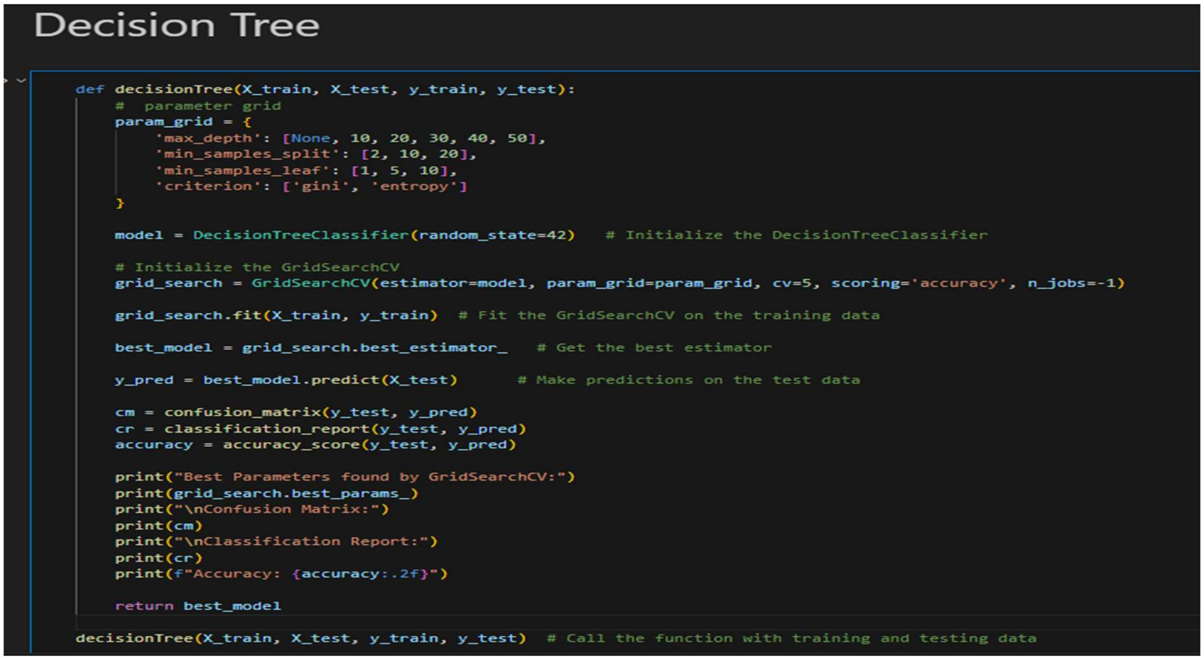
Model Development Phase Template

Date	15 March 2024
Team ID	xxxxxx
Project Title	Human Resource Management: Predicting Employee Promotions Using Machine Learning
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

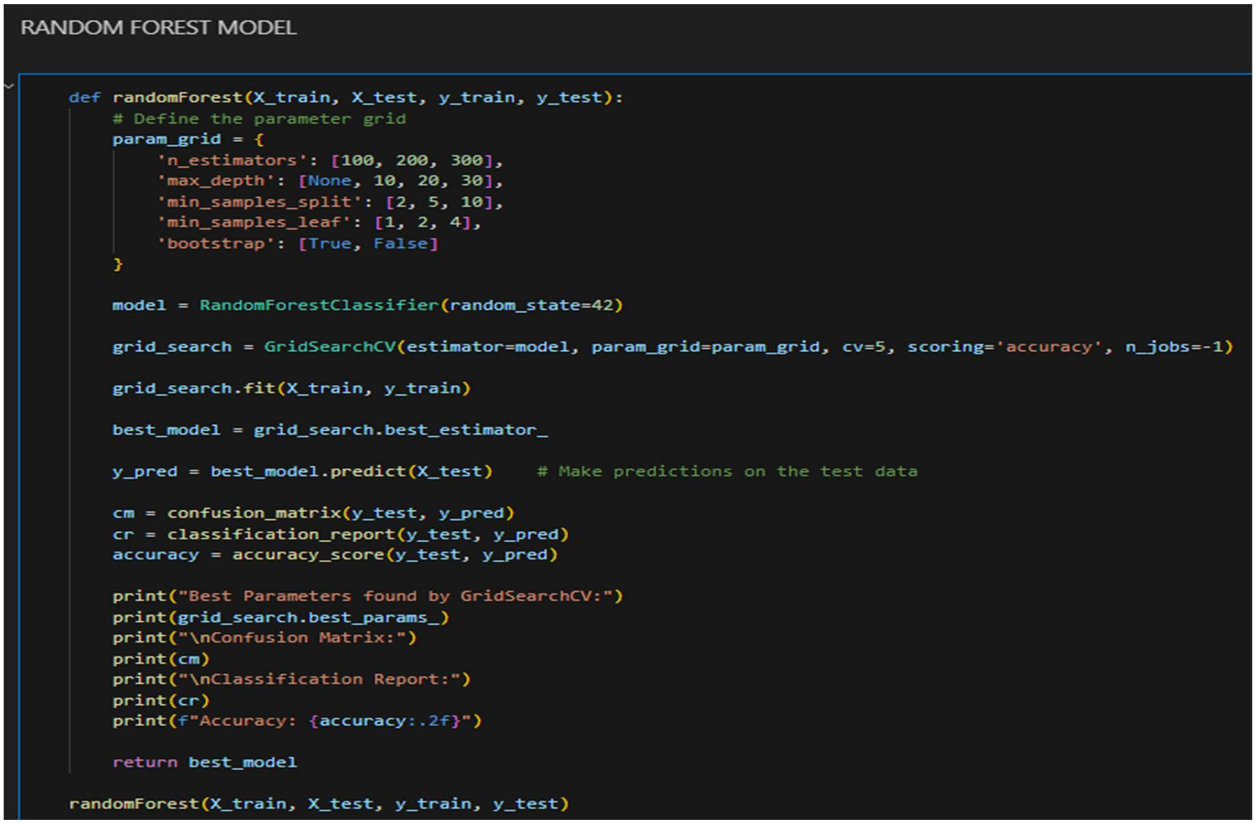
The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:



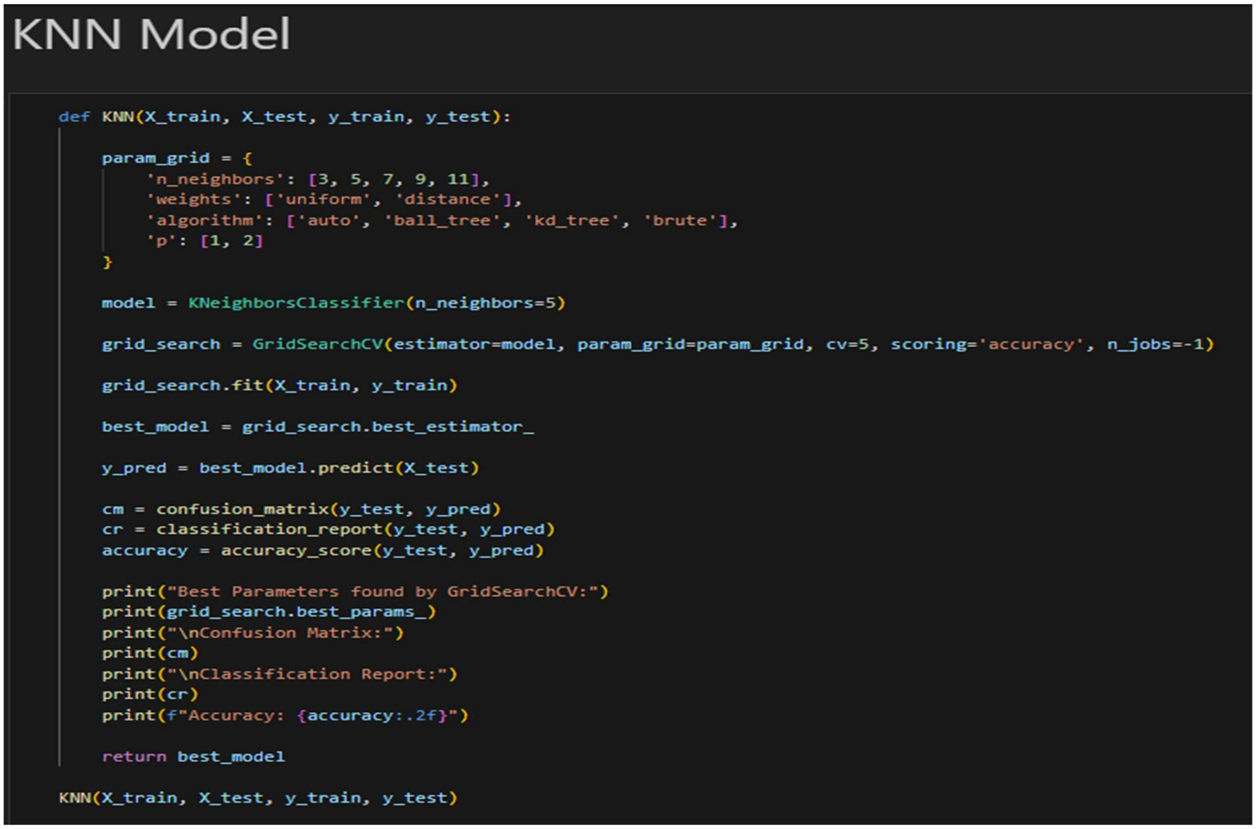
Decision Tree

```
def decisionTree(X_train, X_test, y_train, y_test):
    #  parameter grid
    param_grid = {
        'max_depth': [None, 10, 20, 30, 40, 50],
        'min_samples_split': [2, 10, 20],
        'min_samples_leaf': [1, 5, 10],
        'criterion': ['gini', 'entropy']
    }

    model = DecisionTreeClassifier(random_state=42)   # Initialize the DecisionTreeClassifier

    # Initialize the GridSearchCV
    grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy', n_jobs=-1)

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

    best_model = grid_search.best_estimator_   # Get the best estimator

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

    cm = confusion_matrix(y_test, y_pred)
    cr = classification_report(y_test, y_pred)
    accuracy = accuracy_score(y_test, y_pred)

    print("Best Parameters found by GridSearchCV:")
    print(grid_search.best_params_)
    print("\nConfusion Matrix:")
    print(cm)
    print("\nClassification Report:")
    print(cr)
    print(f"Accuracy: {accuracy:.2f}")

    return best_model

decisionTree(X_train, X_test, y_train, y_test)  # Call the function with training and testing data
```



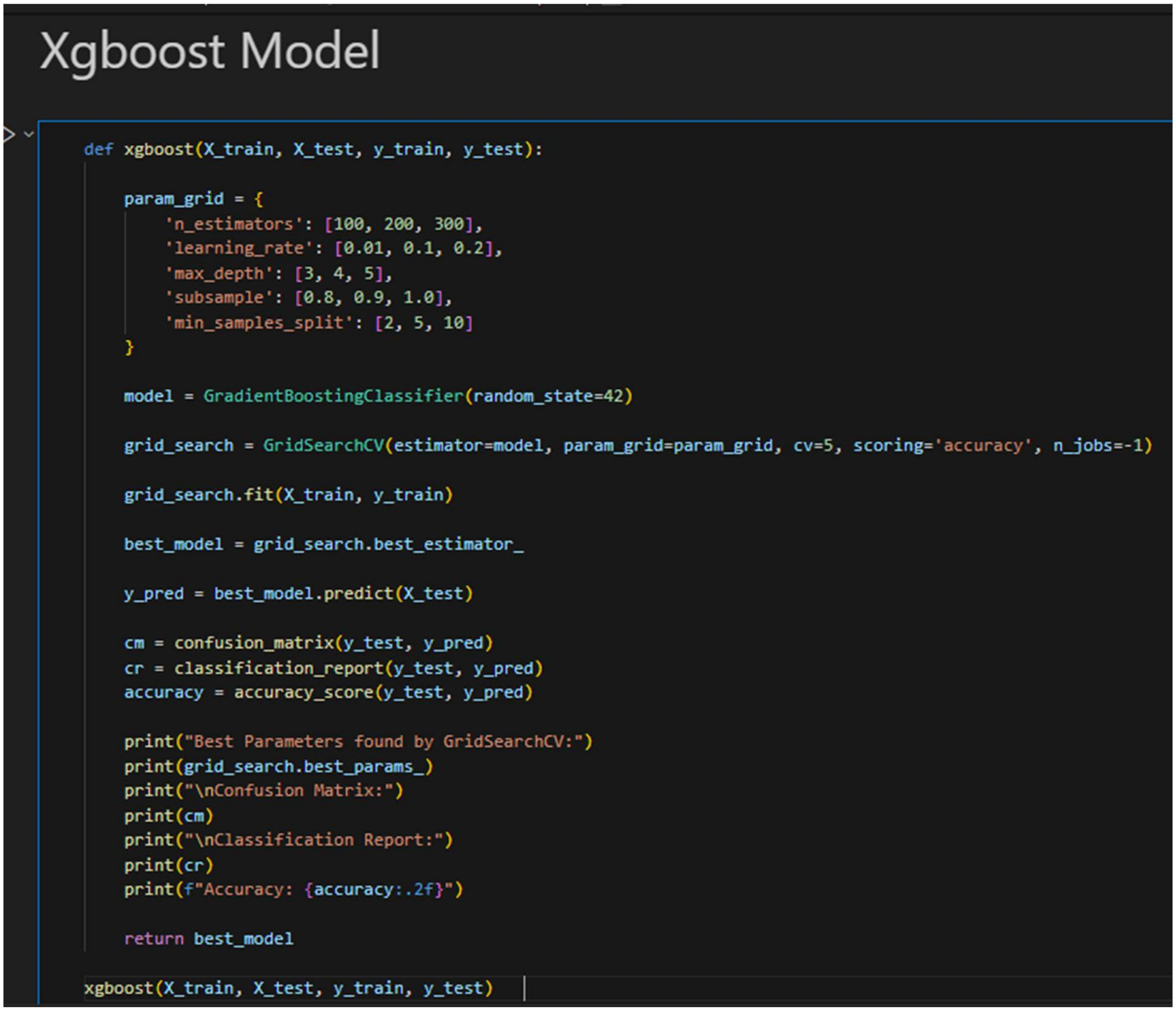
RANDOM FOREST MODEL

```
def randomForest(X_train, X_test, y_train, y_test):
    # Define the parameter grid
    param_grid = {
        'n_estimators': [100, 200, 300],
        'max_depth': [None, 10, 20, 30],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
        'bootstrap': [True, False]
    }

    model = RandomForestClassifier(random_state=42)

    grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy', n_jobs=-1)

    grid_search.fit(X_train, y_train)

    best_model = grid_search.best_estimator_

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

    cm = confusion_matrix(y_test, y_pred)
    cr = classification_report(y_test, y_pred)
    accuracy = accuracy_score(y_test, y_pred)

    print("Best Parameters found by GridSearchCV:")
    print(grid_search.best_params_)
    print("\nConfusion Matrix:")
    print(cm)
    print("\nClassification Report:")
    print(cr)
    print(f"Accuracy: {accuracy:.2f}")

    return best_model

randomForest(X_train, X_test, y_train, y_test)
```



KNN Model

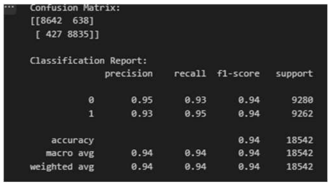
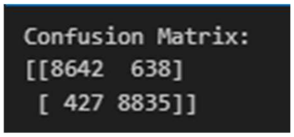
```
def KNN(X_train, X_test, y_train, y_test):

    param_grid = {
        'n_neighbors': [3, 5, 7, 9, 11],
        'weights': ['uniform', 'distance'],
        'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
        'p': [1, 2]
    }

    model = KNeighborsClassifier(n_neighbors=5)

    grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy', n_jobs=-1)

    grid_search.fit(X_train, y_train)

    best_model = grid_search.best_estimator_

    y_pred = best_model.predict(X_test)

    cm = confusion_matrix(y_test, y_pred)
    cr = classification_report(y_test, y_pred)
    accuracy = accuracy_score(y_test, y_pred)

    print("Best Parameters found by GridSearchCV:")
    print(grid_search.best_params_)
    print("\nConfusion Matrix:")
    print(cm)
    print("\nClassification Report:")
    print(cr)
    print(f"Accuracy: {accuracy:.2f}")

    return best_model

KNN(X_train, X_test, y_train, y_test)
```

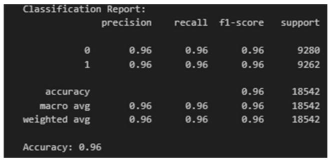
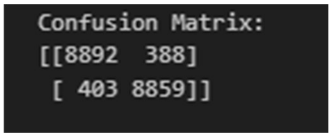


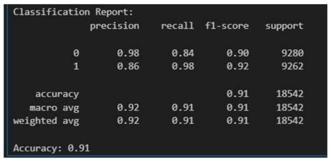
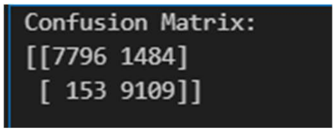
Xgboost Model

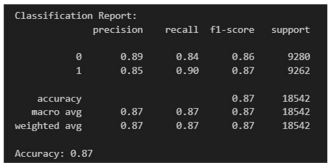
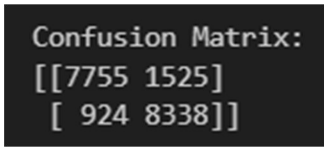
```
def xgboost(X_train, X_test, y_train, y_test):

    param_grid = {
        'n_estimators': [100, 200, 300],
        'learning_rate': [0.01, 0.1, 0.2],
        'max_depth': [3, 4, 5],
        'subsample': [0.8, 0.9, 1.0],
        'min_samples_split': [2, 5, 10]
    }

    model = GradientBoostingClassifier(random_state=42)

    grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy', n_jobs=-1)

    grid_search.fit(X_train, y_train)

    best_model = grid_search.best_estimator_

    y_pred = best_model.predict(X_test)

    cm = confusion_matrix(y_test, y_pred)
    cr = classification_report(y_test, y_pred)
    accuracy = accuracy_score(y_test, y_pred)

    print("Best Parameters found by GridSearchCV:")
    print(grid_search.best_params_)
    print("\nConfusion Matrix:")
    print(cm)
    print("\nClassification Report:")
    print(cr)
    print(f"Accuracy: {accuracy:.2f}")

    return best_model

xgboost(X_train, X_test, y_train, y_test)
```

Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix
Decision Tree	 Confusion Matrix: [[8642 638] [427 8835]] Classification Report: precision recall f1-score support 0 0.95 0.93 0.94 9280 1 0.93 0.95 0.94 9262 accuracy 0.94 18542 macro avg 0.94 0.94 0.94 18542 weighted avg 0.94 0.94 0.94 18542	94%	 Confusion Matrix: [[8642 638] [427 8835]]

Random Forest	
	Classification Report:
	              precision recall f1-score support
	
	           0 0.96 0.96 0.96 9280
	           1 0.96 0.96 0.96 9262
	
	    accuracy 0.96 18542
	   macro avg 0.96 0.96 0.96 18542
	weighted avg 0.96 0.96 0.96 18542
	
	Accuracy: 0.96
		96%	
			Confusion Matrix:
			[[8892 388]
			 [403 8859]]

KNN	
	Classification Report:
	              precision recall f1-score support
	
	           0 0.98 0.84 0.90 9280
	           1 0.86 0.98 0.92 9262
	
	    accuracy 0.91 18542
	   macro avg 0.92 0.91 0.91 18542
	weighted avg 0.92 0.91 0.91 18542
	
	Accuracy: 0.91
		91%	
			Confusion Matrix:
			[[7796 1484]
			 [153 9109]]

Xgboost	
	Classification Report:
	              precision recall f1-score support
	
	           0 0.89 0.84 0.86 9280
	           1 0.85 0.90 0.87 9262
	
	    accuracy 0.87 18542
	   macro avg 0.87 0.87 0.87 18542
	weighted avg 0.87 0.87 0.87 18542
	
	Accuracy: 0.87
		87%	
			Confusion Matrix:
			[[7755 1525]
			 [924 8338]]