Y.SAKETH

21BBS0123

CSE(BUSINESS SYSTEMS)

VELLORE INSTITUTE OF TECHNOLOGY

Disease prediction using machine learning is used in healthcare to provide accurate and early diagnosis based on patient symptoms. We can build predictive models that identify diseases efficiently. In this article, we will explore the end-to-end implementation of such a system.

**Step 1: Import Libraries**

We will import all the necessary libraries like [pandas](https://www.geeksforgeeks.org/pandas-tutorial/), [Numpy](https://www.geeksforgeeks.org/numpy-tutorial/" \t "_blank), [scipy](https://www.geeksforgeeks.org/how-to-install-scipy-in-python-on-linux/" \t "_blank), [matplotlib](https://www.geeksforgeeks.org/python-introduction-matplotlib/), [seaborn](https://www.geeksforgeeks.org/introduction-to-seaborn-python/) and [scikit learn](https://www.geeksforgeeks.org/learning-model-building-scikit-learn-python-machine-learning-library/).

**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

**Step 2: Reading the dataset**

In this step we load the dataset and encode disease labels into numbers and visualize class distribution to check for imbalance. We then use [RandomOverSampler](https://www.geeksforgeeks.org/imbalanced-learn-module-in-python/" \t "_blank)to balance the dataset by duplicating minority classes and ensuring all diseases have equal samples for fair and effective model training.

**You can download dataset from here :**[**Click here**](https://media.geeksforgeeks.org/wp-content/uploads/20250415160347722782/improved_disease_dataset.csv)**.**

data = pd.read\_csv('improved\_disease\_dataset.csv')

encoder = LabelEncoder()

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

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

y = data.iloc[:, -1]

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

sns.countplot(x=y)

plt.title("Disease Class Distribution Before Resampling")

plt.xticks(rotation=90)

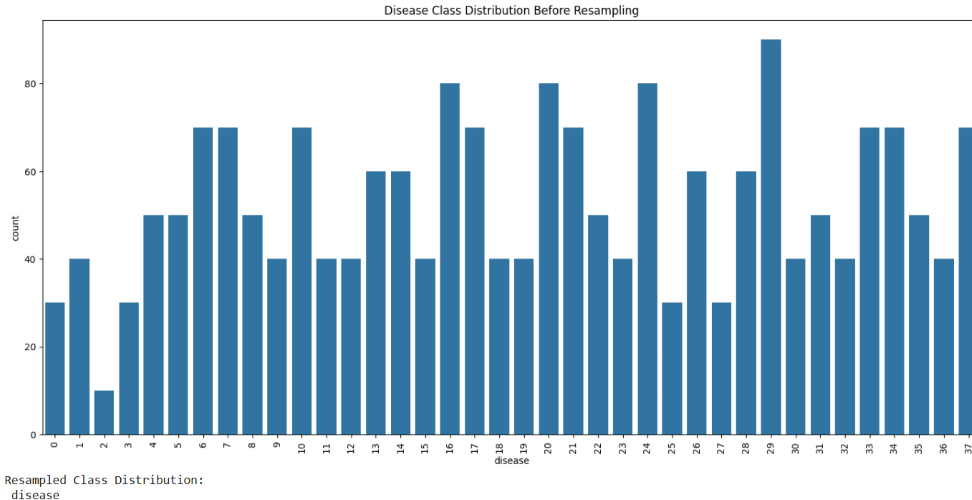
plt.show()

ros = RandomOverSampler(random\_state=42)

X\_resampled, y\_resampled = ros.fit\_resample(X, y)

print("Resampled Class Distribution:**\n**", pd.Series(y\_resampled).value\_counts())

**Output:**

Dataset Reading

**Step 3: Cross-Validation with Stratified K-Fold**

We use [**Stratified K-Fold Cross-Validation**](https://www.geeksforgeeks.org/stratified-k-fold-cross-validation/)to evaluate three machine learning models. The number of splits is set to 2 to accommodate smaller class sizes

**if** 'gender' **in** X\_resampled.columns:

le = LabelEncoder()

X\_resampled['gender'] = le.fit\_transform(X\_resampled['gender'])

X\_resampled = X\_resampled.fillna(0)

**if** len(y\_resampled.shape) > 1:

y\_resampled = y\_resampled.values.ravel()

models = {

"Decision Tree": DecisionTreeClassifier(),

"Random Forest": RandomForestClassifier()

}

cv\_scoring = 'accuracy' *# you can also use 'f1\_weighted', 'roc\_auc\_ovr' for multi-class*

stratified\_kfold = StratifiedKFold(n\_splits=5, shuffle=**True**, random\_state=42)

**for** model\_name, model **in** models.items():

**try**:

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**}**")

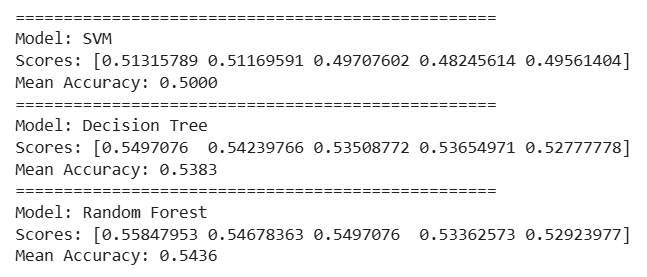
**except** **Exception** **as** e:

print("=" \* 50)

print(f"Model: **{**model\_name**}** failed with error:")

print(e)

**Output:**

Cross Validation Output

The output shows the evaluation results for three models SVC, Gaussian Naive Bayes and Random Forest using [cross-validation](https://www.geeksforgeeks.org/cross-validation-machine-learning/). Each model has two accuracy scores: 1.0 and approximately 0.976 indicating consistently high performance across all folds.

**Step 4: Training Individual Models and Generating Confusion Matrices**

After evaluating the models using cross-validation we train them on the resampled dataset and generate [confusion matrix](https://www.geeksforgeeks.org/confusion-matrix-machine-learning/) to visualize their performance on the test set.

[**Support Vector Classifier (SVC)**](https://www.geeksforgeeks.org/support-vector-machine-algorithm/)

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")

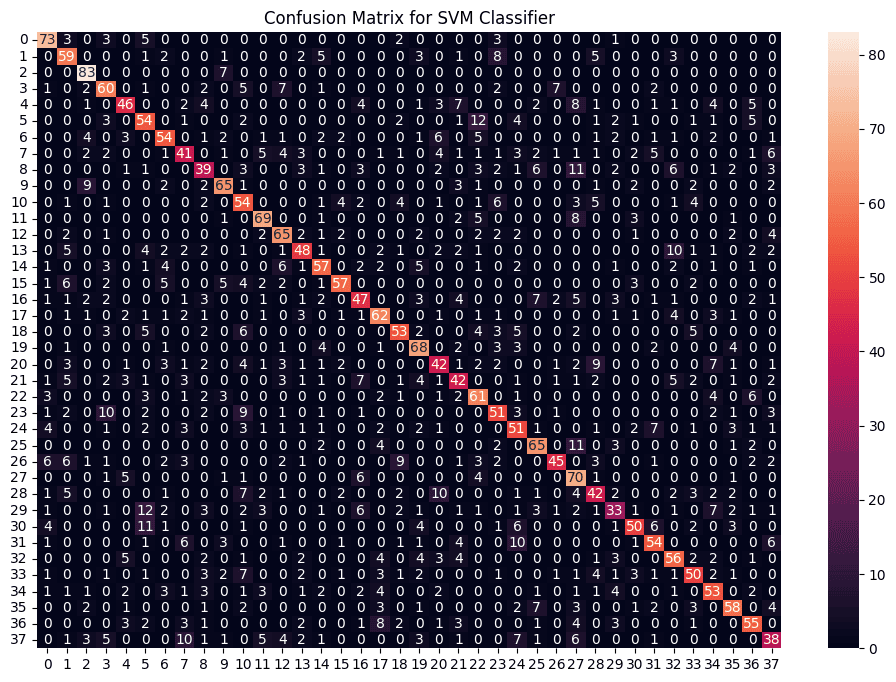
plt.title("Confusion Matrix for SVM Classifier")

plt.show()

print(f"SVM Accuracy: **{**accuracy\_score(y\_resampled, svm\_preds) \* 100**:**.2f**}**%")

**Output:**

*SVM Accuracy: 60.53%*

Confusion Matrix of SVM

The matrix shows good accuracy with most values along the diagonal meaning the SVM model predicted the correct class most of the time.

[**Gaussian Naive Bayes**](https://www.geeksforgeeks.org/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")

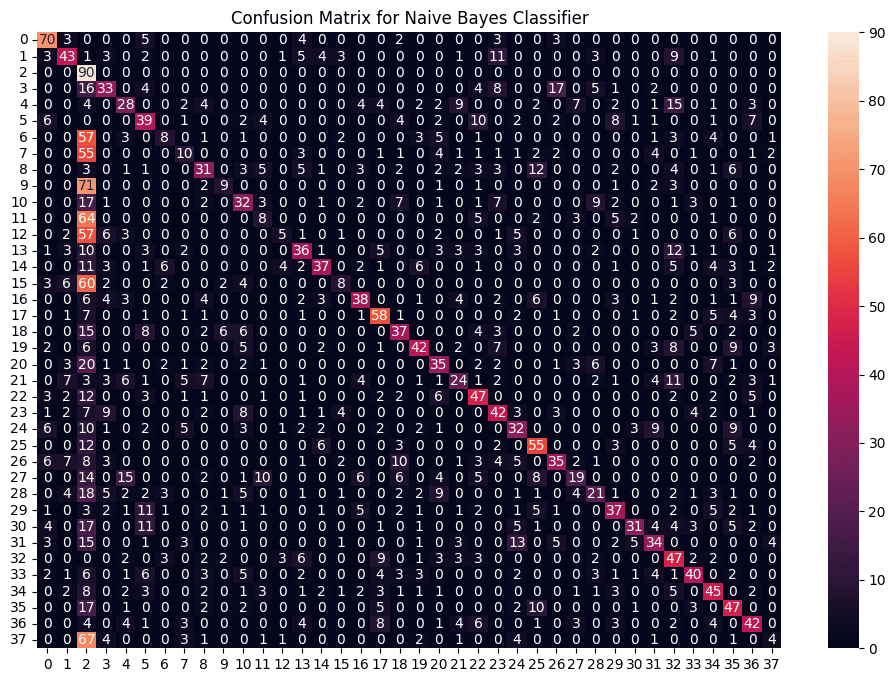
plt.show()

​

print(f"Naive Bayes Accuracy: {accuracy\_score(y\_resampled, nb\_preds) \* 100:.2f}%")

**Output:**

*Naive Bayes Accuracy: 37.98%*

Naive bayes Confusion Matrix

This matrix shows many off-diagonal values meaning the Naive Bayes model made more errors compared to the SVM. The predictions are less accurate and more spread out across incorrect classes.

[**Random Forest Classifier**](https://www.geeksforgeeks.org/random-forest-classifier-using-scikit-learn/)

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")

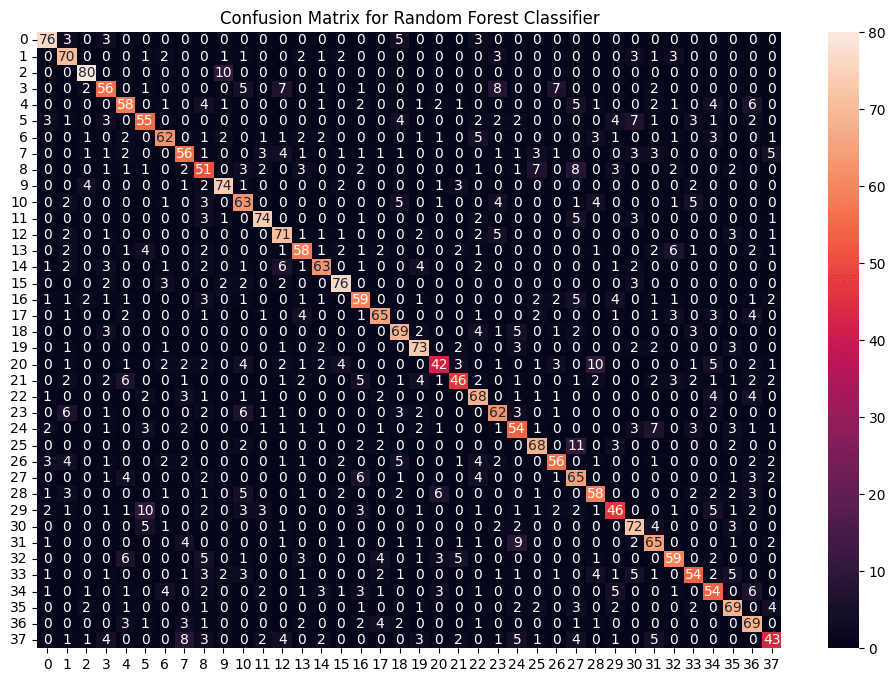
plt.show()

​

print(f"Random Forest Accuracy: {accuracy\_score(y\_resampled, rf\_preds) \* 100:.2f}%")

**Output:**

*Random Forest Accuracy: 68.98%*

Random Forest Confusion Matrix

This confusion matrix shows strong performance with most predictions correctly placed along the diagonal. It has fewer misclassifications than Naive Bayes and is comparable or slightly better than SVM.

**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\_preds, nb\_preds, rf\_preds)]

​

cf\_matrix\_combined = confusion\_matrix(y\_resampled, final\_preds)

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

sns.heatmap(cf\_matrix\_combined, annot=True, fmt="d")

plt.title("Confusion Matrix for Combined Model")

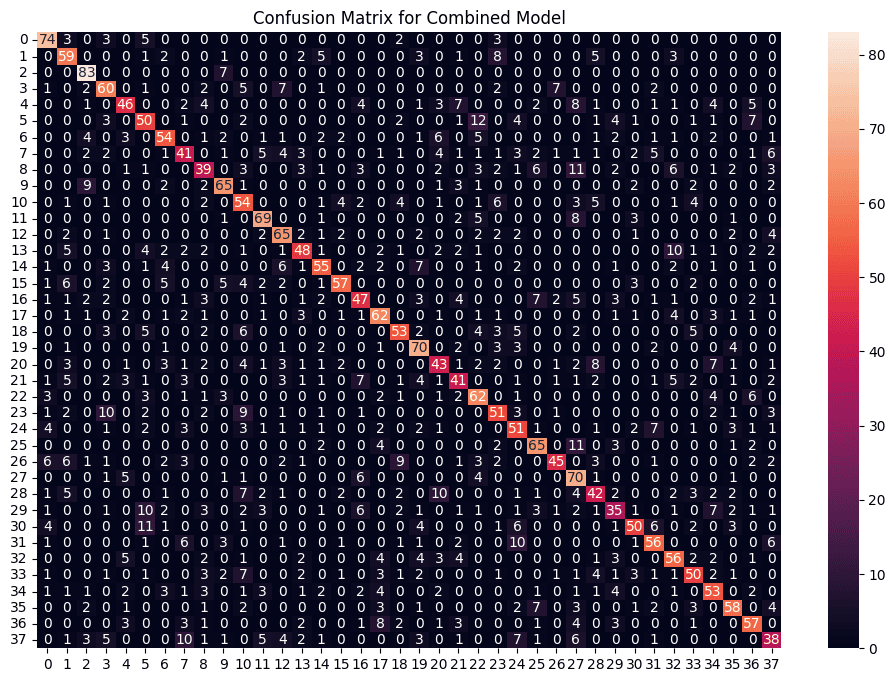
plt.show()

​

print(f"Combined Model Accuracy: {accuracy\_score(y\_resampled, final\_preds) \* 100:.2f}%")

**Output:**

*Combined Model Accuracy: 60.64%*

Combined Model Confusion matrix

Each cell shows how many times a true class (rows) was predicted as another class (columns) with high values on the diagonal indicating correct predictions.

**Step 6: Creating Prediction Function**

Finally, we create a function that takes symptoms as input and predicts the disease using the combined model. The input symptoms are encoded into numerical format and predictions are generated using the trained models.

symptoms = X.columns.values

symptom\_index = {symptom: idx for idx, symptom in enumerate(symptoms)}

​

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

​

rf\_pred = encoder.classes\_[rf\_model.predict(input\_data)[0]]

nb\_pred = encoder.classes\_[nb\_model.predict(input\_data)[0]]

svm\_pred = encoder.classes\_[svm\_model.predict(input\_data)[0]]

​

final\_pred = mode([rf\_pred, nb\_pred, svm\_pred])

return {

"Random Forest Prediction": rf\_pred,

"Naive Bayes Prediction": nb\_pred,

"SVM Prediction": svm\_pred,

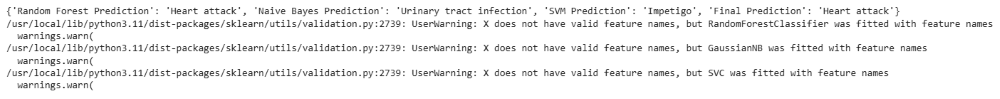
"Final Prediction": final\_pred

}

​

print(predict\_disease("Itching,Skin Rash,Nodal Skin Eruptions"))

**Output:**

Prediction

The output shows predictions from three classifiers where Random Forest predicted "Heart Attack", Naive Bayes Predicted "Urinary tract Infection" and SVM predict "Impetigo". The final combined prediction was "Heart Attack". We can further fine tune this model to make predictions more accurate.