```
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, cros
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matr
from imblearn.over_sampling import RandomOverSampler
%matplotlib inline
```

```
import os
os.getcwd()
```

```
data = pd.read_csv('/content/improved_disease_dataset.csv'
```

```
data.head(2)
```

```
data.columns
```

```
data["disease"]
```

```
data.info()
```

```
encoder = LabelEncoder()
data["disease"] = encoder.fit_transform(data["disease"])
data["disease"]
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
```

```
data
```

iske ander jis diseases ka graph sabse jada hoga vo sabse uper aa jayega table mein

```
plt.figure(figsize=(18, 8))
sns.countplot(x=y)
plt.title("Disease Class Distribution Before Resampling")
plt.show()
```

Agar dataset imbalanced hai (unequal class counts), to ye minority class ko duplicate karke balance karta hai.

e.g. Agar “Flu” ke 1000 aur “Cancer” ke 50 hain → ye “Cancer” ko 1000 kar dega (duplicate karke).

```
ros = RandomOverSampler(random_state=42)
X_resampled, y_resampled = ros.fit_resample(X, y)
```

```
y.head().shape
```

```
if len(y_resampled.shape) > 1:
    y_resampled = y_resampled.values.ravel()
```

DecisionTreeClassifier: ek tree jaisa model jo decision rules banata hai.

RandomForestClassifier: multiple trees ka ensemble (zyada accurate).

SVC (Support Vector Classifier): plane (hyperplane) banata hai classes ke beech.

```
models = {
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "SVM": SVC()
}
```

Dataset ko 5 equal parts (folds) me divide karta hai.

Har baar 4 folds training ke liye aur 1 testing ke liye use hoti hai.

Stratified = ensures har fold me same class ratio ho.

```
cv_scoring = 'accuracy'
stratified_kfold = StratifiedKFold(n_splits=5, shuffle=Tru
```

ye teeno model ko compare kr raha hai

```
for model_name, model in models.items():
      scores = cross_val_score(
          model,
          X_resampled,
          y_resampled,
          cv=stratified_kfold,
          scoring=cv_scoring,
          n_jobs=-1,
          error_score='raise'
      )
      print("=" * 50)
      print(f"Model: {model_name}")
      print(f"Scores: {scores}")
      print(f"Mean Accuracy: {scores.mean():.4f}")
```

Support Vector Classifier (SVC)

```
svm_model = SVC()
svm_model.fit(X_resampled, y_resampled)
svm_preds = svm_model.predict(X_resampled)

cf_matrix_svm = confusion_matrix(y_resampled, svm_preds)
plt.figure(figsize=(12, 8))
sns.heatmap(cf_matrix_svm, annot=True, fmt="d")
plt.title("Confusion Matrix for SVM Classifier")
plt.show()

print(f"SVM Accuracy: {accuracy_score(y_resampled, svm_pre
```

Gaussian Naive Bayes

```
nb_model = GaussianNB()
nb_model.fit(X_resampled, y_resampled)
nb_preds = nb_model.predict(X_resampled)

cf_matrix_nb = confusion_matrix(y_resampled, nb_preds)
plt.figure(figsize=(12, 8))
sns.heatmap(cf_matrix_nb, annot=True, fmt="d")
plt.title("Confusion Matrix for Naive Bayes Classifier")
plt.show()

print(f"Naive Bayes Accuracy: {accuracy_score(y_resampled,
```

Random Forest Classifier

```
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_resampled, y_resampled)
rf_preds = rf_model.predict(X_resampled)

cf_matrix_rf = confusion_matrix(y_resampled, rf_preds)
plt.figure(figsize=(12, 8))
sns.heatmap(cf_matrix_rf, annot=True, fmt="d")
plt.title("Confusion Matrix for Random Forest Classifier")
plt.show()

print(f"Random Forest Accuracy: {accuracy_score(y_resample
```

Step 5: Combining Predictions for Robustness To build a robust model, we combine the predictions of all three models by taking the mode of their outputs. This ensures that even if one model makes an incorrect prediction the final output remains accurate.

```

```

```
from statistics import mode

final_preds = [mode([i, j, k]) for i, j, k in zip(svm_pred

cf_matrix_combined = confusion_matrix(y_resampled, final_p
plt.figure(figsize=(12, 8))
sns.heatmap(cf_matrix_combined, annot=True, fmt="d")
plt.title("Confusion Matrix for Combined Model")
plt.show()

print(f"Combined Model Accuracy: {accuracy_score(y_resampl
```

```
symptoms = X.columns.values
symptom_index = {symptom: idx for idx, symptom in enumerat
```

ye user se input lega or pridict krega diseases

```
disease_mapping = {
    0: "AIDS",
    1: "Acne",
    2: "Allergy",
    3: "Cervical spondylosis",
    4: "Chicken pox",
    5: "Chronic cholestasis",
    6: "Common Cold",
    7: "Dengue",
    8: "Diabetes ",
    9: "Drug Reaction",
    10: "Fungal infection",
    11: "Gastroenteritis",
    12: "Heart attack",
    13: "Hepatitis A",
    14: "Hepatitis B",
    15: "Hepatitis C",
    16: "Hepatitis D",
    17: "Hepatitis E",
    18: "Hypertension ",
    19: "Hyperthyroidism",
    20: "Hypoglycemia",
    21: "Hypothyroidism",
    22: "Impetigo",
    23: "Jaundice",
    24: "Malaria",
    25: "Migraine",
    26: "Osteoarthristis",
```

```
    27: "Paralysis (brain hemorrhage)",
    28: "Peptic ulcer disease",
    29: "Pneumonia",
    30: "Psoriasis",
    31: "Tuberculosis",
    32: "Typhoid",
    33: "Urinary tract infection",
    34: "Varicose veins",
    35: "Bronchial Asthma",
    36: "Dimorphic hemmorhoids(piles)",
    37: "Viral fever",
    38: "Jaundice",
    39: "Typhoid",
    40: "Pneumonia",
    41: "Acne",

}
```

```
from pickle import decode_long
symptoms = X.columns.values
symptom_index = {symptom: idx for idx, symptom in enumerat

def predict_disease(input_symptoms):
    input_symptoms = input_symptoms.split(",")
    input_data = [0] * len(symptom_index)

    for symptom in input_symptoms:
        if symptom in symptom_index:
            input_data[symptom_index[symptom]] = 1

    input_df = pd.DataFrame([input_data], columns=symptoms

    rf_index = rf_model.predict(input_df)[0]
    nb_index = nb_model.predict(input_df)[0]
    svm_index = svm_model.predict(input_df)[0]
    rf_pred = disease_mapping[rf_index]
    nb_pred = disease_mapping[nb_index]
    svm_pred = disease_mapping[svm_index]
    final_pred = mode([rf_pred, nb_pred, svm_pred])[0]
    return {
        "Random Forest Prediction": rf_pred,
        "Naive Bayes Prediction": nb_pred,
        "SVM Prediction": svm_pred,
        "Final Prediction": final_pred
```

```
    }

print(predict_disease("skin_rash,fever,headache"))
```