Model Development Phase Template

Date	28 June 2024
Team ID	739973
Project Title	A Comprehensive Measure of Well-Being:The Human Development Index 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:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.impute import SimpleImputer

# Assuming 'data' is your original DataFrame
x = data.iloc[:, [2, 5, 6, 7, 67]]
x = pd.DataFrame(x)
y = data.iloc[:, 4].values
y = pd.DataFrame(y)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.1, random_state=0)

# Handle the 'Country' column (assuming it's at index 0)
imputer_numeric = SimpleImputer(strategy='mean')  # Imputer for numeric columns
x_train_numeric = x_train.drop(x_train.columns[0], axis=1)  # Remove 'Country' column
x_test_numeric = x_test.drop(x_test.columns[0], axis=1)    # Remove 'Country' column

x_train_imputed_numeric = imputer_numeric.fit_transform(x_train_numeric)
x_test_imputed_numeric = imputer_numeric.transform(x_test_numeric)

# Handle missing values in y_train
imputer_y = SimpleImputer(strategy='mean') # Use an imputer to fill missing values in y_train
y_train_imputed = imputer_y.fit_transform(y_train)

# Now fit the model with the imputed y_train
model = LinearRegression().fit(x_train_imputed_numeric, y_train_imputed)
```

```python
print(mean_squared_error(y_test,y_pred2))
```

```
0.001971249999999999
```

```python
# MSE for Random Forest
mse_rfc = mean_squared_error(y_test, y_pred2)
print("Random Forest MSE:", mse_rfc)

# R-squared for Random Forest
print("Random Forest Train Score:", rfc.score(x_train_imputed_numeric, y_train_imputed))
print("Random Forest Test Score:", rfc.score(x_test_imputed_numeric, y_test))

# MSE for Decision Tree
mse_dt = mean_squared_error(y_test, y_pred1)
print("Decision Tree MSE:", mse_dt)

# R-squared for Decision Tree
print("Decision Tree Train Score:", model1.score(x_train_imputed_numeric, y_train_imputed))
print("Decision Tree Test Score:", model1.score(x_test_imputed_numeric, y_test))
```

```
Random Forest MSE: 0.001971249999999999
Random Forest Train Score: 0.9947387758915783
Random Forest Test Score: 0.9743873613771092
Decision Tree MSE: 0.0006954529970833289
Decision Tree Train Score: 1.0
Decision Tree Test Score: 0.9274014000987563
```

Model Validation and Evaluation Report:

Model	Classification Report	Mean Square Error	Score
Random Forest	[37] print("Train:",rfc.score(x_train_imputed_numeric, y_train_imputed)) print("Test:",rfc.score(x_test_imputed_numeric, y_test)) Train: 0.9947387758915783 Test: 0.9743873613771092 print(mean_squared_error(y_test,y_pred1)) 0.0006954529970833289	0.0006954529970833289	**Train:**0.9947387758915783 **Test:**0.9743873613771092

Decision Tree	print(mean_squared_error(y_test,y_pred2))
	0.0019712499999999
	# MSE for Random Forest
	mse_rfc = mean_squared_error(y_test, y_pred2)
	print("Random Forest MSE:", mse_rfc)
	# R-squared for Random Forest
	print("Random Forest Train Score:", rfc.score(x_train_imputed_numeric, y_train_imputed))
	print("Random Forest Test Score:", rfc.score(x_test_imputed_numeric, y_test))
	# MSE for Decision Tree
	mse_dt = mean_squared_error(y_test, y_pred1)
	print("Decision Tree MSE:", mse_dt)
	# R-squared for Decision Tree
	print("Decision Tree Train Score:", model1.score(x_train_imputed_numeric, y_train_imputed))
	print("Decision Tree Test Score:", model1.score(x_test_imputed_numeric, y_test))
	Random Forest MSE: 0.0019712499999999
	Random Forest Train Score: 0.9947387758915783
	Random Forest Test Score: 0.9741873611771092
	Decision Tree MSE: 0.0006954529970833289
	Decision Tree Train Score: 1.0
	Decision Tree Test Score: 0.9274014000987563
		0.0006954529970833289	**Train:**1.0
			Test:0.9274014000987563

Linear Regression	[30] from sklearn.metrics import mean_squared_error,accuracy_score [31] mse=mean_squared_error(y_test,y_pred) mse 0.0007921136930643151 print("Train:",model.score(x_train_imputed_numeric, y_train_imputed)) print("Test:",model.score(x_test_imputed_numeric, y_test)) Train: 0.9534809529305541 Test: 0.9708274723758666	0.0007921136930643151	**Train:**0.9534809529305541 **Test:**0.9708274723758666