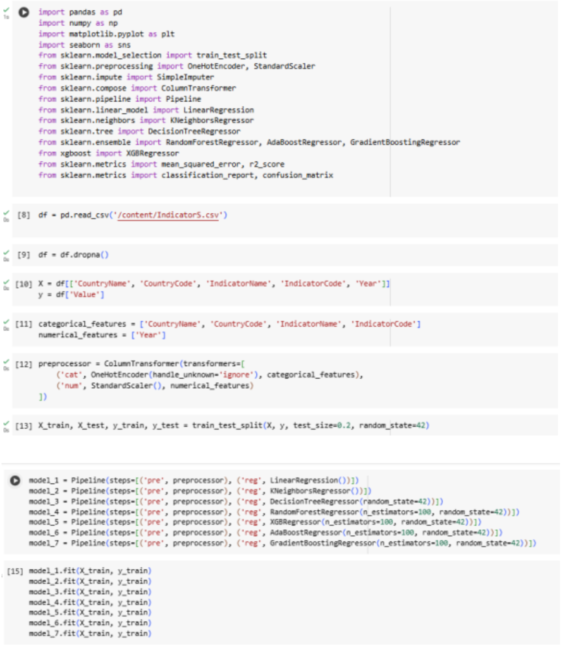
Model Development Phase Template

Date	14 June 2025
Team ID	SWTID1749709340
Project Title	Predicting Co2 Emission by countries 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
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor, GradientBoostingRegressor
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import classification_report, confusion_matrix
```

```
[8] df = pd.read_csv('/content/Indicator5.csv')
```

```
[9] df = df.dropna()
```

```
[10] X = df[['CountryName', 'CountryCode', 'IndicatorName', 'IndicatorCode', 'Year']]
     y = df['Value']
```

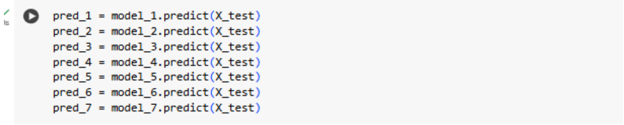
```
[11] categorical_features = ['CountryName', 'CountryCode', 'IndicatorName', 'IndicatorCode']
     numerical_features = ['Year']
```

```
[12] preprocessor = ColumnTransformer(transformers=[
         ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features),
         ('num', StandardScaler(), numerical_features)
     ])
```

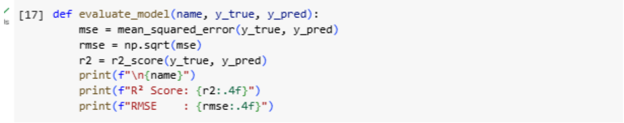
```
[13] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
model_1 = Pipeline(steps=[('pre', preprocessor), ('reg', LinearRegression())])
model_2 = Pipeline(steps=[('pre', preprocessor), ('reg', KNeighborsRegressor())])
model_3 = Pipeline(steps=[('pre', preprocessor), ('reg', DecisionTreeRegressor(random_state=42))])
model_4 = Pipeline(steps=[('pre', preprocessor), ('reg', RandomForestRegressor(n_estimators=100, random_state=42))])
model_5 = Pipeline(steps=[('pre', preprocessor), ('reg', XGBRegressor(n_estimators=100, random_state=42))])
model_6 = Pipeline(steps=[('pre', preprocessor), ('reg', AdaBoostRegressor(n_estimators=100, random_state=42))])
model_7 = Pipeline(steps=[('pre', preprocessor), ('reg', GradientBoostingRegressor(n_estimators=100, random_state=42))])
```

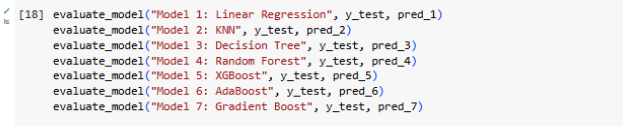
```
[15] model_1.fit(X_train, y_train)
     model_2.fit(X_train, y_train)
     model_3.fit(X_train, y_train)
     model_4.fit(X_train, y_train)
     model_5.fit(X_train, y_train)
     model_6.fit(X_train, y_train)
     model_7.fit(X_train, y_train)
```



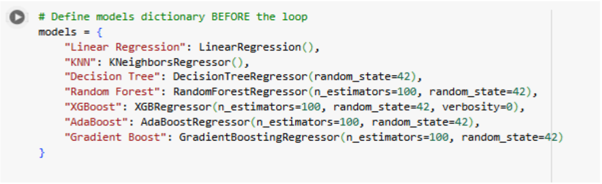
```
pred_1 = model_1.predict(X_test)
pred_2 = model_2.predict(X_test)
pred_3 = model_3.predict(X_test)
pred_4 = model_4.predict(X_test)
pred_5 = model_5.predict(X_test)
pred_6 = model_6.predict(X_test)
pred_7 = model_7.predict(X_test)
```



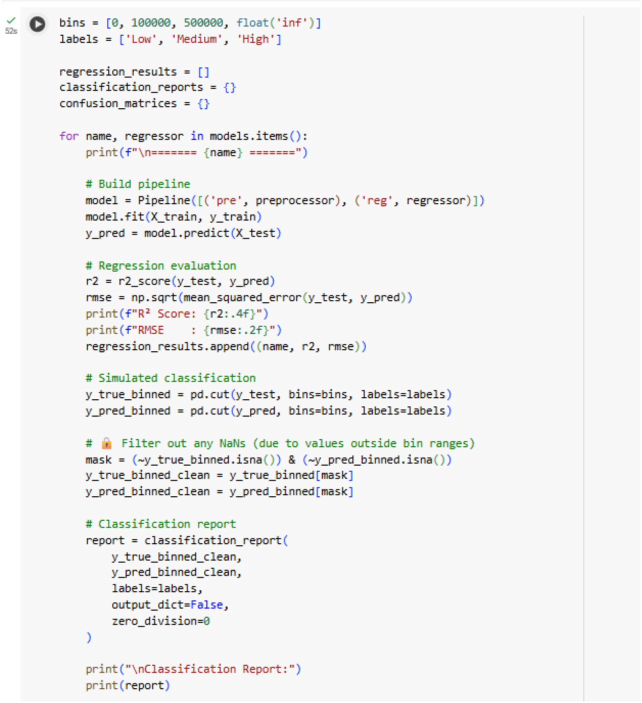
```
def evaluate_model(name, y_true, y_pred):
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_true, y_pred)
    print(f"\n{name}")
    print(f"R² Score: {r2:.4f}")
    print(f"RMSE    : {rmse:.4f}")
```



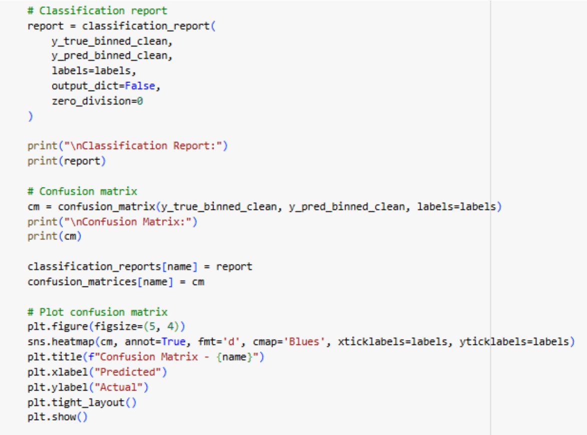
```
evaluate_model("Model 1: Linear Regression", y_test, pred_1)
evaluate_model("Model 2: KNN", y_test, pred_2)
evaluate_model("Model 3: Decision Tree", y_test, pred_3)
evaluate_model("Model 4: Random Forest", y_test, pred_4)
evaluate_model("Model 5: XGBoost", y_test, pred_5)
evaluate_model("Model 6: AdaBoost", y_test, pred_6)
evaluate_model("Model 7: Gradient Boost", y_test, pred_7)
```



```
# Define models dictionary BEFORE the loop
models = {
    "Linear Regression": LinearRegression(),
    "KNN": KNeighborsRegressor(),
    "Decision Tree": DecisionTreeRegressor(random_state=42),
    "Random Forest": RandomForestRegressor(n_estimators=100, random_state=42),
    "XGBoost": XGBRegressor(n_estimators=100, random_state=42, verbosity=0),
    "AdaBoost": AdaBoostRegressor(n_estimators=100, random_state=42),
    "Gradient Boost": GradientBoostingRegressor(n_estimators=100, random_state=42)
}
```

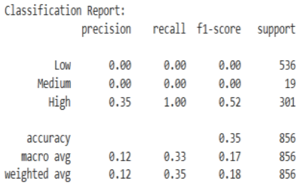
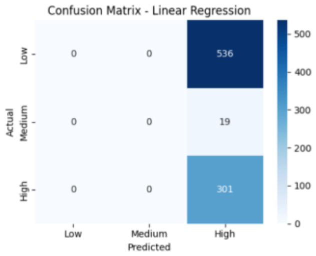
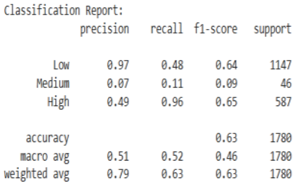
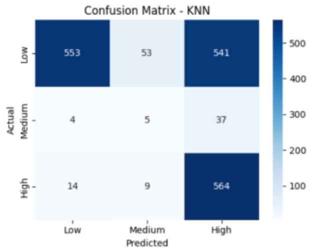
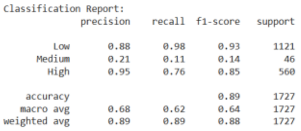
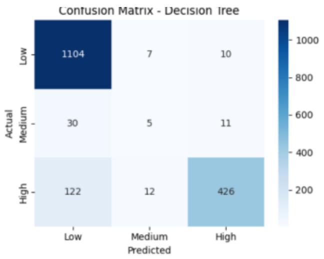


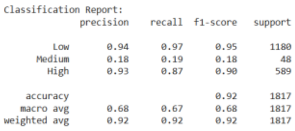
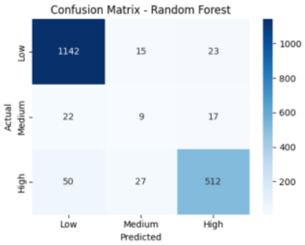
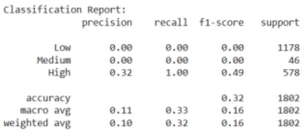
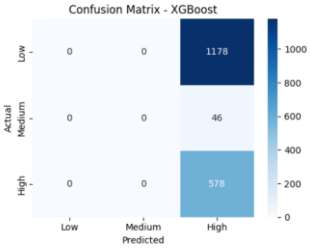
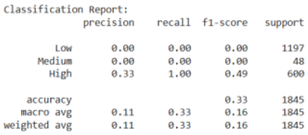
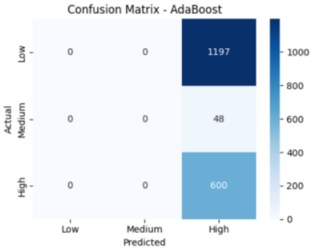
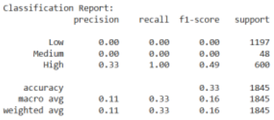
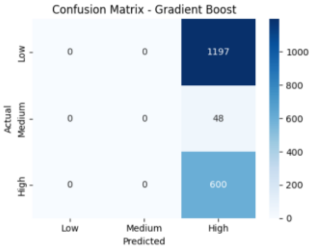
```
bins = [0, 100000, 500000, float('inf')]
labels = ['Low', 'Medium', 'High']

regression_results = []
classification_reports = {}
confusion_matrices = {}

for name, regressor in models.items():
    print(f"\n======= {name} =======")

    # Build pipeline
    model = Pipeline([('pre', preprocessor), ('reg', regressor)])
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Regression evaluation
    r2 = r2_score(y_test, y_pred)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    print(f"R² Score: {r2:.4f}")
    print(f"RMSE    : {rmse:.2f}")
    regression_results.append((name, r2, rmse))

    # Simulated classification
    y_true_binned = pd.cut(y_test, bins=bins, labels=labels)
    y_pred_binned = pd.cut(y_pred, bins=bins, labels=labels)

    # 🔒 Filter out any NaNs (due to values outside bin ranges)
    mask = (~y_true_binned.isna()) & (~y_pred_binned.isna())
    y_true_binned_clean = y_true_binned[mask]
    y_pred_binned_clean = y_pred_binned[mask]

    # Classification report
    report = classification_report(
        y_true_binned_clean,
        y_pred_binned_clean,
        labels=labels,
        output_dict=False,
        zero_division=0
    )

    print("\nClassification Report:")
    print(report)
```



```
# Classification report
report = classification_report(
    y_true_binned_clean,
    y_pred_binned_clean,
    labels=labels,
    output_dict=False,
    zero_division=0
)

print("\nClassification Report:")
print(report)

# Confusion matrix
cm = confusion_matrix(y_true_binned_clean, y_pred_binned_clean, labels=labels)
print("\nConfusion Matrix:")
print(cm)

classification_reports[name] = report
confusion_matrices[name] = cm

# Plot confusion matrix
plt.figure(figsize=(5, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels, yticklabels=labels)
plt.title(f"Confusion Matrix - {name}")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```

Model Validation and Evaluation Report:

Model	Classification Report	Rsquare score	Confusion Matrix
Linear Regression	 Classification Report: precision recall f1-score support Low 0.00 0.00 0.00 536 Medium 0.00 0.00 0.00 19 High 0.35 1.00 0.52 301 accuracy 0.35 856 macro avg 0.12 0.33 0.17 856 weighted avg 0.12 0.35 0.18 856	R² Score: 0.0301 RMSE : 13282867018179.9961	
KNN	 Classification Report: precision recall f1-score support Low 0.97 0.48 0.64 1147 Medium 0.07 0.11 0.09 46 High 0.49 0.96 0.65 587 accuracy 0.63 1780 macro avg 0.51 0.52 0.46 1780 weighted avg 0.79 0.63 0.63 1780	R² Score: 0.2783 RMSE : 11458119062280.3848	
Decision Tree	 Classification Report: precision recall f1-score support Low 0.88 0.98 0.93 1121 Medium 0.21 0.11 0.14 46 High 0.95 0.76 0.85 560 accuracy 0.89 1727 macro avg 0.68 0.62 0.64 1727 weighted avg 0.89 0.89 0.88 1727	R² Score: 0.8586 RMSE : 5071746744942.4941	

Random Forest	 Classification Report: precision recall f1-score support Low 0.94 0.97 0.95 1180 Medium 0.18 0.19 0.18 48 High 0.93 0.87 0.90 589 accuracy 0.92 1817 macro avg 0.68 0.67 0.68 1817 weighted avg 0.92 0.92 0.92 1817	R^2 Score: 0.9985 RMSE : 7863224335477.2188	
XGBoost	 Classification Report: precision recall f1-score support Low 0.00 0.00 0.00 1178 Medium 0.00 0.00 0.00 46 High 0.32 1.00 0.49 578 accuracy 0.32 1802 macro avg 0.11 0.33 0.16 1802 weighted avg 0.10 0.32 0.16 1802	Model 5: XGBoost R^2 Score: 0.8598 RMSE : 5050634315628.5957	
AdaBoost	 Classification Report: precision recall f1-score support Low 0.00 0.00 0.00 1197 Medium 0.00 0.00 0.00 48 High 0.33 1.00 0.49 600 accuracy 0.33 1845 macro avg 0.11 0.33 0.16 1845 weighted avg 0.11 0.33 0.16 1845	R^2 Score: -513.7526 RMSE : 305998691802595.8750	
Gradient Boost	 Classification Report: precision recall f1-score support Low 0.00 0.00 0.00 1197 Medium 0.00 0.00 0.00 48 High 0.33 1.00 0.49 600 accuracy 0.33 1845 macro avg 0.11 0.33 0.16 1845 weighted avg 0.11 0.33 0.16 1845	R^2 Score: 0.6902 RMSE : 7506340843913.5820	 Confusion Matrix - Gradient Boost Actual Predicted Low Medium High 0 0 1197 0 0 48 0 0 600 1000 800 600 400 200 0