SOURCE CODE

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

data = pd.read_csv('/content/test_data (1).csv
')

print(data.head())

print(data.info())

print(data.describe())

data = data.dropna()

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

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)

model = LogisticRegression()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
```

print(f'Accuracy: {accuracy}')

cm = confusion_matrix(y_test, y_pred)

print(f'Confusion Matrix:\n{cm}')

report = classification_report(y_test, y_pred)

print(f'Classification Report:\n{report}')

coefficients = model.coef_

print(f'Coefficients:\n{coefficients}')

#LOGISTIC REGRESSION

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix

# Step 1: Load your data
data = pd.read_csv('/content/test_data (1).csv')

# Step 2: Preprocess the data
# Assuming no missing values and all columns are numerical or have been encoded
appropriately
# If there are categorical variables, use pd.get_dummies() or other encoding
techniques

# Separate features and target
X = data.drop(columns=['Disease'])  # Replace 'target' with the name of your
target column
y = data['Disease']

# Normalize/Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 3: Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=0)

# Step 4: Train a classification model (Logistic Regression as an example)
model = LogisticRegression()
model.fit(X_train, y_train)

# Step 5: Evaluate the model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='macro')  # Use 'macro',
'micro', or 'weighted' for multi-class
recall = recall_score(y_test, y_pred, average='macro')       # Use 'macro',
'micro', or 'weighted' for multi-class
f1 = f1_score(y_test, y_pred, average='macro')

print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')

# Optionally, display the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print('Confusion Matrix:')
print(conf_matrix)
```

#RANDOM FOREST

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix

# Step 1: Load your data
data = pd.read_csv('/content/test_data (1).csv')

# Step 2: Preprocess the data
# Assuming no missing values and all columns are numerical or have been encoded
appropriately
# If there are categorical variables, use pd.get_dummies() or other encoding
techniques
```

```
# Separate features and target
X = data.drop(columns=['Disease'])  # Replace 'target' with the name of your
target column
y = data['Disease']

# Normalize/Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 3: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=0)

# Step 4: Train a Random Forest classifier
model = RandomForestClassifier(n_estimators=100, random_state=0)
model.fit(X_train, y_train)

# Step 5: Evaluate the model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='macro')  # Changed to
'macro'
recall = recall_score(y_test, y_pred, average='macro')        # Changed to
'macro'
f1 = f1_score(y_test, y_pred, average='macro')                # Changed to
'macro'

print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')

# Optionally, display the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print('Confusion Matrix:')
print(conf_matrix)
```

#DECISION TREE

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix

# Step 1: Load your data
data = pd.read_csv('/content/test_data (1).csv')

# Step 2: Preprocess the data
# Assuming no missing values and all columns are numerical or have been encoded
appropriately
# If there are categorical variables, use pd.get_dummies() or other encoding
techniques

# Separate features and target
X = data.drop(columns=['Disease'])  # Replace 'target' with the name of your
target column
y = data['Disease']

# Normalize/Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 3: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=0)

# Step 4: Train a Decision Tree classifier
model = DecisionTreeClassifier(random_state=0)
model.fit(X_train, y_train)

# Step 5: Evaluate the model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='macro')  # Changed to
'macro'
recall = recall_score(y_test, y_pred, average='macro')        # Changed to
'macro'
f1 = f1_score(y_test, y_pred, average='macro')                # Changed to
'macro'

print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
```

```
print(f'F1 Score: {f1}')

# Optionally, display the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print('Confusion Matrix:')
print(conf_matrix)
```

#Support Vector Machines (SVM):

#Naive Bayes:

#K-Nearest Neighbors (KNN):

#Neural Networks:

#Gradient Boosting Machines:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix

# Step 1: Load your data
data = pd.read_csv('/content/test_data (1).csv')  # Ensure your file is in the
same directory or provide the correct path

# Step 2: Preprocess the data
# Assuming no missing values and all columns are numerical or have been encoded
appropriately

# Separate features and target
X = data.drop(columns=['Disease'])  # Replace 'Disease' with the name of your
target column
y = data['Disease']
```

```
# Normalize/Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 3: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=0)

# Define models
models = {
    'Support Vector Machine': SVC(random_state=0),
    'Naive Bayes': GaussianNB(),
    'K-Nearest Neighbors': KNeighborsClassifier(),
    'Neural Network': MLPClassifier(random_state=0, max_iter=300),
    'Gradient Boosting': GradientBoostingClassifier(random_state=0)
}

# Train and evaluate each model
results = []
conf_matrices = {}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='macro')
    recall = recall_score(y_test, y_pred, average='macro')
    f1 = f1_score(y_test, y_pred, average='macro')
    results.append({
        'Model': name,
        'Accuracy': accuracy,
        'Precision': precision,
        'Recall': recall,
        'F1 Score': f1
    })
    conf_matrices[name] = confusion_matrix(y_test, y_pred)

results_df = pd.DataFrame(results)

# Displaying the results and confusion matrices
print(results_df)
for name, conf_matrix in conf_matrices.items():
```

```
    print(f'Confusion Matrix for {name}:')
    print(conf_matrix)
```