

# Prediction Of Oil Price

Present by team 4

## Team members

- Sanket raut
- Mangalparthy Mahesh
- R.Vishnu Vardhan Reddy
- S Muhaideen abdul kader
- D.Vikram kanth
- Mohamed Rehnaz Ashraf

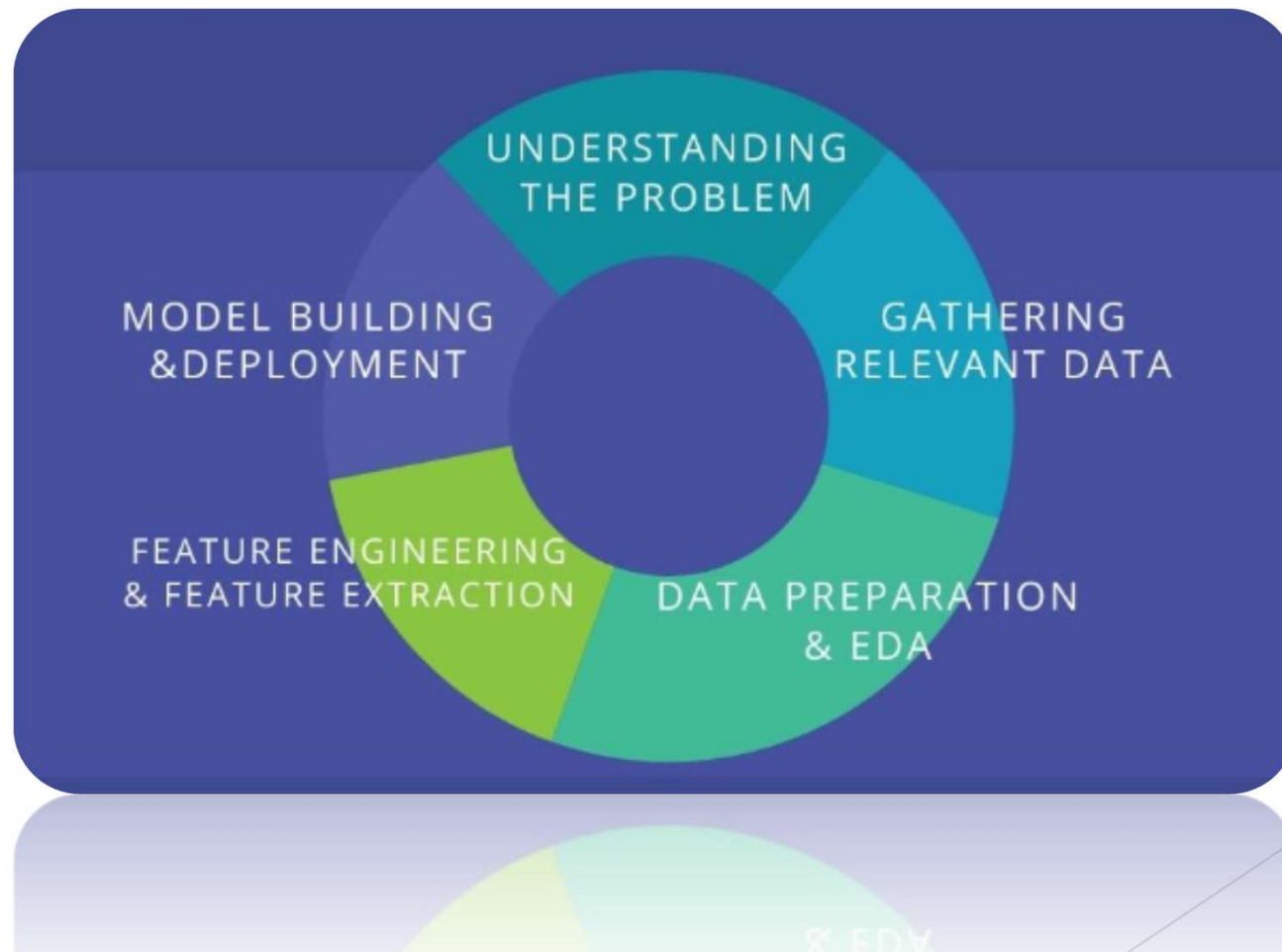
Date : 27/06/2023

## Objective :

Oil prices are often influenced by factors outside of immediate information, which makes it difficult to predict them accurately. These prices can have a significant impact on the economy. Our model aims to analyze the patterns in oil prices to assist customers and businesses in making informed decisions.



# Project Architecture / Project Flow



# Dataset



Source: U.S. Energy  
Information  
Administration

## Link

[Crude Oil Prices: West Texas Intermediate \(WTI\) - Cushing, Oklahoma \(DCOILWTICO\) | FRED | St. Louis Fed \(stlouisfed.org\)](https://fred.stlouisfed.org/series/DCOILWTICO)

# Exploratory Data Analysis (EDA)

The background of the slide features abstract, overlapping geometric shapes in various shades of green, ranging from light lime to dark forest green. These shapes are primarily located on the right side and bottom, creating a modern, layered effect. A thin, light gray line also extends diagonally across the lower right portion of the slide.

# In this dataset

- ✓ The dataset consists of 2 columns: "Date" and "Price".
- ✓ The dataset contains 2,921 entries.
- ✓ The "Date" column is of the datetime data type.
- ✓ The "Price" column is of the float data type.
- ✓ The "Price" column has 2,812 non-null values, indicating some missing data.
- ✓ The average price in the dataset is approximately 67.92.
- ✓ The standard deviation of the prices is around 22.58.
- ✓ The minimum price recorded in the dataset is 36.98.
- ✓ The 25th percentile of prices is 49.59, meaning 25% of the prices are below this value.
- ✓ The median price is 63.45, indicating that 50% of the prices are below this value.
- ✓ The 75th percentile of prices is 88.92, meaning 75% of the prices are below this value.
- ✓ The maximum price recorded in the dataset is 123.64.

	Date	Price
0	2012-04-02	105.25
1	2012-04-03	104.02
2	2012-04-04	101.53
3	2012-04-05	103.29
4	2012-04-06	NaN
...	...	...
2916	2023-06-06	71.71
2917	2023-06-07	72.52
2918	2023-06-08	71.28
2919	2023-06-09	70.16
2920	2023-06-12	67.08

2921 rows × 2 columns

df.describe()	
	Price
count	2812.000000
mean	67.922219
std	22.579956
min	-36.980000
25%	49.590000
50%	63.455000
75%	88.922500
max	123.640000

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2921 entries, 0 to 2920
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  -
 0    Date   2921 non-null   datetime64[ns]
 1    Price  2812 non-null   float64
dtypes: datetime64[ns](1), float64(1)
memory usage: 45.8 KB
```

# Feature Engineering

```
In [2]: # Loading dataset

file_path = 'D:\\projects\\datasets\\DCOILWTICO (1).xls'

# Read the .xls file using pandas
df = pd.read_excel(file_path)
df['Date'] = pd.to_datetime(df['Date'])
```

- Assigns the converted datetime values back to the 'Date' column in the DataFrame.
- Conversion to datetime format allows for easier manipulation and analysis of dates.
- The code prepares the dataset for further analysis and manipulation by ensuring the 'Date' column is treated as a date/time data type.

# Handle Missing Data

## ✓ Before handling missing values:

1. The "Date" variable has no missing values.
2. The "Price" variable has 109 missing values.

## ✓ After handling missing values:

1. Both the "Date" and "Price" variables have no missing values.
2. The "Year" variable also has no missing values.

Total number of values: 2921 and Number of null values in the "Price" variable: 109  
Percentage of null values in the "Price" variable:  $(109 / 2921) * 100 = 3.73\%$   
approximately 3.73% of the values in the "Price" variable are null.

```
In [19]: df.isnull().sum()
Out[19]: Date      0
         Price    109
         Year      0
         dtype: int64

In [20]: # Drop rows with missing values
         df.dropna(inplace=True)

In [22]: df.isnull().sum()
Out[22]: Date      0
         Price      0
         Year      0
         dtype: int64
```



# Library

## **Pandas**

- Pandas is used for data manipulation, indexing, filtering, updating, and preprocessing
- It allows operations such as data alignment, column selection, visualization, and prioritized splitting.

## **Matplotlib**

- Matplotlib is used in Python for generating graphics and visualizations, including plots, charts, and figures.
- It supports various types of plots, including bar graphs, line graphs, pie charts, scatter plots, histograms, and more.

## **Seaborn**

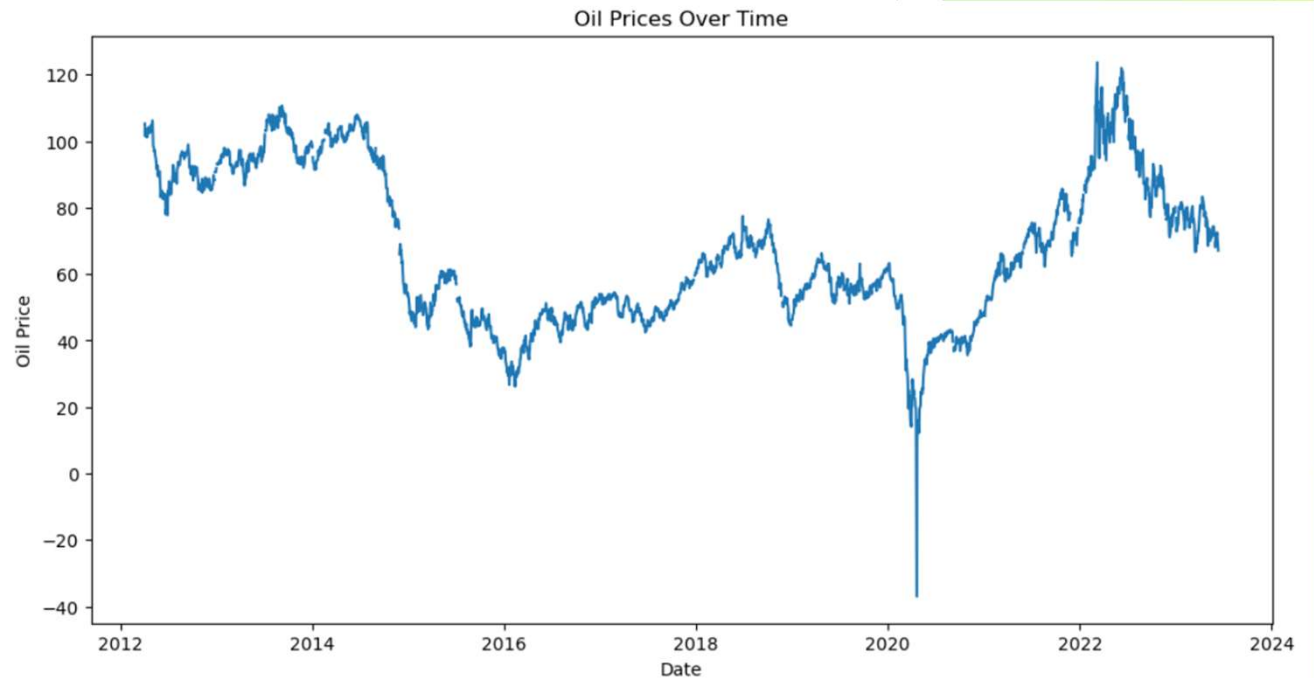
- Seaborn is used for data visualization, data exploration, and chart customization in data science.
- It enables dynamic updating, exploratory development, partitioning usage, and the creation of various plots like box plots, violin plots, swarm plots, and others.

## **Scikit-learn (sklearn)**

- Scikit-learn is a powerful machine learning library used for various tasks like classification, regression, clustering, and dimensionality reduction.
- It provides tools for model selection, preprocessing, evaluation, and scoring, along with a wide range of machine learning algorithms and utilities.

# Visualization [ Trend ]

- ✓ Oil prices were relatively high, ranging between 100 to 120 units, in 2012.
- ✓ Subsequently, there was a consistent downward trend in oil prices.
- ✓ In the first month of 2020, there was a sharp decline, with oil prices hitting a low point of around 30 units.
- ✓ However, in 2023, there has been an upward trend, suggesting a recent increase in oil prices.



# Outlier's

## 1.Outlier Percentage:

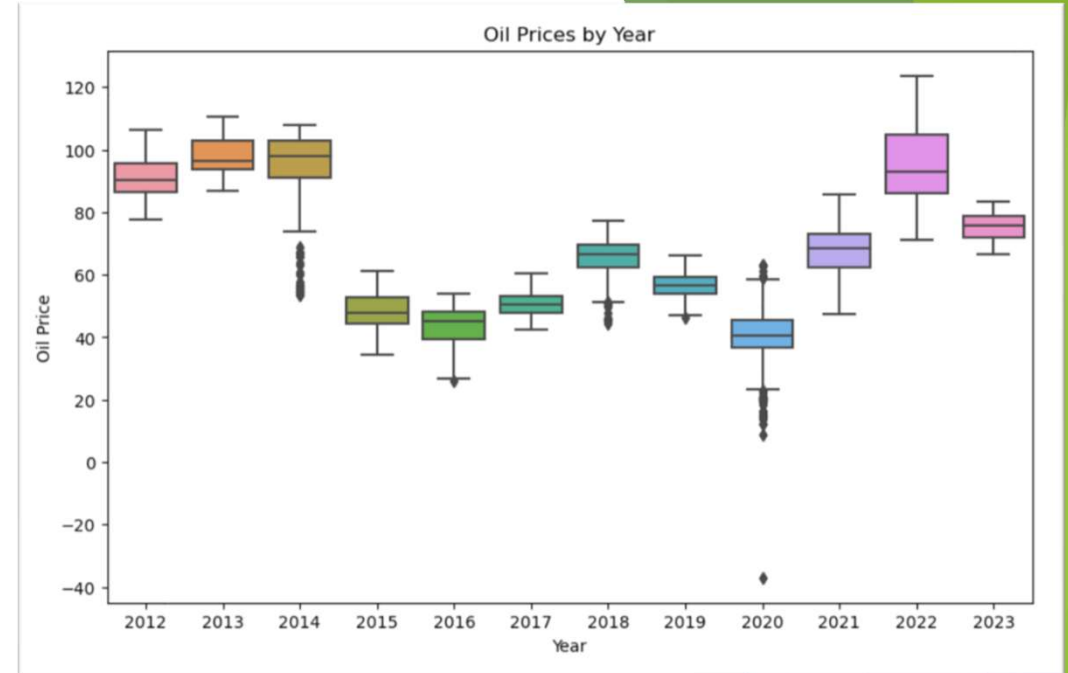
After analyzing the data, it has been determined that only 0.04% of the data points are considered outliers.

## 1.Definition of Outliers:

Outliers are data points that significantly deviate from the rest of the dataset. They can be unusually high or low values compared to the majority of the data.

## 1.Impact on Data:

2.Since the percentage of outliers is extremely low (0.04%), their presence has a minimal effect on the overall dataset.



```
import numpy as np
```

```
Q1 = np.percentile(df['Price'], 25)
```

```
Q3 = np.percentile(df['Price'], 75)
```

```
IQR = Q3 - Q1
```

```
lower_bound = Q1 - (1.5 * IQR)
```

```
upper_bound = Q3 + (1.5 * IQR)
```

```
outliers = df[(df['Price'] < lower_bound) | (df['Price'] > upper_bound)]
```

```
percentage_of_outliers = (len(outliers) / len(df)) * 100
```

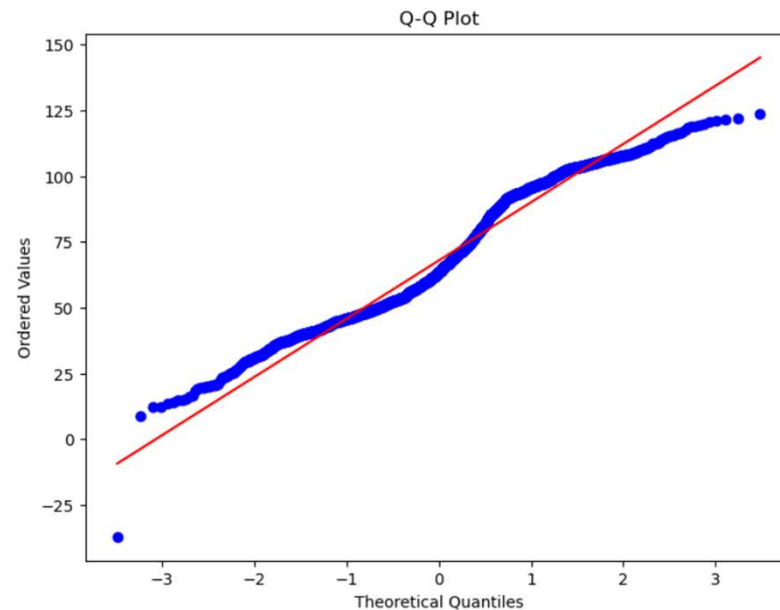
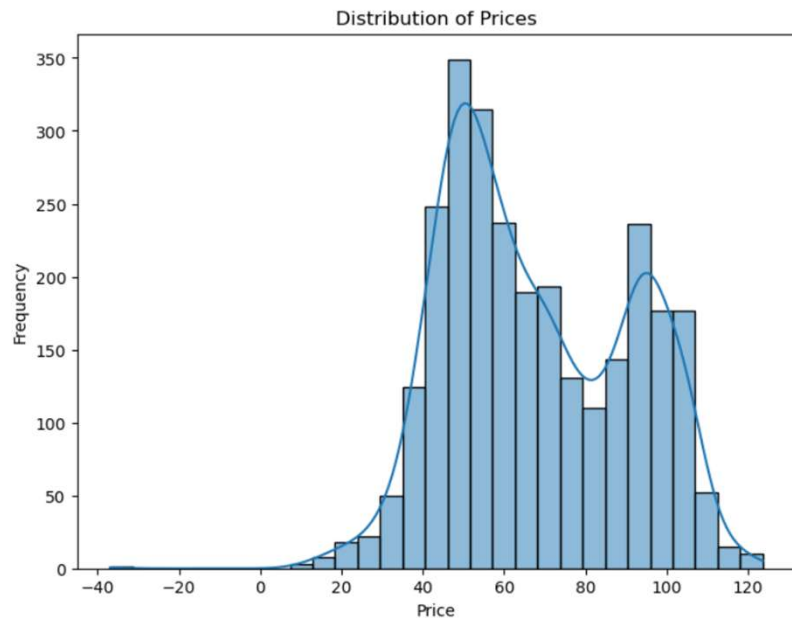
```
print("Percentage of outliers: {:.2f}%".format(percentage_of_outliers))
```

```
Percentage of outliers: 0.04%
```

```
percentage of outliers: 0.04%
```

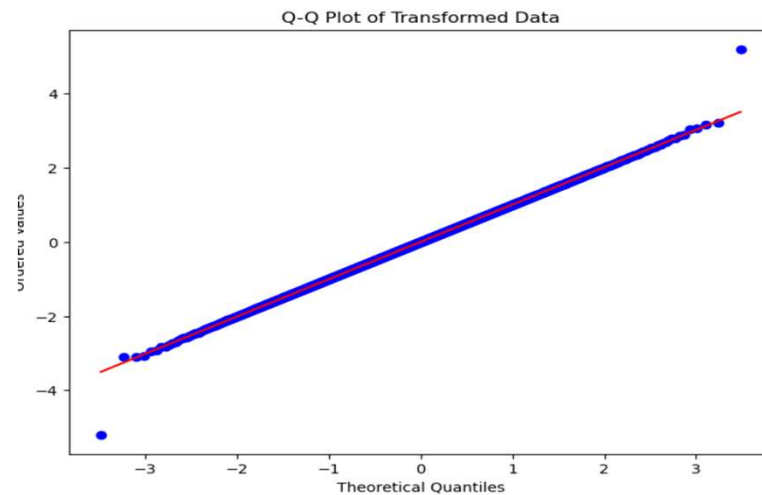
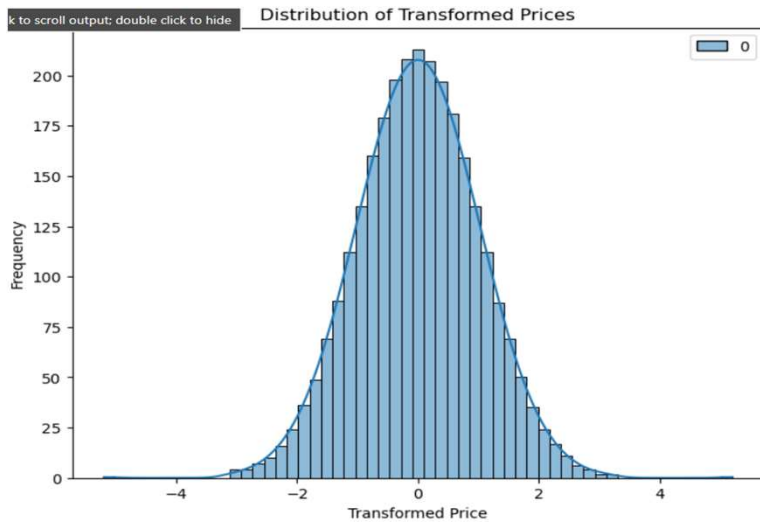
# Distribution of data

Data is not normally distributed (reject  $H_0$ )



The dataset is not normally distributed, as indicated by the non-linear trend in the Q-Q plot, deviating from the expected line for a normal distribution.

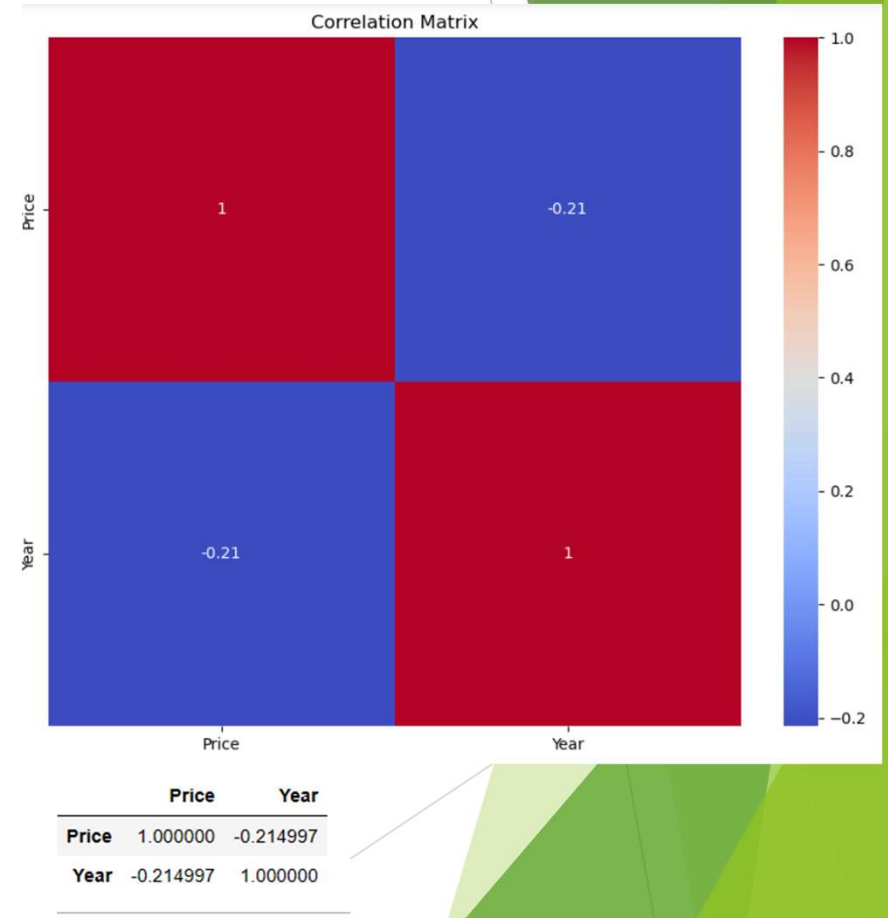
# Normal distribution

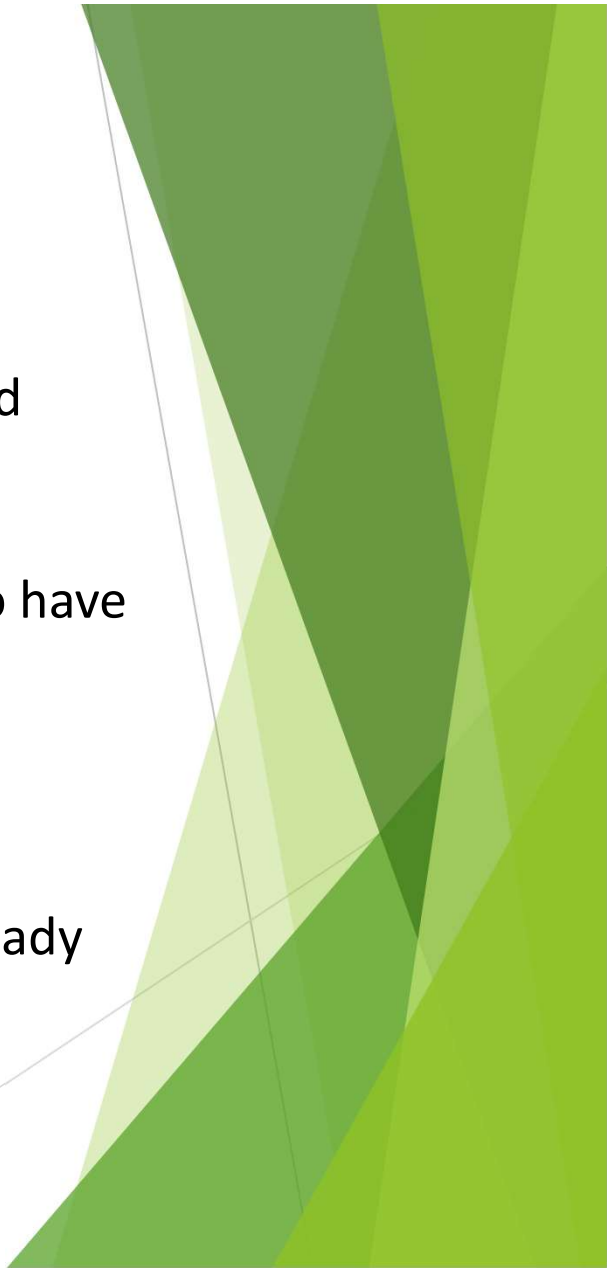


- ✓ Use scikit-learn's **QuantileTransformer** to transform 'Price' column.
- ✓ Reshape 'Price' column to match the input format.
- ✓ Transformed data now follows a normal distribution.
- ✓ Use transformed data for analysis or modeling, considering normal distribution assumptions.
- ✓ Apply quantile transformation using **fit\_transform** method.

# Correlation

- ✓ Weak negative correlation (-0.215) between 'Price' and 'Year'.
- ✓ 'Price' tends to decrease slightly as 'Year' increases.
- ✓ Correlation is not strong or significant.
- ✓ Correlation does not imply causation.
- ✓ Visualization helps identify potential relationships and patterns.



- 
- EDA (Exploratory Data Analysis) was performed on the dataset.
  - Missing or null values in the dataset were identified and replaced with appropriate values.
  - Outliers in the data were detected, but they were determined to have no significant impact on the dataset, so they were retained.
  - The data was transformed to achieve a normal distribution.
  - With these steps completed, the dataset is now prepared and ready
    - for further processing or analysis.

**Thank you**

