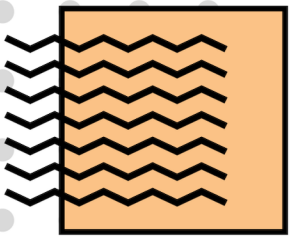
ASSIGNMENT PRESENTATION



YEARLY CARBON DIOXIDE EMISSION

ABDELKARIM ALI HASSAN (1211302792)
NUR AINA MAISARAH BINTI ZULKIFLI (1211303950)

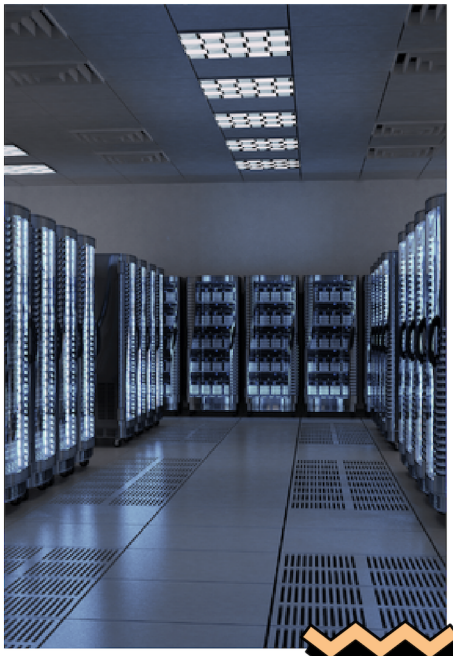
INTRODUCTION

PROBLEM STATEMENT

The release of carbon dioxide from the combustion of fuels will increase the concentration of carbon dioxide to the atmosphere. Which will supercharge the natural greenhouse effect, causing the temperature to rise. This will also lead to the increasing of sea temperature which will melt the glacier and sea ice.

OBJECTIVES:

1. To predict the carbon dioxide emission in the world.
2. To reduce the usage of combustible fuels.



DATA PREPROCESSING

The goal of the data preprocessing is to prepare raw data for further analysis. This report covers the data preprocessing steps of cleaning, transforming, and reducing the data.

CONTENT

- Loading data
- Data cleaning
- Data transformation
- Data reduction

LOADING DATA

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.model_selection import train_test_split as split
import warnings
warnings.filterwarnings('ignore')

plt.style.use('fivethirtyeight')
%matplotlib inline
pd.set_option('display.max_columns', 26)
```

→ import the necessary function

```
1 df = pd.read_csv('owid-co2-data.csv')
```

→ used to load our data

DATA CLEANING

```
# checking for any missing values
df.isna().sum()
```

```
iso_code                                4029
country                                    0
year                                       0
co2                                     1319
```

```
df = df.dropna()
print(df)
```

```
      iso_code    country  year        co2  co2_per_capita  trade_co2  \
1565       AUS  Australia  1990    279.365          16.471    -33.458
1566       AUS  Australia  1991    281.872          16.398    -34.355
1567       AUS  Australia  1992    286.332          16.454    -36.551
1568       AUS  Australia  1993    291.325          16.550    -40.025
1569       AUS  Australia  1994    300.084          16.860    -37.466
...        ...        ...   ...        ...             ...        ...
```

- Data cleaning involves identifying and correcting errors, inconsistencies, and missing values in the data

EXTRA CODING

```
df.describe().style.background_gradient(cmap = 'copper')
```

	year	co2	co2_per_capita	trade_co2	cement_co2	cement_co2_per_capita	
count	1128.000000	1128.000000	1128.000000	1128.000000	1128.000000	1128.000000	11
mean	2004.093085	550.405057	9.073804	20.266298	12.493159	0.210470	1
std	8.376522	2480.059924	4.609500	100.931134	90.254167	0.160924	9
min	1990.000000	7.081000	1.468000	-1167.005000	0.001000	0.000000	
25%	1997.000000	44.084750	5.963500	-0.854750	0.832750	0.126000	
50%	2004.000000	95.665000	8.190500	7.626000	2.108000	0.176000	
75%	2011.000000	360.852750	11.025250	31.850750	6.990000	0.249000	1
max	2018.000000	36646.140000	32.206000	543.504000	1566.740000	1.413000	147

*refer to the coding slide to see the result in details

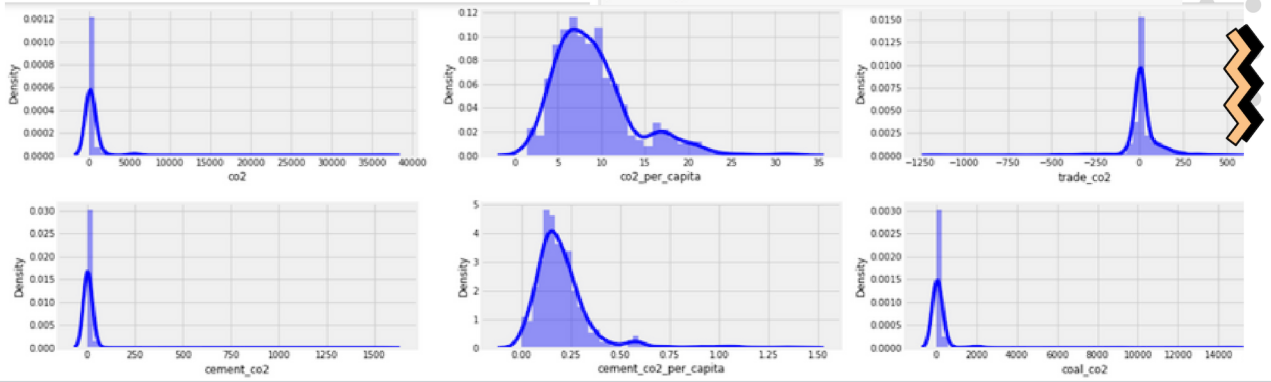
- For this step it is just to show the output for: standard deviation, minimum, at (25%), at (50%), at (75%),and at maximum

DATA TRANSFORMATION

Data transformation involves converting the data into a format that is suitable for analysis.

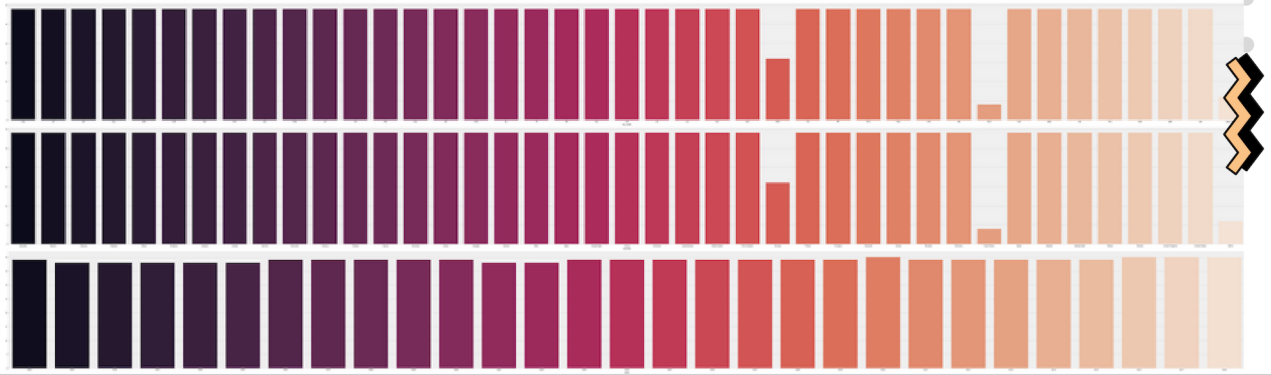
```
num_cols = [col for col in df.columns if df[col].dtype == 'float64']
cat_cols = [col for col in df.columns if df[col].dtype != 'float64']
```

```
# checking numerical features distribution
plt.figure(figsize = (20, 80))
plotnumber = 1
for column in num_cols:
    if plotnumber <= 80:
        ax = plt.subplot(27, 3, plotnumber)
        sns.distplot(df[column],color='blue',)
        plt.xlabel(column)


    plotnumber += 1

plt.tight_layout()
plt.show()
```



DATA TRANSFORMATION

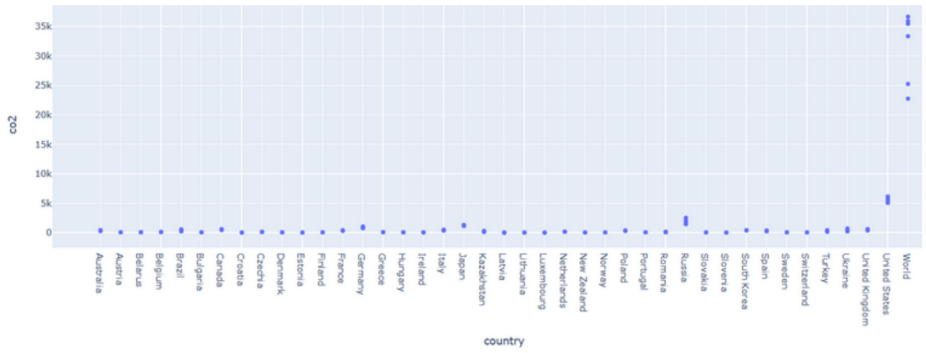
```
# looking at categorical columns

plt.figure(figsize = (100, 30))
plotnumber = 1
for column in cat_cols:
    if plotnumber <= 11:
        ax = plt.subplot(3, 1, plotnumber)
        sns.countplot(df[column], palette = 'rocket',color='black')
        plt.xlabel(column)

    plotnumber += 1

plt.tight_layout()
plt.show()
```



DATA REDUCTION

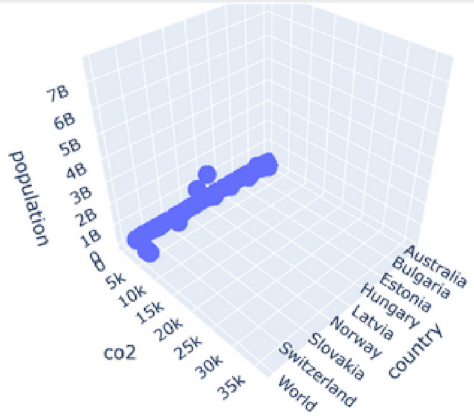
Data reduction involves reducing the dimensionality of the data by removing redundant or irrelevant information.

```
px.scatter(df, x="country", y="co2")
```



DATA REDUCTION

```
px.scatter_3d(df, x='country', y='co2', z='population')
```




```
df2 = df[['country', 'year', 'co2', 'population']]
df2
```

	country	year	co2	population
1565	Australia	1990	279.365	1.696060e+07
1566	Australia	1991	281.872	1.718924e+07
1567	Australia	1992	286.332	1.740218e+07
1568	Australia	1993	291.325	1.760321e+07
1569	Australia	1994	300.084	1.779853e+07
...	...	...	...	...
25708	World	2000	25234.207	6.143777e+09
25718	World	2010	33343.300	6.957138e+09
25724	World	2016	35452.459	7.464344e+09
25725	World	2017	35925.738	7.548183e+09
25726	World	2018	36646.140	7.631091e+09

1128 rows × 4 columns

MODEL BUILDING

CONTENT

- Handling categorical data
- Data splitting
- Train regression model with linear regression and k-NN
- Validate the performance of the model

LETS GET STARTED

HANDLING CATEGORICAL DATA

```
1 # convert categorical data into numerical values
2 df3 = pd.get_dummies(df2)
3 df3
```

- Convert catogorical data into numerical values

country	year	co2	population
Australia	1990	279.365	1.696060e+07
Australia	1991	281.872	1.718924e+07
Australia	1992	286.332	1.740218e+07
Australia	1993	291.325	1.760321e+07
Australia	1994	300.084	1.779853e+07

year	co2	population	country_Australia	country_Austria	country_Belarus
1990	279.365	1.696060e+07	1	0	0
1991	281.872	1.718924e+07	1	0	0
1992	286.332	1.740218e+07	1	0	0
1993	291.325	1.760321e+07	1	0	0
1994	300.084	1.779853e+07	1	0	0

SPLITTING DATASET

- We split the data into features (X) and target (y)

```
#split data into features (X) and target (y)
y = df3['co2'].values
del df3['co2']
X = df3.values
```

```
# split data into training and testing sets
X_train, X_test, y_train, y_test = split(X, y, test_size=0.2, random_state=42)
print(X_train.shape, X_test.shape)
```

```
(902, 43) (226, 43)
```

- We split the dataset into training and testing sets

WHY DOES THE TRAIN SET HAS MORE DATA ?

We include more training data to avoid overfitting.

TRAIN AND VALIDATE MODEL

- Train a linear regression model with training set
- Then, validate the performance of the model with testing set

```
# train and validate a linear regression model
lnr = LinearRegression().fit(X_train, y_train)

print(f"lnr score: {lnr.score(X_test, y_test)}")
```

```
lnr score: 0.999267287531636
```

TRAIN AND VALIDATE MODEL

- Train a k-NN regression model with training set
- Then, validate the performance of the model with testing set

```
# train and validate k-NN regression model with k from 1 to 10
for k in range(1, 11):
    knn = KNeighborsRegressor(n_neighbors=k).fit(X_train, y_train)
    print(f"knn {k} score: {knn.score(X_test, y_test)}")
```

```
knn 1 score: 0.9986006692636915
knn 2 score: 0.9986131274374814
knn 3 score: 0.9991048611656329
knn 4 score: 0.9939522281270994
knn 5 score: 0.9832276361009097
knn 6 score: 0.9409355071503079
knn 7 score: 0.8941513390349918
knn 8 score: 0.8505591268859661
knn 9 score: 0.8116259210501161
knn 10 score: 0.7768625922721673
```