1. INTRODUCTION

1.1. OVERVIEW

Natural gas is a ‘clean fuel’. Burning natural gas emits less carbon dioxide into the environment as compared to coal or other similar fossil fuels. Thus, natural gas has become a popular alternative fuel in many homes and industries to power various household appliances, lighting, heating, in turbines to produce solar as well as wind energies, transport vehicles etc.

This is why forecasting natural gas prices is increasingly becoming an instrumental and crucial tool for various stakeholders in the natural gas market. It has enabled stakeholders to make better and more effective decisions for a variety of issues concerning the natural gas market. Decisions about managing the potential risk, reducing the gap between demand and supply, and optimising the use of resources are a few examples that accurate forecasting of gas prices aids in.

An accurate forecast of the price of natural gas provides a crucial reference for the efficient execution of energy policy and planning. It is also of the utmost significance in terms of economic planning, energy investment, and the preservation of the environment.

1.2. PURPOSE

The main objective of this project is to build a model that predicts natural gas prices for the near future using Machine Learning models.The methods to be adopted are regression techniques that are tuned to match our needs and perform accurately under a variety of conditions.

The final model is deployed as an interactive website where a simple date input can give a prediction of the gas prices for that day.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

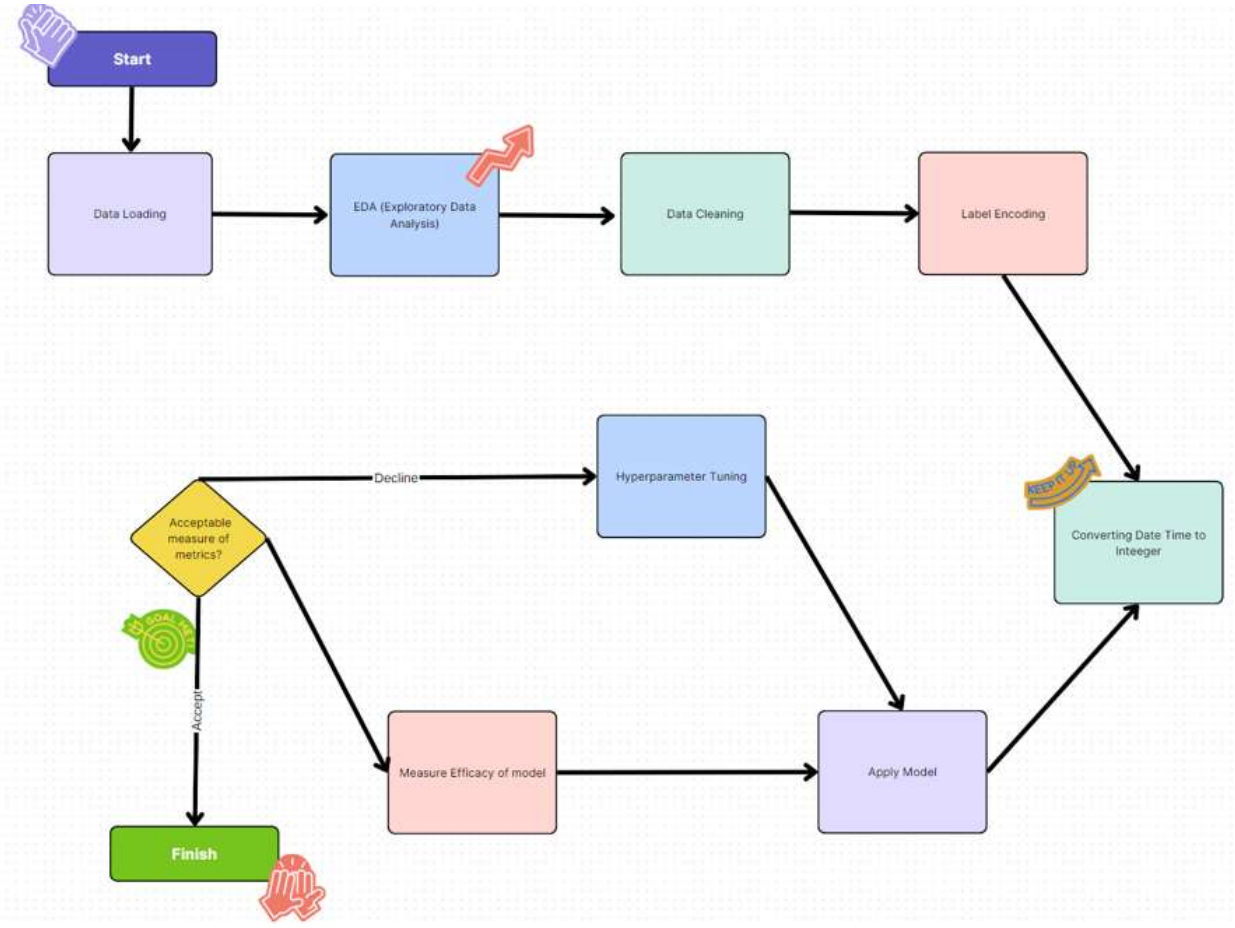
Analysts now primarily rely on energy futures markets for information regarding future energy prices due to the poor track record of energy price forecasting models. Natural gas pricing and marketing "hubs" are energy futures marketplaces. Walls (1995), looking at several years' worth of spot prices, concludes that natural gas futures are generally unbiased predictors of future spot prices, in contrast to Herbet (1993), who finds bias in natural gas futures prices when futures prices are higher than actual spot prices. With the exception of those in the natural gas market at the three-month horizon, futures prices are found by Chinn et al. (2005) to be fair predictors of future spot prices and to slightly outperform time series models.

2.2 PROPOSED SOLUTION

We will be attempting to predict natural gas prices using the data set of prices provided from the 7th of January 1997 to the 11th of August 2020. To do this, we will test various machine learning models and provide a real-time web-based graphical user interface (GUI) that will ask the user to enter the desired date in order to predict the rate of the natural gas on that specific day. By comparing the precision of several methods until the best match value is found, this report adds to the body of current literature.

3. THEORITICAL ANALYSIS

3.1 BLOCK DIAGRAM



3.2 HARDWARE / SOFTWARE DESIGNING

The following modules and softwares are used to build the Natural gas price prediction project.

Python Kernel:
jupyter -1.0.0
ipykernel - 6.15.2
ipython -7.31.1

Data Preprocessing:
Pandas-1.4.4
Numpy-1.21.5
Scipy-1.9.1

Data Visualization:
matplotlib -3.5.2

seaborn- 0.11.2

ML Model:
scikit-learn-1.0.2

Pickle File:
Joblib-1.1.0

Framework:
Flask-1.1.2

IDE:
Jupyter Notebook
Spyder -5

4.EXPERIMENTAL INVESTIGATIONS

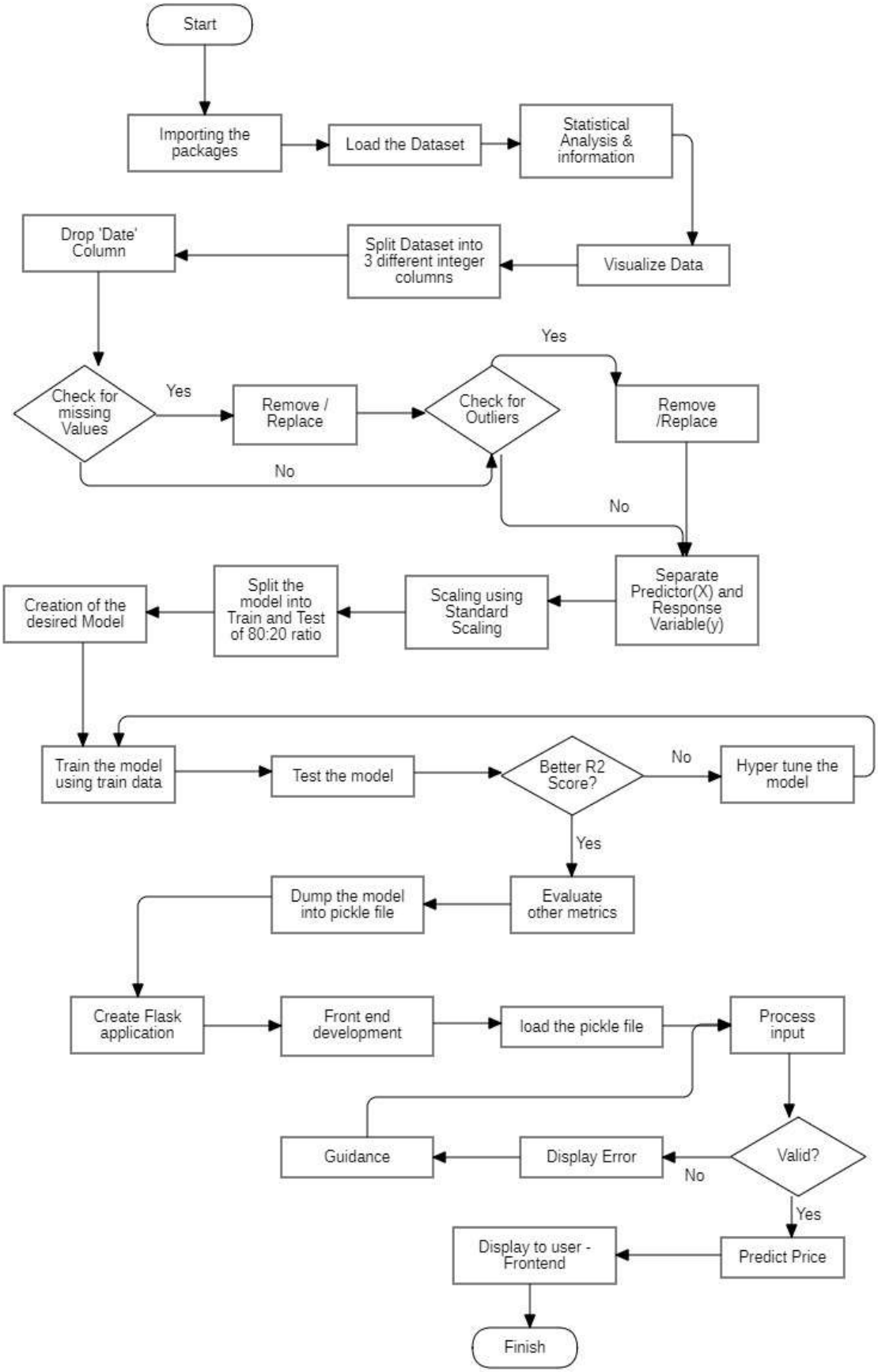
Upon exploring the dataset we find that there were 5,938 samples. The prices were recorded as they were on a particular day. The time period of recording the natural gas prices were from 7th January, 1997 to 11th August, 2020. The prices recorded are in USD($).

The data shows that the gas prices trend from 1997 to 2020 are fairly seasonal. There are frequent highs and lows showing how people consume natural gas with the changing seasons. For example, natural gas may be in higher demand during the winters due to being used as fuel for heating homes. Because supply frequently finds it difficult to respond promptly to temporary surges in demand, the effect on pricing can become more pronounced when cold or severe weather strikes unexpectedly. We can also see a steady decrease in the prices from around the year 2006. Markets for natural gas are influenced by the health of the economy. Increases in the demand for goods and services from the commercial and industrial sectors may result in an increase in natural gas consumption during times of economic expansion. In the industrial sector, which uses natural gas as a fuel and a feedstock for numerous items like fertiliser and pharmaceuticals, economic-related rises in consumption can be particularly high.

Also a higher demand for other fuels also affects the prices of natural gas. The demand for natural gas may decline if the price of other fuels drops, which could result in a drop in natural gas prices. Switching to natural gas may raise demand and prices for natural gas when the cost of competing fuels increases relative to the cost of natural gas.

The data also showed a very low correlation between the date and the prices suggesting that other variables might also be required to make accurate predictions. A linear regression model on the data performed poorly giving an RMSE of 2.45 and a R-squared value of 0.03. Following this a random forest regressor was applied with default hyperparameters. This also resulted in poor performance with an RMSE of 1.83 and a R-squared value of 0.45. This suggested a strong need for hyperparameter tuning to get our ideal model.

5. FLOWCHART


Start
Importing the packages
Load the Dataset
Statistical Analysis & information
Visualize Data
Split Dataset into 3 different integer columns
Drop 'Date' Column
Check for missing Values
Yes
Remove / Replace
No
Check for Outliers
Yes
Remove /Replace
No
Separate Predictor(X) and Response Variable(y)
Scaling using Standard Scaling
Split the model into Train and Test of 80:20 ratio
Creation of the desired Model
Train the model using train data
Test the model
Better R2 Score?
No
Hyper tune the model
Yes
Evaluate other metrics
Dump the model into pickle file
Create Flask application
Front end development
load the pickle file
Process input
Valid?
No
Display Error
Guidance
Yes
Predict Price
Display to user - Frontend
Finish

6. RESULTS

The ideal model was constructed after performing the hyperparameter tuning usingGrid Search. The parameters n_extimators and max_depth were searched over the following domains:

n_estimators	100, 200, 250, 300
max_depth	2, 6, 8, 10

The best parameters were found to be n_estimators: 250 and max_depth: 8 with a score of 0.97. The parameters were used to predict for the test set. The RMSE value turned out to be 1.52 and the R-squared value was 0.97.

Model	**MAE**	**RMSE**	**R^2**
Linear Regression	1.73	2.4	0.01641
Random Forest Regressor	1.16	1.84	0.975
Decision Tree Regressor (w/ hyperparameter tuning)	0.15	0.58	0.9779

7. ADVANTAGES & DISADVANTAGES

Models	Advantages	Disadvantages
Linear Regression	Despite being straightforward, the model is quick. predicts accurately and may show the intensity of the relationship and influence between the independent and dependent variables.	Sometimes, the selection and use of the criteria are merely hypothetical and constrained.
Random forest Regressor	Random Forest Regressor can effectively handle datasets with a large number of input features. It automatically selects a subset of features at each split, reducing the risk of overfitting and improving generalization performance.	It's not explicitly designed to handle the temporal nature of time series data. It does not inherently capture the autocorrelation and temporal dependencies that are often present in natural gas price data. Additional feature engineering or the use of specialized time series models may be necessary to address this limitation.
Decision Tree (w/ hyperparameter tuning)	The model can be trained quickly, has good generalisation properties, and is not sensitive to missing data.	Overfitting is more likely if the data are noisy.

7. APPLICATIONS

About one-third of the energy demand in the US and one-fourth of the global demand are met by natural gas. Natural gas is the most prevalent type of energy after oil. Being on the verge of improving natural gas demand prediction is therefore very beneficial. The energy market orientation depends on the accuracy of energy price predictions, which can serve as a guide for regulators and market participants. Energy costs are influenced by outside forces in real life, making it difficult to estimate them accurately. The ability to forecast natural gas prices is advantageous to a variety of market participants and has grown to be an extremely useful tool for all market participants in highly competitive natural gas marketplaces. The use of machine learning algorithms for predicting natural gas prices has gradually gained popularity.

8. CONCLUSION

Predicting the precise daily price of natural gas has always been tricky. Politics, the economy as a whole, and traders' expectations are just a few of the variables that could affect it. But in this case, based on the past and present characteristics, we were able to estimate the price of any given date with up to 97.79% accuracy. Unexpected events like acts of war or terrorism cannot be predicted, despite this. However, it is crucial to have accurate knowledge about the potential price of natural gas because it can make or break economies. Data-driven machine learning models in this situation merit all the attention they can possibly receive, as this experiment demonstrates.

9. FUTURE SCOPE

The framework may be worked focused on forecasting the price of natural gas for the entire day. A few improvements may be done in the future to forecast the price the natural gas at different hours of a particular day. This model can also be extended to predict the prices of the other gases that are necessary.

10. BIBLIOGRAPHY

For the whole Project:
https://smartinternz.com/Student/guided_project_info/524395#

Creation of the model:
https://towardsdatascience.com/power-of-xgboost-lstm-in-forecasting-natural-gas-price-f426fada80f0
https://towardsdatascience.com/machine-learning-basics-decision-tree-regression-1d73ea003fda
https://scikit-learn.org/stable/modules/tree.html
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

Book:
Aurelion Geron, Hands-On Machine Learning with Scikit – Learn, Keras and Tensorflow, 2nd Edition, O.Reilly, 2019

For Integration:
https://stackoverflow.com/questions/43645790/passing-javascript-variable-to-python-flask
https://www.geeksforgeeks.org/pass-javascript-variables-to-python-in-flask/

APPENDIX

A. Source Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

dataset = pd.read_csv('../data/daily_csv.csv')
dataset.head(10)

# ## Data Preprocessing
# ### Analysing Data
dataset.info()

# Statistical Analysis
dataset.describe()
dataset.shape
    dataset.dtypes
# ### Feature Engineering - Feature Extraction

# ##### Splitting the Date column into 3 different integer columns respectly Year,
    Month and Day
dataset['year'] = pd.DatetimeIndex(dataset['Date']).year
```

```
dataset['month'] = pd.DatetimeIndex(dataset['Date']).month
dataset['day'] = pd.DatetimeIndex(dataset['Date']).day
print(dataset.head())


# ### Feature Selection
# dropping the Date Column
dataset.drop('Date', axis=1, inplace=True)
dataset.head()


# ### Handling Missing Values

dataset.isnull().any()
# From this we could identify that the PRICE column contains null values. So replace
    them with the mean of the column

sns.distplot(dataset['Price'].dropna())
plt.title("Distribution of Price")

dataset['Price'].fillna(dataset['Price'].mean(), inplace=True)
dataset.head()
# Re-checking for the NULL values
dataset.isnull().any()

# We have successfully replaced the missing values of the PRICE column
dataset.describe()

# ### Handling Outliers
plt.figure(figsize=(13,8))
l = ['Price','year','month','day']
for i in range(len(l)):
   plt.subplot(2,2,i+1)
   sns.boxplot(dataset[l[i]])

# This Price column contains outliers. So, we are removing them using IQR method

from scipy import stats
z = np.abs(stats.zscore(dataset))
print(z)

print('Outliers')
print(np.where(z>3))
print('Number of outliers',len(np.where(z>3)[0]))

# The dataset contains only 117 rows of Outliers, Thus we can just remove them
    instead of replacing.
```

```
dataset_updated = dataset[(z<=3).all(axis=1)]
dataset_updated

dataset_updated.shape

# ## Data Visualization

plt.figure(figsize=(13,5))
plt.subplot(1,2,1)
plt.scatter(x=dataset['year'],
             y=dataset['Price'],
            color='red')
plt.xlabel('Year')
plt.ylabel('Price')
plt.title('Price based on the Year')

plt.subplot(1,2,2)
plt.bar(x=dataset['year'],
       height=dataset['Price'],
            color='red')
plt.xlabel('Year')
plt.ylabel('Price')
plt.title('Price based on the Year')
     # in between 2000-2005, the price of the natural gas was at its high.
plt.figure(figsize=(13,5))
plt.subplot(1,2,1)
plt.bar(x=dataset['month'],
       height=dataset['Price'],
            color='green')
plt.xlabel('Month')
plt.ylabel('Price')
plt.title('Price based on the Month')

plt.subplot(1,2,2)
plt.scatter(x=dataset['month'],
          y=dataset['Price'],
            color='green')
plt.xlabel('Month')
plt.ylabel('Price')
plt.title('Price based on the Month')

# From the overall data, the 2nd month has the highest price rate.


plt.figure(figsize=(6,5))
sns.scatterplot(x=dataset['day'],
             y=dataset['Price'])
```

```
plt.title('Price based on Day of the month')
# Line Graph - Trend Line for Time Series
plt.figure(figsize=(6,5))
sns.lineplot(x='year', y='Price',
             data=dataset, color='red')
plt.title('Year wise Price')


plt.figure(figsize=(6,5))
sns.lineplot(x='month', y='Price',
             data=dataset, color='green')
plt.title('Month wise Price')


plt.figure(figsize=(6,5))
sns.lineplot(x='day', y='Price',
             data=dataset)
plt.title('Day wise Price')

sns.heatmap(dataset.corr(), annot=True)

# Pair Plot
sns.pairplot(dataset)
plt.show()


    # ## Splitting the Dataset into Train and Test
X = dataset.iloc[:,1:4].values
y = dataset.iloc[:,0:1].values
print(X)
print(y)


# ### Encoding
# There are no Categorical data available in our dataset. Thus, we can skip this step of
    preprocessing.

# ### Train - Test Split

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,
                          y, test_size=0.2, random_state=42)

X_train.shape, X_test.shape

# ## Model Building
#
```

```
    # Since our dataset predicts the numerical output, we can use any of the Regression
    Models.
# ### Decision Tree Regressor Model
from sklearn.tree import DecisionTreeRegressor
dt = DecisionTreeRegressor()
dt.fit(X_train, y_train)
dt_pred = dt.predict(X_test)
print(dt_pred)


# ## Random Forest Regression Model
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor()
rf.fit(X_train, np.ravel(y_train))
rf_pred = rf.predict(X_test)
print(rf_pred)


# ## SVM Regression
from sklearn.svm import SVR
svr = SVR()
svr.fit(X_train, np.ravel(y_train))
svr_pred = svr.predict(X_test)
print(svr_pred)
# ### Linear Regression Model
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)
lr_pred = lr.predict(X_test)
print(lr_pred)

# ### Finding the R2 - Score for each models created above

from sklearn.metrics import r2_score
print('R2 Scores:\n')

print("Decision Tree:\t%.5f"%r2_score(y_test, dt_pred))
print("Linear Regression:%.5f"%r2_score(y_test, lr_pred))
print("Random Forest:\t%.5f"%r2_score(y_test, rf_pred))
print("Support Vector:\t%.5f"%r2_score(y_test, svr_pred))

# From these, we could identify that, the DECISION TREE REGRESSION gives the
    better R2-Score. Thus, we can say this is the best model

    # #### Since Linear Regression is not robust to outliers and not standard values, we
    have to standardize the values
from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
X = dataset_updated.iloc[:,1:4].values
y = dataset_updated.iloc[:,0:1].values

X_train, X_test, y_train, y_test = train_test_split(X,
                            y, test_size=0.2, random_state=42)

X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)

# training the Linear Regressor with the standardised values
lr.fit(X_train, y_train)
lr_pred = lr.predict(X_test)
print(lr_pred)

print("Linear Regression:%.5f"%r2_score(y_test, lr_pred))

    # #### Still the Decision Tree Regressor is the best model for our dataset
# ## Model Evaluation
X = dataset.iloc[:,1:4].values
y = dataset.iloc[:,0:1].values

X_train, X_test, y_train, y_test = train_test_split(X,
                            y, test_size=0.2, random_state=42)

from sklearn.metrics import mean_squared_error, mean_absolute_error
print(mean_absolute_error(y_test, dt_pred))

print(mean_squared_error(y_test, dt_pred))

print("RMSE:",(mean_squared_error(y_test,dt_pred))**(1/2))


# ## Creating the Pickle File
import joblib
    joblib.dump(dt,'DTR.pkl')
```