```python
# Required libraries
import numpy as np
import pandas as pd
from scipy import stats
from scipy.stats import mode
import matplotlib.pyplot as plt
import seaborn as sb
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

#Reading Training Data
data_train = pd.read_csv("Dataset/Training.csv").
dropna(axis=1)

#Converting Prognosis Object part into numerical form
 using sklearn LabelEncoder()
le = LabelEncoder()
detected = le.fit_transform(data_train["prognosis"])

#Splitting Data For Training And Testing
X = data_train.iloc[:, :-1]
y = data_train.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X
, y, test_size=0.2, random_state=42)


#Implementing K-Fold Cross Validation, K=12
def cv_scoring(estimator, X, y):
    return accuracy_score(y, estimator.predict(X))


model_set = {
    "SVC": SVC(),
    "Gaussian NB": GaussianNB(),
    "Random Forest": RandomForestClassifier(
```

```
36 random_state=16)
37 }
38 for i in model_set:
39     current_model = model_set[i]
40     scores = cross_val_score(current_model, X, y, cv=
   12, n_jobs=-1, scoring=cv_scoring)
41
42
43 #Reading Test Data
44 data_test = pd.read_csv("Dataset/Testing.csv").dropna
   (axis=1)
45
46 #Selct Test Data
47 test_X = data_test.iloc[:, :-1]
48 test_Y = data_test.iloc[:, -1]
49
50 #Training Using SVM Algorithm
51 main_model_SVC = SVC()
52 main_model_SVC.fit(X, y)
53
54 #Training The Model Using Naive Bayes Algorithm
55 main_gnb = GaussianNB()
56 main_model_NB = main_gnb.fit(X, y)
57
58 #Training The Model Using RandomForestClassifier -
   Decision Tree Algorithm
59 main_RFC = RandomForestClassifier(n_estimators=100,
   random_state=16)
60 main_model_RFC = main_RFC.fit(X, y)
61
62
63
64 ##########################################################
   ##########################################################
   ################
65
66
67 ##GUI For App
68
69 #Required Libraries
70 from flask import Flask, render_template, request,
```

```
 70 send_from_directory
 71
 72 symptoms = X.columns.values
 73
 74 # Creating a symptom index dictionary to encode the
 75 # input symptoms into numerical form
 76 symptom_index = {}
 77 for index, value in enumerate(symptoms):
 78     symptom = "".join(value)
 79     symptom_index[symptom] = index
 80
 81 data_dict = {
 82     "symptom_index":symptom_index,
 83     "detection_classes":le.classes_
 84 }
 85
 86 #Function For Disease Detection
 87 def detect_Disease(symptoms):
 88     # creating input data for the models
 89     input_data = [0] * len(data_dict["symptom_index"
    ])
 90     for symptom in symptoms:
 91         index = data_dict["symptom_index"][symptom]
 92         input_data[index] = 1
 93
 94     # reshaping the input data and converting it
 95     # into suitable format for model predictions
 96     input_data = np.array(input_data).reshape(1,-1)
 97
 98     #Using Models For Detections As Per User Given
    Symptoms
 99     SVM_detection = main_model_SVC.predict(
    input_data)[0]
100     NB_detection = main_model_NB.predict(input_data
    )[0]
101     RFC_detection = main_model_RFC.predict(
    input_data)[0]
102
103     # making final detection by taking mode of all
    detection from all algorithms
104     import statistics
```

```
105     final_result = statistics.mode([RFC_detection,
    NB_detection, SVM_detection])
106     detected_result = {
107         "RandomForestClassifier Detected":
    RFC_detection,
108         "Naive Bayes Classifier Detected": str(
    NB_detection),
109         "SVM Classifier Detected": SVM_detection,
110         "Thus You Have": final_result
111     }
112     RandomForestClassifier = RFC_detection
113     NaiveBayes = str(NB_detection)
114     SVM = SVM_detection
115     fin_det = final_result
116     return RandomForestClassifier,NaiveBayes,SVM,
    fin_det
117
118
119 app = Flask(__name__)
120
121 @app.route('/', methods=['GET','POST'])
122 def home():
123     result_RFC="None"
124     result_NB="None"
125     result_SVM="None"
126     fin = "None"
127     pat_name = ""
128     result = {}
129     #Form Input From Checkboxes
130     if request.method=='POST':
131         symptoms_values = np.array(request.form.
    getlist('sympt'))
132         result_RFC, result_NB, result_SVM, fin =
    detect_Disease(symptoms_values)
133         pat_name = request.form.get("patient_name")
134
135     return render_template('index.html', res_RFC=
    result_RFC, res_NB= result_NB, res_SVM=result_SVM,
    fin=fin,name=pat_name)
136
137 @app.route('/files/Input.txt')
```

```
138 def serve_file(filename):
139     # Ensure the directory is correct
140     return send_from_directory('Code/Templates',
    filename)
141
142 if __name__ == '__main__':
143     app.run(debug=True)
144
```