_encoded, pred\_y\_encoded)

# Print the evaluation metrics

print('Accuracy:', accuracy)

print('Recall:', recall)

print('Precision:', precision)

print('F1\_score:', f1)

Result:

Accuracy: 0.9053476443768997

Recall: 0.12538540596094552

Precision: 0.45607476635514016

F1\_score: 0.1966948810963321

Brief:

Logistic regression is a statistical model used for binary classification tasks, where the goal is to predict the probability of an outcome belonging to a specific class.

It has several applications including medical diagnoses, credit scoring, and market analysis. Fraud detection and natural language processing