**Automated Disease Prediction Using Machine Learning Ensembles**

**Phase 3: Model Training and Evaluation**

**3.1 Overview of Model Training and Evaluation [Radha]**

In this stage, we concentrate on developing and examining ensemble machine learning models tailored for the prediction of diseases. The main goal is to ensure that the chosen models Random Forest, Gradient Boosting, and Logistic Regression are well-trained on the processed data and tuned to be as accurate as they can be. Hyper-parameter tuning is done to increase the model's prediction section and various evaluation metrics are employed to measure the system's accuracy and robustness. Cross-validation is used to ensure that the models correctly generalize over new data, boosting confidence in them and their reliability.

**3.2 Choosing Suitable Algorithms [Vasavi C Kulkarni]**

For the Automated Disease Prediction Using Machine Learning Ensembles project, the following algorithms are selected:

**Random Forest Classifier**

A tree-based ensemble algorithm which constructs many decision trees during the training and combines their outputs for the final forecast. It is capable of coping with noise and overfitting.

**Gradient Boosting Classifier**

A boosting algorithm which gradually corrects the errors of previous models. Its ability to handle complicated relationships in data makes it suitable for medical datasets.

**Logistic Regression**

A linear mode employed as a benchmark to its simplicity and interpretability. Besides the ensemble, it also contributes additional insights into the data, providing linear trends in the data.

**Source Code:**

**python**

# Import necessary libraries

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import train\_test\_split

# Assume 'data' is the preprocessed dataset, and 'labels' are the target variables

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data, labels, test\_size=0.2, random\_state=42)

# Train Random Forest

rf\_model = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf\_model.fit(X\_train, y\_train)

# Train Gradient Boosting

gb\_model = GradientBoostingClassifier(n\_estimators=100, learning\_rate=0.1, random\_state=42)

gb\_model.fit(X\_train, y\_train)

# Train Logistic Regression

lr\_model = LogisticRegression(random\_state=42)

lr\_model.fit(X\_train, y\_train)

**3.3 Hyperparameter Tuning [Sumeet U Pattan]**

One of the most crucial tasks in the process of optimizing model performance is the tuning of the hyperparameters. Grid Search and Random Search Techniques were used for the key parameters for each model to be adjusted.

**Example of Grid Search for Random Forest:**

For the hyperparameter, the number of estimators (n\_estimators) and the maximum depth of trees (max\_depth) were tuned to find the best configuration to be used in the overall model.

**Source Code:**

**Python**

from sklearn.model\_selection import GridSearchCV

# Define parameter grid for Random Forest

param\_grid = {

'n\_estimators': [50, 100, 150],

'max\_depth': [10, 20, None]

}

# Perform grid search

grid\_search = GridSearchCV(RandomForestClassifier(random\_state=42), param\_grid, cv=5, scoring='accuracy')

grid\_search.fit(X\_train, y\_train)

# Best parameters

print("Best Parameters:", grid\_search.best\_params\_)

**3.4 Model Evaluation Metrics [Shreya Kulkarni and Sumeet U Pattan]**

Various metrics are used to evaluate the performance of the models in the project of disease predictions. This is done to confirm their accuracy and reliability, and to examine the validity of the predictions.

**Accuracy Score**

The exactness of the predictions is the measure of the overall correctness. A higher accuracy gives more theoretical confidence in the results.

**Source Code:**

**Python**

from sklearn.metrics import accuracy\_score

y\_pred\_rf = rf\_model.predict(X\_test)

y\_pred\_gb = gb\_model.predict(X\_test)

y\_pred\_lr = lr\_model.predict(X\_test)

print("Random Forest Accuracy:", accuracy\_score(y\_test, y\_pred\_rf))

print("Gradient Boosting Accuracy:", accuracy\_score(y\_test, y\_pred\_gb))

print("Logistic Regression Accuracy:", accuracy\_score(y\_test, y\_pred\_lr))

**Precision and Recall**

The proportion of the number of true positives to the number of predicted positives is specified by precision, while the proportion of the number of true positives to the number of actual positives is given by recall.

**Source Code:**

**Python**

from sklearn.metrics import precision\_score, recall\_score

print("Random Forest Precision:", precision\_score(y\_test, y\_pred\_rf))

print("Gradient Boosting Recall:", recall\_score(y\_test, y\_pred\_gb))

**F1-Score**

The F-score combines both precision and recall into a single metric, this is especially useful for imbalanced datasets.

**Source Code:**

**Python**

from sklearn.metrics import f1\_score

print("Random Forest F1-Score:", f1\_score(y\_test, y\_pred\_rf))

**ROC-AUC Score**

It is a model's ability to differentiate between classes by calculating the area under the ROC curve that is assessed with a ROC-AUC score.

**Source Code:**

**Python**

from sklearn.metrics import roc\_auc\_score

print("Random Forest ROC-AUC Score:", roc\_auc\_score(y\_test, rf\_model.predict\_proba(X\_test)[:, 1]))

**3.5 Cross-Validation [Radha and Shreya Kulkarni]**

The assessment of the generalizability of the models was done through cross-validation. The process of cross-validation is the one that splits the data into a few subsets and then verifies the accuracy of the models by each of the unique datasets. It is a method of K-Fold cross-validation. Also, it ensures a robust evaluation.

**Source Code:**

**Python**

from sklearn.model\_selection import cross\_val\_score

# Perform cross-validation for Random Forest

cv\_scores\_rf = cross\_val\_score(rf\_model, data, labels, cv=5, scoring='accuracy')

print("Random Forest Cross-Validation Accuracy:", cv\_scores\_rf.mean())

**3.6 Conclusion of Phase 3 [Vasavi C Kulkarni]**

During Phase 3, The ensemble models (Random Forest, Gradient Boosting, and Logistic Regression) underwent the process of being trained and evaluated on the preprocessed dataset. The study on this subject revealed that hyperparameter tuning was the most effective method for this. This method was utilized and, in this way, the models’ configuration was the major focus of the study. Besides, the models’ performance was evaluated using metrics like accuracy, precision, recall, and ROC-AUC. For instance, Cross-validation guaranteed that the models were generalized data which was obvious with the robustness and reliability. These statistical models are now in working order and are the reason why automated disease risk prediction.