Disease Prediction Model Documentation

Overview

This project implements a machine learning model for predicting diseases based on user-provided symptoms. The model uses multiple classifiers, including Support Vector Machine (SVM), Naive Bayes, and Random Forest, to enhance prediction accuracy.

Libraries Used

- **NumPy**: For numerical computations.
- **Pandas**: For data manipulation and analysis.
- **SciPy**: For statistical functions.
- **Matplotlib**: For plotting graphs.
- **Seaborn**: For advanced data visualization.
- **scikit-learn**: For implementing machine learning algorithms and evaluation metrics.

Data Preparation

1. Load Dataset

data = pd.read_csv('path/to/Training.csv').dropna(axis=1)

- **Description**: Loads the training dataset and removes columns with missing values.

2. Check Class Distribution

disease_counts = data["prognosis"].value_counts()

- **Description**: Counts occurrences of each disease in the dataset and visualizes the distribution.

Data Encoding and Splitting

3. Encode Target Variable

encoder = LabelEncoder()

data["prognosis"] = encoder.fit_transform(data["prognosis"])

- **Description**: Converts the categorical target variable (prognosis) into numerical format.

4. Split Data

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=24)

- **Description**: Splits the dataset into training (80%) and testing (20%) sets.

Model Training and Evaluation

5. Initialize Classifiers

```
models = {
   "SVC": SVC(),
   "Gaussian NB": GaussianNB(),
   "Random Forest": RandomForestClassifier(random_state=18)
}
```

- **Description**: Initializes the models to be evaluated.

6. Evaluate Models

```
scores = cross_val_score(model, X, y, cv=skf, n_jobs=-1, scoring=cv_scoring)
```

- **Description**: Performs cross-validation to evaluate the models and prints accuracy scores.

Train and Test Models

7. Train Each Classifier

- **Support Vector Classifier**

```
svm_model.fit(X_train, y_train)
```

- **Naive Bayes Classifier**

```
nb_model.fit(X_train, y_train)
```

- **Random Forest Classifier**

```
rf_model.fit(X_train, y_train)
```

- **Description**: Each classifier is trained on the training data and evaluated for accuracy on both the training and testing sets.

Final Model Training

8. Retrain Models on Full Dataset

```
final_svm_model.fit(X, y)
final_nb_model.fit(X, y)
final_rf_model.fit(X, y)
```

- **Description**: Retrains all models on the complete dataset for final predictions.

Making Predictions

9. Load Test Data

```
test_data = pd.read_csv("path/to/Testing.csv").dropna(axis=1)
```

- **Description**: Loads the test dataset for evaluation.

10. Combine Predictions

final_preds = [stats.mode([i, j, k]).mode[0] for i, j, k in zip(svm_preds, nb_preds, rf_preds)]

- **Description**: Combines predictions from all classifiers using the mode for a final decision.

Prediction Function

11. predictDisease(symptoms)

def predictDisease(symptoms):

...

- **Parameters**:
    - symptoms (str): A comma-separated string of symptoms (e.g., "Itching, Skin Rash").
- **Returns**: A dictionary with predictions from each model and a final combined prediction.

Example Usage

print(predictDisease("Itching,Skin Rash,Nodal Skin Eruptions"))

- **Description**: Calls the prediction function with input symptoms and returns predictions.

Conclusion

This documentation outlines the process of building a disease prediction model using machine learning. The code encompasses data loading, preprocessing, model training, evaluation, and a function for making predictions based on user input.