UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

ANALYSE TREND AND CHANGES IN PRICES OF GASOLINE

Presented by:
Phạm Mạnh Hùng - 21520901
Phùng Thiên Phúc - 21521297
Chu Ngọc Thăng - 20521891

OVERVIEW

01

Overview

02

Application algorithm for dataset

03

Economic Implications

04

Conclusion

I. OVERVIEW

1. Reason for choosing the topic

Background on gasoline as a crucial commodity

- Gasoline plays a pivotal role in modern society
- Its availability and price significantly affect global economies, individual households.
- Understanding these factors is essential to grasp why gasoline prices fluctuate and how they influence broader economic patterns.

1. Reason for choosing the topic

Importance of Understanding Price Trends

- Understanding gasoline price trends is crucial because fluctuations in prices directly impact consumers and businesses.
- For consumers, higher gasoline prices lead to increased transportation costs, affecting household budgets.
- For businesses, especially those dependent on transportation, changes in gasoline prices can increase operating costs, potentially resulting in higher prices for goods and services passed on to consumers.

1. Reason for choosing the topic

Brief Overview of Factors Influencing Gasoline Prices

- Gasoline prices are influenced by a range of factors, including economic and geopolitical conditions.
- Price increases may indicate rising demand, supply disruptions, or instability in oil-producing regions.
- Recognizing these trends helps consumers, businesses, and policymakers make informed decisions and prepare for potential economic shifts.

1. Reason for choosing the topic

About Dataset

- This is a dataset of gasoline prices in the US market from 1995 to 2021.
- Regular, Midgrade, Premium: These are classifications based on the octane rating of the gasoline:
  - R2: Regular gasoline. Typically has an octane rating of 87
  - M2: Midgrade gasoline. Usually has an octane rating of 89.
  - P2: Premium gasoline. Has a higher octane rating, usually 91 or 93.

Date	R2	M2	P2
1/2/1995	1.063	1.159	1.25
1/9/1995	1.07	1.164	1.256
1/16/1995	1.062	1.155	1.249
1/23/1995	1.068	1.165	1.256
1/30/1995	1.068	1.163	1.255
2/6/1995	1.062	1.157	1.25
2/13/1995	1.058	1.153	1.243
2/20/1995	1.052	1.148	1.239
2/27/1995	1.06	1.153	1.246
3/6/1995	1.063	1.157	1.244
3/13/1995	1.056	1.15	1.238
3/20/1995	1.055	1.149	1.236
3/27/1995	1.063	1.153	1.241
4/3/1995	1.077	1.167	1.255
4/10/1995	1.094	1.186	1.273
4/17/1995	1.11	1.201	1.29
4/24/1995	1.133	1.224	1.315
5/1/1995	1.141	1.234	1.323
5/8/1995	1.164	1.257	1.349
5/15/1995	1.173	1.267	1.357
5/22/1995	1.191	1.285	1.376
5/29/1995	1.193	1.288	1.379
6/5/1995	1.194	1.288	1.38

II. APPLICATION ALGORITHM FOR DATASET

1. STATISTICAL DESCRIPTIVE METHODS

Step 1: Read csv file

```python
import pandas as pd

# Load the data
file_path = "C:/Users/cnt02/OneDrive/Máy tính/Phân tích dữ liệu kinh doanh/Project/PET_PRI_GND_DCUS_NUS_W.csv"
data = pd.read_csv(file_path)

# Display the first few rows to understand its structure
data.head()
```

	Date	R2	M2	P2
0	1/2/1995	1.063	1.159	1.250
1	1/9/1995	1.070	1.164	1.256
2	1/16/1995	1.062	1.155	1.249
3	1/23/1995	1.068	1.165	1.256
4	1/30/1995	1.068	1.163	1.255

1. STATISTICAL DESCRIPTIVE METHODS

Step 2: Data description

```
data.info()
```

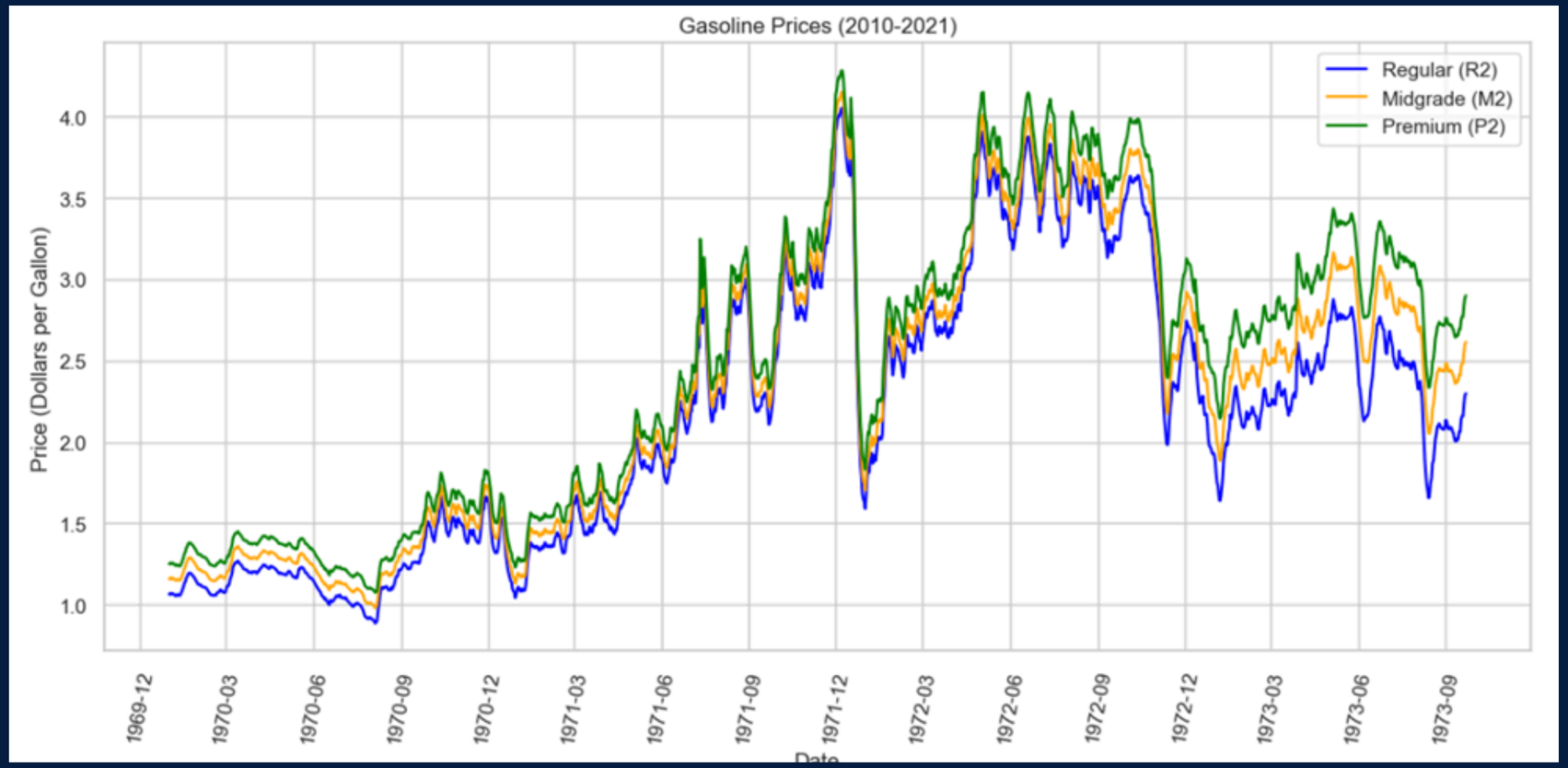
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1361 entries, 0 to 1360
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Date    1361 non-null   object 
 1   R2      1361 non-null   float64
 2   M2      1361 non-null   float64
 3   P2      1361 non-null   float64
dtypes: float64(3), object(1)
memory usage: 42.7+ KB
```

```
data.describe()
```

	R2	M2	P2
count	1361.000000	1361.000000	1361.000000
mean	2.178511	2.320970	2.472096
std	0.835549	0.858521	0.894472
min	0.885000	0.979000	1.074000
25%	1.393000	1.482000	1.573000
50%	2.175000	2.404000	2.640000
75%	2.765000	2.930000	3.127000
max	4.054000	4.153000	4.283000

1. STATISTICAL DESCRIPTIVE METHODS

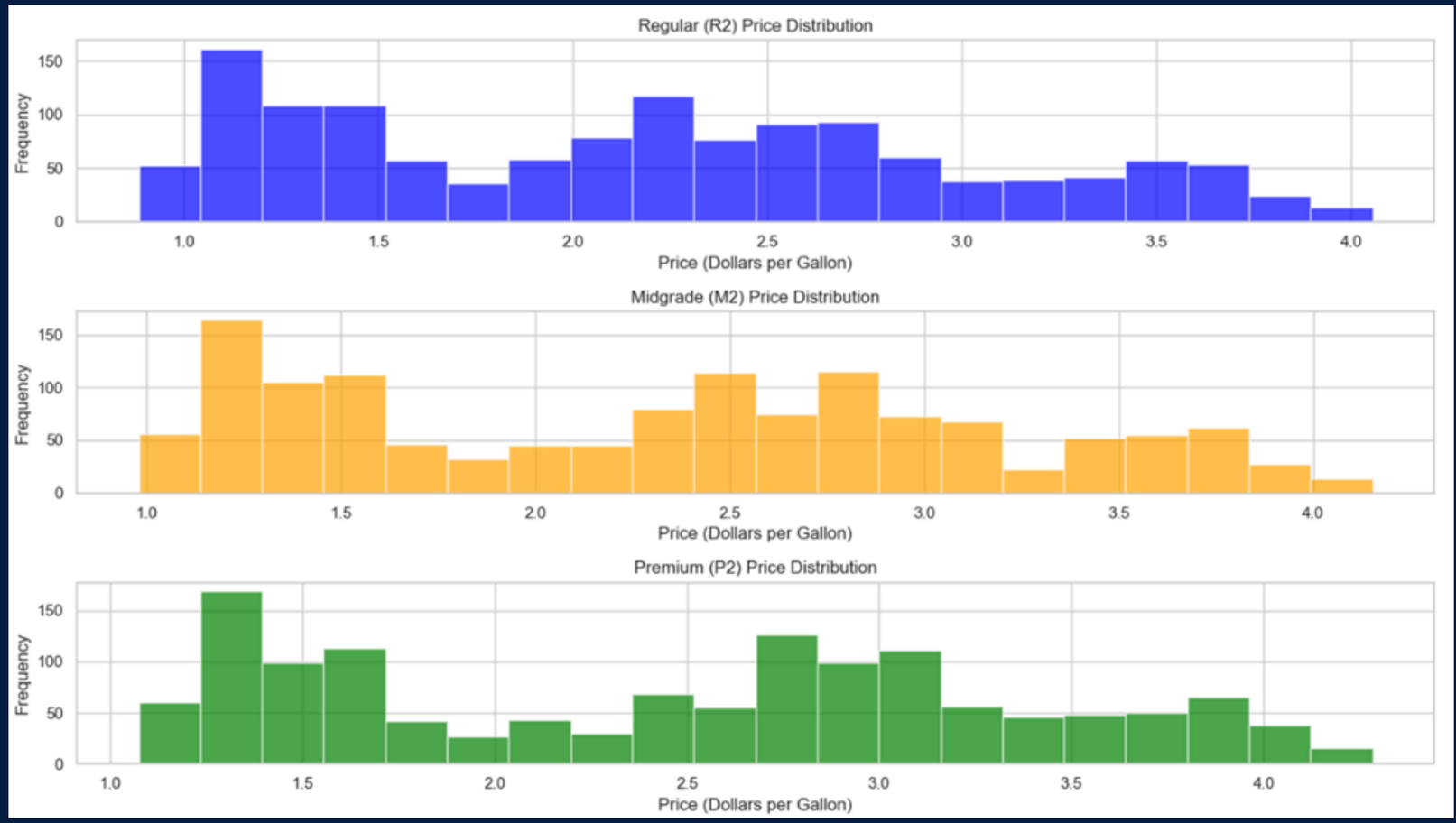
Step 3: Data statistics



Line plot for price trends over time

1. STATISTICAL DESCRIPTIVE METHODS

Step 3: Data statistics



Histogram for frequency distribution of gas prices

2. HYPOTHESIS TESTING METHODS

- T-Test

I will perform an Independent samples t-test and choose confidence interval =95% to compare the average prices of each pair of gasoline grades (Regular, Midgrade, Premium). I will test each pair as follows:

- Regular vs. Midgrade
- Regular vs. Premium
- Midgrade vs. Premium

2. HYPOTHESIS TESTING METHODS

- T-Test

```python
from scipy.stats import ttest_ind

# Extracting data columns
regular_prices = data['R2'].dropna()
midgrade_prices = data['M2'].dropna()
premium_prices = data['P2'].dropna()

# Conducting t-tests
t_test_regular_midgrade = ttest_ind(regular_prices, midgrade_prices)
t_test_regular_premium = ttest_ind(regular_prices, premium_prices)
t_test_midgrade_premium = ttest_ind(midgrade_prices, premium_prices)

t_test_results = {
    'Regular vs Midgrade': t_test_regular_midgrade,
    'Regular vs Premium': t_test_regular_premium,
    'Midgrade vs Premium': t_test_midgrade_premium
}

t_test_results
```

2. HYPOTHESIS TESTING METHODS

- T-Test

```python
from scipy.stats import ttest_ind

# Extracting data columns
regular_prices = data['R2'].dropna()
midgrade_prices = data['M2'].dropna()
premium_prices = data['P2'].dropna()

# Conducting t-tests
t_test_regular_midgrade = ttest_ind(regular_prices, midgrade_prices)
t_test_regular_premium = ttest_ind(regular_prices, premium_prices)
t_test_midgrade_premium = ttest_ind(midgrade_prices, premium_prices)

t_test_results = {
    'Regular vs Midgrade': t_test_regular_midgrade,
    'Regular vs Premium': t_test_regular_premium,
    'Midgrade vs Premium': t_test_midgrade_premium
}

t_test_results
```

2. HYPOTHESIS TESTING METHODS

- **T-Test**

Result :

```
{'Regular vs Midgrade': TtestResult(statistic=-4.386932072771401, pvalue=1.1933482054833031e-05, df=2720.0),
 'Regular vs Premium': TtestResult(statistic=-8.8485970535252, pvalue=1.562438179642916e-18, df=2720.0),
 'Midgrade vs Premium': TtestResult(statistic=-4.49689374064071, pvalue=7.184203642719685e-06, df=2720.0)}
```

Conclusion: With very small p-values ($p < 0.05$) for all pairs, we can conclude that the difference in average price between the pairs of gasoline grades (Regular, Midgrade, and Premium) is significant.

2. HYPOTHESIS TESTING METHODS

- **ANOVA**

To perform an ANOVA test to compare the prices of these three gasoline types over time, we'll follow these steps:

1. Reshape the data to combine all prices into a single column, with a separate column indicating the type of gasoline. Choose confidence interval =95%.
2. Conduct a one-way ANOVA to determine if there is a statistically significant difference in prices among the gasoline types.

2. HYPOTHESIS TESTING METHODS

- ANOVA

```python
import numpy as np
import pandas as pd
from scipy.stats import f


# Calculate summary statistics for each group
group_counts = [len(regular_prices), len(midgrade_prices), len(premium_prices)]
group_sums = [regular_prices.sum(), midgrade_prices.sum(), premium_prices.sum()]
group_averages = [regular_prices.mean(), midgrade_prices.mean(), premium_prices.mean()]
group_variances = [regular_prices.var(ddof=1), midgrade_prices.var(ddof=1), premium_prices.var(ddof=1)]

# Calculate ANOVA table values
SS_between = sum(count * (avg - np.mean(group_averages)) ** 2 for count, avg in zip(group_counts, group_averages))
SS_within = sum((group - avg).var(ddof=1) * (len(group) - 1) for group, avg in
                zip([regular_prices, midgrade_prices, premium_prices], group_averages))
SS_total = SS_between + SS_within

df_between = len(group_counts) - 1
df_within = sum(count - 1 for count in group_counts)
df_total = df_between + df_within

MS_between = SS_between / df_between
MS_within = SS_within / df_within

F_value = MS_between / MS_within

# Critical F-value (for alpha = 0.05)
alpha = 0.05
F_critical = f.ppf(1 - alpha, df_between, df_within)

# P-value from F-distribution
p_value = 1 - f.cdf(F_value, df_between, df_within)
```

2. HYPOTHESIS TESTING METHODS

- ANOVA

```
# Display results in table format using pandas
anova_summary = pd.DataFrame({
    "Groups": ["Regular", "Midgrade", "Premium"],
    "Count": group_counts,
    "Sum": group_sums,
    "Average": group_averages,
    "Variance": group_variances
})

anova_table = pd.DataFrame({
    "Source of Variation": ["Between Groups", "Within Groups", "Total"],
    "SS": [SS_between, SS_within, SS_total],
    "df": [df_between, df_within, df_total],
    "MS": [MS_between, MS_within, None],
    "F": [F_value, None, None],
    "P-value": [p_value, None, None],
    "F crit": [F_critical, None, None]
})

# Display the tables
print("ANOVA Summary Table:")
print(anova_summary.to_string(index=False))
print("\n")
print("ANOVA Table:")
print(anova_table.to_string(index=False))
```

2. HYPOTHESIS TESTING METHODS

- **ANOVA**

Result:

ANOVA Summary Table:

Groups	Count	Sum	Average	Variance
Regular	1361	2964.954	2.178511	0.698143
Midgrade	1361	3158.840	2.320970	0.737059
Premium	1361	3364.523	2.472096	0.800081

ANOVA Table:

Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	58.670748	2	29.335374	39.371362	1.110223e-16	2.997933
Within Groups	3039.984378	4080	0.745094	NaN	NaN	NaN
Total	3098.655126	4082	NaN	NaN	NaN	NaN

Conclusion: Since the p-value is extremely small (significantly less than 0.05), we reject the null hypothesis, suggesting that there is a statistically significant difference in prices among the three gasoline types (regular, midgrade, and premium) over the period from 1995 to 2021.

4. Random forest

April February

- import dataset into data variable, we convert the Data column data type to datatime data type, at this time, the data will be formatted YYYY-MM-DD
- Split the dataset into 2 sets, 1 for train and 1 for test (with the division of train before 2015 and test after 2016)
- Create columns 'year', 'month', 'day' from Date column
- Determine the independent variables and dependent variables X_train, X_test, y_train, y_test

R2 - 2

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Read data from CSV file
data = pd.read_csv('PET_PRI_GND_DCUS_NUS_W.csv')
data['Date'] = pd.to_datetime(data['Date'])  # Ensure 'Date' column is datetime type

# Assuming 'Price' is the column containing oil prices, adjust as needed for your dataset
datar2 = data.copy()  # Keep a copy of the entire dataset

# Split data from 1995 to 2015 for training, and 2015 to 2021 for testing
train_datar2 = datar2.loc[(datar2['Date'].dt.year >= 1995) & (datar2['Date'].dt.year <= 2015)]
test_datar2 = datar2.loc[(datar2['Date'].dt.year > 2015) & (datar2['Date'].dt.year <= 2021)]

# Convert 'Date' column to numeric features (e.g., year, month, day)
train_datar2['Year'] = train_datar2['Date'].dt.year
train_datar2['Month'] = train_datar2['Date'].dt.month
train_datar2['Day'] = train_datar2['Date'].dt.day

test_datar2['Year'] = test_datar2['Date'].dt.year
test_datar2['Month'] = test_datar2['Date'].dt.month
test_datar2['Day'] = test_datar2['Date'].dt.day

# Remove 'Date' column from X_train and X_test
X_train = train_datar2.drop(columns=['R2', 'Date'])
y_train = train_datar2['R2']
X_test = test_datar2.drop(columns=['R2', 'Date'])
```

Train the model

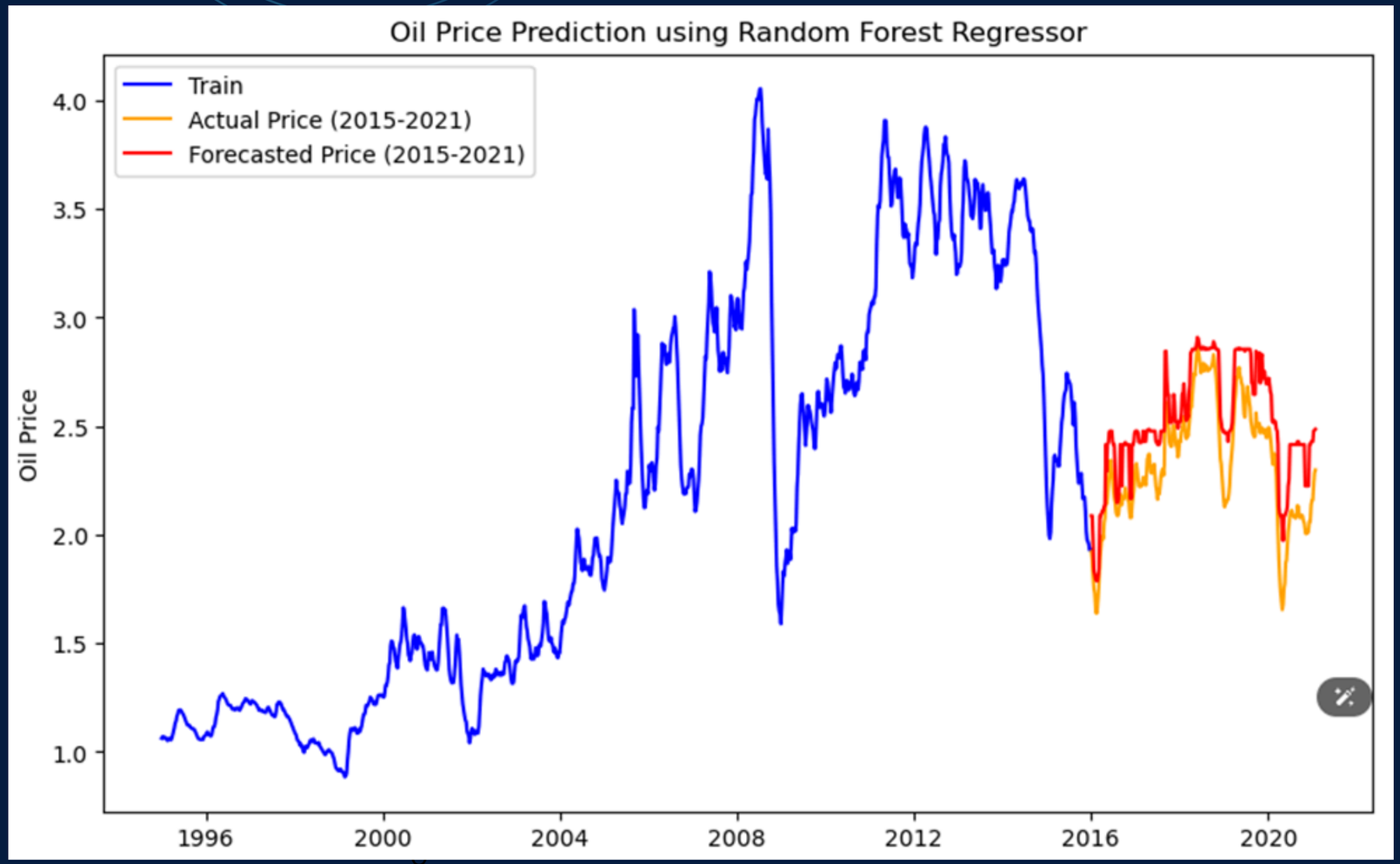
Modeling Implementation:

- Create RandomForestRegressor constructor
- Fit X_train and y_train to predict
- Get y_predict value from X_test
- Visualize data

```
# Train the model
rdf_regressor = RandomForestRegressor(
    max_depth=3,
    max_leaf_nodes=16,
    min_samples_split=10,
    min_samples_leaf=10
)
rdf_regressor.fit(X_train, y_train)

# Predict oil prices from 2015 to 2021
y_pred = rdf_regressor.predict(X_test)

# Plot results
plt.figure(figsize=(10, 6))

# Plot training set
plt.plot(train_datar2['Date'], y_train, label='Train', color='blue')

# Plot actual prices from 2015 to 2021
plt.plot(test_datar2['Date'], y_test, label='Actual Price (2015-2021)', color='orange')

# Plot forecasted prices from 2015 to 2021
plt.plot(test_datar2['Date'], y_pred, label='Forecasted Price (2015-2021)', color='red')

plt.xlabel('Year')
plt.ylabel('Oil Price')
plt.title('Oil Price Prediction using Random Forest Regressor')
plt.legend()
plt.show()
```

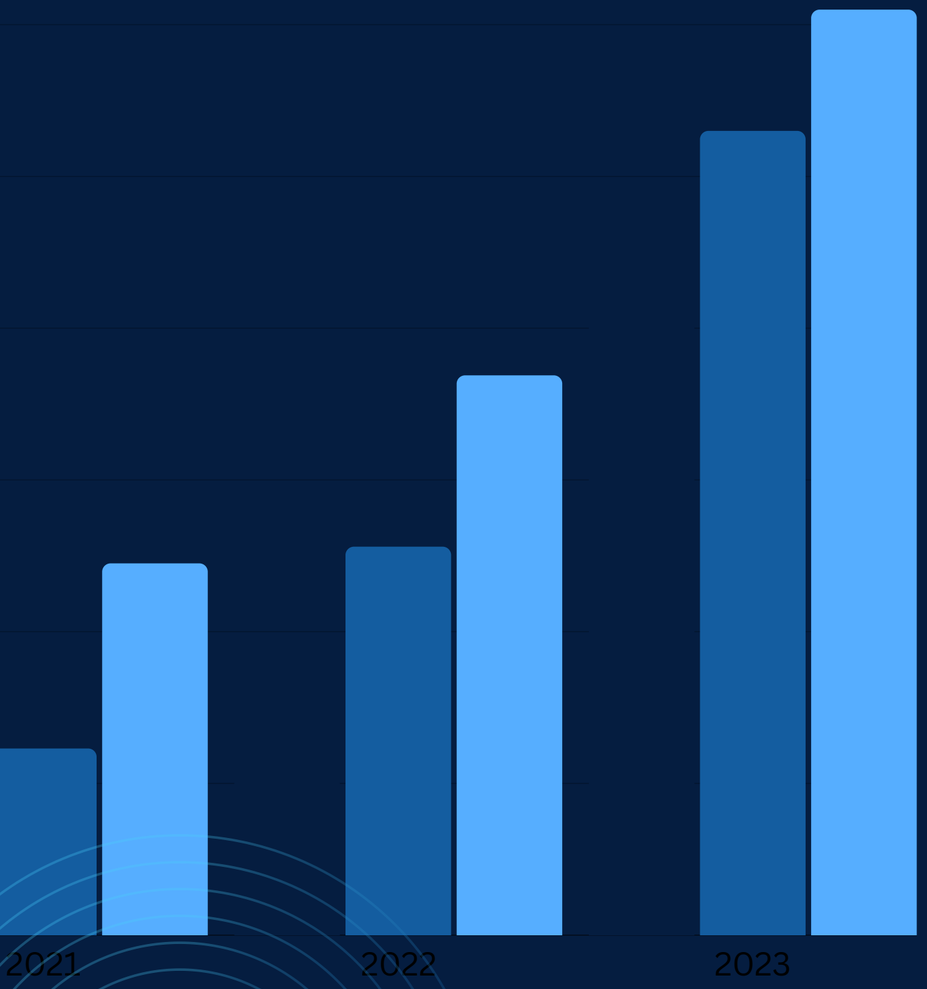
look the line graph

Price comparison chart y_test and y_pred



the forcast price quitely familiar with the actual price

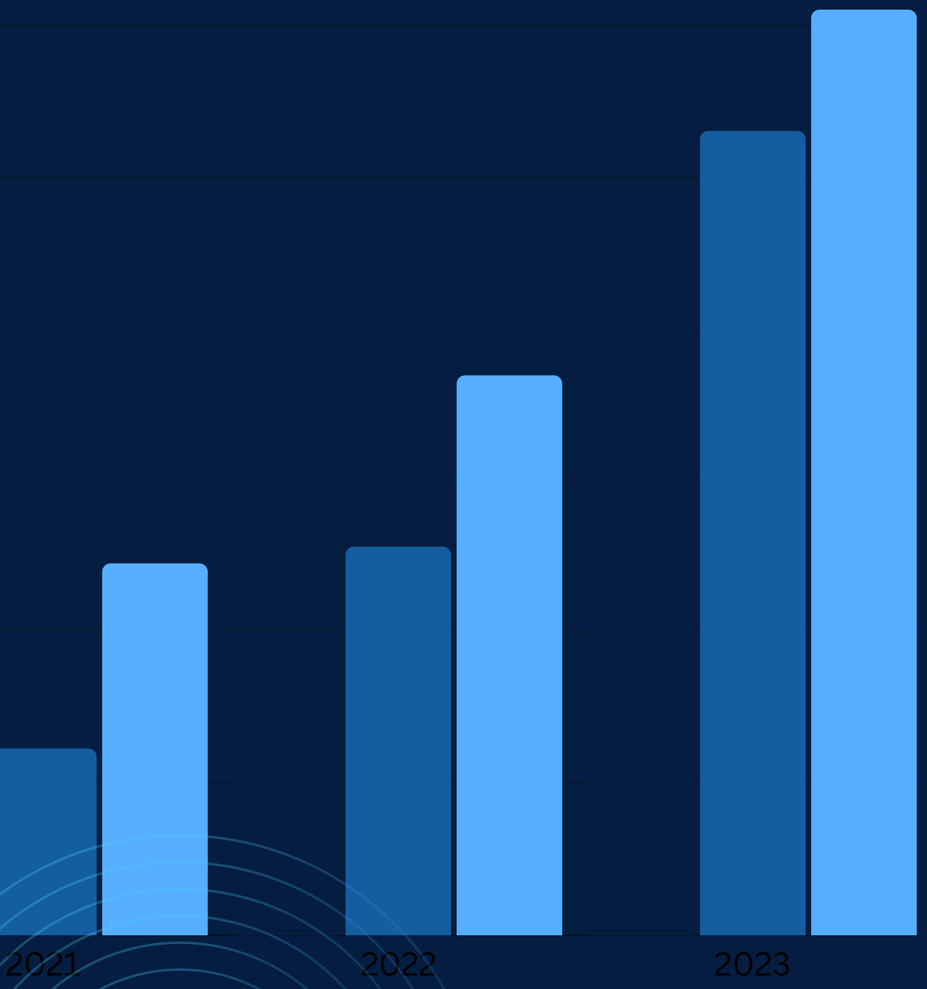
avaluate the efficiency of the model



use mae, mse, r2 to evaluate

```
# Evaluate model on test set
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print evaluation metrics
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R2): {r2}")

# Prepare data for output to Excel
output_data = test_datar2[['Date', 'R2']].copy()  # Copy relevant columns from the test set
output_data['Predicted R2'] = y_pred  # Add predicted values to the data
output_data['MAE'] = mae  # Add MAE to the data
output_data['MSE'] = mse  # Add MSE to the data
```

The smaller both MAE and MSE, the better the model, i.e. the model is predicting close to the actual value.
for r2, if r2 closer to 1, the more precise that model



avaluate the efficiency of the model

use mae, mse, r2 to evaluate

```
Mean Absolute Error (MAE): 0.1944100782017826
Mean Squared Error (MSE): 0.04480607268650949
R-squared (R2): 0.4283124368472991
```

In this case, MAE is 0.1944, which means that the model predicts an average error of about 0.1944 units
In this case, MSE is 0.0449, which indicates that the prediction errors are not too large.
The R2 value is 0.4283, which means that the model can explain about 42.83% of the variation in the data.

Conclusion:
The model has a fairly good accuracy with relatively low MAE and MSE, however, the R2 value shows that the model only explains a part (about 42.83%) of the variation in the data. This means that the model may need to be improved or additional features added to increase its predictive ability..

Linear Regression

```python
import pandas as pd
from sklearn.linear_model import LinearRegression

# Bước 1: Đọc dữ liệu từ file CSV
df = pd.read_csv('PET_PRI_GND_DCUS_NUS_W.csv')
df['Date'] = pd.to_datetime(df['Date'])  # Chuyển 'Date' sang dạng datetime

# Tách dữ liệu thành tập huấn luyện và tập kiểm tra dựa trên năm
subdataset_for_train = df.loc[(df['Date'].dt.year >= 1995) & (df['Date'].dt.year <= 2015)]
subdataset_for_test = df.loc[(df['Date'].dt.year > 2015) & (df['Date'].dt.year <= 2021)]

# Sau khi tách tập dữ liệu. Chuyển đổi 'Date' sang giá trị số (ví dụ: số thứ tự ngày) để mô hình có thể sử dụng (linear regression
# ...ko xử lý trực tiếp dữ liệu kiểu datetime)
subdataset_for_train['Date'] = subdataset_for_train['Date'].map(pd.Timestamp.toordinal)
subdataset_for_test['Date'] = subdataset_for_test['Date'].map(pd.Timestamp.toordinal) # 1995-1-2 thành 728295
# Chuyển từ số thứ tự về dạng datetime: date = pd.to_datetime(728295, origin='julian', unit='D') ==> lúc này 728295 thành 1995-1-2
```

First, import dataset into dataframe df (convert datatype of Date column to datetime)
We split dataset into two part, the first one is for train and the other one for test. To dataset for train, data from 1995 to 2015 and for test, data from 2016 to 2021
Note: linear regression cant handle datetime, so I transform it to time type of julian that linear regression can use it

Train the model

```
# Tạo tập huấn luyện và tập kiểm tra
X_train = subdataset_for_train.drop(columns=['R2']).astype(float)  # Chuyển toàn bộ X_train sang float
y_train = subdataset_for_train['R2']
X_test = subdataset_for_test.drop(columns=['R2']).astype(float)    # Chuyển toàn bộ X_test sang float
y_test = subdataset_for_test['R2']

# Huấn luyện mô hình hồi quy tuyến tính
lm = LinearRegression()
lm.fit(X_train, y_train)

# Dự đoán với tập kiểm tra
y_pred = lm.predict(X_test)
print(y_pred)
```

After we split into test and train dataset, I define the features which is independent and which is dependent feature.
Next, I train model by use fit. Finally, I can predict y_test(this is predicted y_test, not real) by using predict and parameter is X_test

linear regression

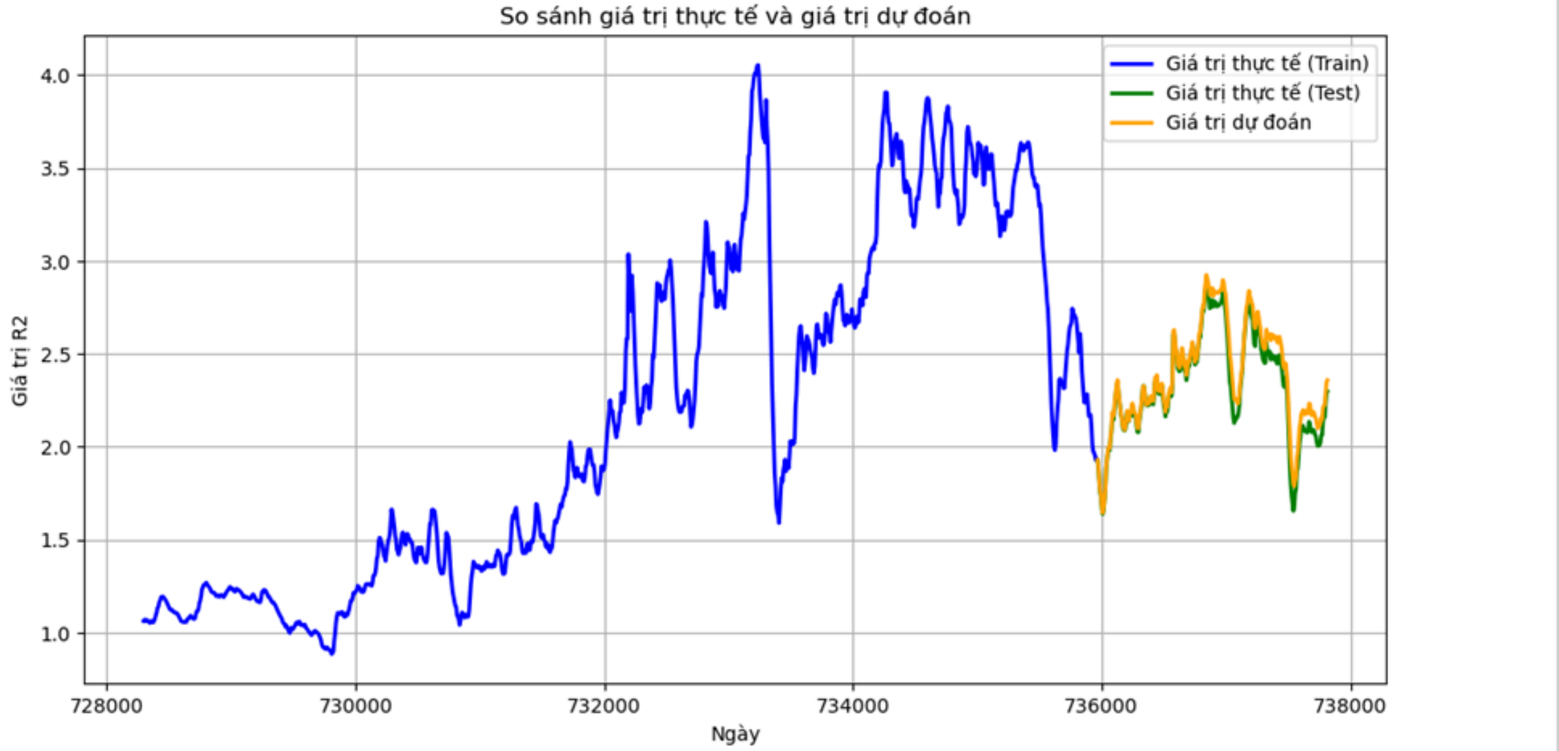
Evaluate the model

In this case, I use mean squared error to evaluate the output,
mse is 0.004 point out the model is reliable

```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test, y_pred)
print(f'mse là: {mse}')
```

```
mse là: 0.004407965802330199
```

```python
import matplotlib.pyplot as plt

# Trực quan hóa kết quả
plt.figure(figsize=(12, 6))

# Vẽ đường cho giá trị thực tế của tập huấn luyện
plt.plot(subdataset_for_train['Date'].reset_index(drop=True), y_train.reset_index(drop=True), label='Giá trị thực tế (Train)', color='blue', linewidth=2)

# Vẽ đường cho giá trị thực tế của tập kiểm tra
plt.plot(subdataset_for_test['Date'].reset_index(drop=True), y_test.reset_index(drop=True), label='Giá trị thực tế (Test)', color='green', linewidth=2)

# Vẽ đường cho giá trị dự đoán
plt.plot(subdataset_for_test['Date'].reset_index(drop=True), y_pred, label='Giá trị dự đoán', color='orange', linewidth=2)

plt.xlabel('Ngày')
plt.ylabel('Giá trị R2')
plt.title('So sánh giá trị thực tế và giá trị dự đoán')
plt.legend()
plt.grid(True)
plt.show()
```



plot the diagram to assess

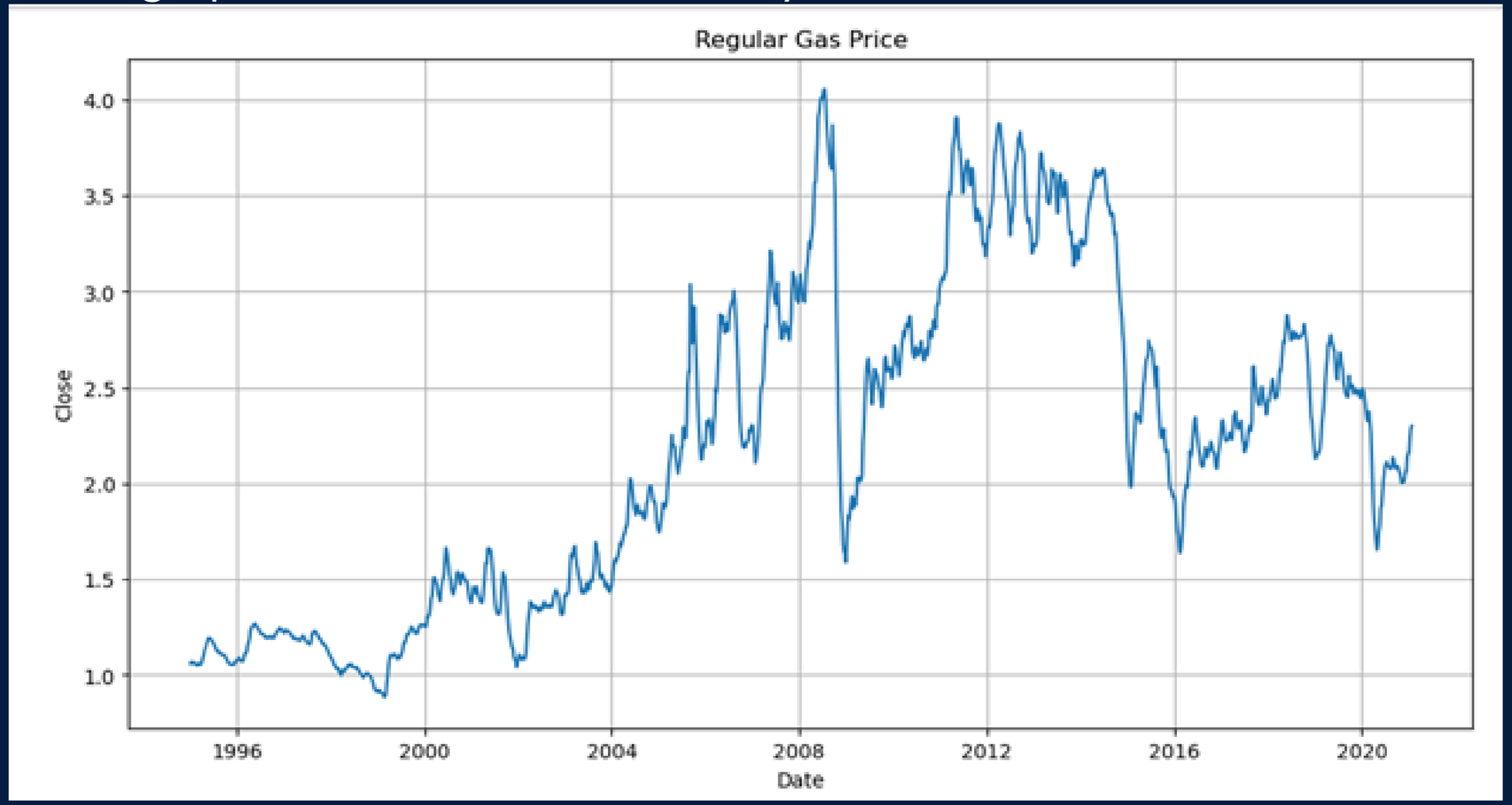
As we can see, the prediction quitely familiar with the actual test

5. ARIMA

1.Python

Step 1: Check the dataset information

Step 2: Plot a graph to check for seasonality



5. ARIMA

1.Python

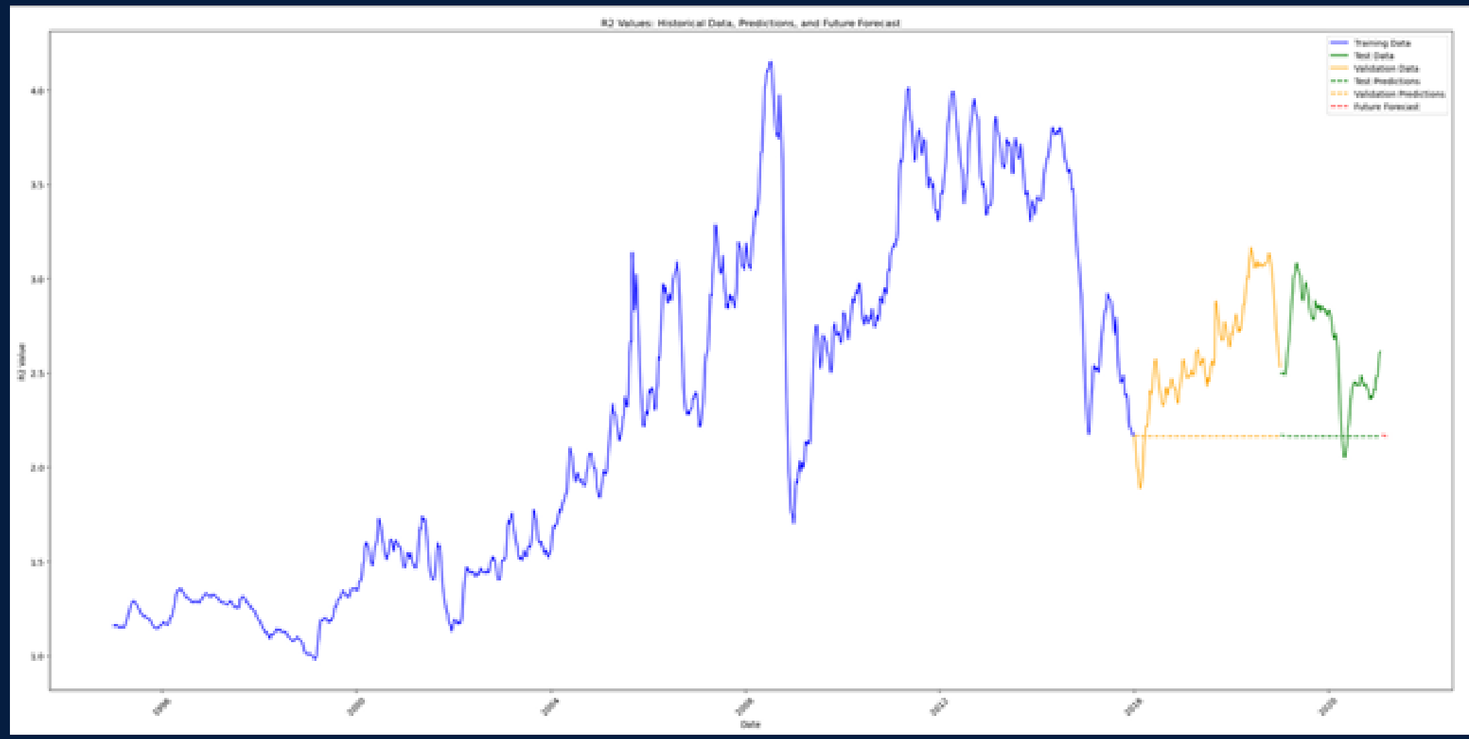
Step 3: ADF Test

ADF Test

```
# ADF Test
series = df.loc[:, 'R2'].values
result = adfuller(series, autolag='AIC')
output = pd.Series(result[0:4],index=['Test St
for key,values in result[4].items():
    output['critical value (%s)'%key] =  value
print(output)
```

```
Test Statistics                  -2.566204
p-value                           0.100185
No. of lags used                  3.000000
Number of observations used    1357.000000
critical value (1%)              -3.435178
critical value (5%)              -2.863672
critical value (10%)             -2.567905
dtype: float64
```

Step 4: Split data and perform Auto ARIMA



5. ARIMA

1.Python

Step 5: Identify RMSE and MAE of the model

```
Validate result:

RMSE: 1.0740095296564787
MAE: 0.8745788306674801
Test result:

RMSE: 1.2369044659265065
MAE: 1.2072400013557139
```

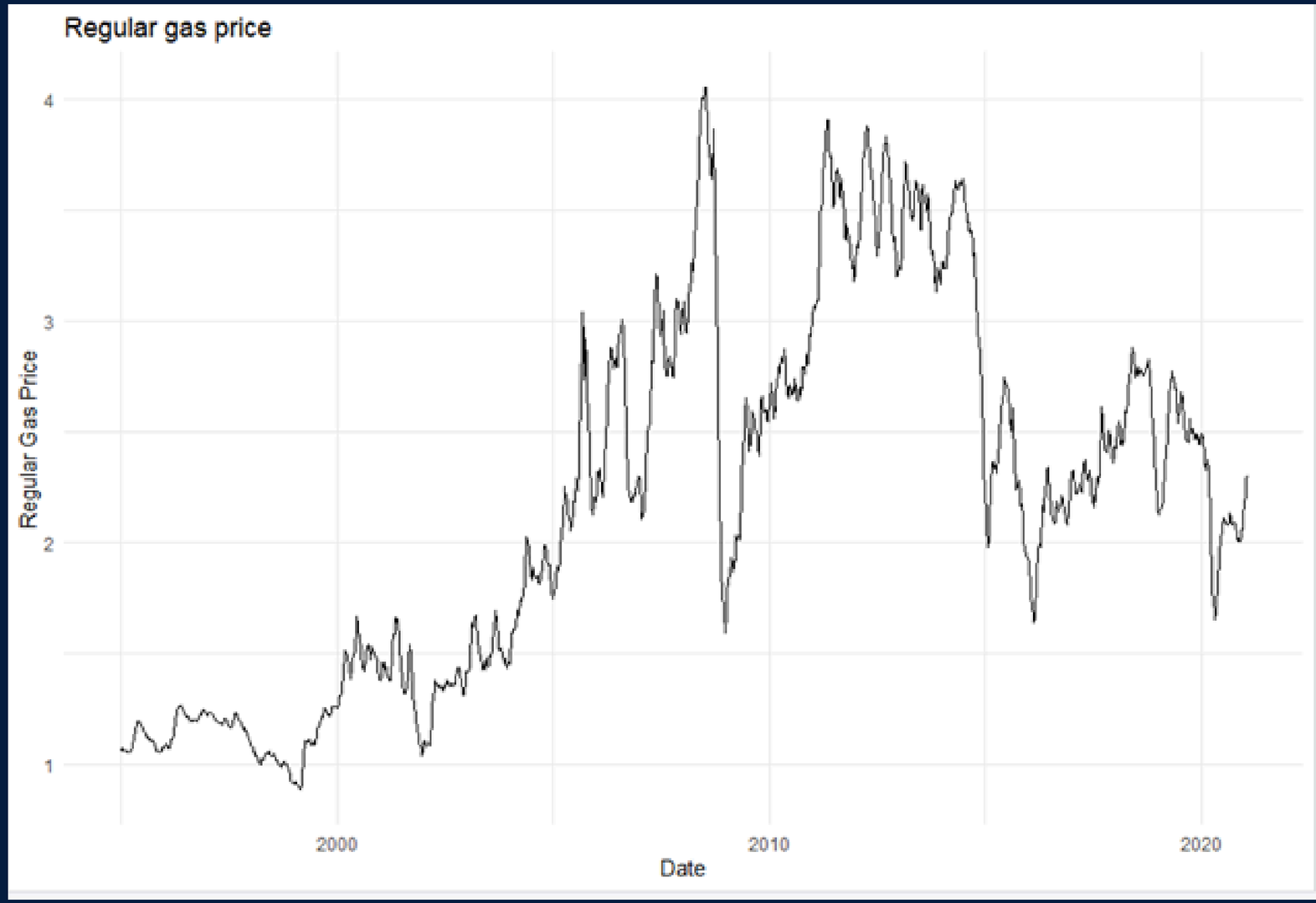
Based on the graph and RMSE,MAE we can see the predicted value as well as the forecast are much lower than the predicted value not to mention being much more linear despite the original data being more volatile.

5. ARIMA

2 R:

Step 1: Check the dataset information

Step 2: Plot a graph to check for seasonality

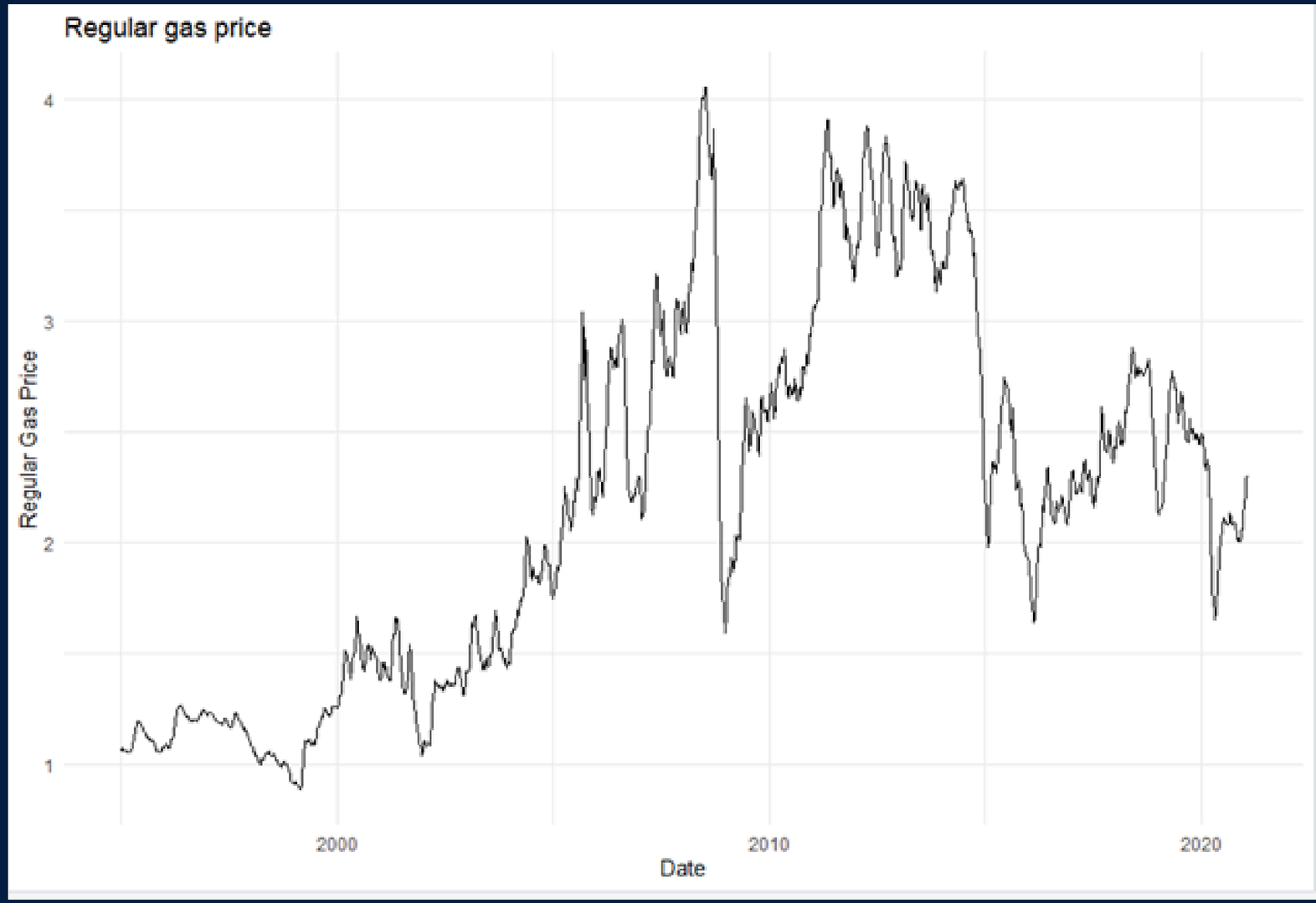


5. ARIMA

2 R:

Step 1: Check the dataset information

Step 2: Plot a graph to check for seasonality



5. ARIMA

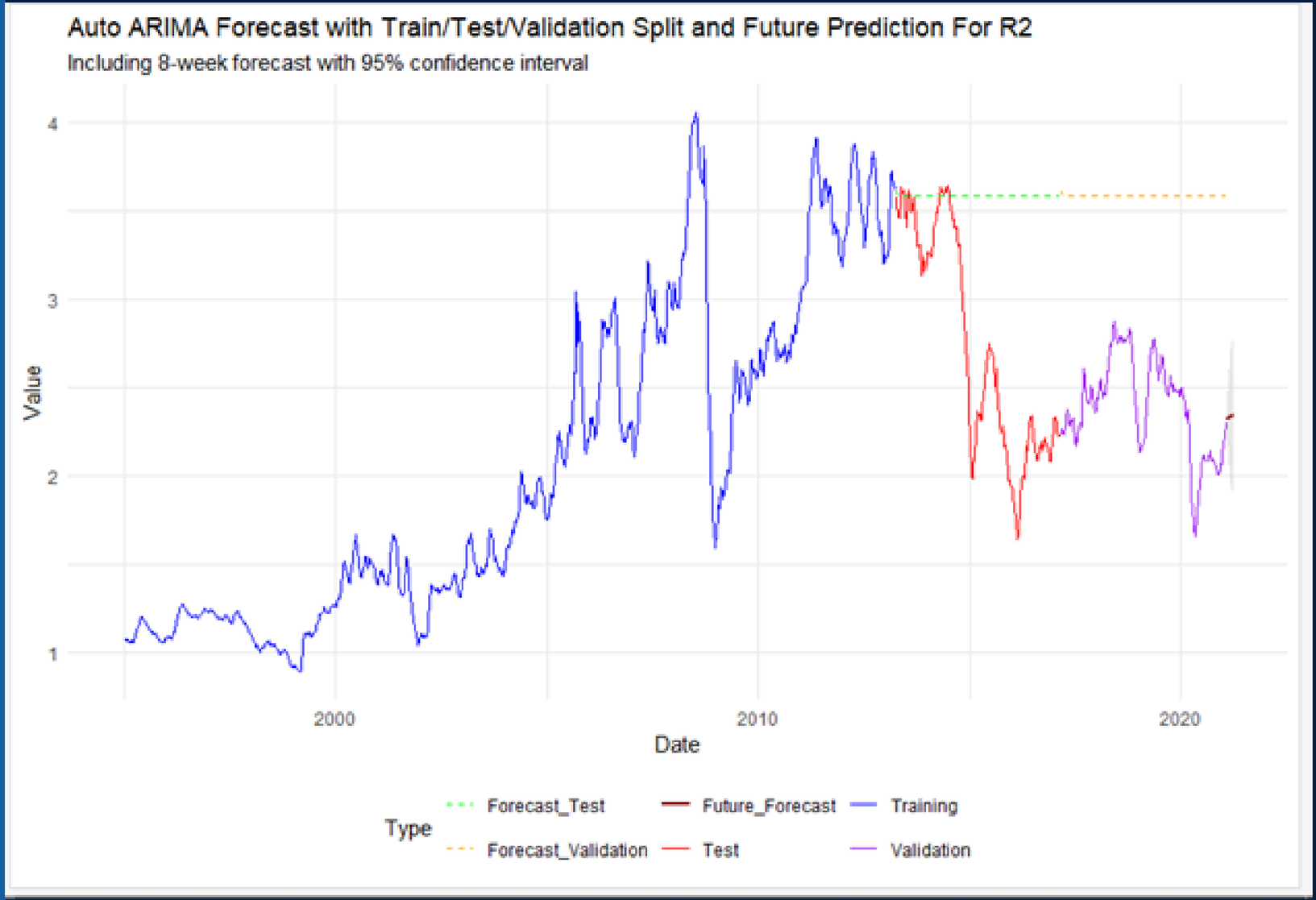
2 R:

Step 3: ADF Test

```
[1] "REGULAR GAS PRICE"

        Augmented Dickey-Fuller Test

data:  df$R2
Dickey-Fuller = -2.5229, Lag order = 1, p-value = 0.357
alternative hypothesis: stationary

Conclusion: Data's non stationary
Use the differencing method.

        Augmented Dickey-Fuller Test

data:  diff_data
Dickey-Fuller = -10.217, Lag order = 11, p-value = 0.01
alternative hypothesis: stationary

ADF Statistic: -2.522897
p-value: 0.3569617
Because p-value > 5%, data is not stationary and need to
Series: df$R2
ARIMA(3,1,3)
```

5. ARIMA

2 R:



Step 4: Split data and perform Auto ARIMA

5. ARIMA

2 R:

Step 5: Identify RMSE and MAE of the model

```
Test Set Metrics:
RMSE: 0.9869
MAE: 0.7994

Validation Set Metrics:
RMSE: 1.0362
MAE: 1.0041
```

From the graph and RMSE & MAE, we can judge that the inaccuracy is still high, with the forcast being much higher than the actual value. Consideration should be taken when looking at the forecast for the upcoming 8 weeks.

4.ECONOMIC IMPLICATIONS OF GASOLINE PRICE CHANGES

- Fuel is an essential source of energy in the modern world and serves as the lifeblood of economies. People and businesses depend on fuel for their daily lives, production, and trade
- When gas prices rise, it can be a drag on the economy—impacting everything from consumer spending to the price of airline tickets to hiring practices.Gas is an important input for transportation, which directly impacts households as they drive, but also businesses that rely on logistics and transportation chains around the globe.
- A side effect of high gas prices is that the discretionary spending of consumers drops as they spend a relatively larger portion of their income on gasoline.

4.ECONOMIC IMPLICATIONS OF GASOLINE PRICE CHANGES

- Higher gas prices can result in noticeable increases in some public transportation ridership.
- Though economists and analysts may argue about the extent to which gas prices have an effect on the economy, there is, at the least, a correlation between consumer confidence, spending habits, and gas prices.

CONCLUSION

- This project uses analytical methods to monitor and analyze historical gas price fluctuations, providing insights into market trends and enabling accurate future predictions.
- This helps businesses plan and adjust strategies in response to price changes.
- Ongoing research and development aim to further improve the model's reliability and accuracy.

Thank's For Watching