Model Development Phase Template

Date	10 July 2024
Team ID	739688
Project Title	Revolutionising Liver Care- Predicting Liver Cirrhosis using advanced Machine Learning
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:

```
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

# Print the evaluation metrics
print(f'Accuracy: {accuracy}')
print('Confusion Matrix:')
print(conf_matrix)
print('Classification Report:')
print(class_report)
```

```
# Define the parameter grid
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

# Initialize the RandomForestClassifier
rf = RandomForestClassifier(random_state=42)

# Initialize GridSearchCV
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=3, n_jobs=-1

# Fit the GridSearchCV to the data
grid_search.fit(X_train, y_train)

# Get the best parameters
best_params = grid_search.best_params_
print(f'Best parameters: {best_params}')

# Use the best estimator to make predictions
best_rf = grid_search.best_estimator_
y_pred = best_rf.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
```

```
knn = KNeighborsClassifier()

# Fit the model
knn.fit(X_train, y_train)

# Predict on the test set
y_pred = knn.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print(f'Baseline KNN Accuracy: {accuracy}')
print('Confusion Matrix:')
print(conf_matrix)
print('Classification Report:')
print(class_report)
```

```
gnb = GaussianNB()

# Fit the model to the training data
gnb.fit(X_train, y_train)

# Make predictions on the test data
y_pred = gnb.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

# Print the evaluation metrics
print(f'Naive Bayes Accuracy: {accuracy}')
print('Confusion Matrix:')
print(conf_matrix)
print('Classification Report:')
print(class_report)
```

```
xgb_model = xgb.XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')

# Define the parameter grid for hyperparameter tuning
param_grid = {
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.1, 0.2],
    'n_estimators': [100, 200, 300],
    'subsample': [0.8, 0.9, 1.0],
    'colsample_bytree': [0.8, 0.9, 1.0]
}

# Initialize GridSearchCV
grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid, cv=5, n_jobs=-1, verb

# Fit the GridSearchCV to the data
grid_search.fit(X_train, y_train)

# Get the best parameters
best_params = grid_search.best_params_
print(f'Best parameters: {best_params}')

# Use the best estimator to make predictions
best_xgb = grid_search.best_estimator_
y_pred = best_xgb.predict(X_test)
```

Model	Classification Report	F1 Score	Confusion Matrix
Random Forest	Classification Report: precision recall f1-score support NO 0.81 0.74 0.78 89 YES 0.89 0.92 0.91 196 accuracy 0.87 285 macro avg 0.85 0.83 0.84 285 weighted avg 0.86 0.87 0.86 285	86%	Confusion Matrix: [[66 23] [15 181]]

Model Validation and Evaluation Report:

Naïve Bayes	Classification Report: precision recall f1-score support NO 0.58 0.75 0.65 55 YES 0.88 0.78 0.83 135 accuracy 0.77 190 macro avg 0.73 0.76 0.74 190 weighted avg 0.79 0.77 0.78 190	78%	Confusion Matrix: [[41 14] [30 105]]
KNN	Classification Report: precision recall f1-score support NO 0.86 0.80 0.83 89 YES 0.91 0.94 0.92 196 accuracy 0.89 285 macro avg 0.88 0.87 0.88 285 weighted avg 0.89 0.89 0.89 285	89%	Confusion Matrix: [[71 18] [12 184]]

Xg Boost	Classification Report: precision recall f1-score support NO 0.58 0.75 0.65 55 YES 0.88 0.78 0.83 135 accuracy 0.77 190 macro avg 0.73 0.76 0.74 190 weighted avg 0.79 0.77 0.78 190	78%	Confusion Matrix: [[41 14] [30 105]]