# This Python 3 environment comes with many helpful analytics libraries installed

# It is defined by the kaggle/python Docker image: <https://github.com/kaggle/docker-python>

# For example, here’s several helpful packages to load

!pip install gradio

Import numpy as np # linear algebra

Import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

# Input data files are available in the read-only “../input/” directory

# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

Import os

For dirname, \_, filenames in os.walk(‘Testing.csv’):

For filename in filenames:

Print(os.path.join(dirname, filename))

Test = pd.read\_csv(‘Testing.csv’)

Train = pd.read\_csv(‘Training.csv’)

Test.sample(5)

Train.sample(10)

Test.describe()

Train.describe()

Train.shape

Test.shape

!pip install matplotlib seaborn

Import matplotlib.pyplot as plt

Import seaborn as sns

Import pandas as pd

Import numpy as np

From sklearn.metrics import accuracy\_score

From sklearn.ensemble import RandomForestClassifier

From sklearn.preprocessing import LabelEncoder

Import joblib

Train\_X = train.iloc[:, :-1] # First 132 columns as features

Train\_y = train.iloc[:, -1] # Last column as target (disease names)

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

Test\_y = test.iloc[:, -1]

Label\_encoder = LabelEncoder()

Y\_encoded\_train = label\_encoder.fit\_transform(train\_y)

Y\_encoded\_test = label\_encoder.transform(test\_y)

Model = RandomForestClassifier(n\_estimators=1, random\_state=0)

Model.fit(train\_X, y\_encoded\_train)

Pred\_y = model.predict(test\_X)

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

Def evaluate\_model(y\_encoded\_test, pred\_y, average=”weighted”):

Print(f”Accuracy: {accuracy\_score(y\_encoded\_test, pred\_y):.1f}”)

Print(f”Precision: {precision\_score(y\_encoded\_test, pred\_y, average=average):.1f}”)

Print(f”Recall: {recall\_score(y\_encoded\_test, pred\_y, average=average):.1f}”)

Print(f”F1-Score: {f1\_score(y\_encoded\_test, pred\_y, average=average):.1f}”)

Evaluate\_model(y\_encoded\_test, pred\_y)

Joblib.dump(model, “trained\_model”)

Feature\_importance = model.feature\_importances\_

Features = np.array(train\_X.columns)

Sorted\_idx = np.argsort(feature\_importance)[::-1]

Top\_n = 10

Plt.figure(figsize=(10, 6))

Sns.barplot(x=feature\_importance[sorted\_idx][:top\_n], y=features[sorted\_idx][:top\_n])

Plt.xlabel(“Feature Importance”)

Plt.ylabel(“Features”)

Plt.title(“Top Feature Importance (Random Forest)”)

Plt.show()

#dly

Import gradio as gr

Import joblib

Import pandas as pd

Import numpy as np

From sklearn.preprocessing import LabelEncoder

# Load trained model and label encoder

Model = joblib.load(“trained\_model”)

Train = pd.read\_csv(“Training.csv”)

Feature\_names = train.columns[:-1] # First 132 columns

Label\_encoder = LabelEncoder()

Label\_encoder.fit(train.iloc[:, -1]) # Fit encoder on disease names

# Define prediction function

Def predict\_disease(\*input\_features):

Input\_array = np.array(input\_features).reshape(1, -1)

Prediction = model.predict(input\_array)[0]

Predicted\_label = label\_encoder.inverse\_transform([prediction])[0]

Return f”Predicted Disease: {predicted\_label}”

# Dynamically create inputs for Gradio interface

Inputs = [gr.Slider(0, 1, step=1, label=feature) for feature in feature\_names]

# Create Gradio Interface

Interface = gr.Interface(

Fn=predict\_disease,

Inputs=inputs,

Outputs=”text”,

Title=”Disease Prediction from Symptoms”,

Description=”Input 0 or 1 for each symptom. Model predicts the most likely disease.”

)

Interface.launch()