Model Development Phase Template

Date	20 july 2024
Team ID	739716
Project Title	Predicting Baseline Histological staging in HCV patients using ML
Maximum Marks	10 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 a summary and training and validation performance metrics for multiple models, presented through respective screenshots.

Initial Model Training Code (5 marks):

Model Validation and Evaluation Report (5 marks):

Model	Summary	Training and Validation Performance Metrics

Knn model	The K Nearest Neighbors Classifier (KNN) is a versatile and simple machine learning algorithm used for both classification and regression tasks. It operates on the principle of proximity, assigning labels to data points based on the majority class among their k-nearest neighbors. KNN is non-parametric and requires no training phase, making it particularly suitable for applications	: `knn = KNeighborsClassifier()`
		`knn = knn.fit(X_train,y_train)`
		: `pred = knn.predict(X_test)`
		: `from sklearn.metrics import confusion_matrix, accuracy_score`
		`confusion_matrix(pred,y_test)`
		`print(accuracy_score(pred, y_test))`
		`0.653968253968254`

	where data relationships are not well-defined or change dynamically.	

Decision tree model	The decision tree classifier is apopular machine learning algorithm used for both classification and regression tasks	`dt_model = DecisionTreeClassifier(random_state=42)`
		`dt_model.fit(X_train, y_train)`
		`y_pred = dt_model.predict(X_test)`
		
		`accuracy = accuracy_score(y_test, y_pred)`
		`conf_matrix = confusion_matrix(y_test, y_pred)`
		`classification_rep = classification_report(y_test, y_`
		
		`print("Accuracy:", accuracy)`
		`print("\nConfusion Matrix:\n", conf_matrix)`
		
		`Accuracy: 0.6349206349206349`
		
		`Confusion Matrix:`
		`[[209 106]`

Random forest classifier	It is widely used for its ablity to handle complex data sets,and resistance to overfitting	RF = RandomForestClassifier(random_state=42) RF.fit(X_train, y_train) RandomForestClassifier(random_state=42) pred1 = RF.predict(X_test) score = RF.score(X_test,y_test) score 0.7746031746031746