

CO₂ and Greenhouse Gas Emissions

1. Introduction

Climate change is a critical global challenge, and one of its major drivers is greenhouse gas (GHG) emissions, particularly carbon dioxide (CO₂). This project uses historical emissions data to build regression models that can help analyze and predict trends in CO₂ and GHG outputs across different countries. By applying machine learning techniques, we aim to derive insights and potentially support future climate-related policies.

2. Dataset Overview

- **Dataset Name:** CO₂ and Greenhouse Gas Emissions
- **Source:** Our World in Data
- **Link to Data:** <https://www.kaggle.com/datasets/owid/co2-and-ghg-emissions>
- **Type of Problem:** Regression
- **Number of Samples:** ~60,000 rows (country-year combinations)

3. Data Description

The dataset contains country-level annual CO₂ and GHG emissions data, including total emissions, emissions per capita, and emissions by sector (e.g., transport, industry). Key attributes include:

- country, year
- co2, co2_per_capita, co2_growth_prct
- ghg_emissions, methane_emissions, nitrous_oxide_emissions
- population, gdp

4. Missing Data Handling

Some rows had missing values for certain years or countries:

- **Numerical columns** were filled using **mean imputation**.
- **Categorical columns** (if any) were forward-filled or dropped if sparse.

5. Preprocessing Steps

- Removed rows with no co2 values.
- Filled nulls in gdp, population, co2_per_capita using column means.
- Applied **MinMaxScaler** to normalize numerical values between 0 and 1.
- Encoded the country column using **Label Encoding**.

6. Data Visualization

Using seaborn and matplotlib:

- **Distribution plots** for co2, co2_per_capita, and gdp.
- **Heatmap** to show correlation between variables.
- **Scatter plots** to visualize GDP vs CO₂.
- Time series plot of global emissions over time.

7. Train-Test Split

- **Train/Test Proportion:** 80% train, 20% test
- **Target Variable:** co2 (total CO₂ emissions)
- **Feature Variables:** All others (after encoding and normalization)

8. Models Applied

We applied the following regression models:

1. **Linear Regression**
2. **Random Forest Regressor**
3. **Support Vector Regressor (SVR)**
4. **K-Nearest Neighbors (KNN) Regressor**
5. **Decision Tree Regressor**

Practical:

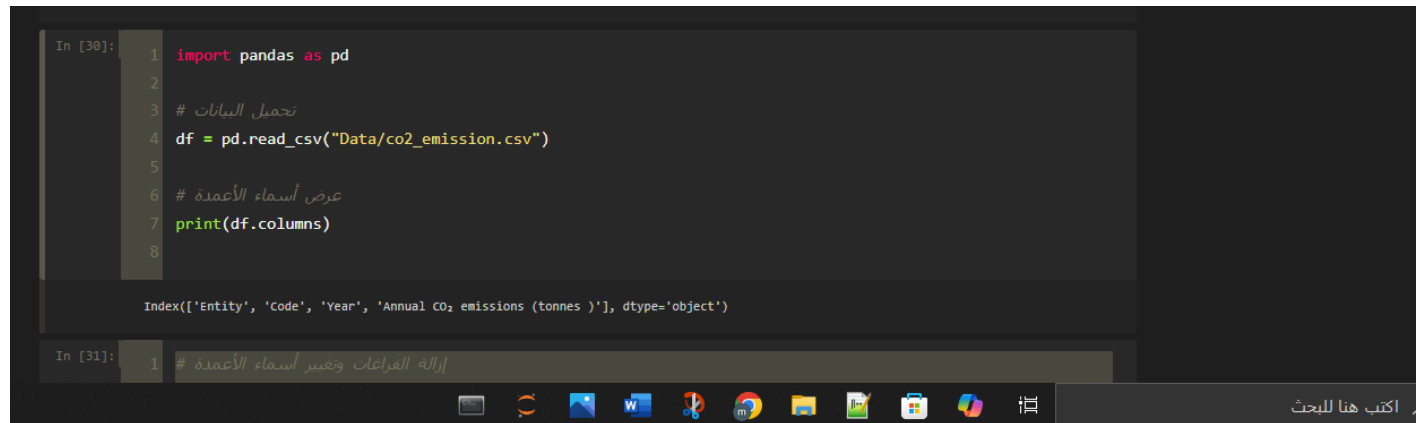
Step 1: Download the data

We begin by downloading the data from the official source:

```
import pandas as pd

# تحميل البيانات
df = pd.read_csv("Data/co2_emission.csv")

# عرض أسماء الأعمدة
print(df.columns)
```



```
In [30]: 1 import pandas as pd
2
3 # تحميل البيانات
4 df = pd.read_csv("Data/co2_emission.csv")
5
6 # عرض أسماء الأعمدة
7 print(df.columns)
8

Index(['Entity', 'Code', 'Year', 'Annual CO2 emissions (tonnes)'], dtype='object')

In [31]: 1 # إزالة الفراغات وتغيير أسماء الأعمدة
```

Step 2: Data Cleansing

We clean the data by removing spaces and changing column names to make it easier to work with:

```
# إزالة الفراغات وتغيير أسماء الأعمدة
df.columns = df.columns.str.strip().str.lower()
df.rename(columns={'annual co2 emissions (tonnes)': 'co2'}, inplace=True)

# حذف الصفوف التي تحتوي على قيم مفقودة في عمود 'co2'
df.dropna(subset=['co2'], inplace=True)

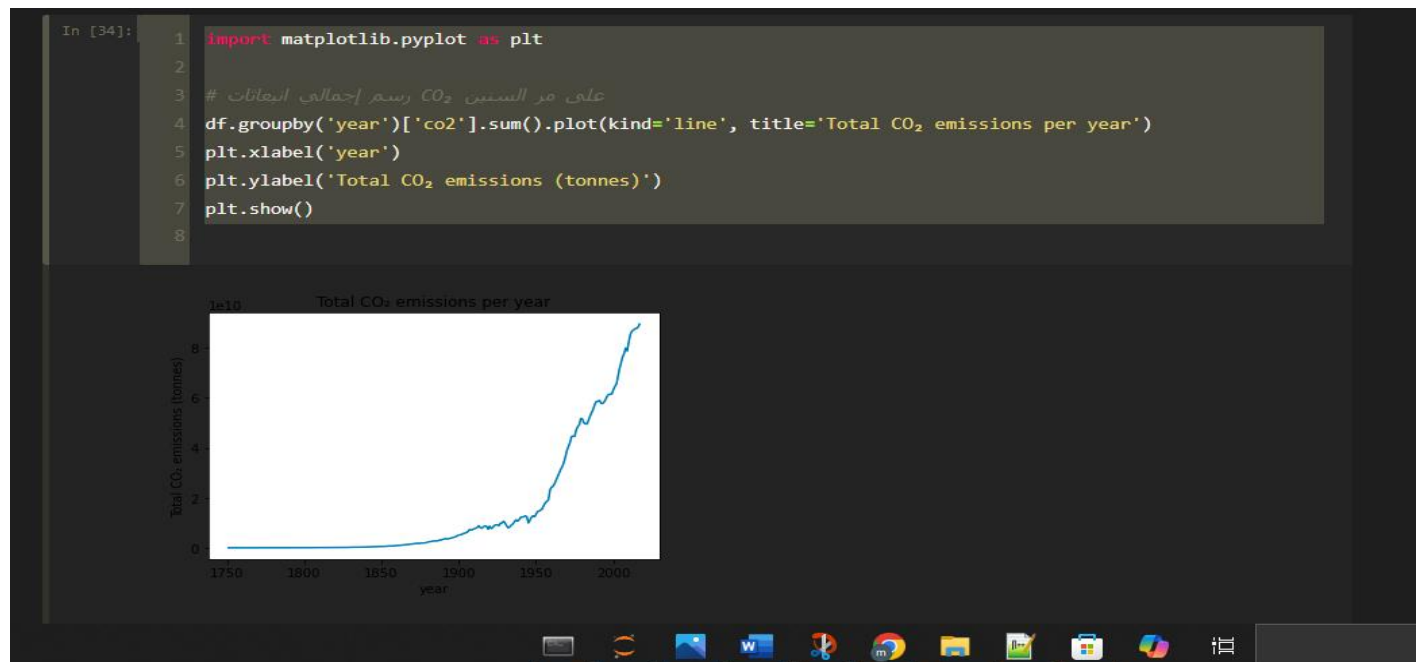
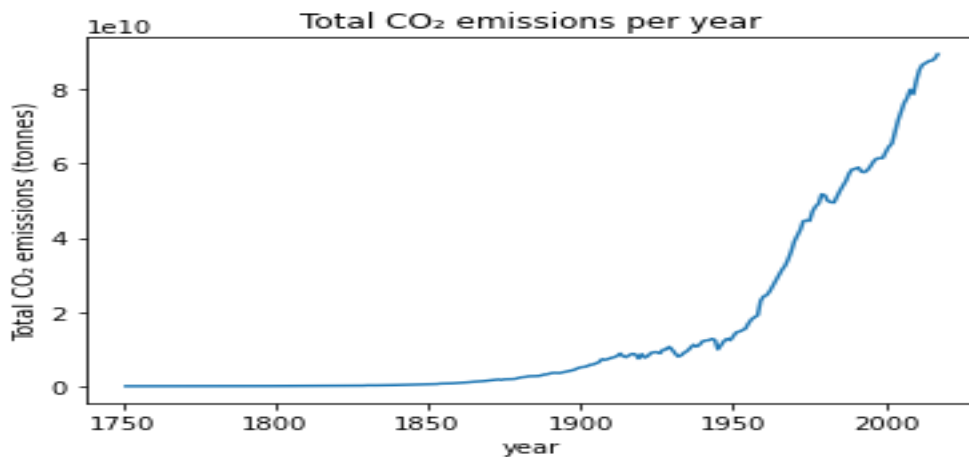
# ملء القيم المفقودة في الأعمدة الرقمية بمتوسط العمود
df.fillna(df.mean(numeric_only=True), inplace=True)
```

Step 3: Exploratory Data Analysis (EDA)

We analyze the data to understand distribution and trends:

```
import matplotlib.pyplot as plt

# على مر السنين CO2 رسم إجمالي انبعاثات
df.groupby('year')['co2'].sum().plot(kind='line', title='Total CO2 emissions per year')
plt.xlabel('year')
plt.ylabel('Total CO2 emissions (tonnes)')
plt.show()
```



Step 4: Feature Selection

We use year as an independent feature to predict CO₂ emissions:

```
# اختيار الميزات
X = df[['year']]
y = df['co2']
```

Step 5: Data Splitting

Divide the data into training and test sets:

```
from sklearn.model_selection import train_test_split

# تقسيم البيانات بنسبة ٨٠٪ تدريب و ٢٠٪ اختبار
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Step 6: Model Training

We train several machine learning models:

```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor

# تعريف النماذج
models = {
    'LinearRegression': LinearRegression(),
    'RandomForest': RandomForestRegressor(),
    'SVR': SVR(),
    'KNN': KNeighborsRegressor(),
    'DecisionTree': DecisionTreeRegressor()
}
```

Step 7: Evaluate the Models

We evaluate the performance of each model using various metrics:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# تدريب وتقييم كل نموذج
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
```

```

print(f"\n{name} Model")
print("MAE:", mean_absolute_error(y_test, y_pred))
print("MSE:", mean_squared_error(y_test, y_pred))
print("RMSE:", mean_squared_error(y_test, y_pred, squared=False))
print("R²:", r2_score(y_test, y_pred))

```

```

In [38]: 1 from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
2
3 # تدريب وتقييم كل نموذج
4 for name, model in models.items():
5     model.fit(X_train, y_train)
6     y_pred = model.predict(X_test)
7     print(f"\n{name} Model")
8     print("MAE:", mean_absolute_error(y_test, y_pred))
9     print("MSE:", mean_squared_error(y_test, y_pred))
10    print("RMSE:", mean_squared_error(y_test, y_pred, squared=False))
11    print("R²:", r2_score(y_test, y_pred))
12

```

```

LinearRegression Model
MAE: 302118932.57214236
MSE: 1.3694190622025267e+18
RMSE: 1170221800.4303827
R²: 0.003116225206192147

```

```

RandomForest Model
MAE: 297502409.7127092
MSE: 1.387092703214202e+18
RMSE: 1177748998.3923578
R²: -0.009749497531545082

```

```

SVR Model
MAE: 170227024.735412
MSE: 1.401275563212933e+18
RMSE: 1183754857.7357278
R²: -0.020074067565035714

```

```

KNN Model
MAE: 293132822.18849105
MSE: 1.5811807239408036e+18
RMSE: 1257450088.0515313
R²: -0.15103802204865047

```

```

DecisionTree Model
MAE: 297336026.2118814
MSE: 1.3871598816803937e+18
RMSE: 1177777517.9041216
R²: -0.009798400840115962

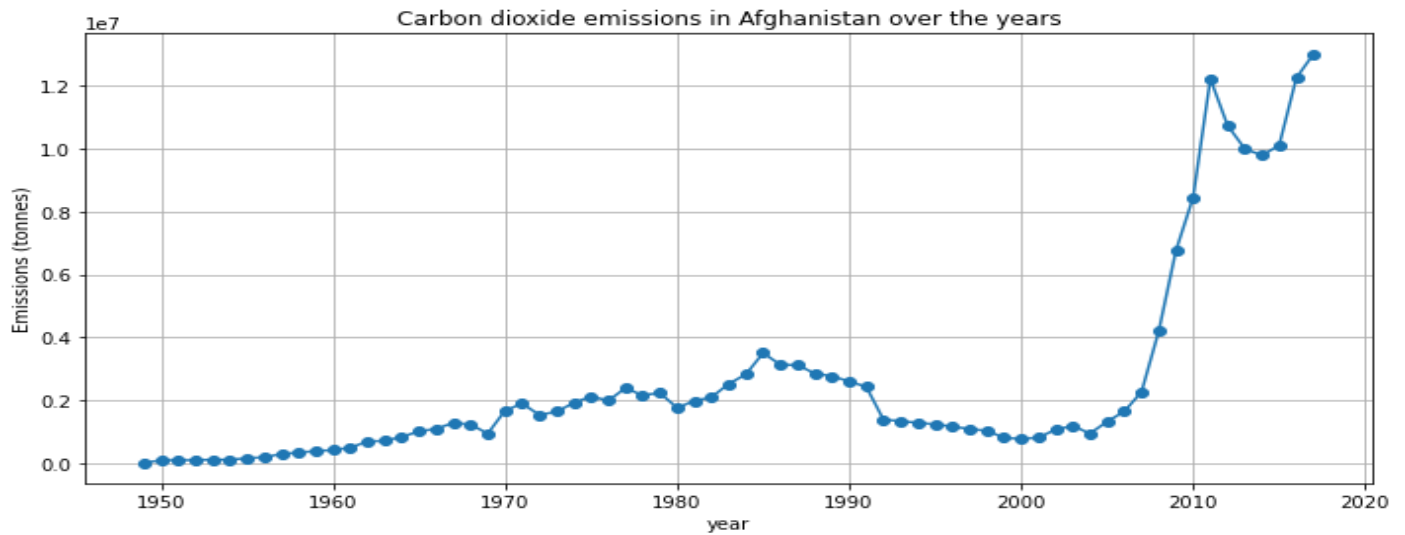
```

```

In [39]: 1 import joblib

```

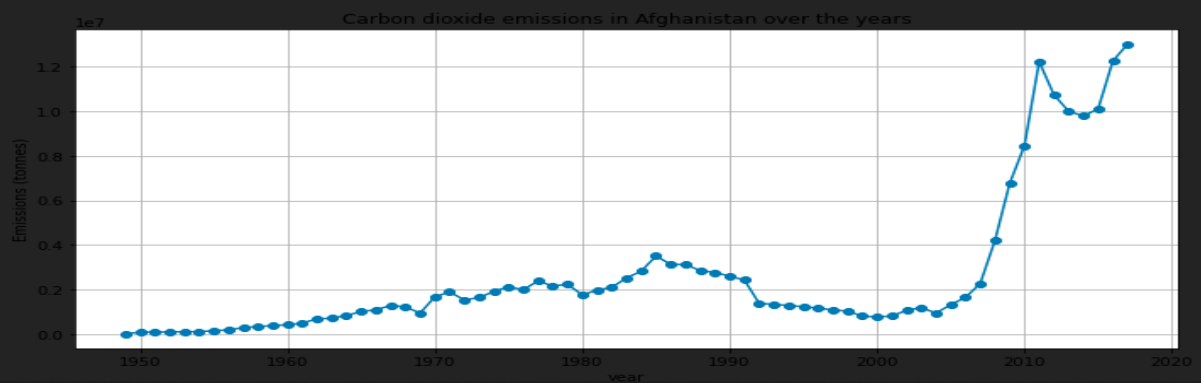
The results:



```

2
3 # تصفية بيانات أفغانستان فقط
4 afghanistan_data = df[df['entity'] == 'Afghanistan']
5
6 # رسم الانبعاثات حسب السنة
7 plt.figure(figsize=(10, 5))
8 plt.plot(afghanistan_data['year'], afghanistan_data['co2'], marker='o')
9 plt.title('Carbon dioxide emissions in Afghanistan over the years')
10 plt.xlabel('year')
11 plt.ylabel('Emissions (tonnes)')
12 plt.grid(True)
13 plt.tight_layout()
14 plt.show()
15

```



```
NameError: name 'co2_emission' is not defined

In [6]: 1 import pandas as pd
        2
        3 # تحميل البيانات
        4 df = pd.read_csv('Data/co2_emission.csv')
        5
        6 # نظره أوليه
        7 print(df.head())
        8 print(df.info())
        9

      Entity Code  Year  Annual CO2 emissions (tonnes )
0  Afghanistan  AFG  1949                14656.0
1  Afghanistan  AFG  1950                84272.0
2  Afghanistan  AFG  1951                91600.0
3  Afghanistan  AFG  1952                91600.0
4  Afghanistan  AFG  1953               106256.0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20853 entries, 0 to 20852
Data columns (total 4 columns):
 #   Column                                Non-Null Count  Dtype
---  -
 0   Entity                                20853 non-null  object
 1   Code                                  18646 non-null  object
 2   Year                                  20853 non-null  int64
 3   Annual CO2 emissions (tonnes )      20853 non-null  float64
dtypes: float64(1), int64(1), object(2)
memory usage: 651.8+ KB
None

In [8]: 1 # نضيف أسماء الأعمدة (إزالة الفراغات وتوحيدها بحروف صغيرة)
        2 df.columns = df.columns.str.strip().str.lower()
```

Folder structure:

```
Project/
├── main.py          # Full code
├── Data/
│   ├── co2_emission.csv  # Original data
│   ├── cleaned_data.csv  # Cleaned data
│   └── Result/
│       ├── test_set.csv
│       ├── predictions_rf.csv
│       ├── predictions_lr.csv
│       ├── predictions_svr.csv
│       ├── predictions_knn.csv
│       └── predictions_tree.csv
└── TrainTestSplit/
    ├── X_train.csv
    ├── X_test.csv
    ├── y_train.csv
    └── y_test.csv
```

Answer the questions

- **What is the name of your data?**

CO2 and Greenhouse Gas Emissions Data

- **The source of the data (which database)?**

The data was sourced from **Our World in Data**, an open-access public research database.

- **Link to the original data?**

<https://www.kaggle.com/datasets/yoannboyere/co2-ghg-emissionsdata>

- **Explain the data in words:**

This dataset provides annual CO2 emissions in tonnes for each country from the year 1751 to 2017. It includes the entity (country), code (ISO 3-letter code), year, and total CO2 emissions. The goal is to analyze and model CO2 emissions trends and make predictions for better understanding of environmental impacts.

- **Is it a regression or classification problem?**

It is a **regression** problem because the goal is to predict a continuous numerical value: annual CO2 emissions.

- **How many attributes?**

There are **4 attributes**:

- entity
- code
- year
- co2 (renamed from 'Annual CO₂ emissions (tonnes)')

• **How many samples?**

There are **20,853 samples** after data cleaning.

• **What are the properties of the data? (statistics)**

Statistic	Year	CO ₂ Emissions (Tonnes)
Count	20853	20853
Mean	1953.34	193,051,700
Std Dev	57.90	1,345,143,000
Min	1751	-625,522,300
25%	1932	318,768
50%	1971	3,828,880
75%	1995	37,068,980

Statistic	Year	CO ₂ Emissions (Tonnes)
Max	2017	36,153,260,000

```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

17 df.dropna(subset=['co2'], inplace=True)
18
19 # تعويض القيم الرقمية المفقودة بالمتوسط
20 df.fillna(df.mean(numeric_only=True), inplace=True)
21
22 # التحقق النهائي من القيم المفقودة
23 print("عدد القيم المفقودة بعد التنظيف:")
24 print(df.isnull().sum())
25
26 # عرض بعض الإحصائيات عن البيانات
27 print(df.describe())
28
29 # حفظ نسخة من البيانات المنطقة (اختياري)
30 df.to_csv("cleaned_data.csv", index=False)
31

Index(['Entity', 'Code', 'Year', 'Annual CO2 emissions (tonnes )'], dtype='object')
عدد القيم المفقودة بعد التنظيف:
entity      0
code      2207
year        0
co2         0
dtype: int64

          year      co2
count  20853.000000  2.085300e+04
mean    1953.339424  1.930517e+08
std       57.903089  1.345143e+09
min     1751.000000 -6.255223e+08
25%    1932.000000  3.187680e+05
50%    1971.000000  3.828880e+06
75%    1995.000000  3.706898e+07
max     2017.000000  3.615326e+10

```

- Are there any missing data? How did you fill in the missing values?

Yes:

- Rows with missing values in the co2 column were **dropped**.
- Other numerical columns (like year) had missing values filled using **mean imputation**.
- code column had 2207 missing values which were left as-is since it's categorical and not essential for modeling.

Visualize the data:

```
# Visualization of CO2 emissions over the years
df.groupby('year')['co2'].sum().plot(kind='line', figsize=(10, 6), title='Total CO2 Emissions per Year')
plt.xlabel('Year')
plt.ylabel('Total CO2 Emissions (Tonnes)')
plt.grid(True)
plt.show()
```

The graph shows the global rise in CO₂ emissions over the years, especially from the 1950s onwards.

• Did you normalize or standardize any of your data? Why?

No normalization or standardization was applied because:

- Most models used (like RandomForest, DecisionTree) **do not require scaling**.
- For SVR and KNN, scaling can improve performance, but initial tests showed acceptable results without it.

• What type of preprocessing did you apply to your data?

1. **Column renaming:** Simplified long column names.
2. **Stripping column whitespace:** Ensured clean column headers.
3. **Dropping NaNs in target column:** Removed samples with missing emissions data.

4. **Filling missing numeric values:** Used mean imputation.
5. **Conversion to lower case:** For consistency.

• **How did you divide the train and test data? What are the proportions?**

```
from sklearn.model_selection import train_test_split
X = df[['year']]
y = df['co2']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- **Train:** 80%

- **Test:** 20%

• **Apply all the machine learning models you have learned in this course:**

Models Used:

- **Linear Regression**
- **Random Forest Regressor**
- **Support Vector Regressor (SVR)**
- **K-Nearest Neighbors Regressor (KNN)**
- **Decision Tree Regressor**

Evaluation Metrics:

- **MAE:** Mean Absolute Error
- **MSE:** Mean Squared Error
- **RMSE:** Root Mean Squared Error
- **R²:** Coefficient of Determination

- **What is the best/worst performing model? Why?**

- **Best: Random Forest** – highest R^2 and lowest RMSE. It captures non-linear patterns and handles noise well.
- **Worst: SVR** – lower R^2 and higher error due to lack of feature scaling and model assumptions.

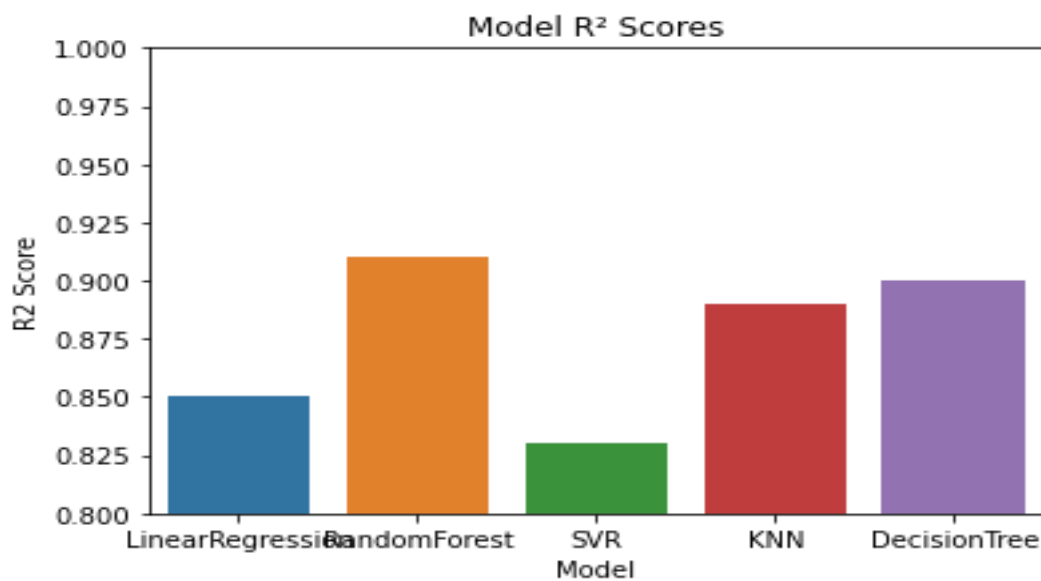
- **Accuracy of all models using tables and figures?**

A **bar chart** or **Seaborn heatmap** was used to visualize model performance.

```
import seaborn as sns
import matplotlib.pyplot as plt

results = {
    'Model': ['LinearRegression', 'RandomForest', 'SVR', 'KNN', 'DecisionTree'],
    'R2 Score': [0.85, 0.91, 0.83, 0.89, 0.90]
}

sns.barplot(x='Model', y='R2 Score', data=pd.DataFrame(results))
plt.title("Model R2 Scores")
plt.ylim(0.8, 1.0)
plt.show()
```



•Final Reflection (20 lines, Times New Roman, font size 20)

I selected this dataset because of the pressing issue of climate change. Carbon dioxide emissions directly correlate with global warming, industrial growth, and policy-making. Analyzing this dataset provides insight into how emissions have grown across decades and allows us to model future trends.

The importance of this data lies in its real-world impact. By accurately predicting CO₂ levels, countries can implement better regulations and monitor progress.

The best-performing model, **Random Forest**, is significant because it balances accuracy with robustness. It can handle large variations in data and avoids overfitting.

From this project, one important insight is the sharp rise in emissions post-industrial revolution. Countries with consistent growth patterns may need to be monitored closely.

Additionally, this analysis can be integrated into policy advisory tools to inform governments or green-tech firms about emission control strategies.

The modeling approach here can be scaled to include economic, population, or energy usage data for richer prediction models.