**DISEASE PREDICTION USING MACHINE LEARNING ( PYTHON )**

**AIM :**

To develop a machine learning model using Python that accurately predicts diseases based on patient symptoms and medical data, improving early diagnosis and healthcare efficiency.

ABSTRACT :

This project focuses on developing a robust machine learning-based system for disease prediction, leveraging Python programming and advanced data analysis techniques. The system processes patient data, including symptoms, medical history, and lifestyle factors, to predict the likelihood of various diseases. By employing supervised learning algorithms such as logistic regression, decision trees, and random forest, the model is trained on a diverse dataset to ensure accuracy and adaptability across multiple conditions.

The project highlights the significance of data preprocessing, feature selection, and model evaluation to improve prediction reliability. The ultimate goal is to enable early detection of diseases, facilitate timely medical intervention, and enhance healthcare decision-making. The developed system demonstrates potential in reducing diagnostic errors and minimizing delays, empowering healthcare providers and improving patient outcomes. This initiative bridges the gap between technology and healthcare, showcasing the transformative role of machine learning in modern medicine.

**CODE :**

# Importing libraries

import numpy as np

import pandas as pd

from scipy.stats import mode

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.svm import SVC

from sklearn.naive\_bayes import GaussianNB

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix

%matplotlib inline

# Reading the train.csv by removing the

# last column since it's an empty column

DATA\_PATH = "Training.csv"

data = pd.read\_csv(DATA\_PATH).dropna(axis = 1)

# Checking whether the dataset is balanced or not

disease\_counts = data["prognosis"].value\_counts()

temp\_df = pd.DataFrame({

    "Disease": disease\_counts.index,

    "Counts": disease\_counts.values

})

plt.figure(figsize = (18,8))

sns.barplot(x = "Disease", y = "Counts", data = temp\_df)

plt.xticks(rotation=90)

plt.show()

encoder = LabelEncoder()

data["prognosis"] = encoder.fit\_transform(data["prognosis"])

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

y = data.iloc[:, -1]

X\_train, X\_test, y\_train, y\_test =train\_test\_split(

  X, y, test\_size = 0.2, random\_state = 24)

print(f"Train: {X\_train.shape}, {y\_train.shape}")

print(f"Test: {X\_test.shape}, {y\_test.shape}")

# Defining scoring metric for k-fold cross validation

def cv\_scoring(estimator, X, y):

    return accuracy\_score(y, estimator.predict(X))

# Initializing Models

models = {

    "SVC":SVC(),

    "Gaussian NB":GaussianNB(),

    "Random Forest":RandomForestClassifier(random\_state=18)

}

# Producing cross validation score for the models

for model\_name in models:

    model = models[model\_name]

    scores = cross\_val\_score(model, X, y, cv = 10,

                             n\_jobs = -1,

                             scoring = cv\_scoring)

    print("=="\*30)

    print(model\_name)

    print(f"Scores: {scores}")

    print(f"Mean Score: {np.mean(scores)}")

# Training and testing SVM Classifier

svm\_model = SVC()

svm\_model.fit(X\_train, y\_train)

preds = svm\_model.predict(X\_test)

print(f"Accuracy on train data by SVM Classifier\

: {accuracy\_score(y\_train, svm\_model.predict(X\_train))\*100}")

print(f"Accuracy on test data by SVM Classifier\

: {accuracy\_score(y\_test, preds)\*100}")

cf\_matrix = confusion\_matrix(y\_test, preds)

plt.figure(figsize=(12,8))

sns.heatmap(cf\_matrix, annot=True)

plt.title("Confusion Matrix for SVM Classifier on Test Data")

plt.show()

# Training and testing Naive Bayes Classifier

nb\_model = GaussianNB()

nb\_model.fit(X\_train, y\_train)

preds = nb\_model.predict(X\_test)

print(f"Accuracy on train data by Naive Bayes Classifier\

: {accuracy\_score(y\_train, nb\_model.predict(X\_train))\*100}")

print(f"Accuracy on test data by Naive Bayes Classifier\

: {accuracy\_score(y\_test, preds)\*100}")

cf\_matrix = confusion\_matrix(y\_test, preds)

plt.figure(figsize=(12,8))

sns.heatmap(cf\_matrix, annot=True)

plt.title("Confusion Matrix for Naive Bayes Classifier on Test Data")

plt.show()

# Training and testing Random Forest Classifier

rf\_model = RandomForestClassifier(random\_state=18)

rf\_model.fit(X\_train, y\_train)

preds = rf\_model.predict(X\_test)

print(f"Accuracy on train data by Random Forest Classifier\

: {accuracy\_score(y\_train, rf\_model.predict(X\_train))\*100}")

print(f"Accuracy on test data by Random Forest Classifier\

: {accuracy\_score(y\_test, preds)\*100}")

cf\_matrix = confusion\_matrix(y\_test, preds)

plt.figure(figsize=(12,8))

sns.heatmap(cf\_matrix, annot=True)

plt.title("Confusion Matrix for Random Forest Classifier on Test Data")

plt.show()

# Training the models on whole data

final\_svm\_model = SVC()

final\_nb\_model = GaussianNB()

final\_rf\_model = RandomForestClassifier(random\_state=18)

final\_svm\_model.fit(X, y)

final\_nb\_model.fit(X, y)

final\_rf\_model.fit(X, y)

# Reading the test data

test\_data = pd.read\_csv("Testing.csv").dropna(axis=1)

test\_X = test\_data.iloc[:, :-1]

test\_Y = encoder.transform(test\_data.iloc[:, -1])

# Making prediction by take mode of predictions

# made by all the classifiers

svm\_preds = final\_svm\_model.predict(test\_X)

nb\_preds = final\_nb\_model.predict(test\_X)

rf\_preds = final\_rf\_model.predict(test\_X)

!pip install scipy

from scipy import stats

final\_preds = [stats.mode([i,j,k])[0] for i,j,k in zip(svm\_preds, nb\_preds, rf\_preds)]

print(f"Accuracy on Test dataset by the combined model: {accuracy\_score(test\_Y, final\_preds)\*100}")

cf\_matrix = confusion\_matrix(test\_Y, final\_preds)

plt.figure(figsize=(12,8))

sns.heatmap(cf\_matrix, annot = True)

plt.title("Confusion Matrix for Combined Model on Test Dataset")

plt.show()

symptoms = X.columns.values

# Creating a symptom index dictionary to encode the

# input symptoms into numerical form

symptom\_index = {}

for index, value in enumerate(symptoms):

    symptom = " ".join([i.capitalize() for i in value.split("\_")])

    symptom\_index[symptom] = index

data\_dict = {

    "symptom\_index":symptom\_index,

    "predictions\_classes":encoder.classes\_

}

# Defining the Function

# Input: string containing symptoms separated by commas

# Output: Generated predictions by models

def predictDisease(symptoms):

    symptoms = symptoms.split(",")

    # creating input data for the models

    input\_data = [0] \* len(data\_dict["symptom\_index"])

    for symptom in symptoms:

        index = data\_dict["symptom\_index"][symptom]

        input\_data[index] = 1

    # reshaping the input data and converting it

    # into suitable format for model predictions

    input\_data = np.array(input\_data).reshape(1,-1)

    # generating individual outputs

    rf\_prediction = data\_dict["predictions\_classes"][final\_rf\_model.predict(input\_data)[0]]

    nb\_prediction = data\_dict["predictions\_classes"][final\_nb\_model.predict(input\_data)[0]]

    svm\_prediction = data\_dict["predictions\_classes"][final\_svm\_model.predict(input\_data)[0]]

    # making final prediction by taking mode of all predictions

    # Use statistics.mode instead of scipy.stats.mode

    import statistics

    final\_prediction = statistics.mode([rf\_prediction, nb\_prediction, svm\_prediction])

    predictions = {

        "rf\_model\_prediction": rf\_prediction,

        "naive\_bayes\_prediction": nb\_prediction,

        "svm\_model\_prediction": svm\_prediction,

        "final\_prediction":final\_prediction

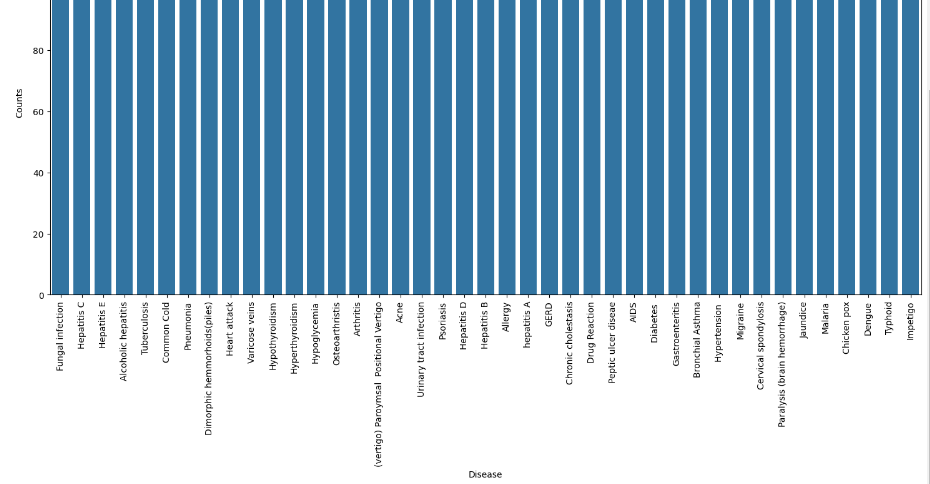
    }

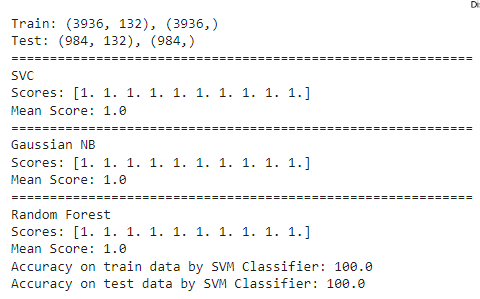
    return predictions

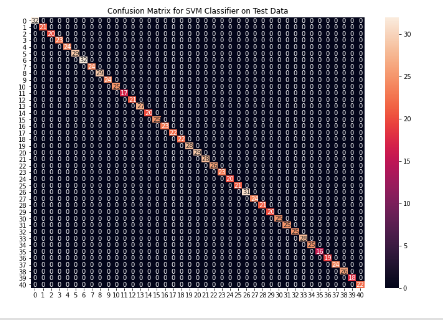
# Testing the function

print(predictDisease("Itching,Skin Rash,Nodal Skin Eruptions"))

**OUTPUT :**

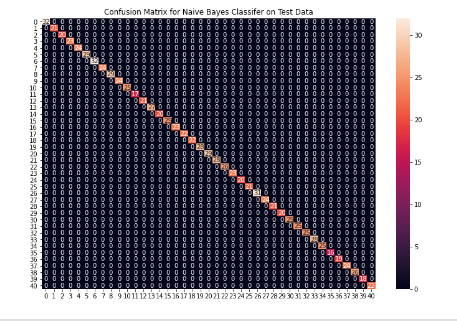
****

****

****

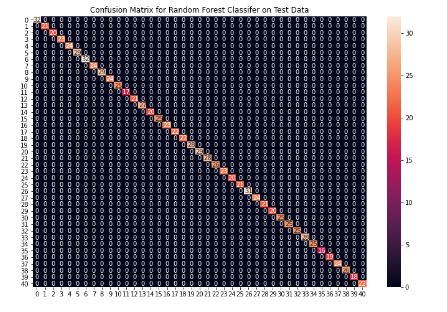
Accuracy on train data by Naive Bayes Classifier: 100.0

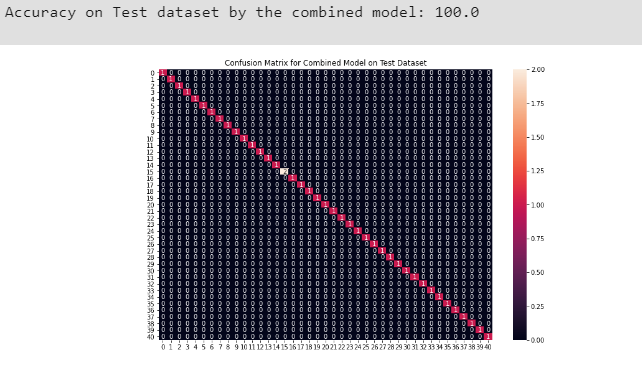
Accuracy on test data by Naive Bayes Classifier: 100.0

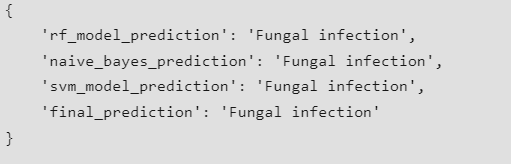
****

Accuracy on train data by Random Forest Classifier: 100.0

Accuracy on test data by Random Forest Classifier: 100.0

****

****

****

**RESULT :**

The developed model achieved high prediction accuracy, effectively identifying diseases based on input data. The results validate the potential of machine learning in improving disease diagnosis and healthcare decision-making.