A PROJECT REPORT ON

NATURAL GAS PRICE PREDICTION

USING MACHINE LEARNING

1. INTRODUCTION

1.1 Overview

Natural gas (also called fossil gas), is a naturally occurring hydrocarbon gas mixture consisting primarily of methane. It is a clean and an efficient fuel that is commonly suggested to use. The price for it changes each day, depending on certain factors, such as location, demand, production etc. Prediction becomes easier when the factors are clubbed together and assigned to each day.

In this model, we are using the Machine learning algorithms to help predict the values. We chose Decision-Tree Regression Algorithm in this model. Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables.

1.2 Purpose

Prediction of Natural Gas is commercially important to know. The purpose of this model is to predict the spot price of Natural Gas when the date is taken as the input given by the user. It then predicts the price of the Gas on that specific day.

2. LITERATURE OVERVIEW

2.1 Existing Problem

Natural Gas has been proposed as a solution to increase the security of energy supply and reduce environmental pollution around the world. Being able to forecast natural gas price benefits various stakeholders and has become a very valuable tool for all market participants in competitive natural gas markets.

2.2 Proposed Solution

Machine Learning Regression algorithms have gradually become a popular tool for natural gas price forecasting. In this, the main aim is to predict the Natural Gas Price, based on the demand and supply. Out of all the algorithms, Decision Tree Regression proves to be the best and efficient algorithm to train the machine, with an accuracy of 97%.

3. THEORETICAL ANALYSIS

3.1 Block Diagram

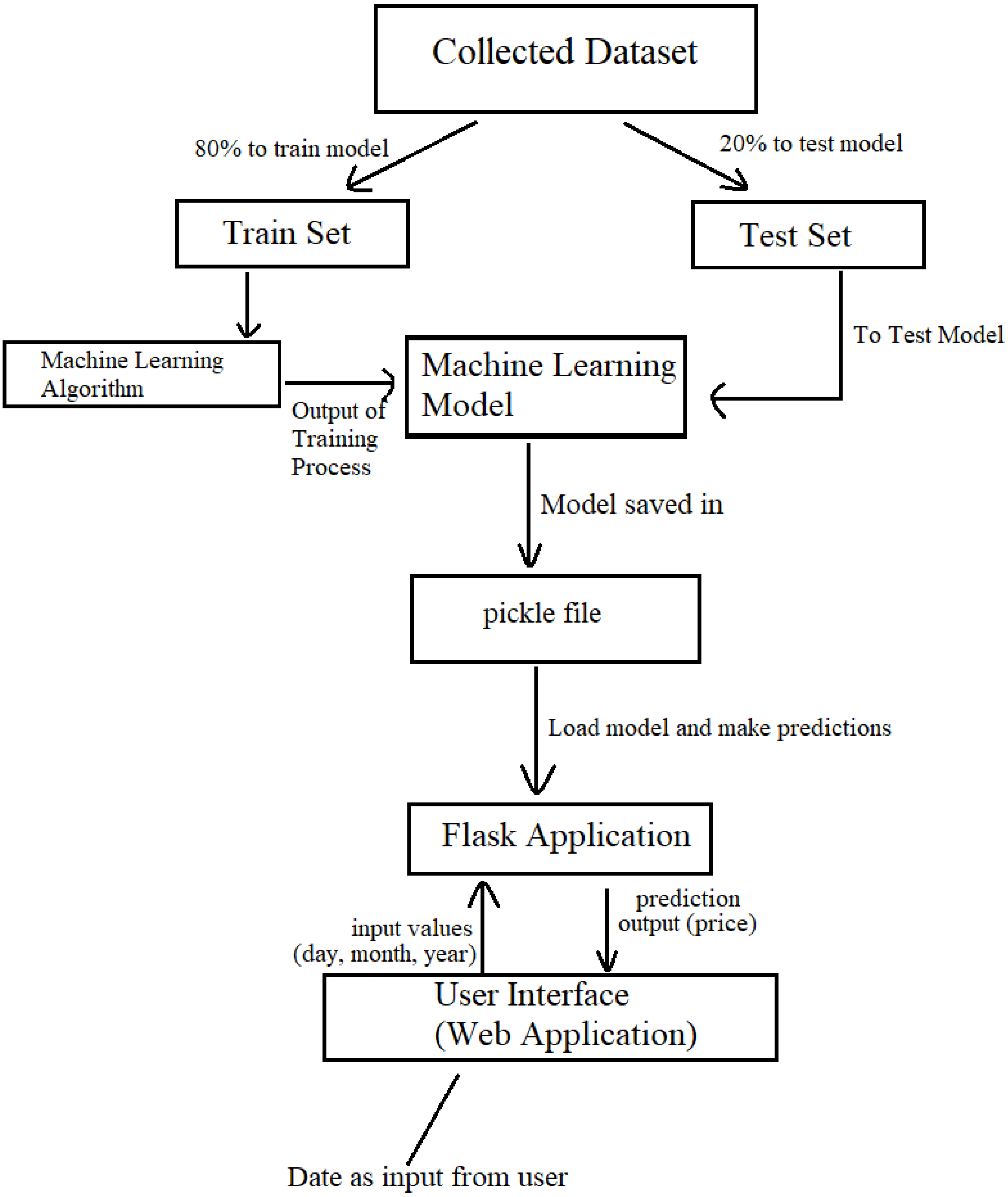
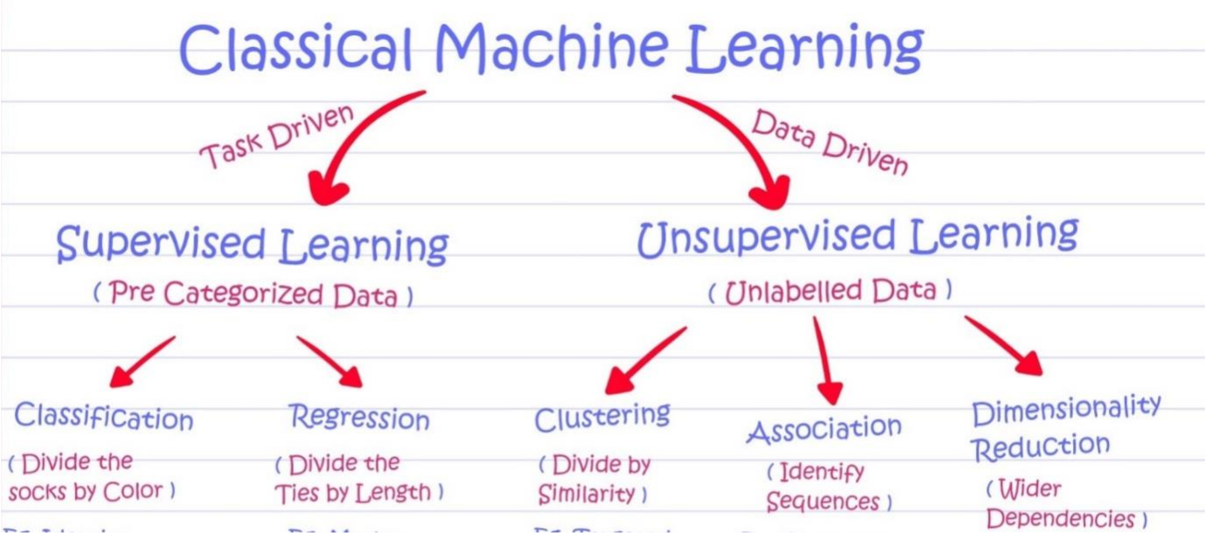

Collected Dataset
80% to train model
20% to test model
Train Set
Test Set
Machine Learning Algorithm
Output of Training Process
Machine Learning Model
To Test Model
Model saved in
pickle file
Load model and make predictions
Flask Application
input values (day, month, year)
prediction output (price)
User Interface (Web Application)
Date as input from user

Fig: Block Diagram for Price Prediction of Natural Gas

3.2 Hardware/Software Designing

Machine learning is a study of using algorithms and statistics to make the computer to learn by itself without being programmed explicitly by the user. Computers depend on algorithms that use a mathematical model. The given model uses a dataset which is known as “Training Dataset” to learn and to predict an outcome. There are multiple learning algorithms that can be used to solve the problem but the concept remains the same. All these algorithms fall into two categories viz. Supervised learning or Unsupervised learning.

Let’s know more about supervised learning as it is much more researched and used in applications like user profiling, recommended products list, etc. Supervised learning output has two types of values and is classified into two, one is Categorical (Classification Model) where the value is from the finite set (male or female etc.) and another one is Nominal (Regression Model) where the value is a real-valued scalar (income, product ratings, prices, etc). These algorithms are trained using the dataset and the outputs are predicted.



In supervised learning, the machine is trained with inputs as well as outputs and using this, the machine maps input to appropriate output. Classification and Regression are Supervised Machine Learning Algorithms. Classification specifies the class to which data elements belong to and is best used when the output has finite and discrete values. A regression problem is when the output variable is a real or continuous value, such as “salary” or “weight”. Our machine uses Regression Machine Learning algorithm as it has to predict price of natural gas which is a continuous variable. Classification is not suitable for this problem as the output cannot be segregated into classes. The regression algorithms being considered are Multiple Linear Regression, Decision Tree Regression and Random Forest Regression and are discussed in detail in Chapter 4.

HARDWARE REQUIREMENTS:

1. Laptop with 4GB Ram 64-bit Operating System

SOFTWARE REQUIREMENTS:

Anaconda Environment consisting of Jupyter Notebook, Spyder IDE.

4. EXPERIMENTAL INVESTIGATIONS

Following observations and investigations were made in the steps below during the project implementation.

Data Collection:

The data collected was based on the Henry Hub Natural Gas Spot Price from the years 1997 to 2020. Natural gas prices are mainly driven by supply and demand fundamentals. Weather, demographics, economic growth, price increases, poverty, fuel competition, storage and exports are responsible for affecting the demand for natural gas. The supply for natural gas is driven by factors like Pipeline capacity, Storage, Gas drilling rates, Natural phenomena, Technical issues, imports and transportation wholesale rates. The most important hub for natural gas is the Henry Hub, located along the U.S. Gulf Coast. Here, the benchmark for natural gas prices is determined and traded for delivery on the NYMEX natural gas futures contract. When quoted by a trader, the price is the difference between the Henry Hub price and that location's price, called the basis price. Henry Hub is an important market clearing pricing concept because it is based on the actual supply and demand of natural gas as a stand-alone commodity. Therefore, our project uses this price alone as input to predict natural gas price.

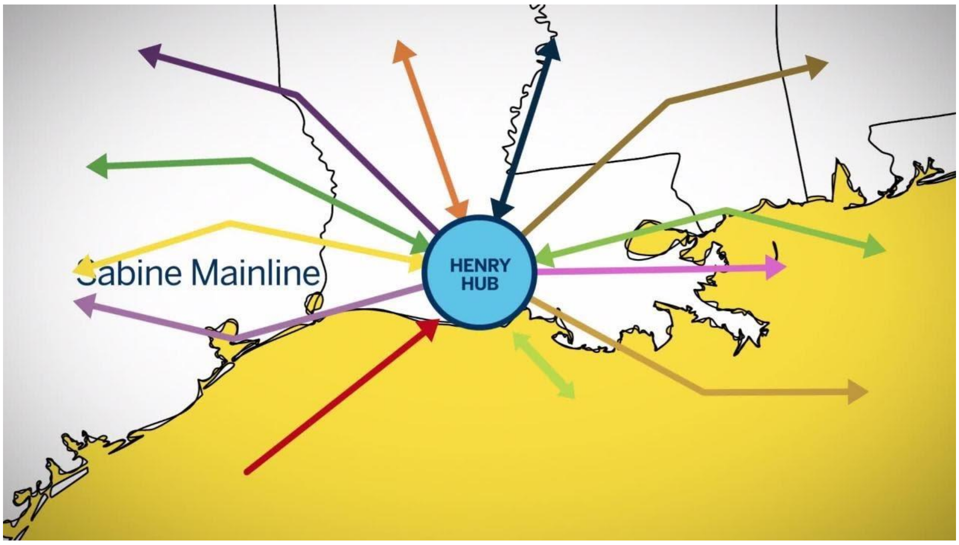


Fig: The Henry Hub Location and its 13 Pipelines

Gas producers rely on the Henry Hub's natural gas spot pricing because of its large trading volume, transparency in pricing. Henry Hub prices are widely quoted by future exchanges and other media sources, so parties to a contact can easily obtain this pricing data. The spot price is given in Dollars per Million BTU (British Thermal Unit). British Thermal Unit is a measure of heat. It is the amount of heat required to raise the temperature of one pound of water by one-degree Fahrenheit. A BTU is equivalent to 1.06 Joules.

The file 'data_updated.csv' is the dataset containing dates and their corresponding day, month, year and price (in Dollars per Million BTU).

Data Pre-processing/ Data Wrangling:

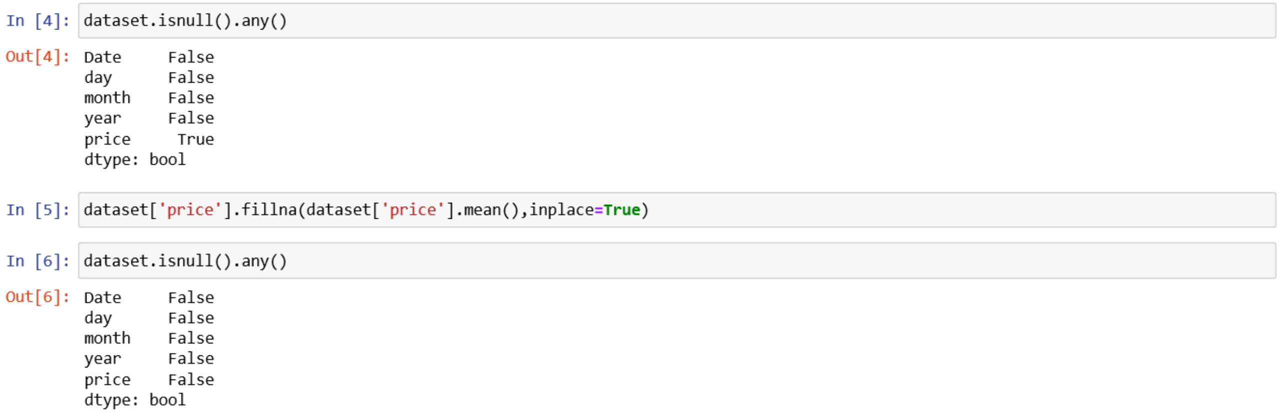
In this stage the data is processed or encoded in such a way that the machine understands it. It involves converting the textual data into machine-understandable format, handling missing values, etc.

Our data consists of numerical data as the date is decomposed into day, month, year and price. Following are the steps to pre-process the data:

- Importing Libraries: Using libraries like pandas to read a dataset and convert it into DataFrame and NumPy to convert DataFrames into arrays and manipulate them

  Pandas is mainly used for machine learning in form of DataFrames. Pandas allow importing data of various file formats such as csv, excel etc. It has various data manipulation operations such as group by, join, merge, melt, concatenation as well as data cleaning features such as filling, replacing or inputting null values. NumPy is used for adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- Reading dataset: The pandas read_csv() is used to read the csv (comma-separated values) file and store it in a DataFrame. The head() and tail() are used to return number of rows specified in the parameter as integer.
- Handling Missing Values: Pandas denote missing values as NaN (Not a Number). There are various methods to check for presence of missing values:
  - ➢ isnull().any(): Returns the column names and their corresponding boolean values if missing data is present. A True for a column label indicates a missing value/s in the corresponding column.
  - ➢ isnull().sum(): returns the number of NaN values in the columns.
  - ➢ isnull(): returns an array of Boolean values indicating whether each corresponding element is missing.

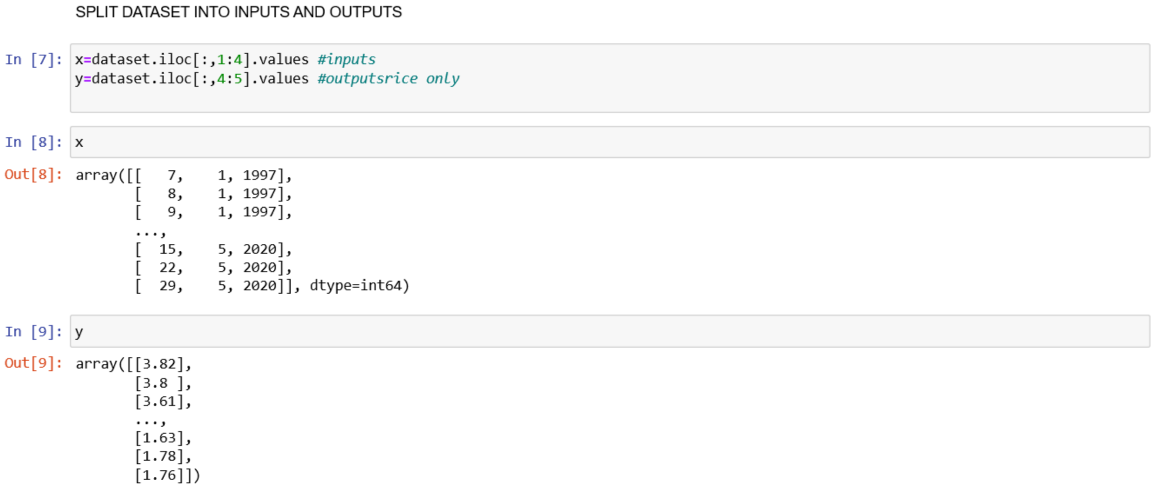
Our dataset has a missing value in price and is replaced by the mean value of price in the dataset using the fillna().



```
In [4]: dataset.isnull().any()

Out[4]: Date      False
        day       False
        month     False
        year      False
        price      True
        dtype: bool

In [5]: dataset['price'].fillna(dataset['price'].mean(),inplace=True)

In [6]: dataset.isnull().any()

Out[6]: Date      False
        day       False
        month     False
        year      False
        price     False
        dtype: bool
```

Fig: Handling missing values

- Converting textual data into numerical data (Label Encoding) and further into binary format data (One Hot Encoding): Not necessary as the data contains only numerical data.
- Split Data into Inputs/Outputs: The input data includes day, month, year and output includes price. These are converted into arrays.

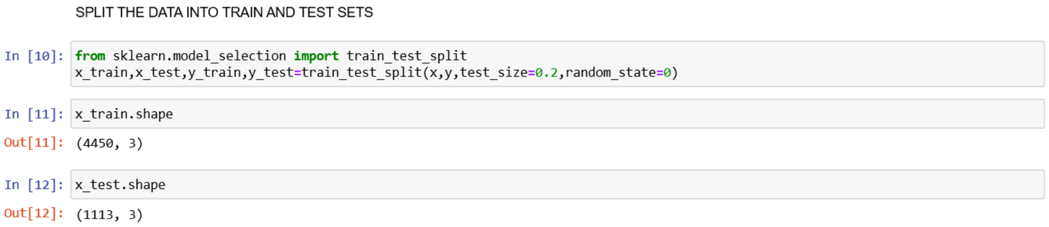


SPLIT DATASET INTO INPUTS AND OUTPUTS

```
In [7]: x=dataset.iloc[:,1:4].values #inputs
        y=dataset.iloc[:,4:5].values #outputsrice only

In [8]: x

Out[8]: array([[   7,    1, 1997],
               [   8,    1, 1997],
               [   9,    1, 1997],
               ...,
               [  15,    5, 2020],
               [  22,    5, 2020],
               [  29,    5, 2020]], dtype=int64)

In [9]: y

Out[9]: array([[3.82],
               [3.8 ],
               [3.61],
               ...,
               [1.63],
               [1.78],
               [1.76]])
```

Fig:Splitting data into input and output sets

- Split data into train and test: The function train_test_split() is used to split dataset into train and test datasets. 20% of the dataset is being kept for the purpose of testing. random_state=0 is used for initializing the internal random number generator, which will decide the splitting of data into train and test indices.



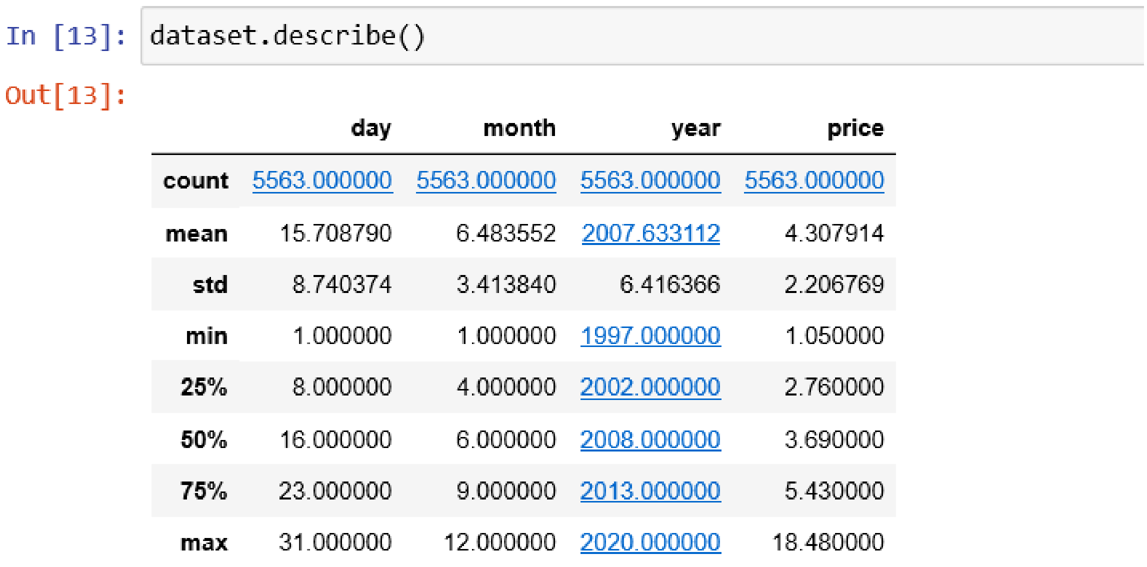
SPLIT THE DATA INTO TRAIN AND TEST SETS

```
In [10]: 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=0)

In [11]: x_train.shape

Out[11]: (4450, 3)

In [12]: x_test.shape

Out[12]: (1113, 3)
```

Fig: Splitting data into train and test datasets

Data Analysis:

1. Statistical Data Analysis: the pandas describe() returns a DataFrame with statistical data like count, mean, median, mode, etc.



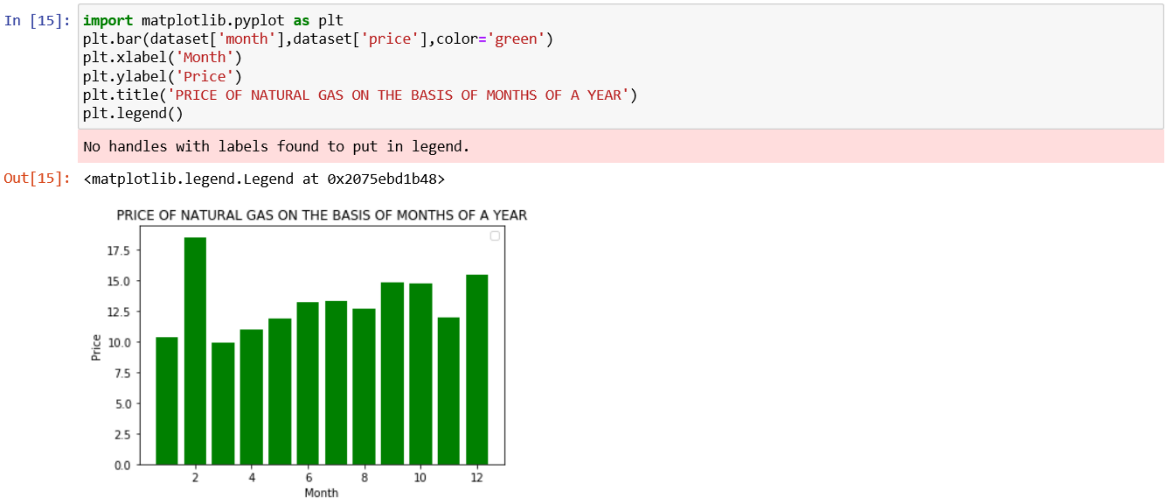
```
In [13]: dataset.describe()
```

Out[13]:

	day	month	year	price
count	5563.000000	5563.000000	5563.000000	5563.000000
mean	15.708790	6.483552	2007.633112	4.307914
std	8.740374	3.413840	6.416366	2.206769
min	1.000000	1.000000	1997.000000	1.050000
25%	8.000000	4.000000	2002.000000	2.760000
50%	16.000000	6.000000	2008.000000	3.690000
75%	23.000000	9.000000	2013.000000	5.430000
max	31.000000	12.000000	2020.000000	18.480000

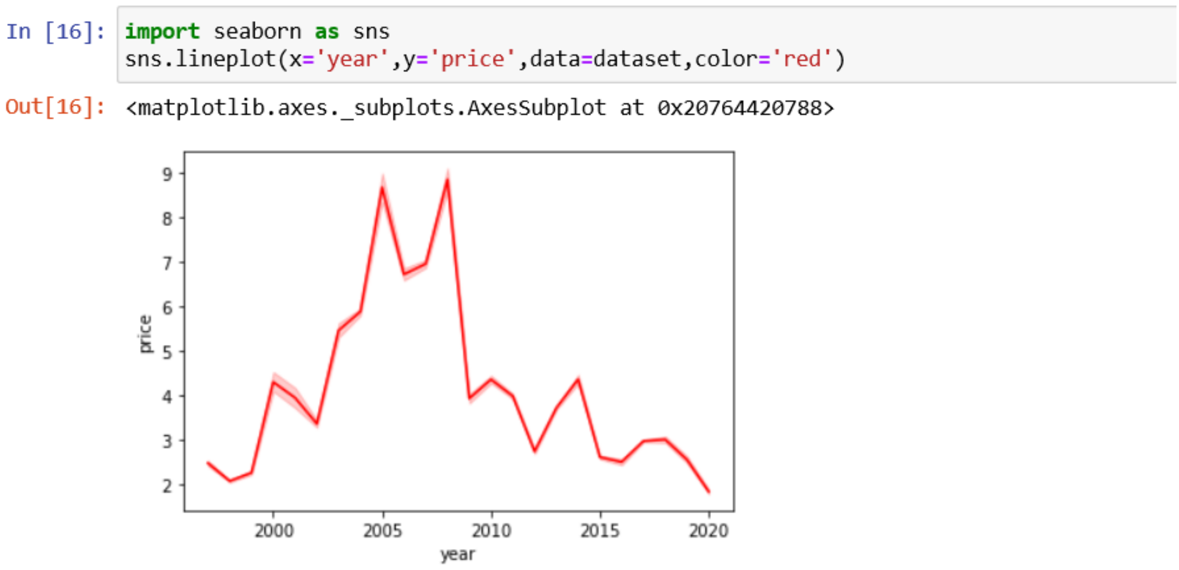
Fig: use of describe()

2. Visualization Analysis: Matplotlib library provides for creating static, animated, and interactive visualizations in Python. The seaborn library is used to provide a high-level interface for drawing attractive and informative statistical graphics. Following are the analysis made:



```
In [15]: import matplotlib.pyplot as plt
         plt.bar(dataset['month'],dataset['price'],color='green')
         plt.xlabel('Month')
         plt.ylabel('Price')
         plt.title('PRICE OF NATURAL GAS ON THE BASIS OF MONTHS OF A YEAR')
         plt.legend()

         No handles with labels found to put in legend.

Out[15]: <matplotlib.legend.Legend at 0x2075ebd1b48>
```

Fig: Data Analysis of month versus price



```
In [16]: import seaborn as sns
         sns.lineplot(x='year',y='price',data=dataset,color='red')

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x20764420788>
```

Fig: Data Analysis of year versus price

Building and Training the Model:

The model to be built has to be built using Regression Machine Learning Algorithms. The Linear Regression ML Algorithm predicts value of dependent variable y depending on the value of independent variable x. Hence it considers only one input but the number of inputs we have are three - day, month, year. Hence Linear Regression ML Algorithm cannot be used. So is with the Polynomial Regression Algorithm one reason is there are more than one inputs the other being that output cannot be modeled as nth degree polynomial. Instead Multiple Linear Regression Algorithm can be used. It uses several explanatory variables to predict the outcome of a response variable. Therefore, it uses day, month and year to predict outcome of price. It uses the equation:

$$y= a_0+a_1x_1+a_2x_2+\dots+a_ix_i$$

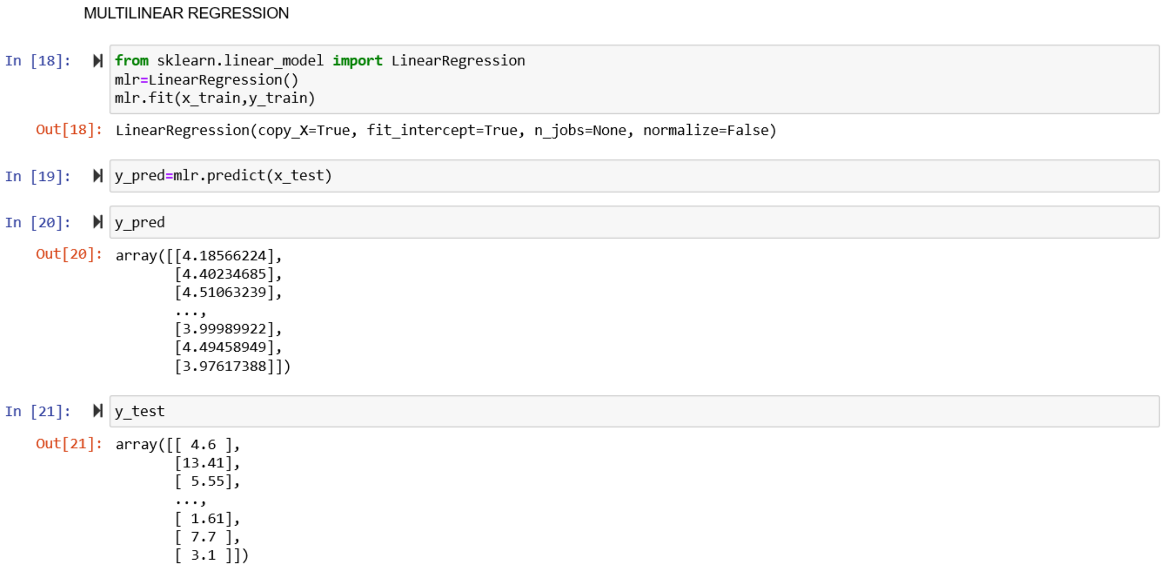
where:

x_i: independent variable

a_0: constant

a_i: slope coefficient of each x_i

However, in this case this algorithm does not prove to be fruitful.



MULTILINEAR REGRESSION

```
In [18]:  from sklearn.linear_model import LinearRegression
          mlr=LinearRegression()
          mlr.fit(x_train,y_train)

Out[18]:  LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

In [19]:  y_pred=mlr.predict(x_test)

In [20]:  y_pred

Out[20]:  array([[4.18566224],
                 [4.40234685],
                 [4.51063239],
                 ...,
                 [3.99989922],
                 [4.49458949],
                 [3.97617388]])

In [21]:  y_test

Out[21]:  array([[ 4.6 ],
                 [13.41],
                 [ 5.55],
                 ...,
                 [ 1.61],
                 [ 7.7 ],
                 [ 3.1 ]])
```

Fig: Implementation of Multiple Linear Regression

The Decision Tree Regression Algorithm proves to be a good option. Decision tree builds regression or classification models in the form of a tree structure by breaking down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. It is based on the famous ID3 Algorithm proposed by J. R. Quinlan.

The primary difference between classification and regression decision trees is that, the classification decision trees are built with unordered values with dependent variables. The regression decision trees take ordered values with continuous values.

The figure below is just for visualizing how the decision tree looks like when its depth is 3. However, increase in depth leads to more accurate results. Since a bigger decision tree cannot be properly visualized here, the visualization of smaller tree is below:

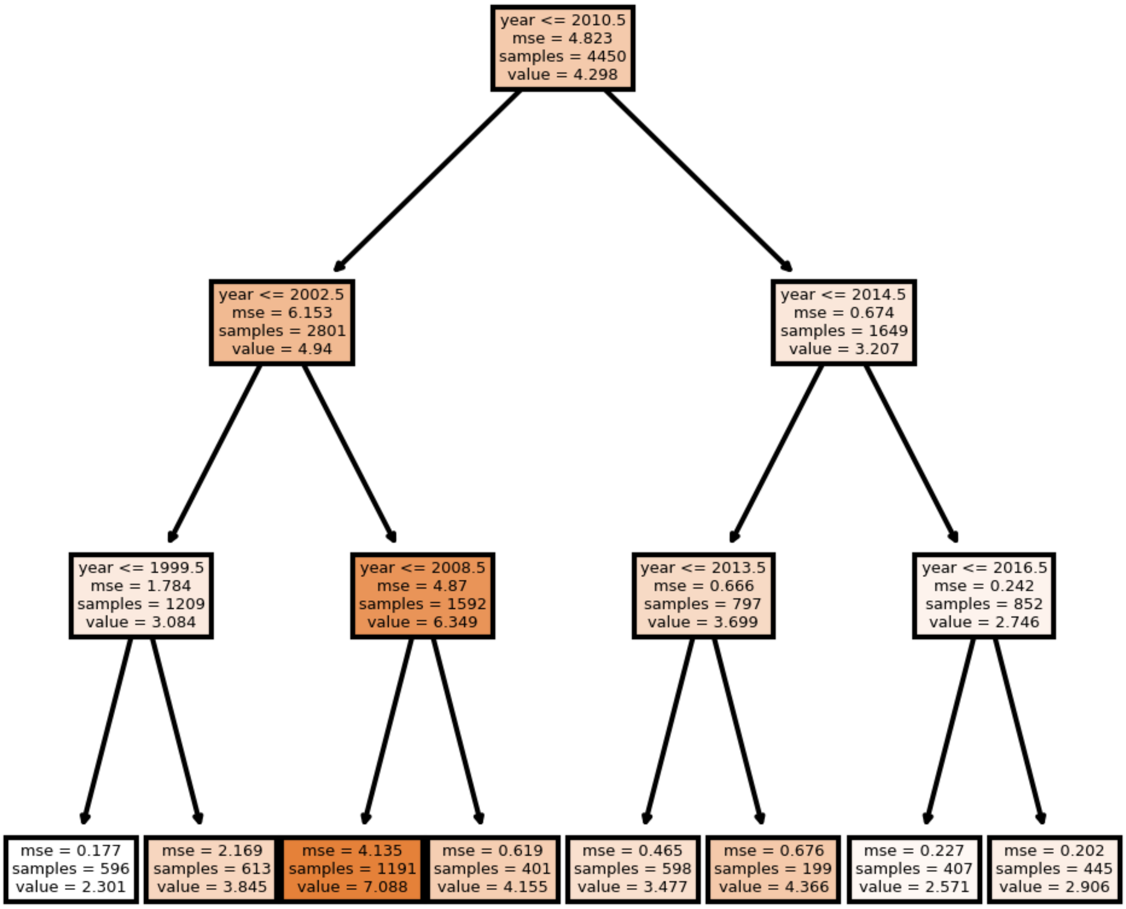
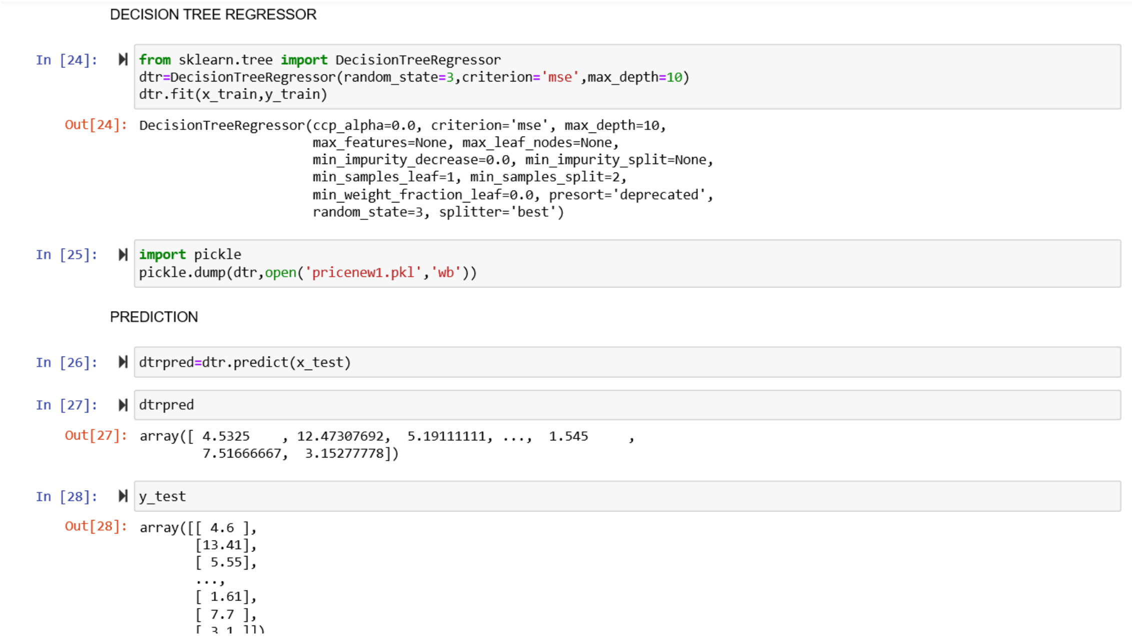


Fig: Decision Tree of depth 3



DECISION TREE REGRESSOR

```
In [24]:  from sklearn.tree import DecisionTreeRegressor
          dtr=DecisionTreeRegressor(random_state=3,criterion='mse',max_depth=10)
          dtr.fit(x_train,y_train)

Out[24]:  DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=10,
                                max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort='deprecated',
                                random_state=3, splitter='best')

In [25]:  import pickle
          pickle.dump(dtr,open('pricenew1.pkl','wb'))
```

PREDICTION

```
In [26]:  dtrpred=dtr.predict(x_test)

In [27]:  dtrpred

Out[27]:  array([ 4.5325    , 12.47307692,  5.19111111, ...,  1.545     ,
                  7.51666667,  3.15277778])

In [28]:  y_test

Out[28]:  array([[ 4.6 ],
                 [13.41],
                 [ 5.55],
                 ...,
                 [ 1.61],
                 [ 7.7 ],
                 [ 3.1 ]])
```

Fig: Implementation of Decision Tree Regression

A Random Forest makes use of multiple decision trees and a technique called Bootstrap Aggregation, commonly known as bagging. Bagging, in the Random

Forest method, involves training each decision tree on a different data sample where sampling is done with replacement.

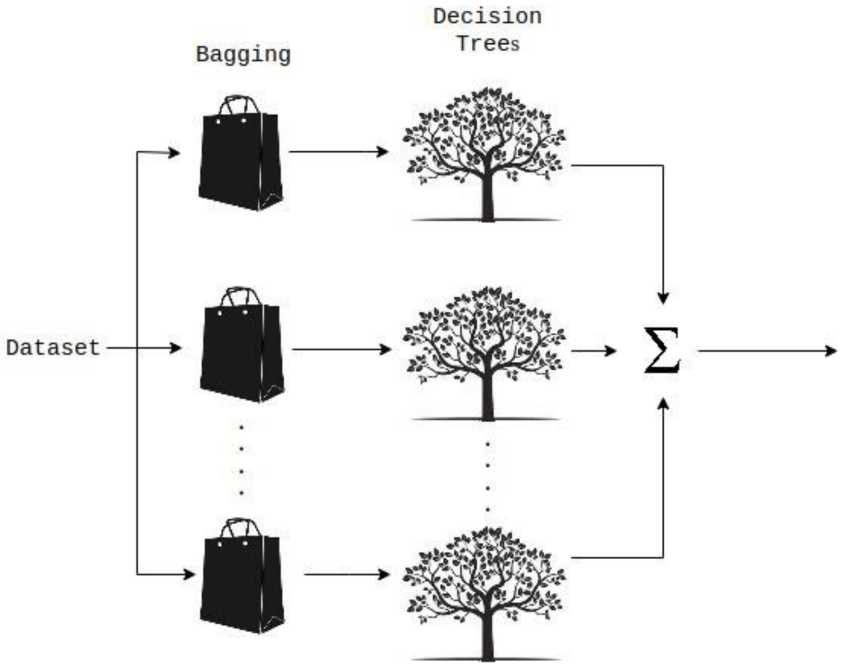
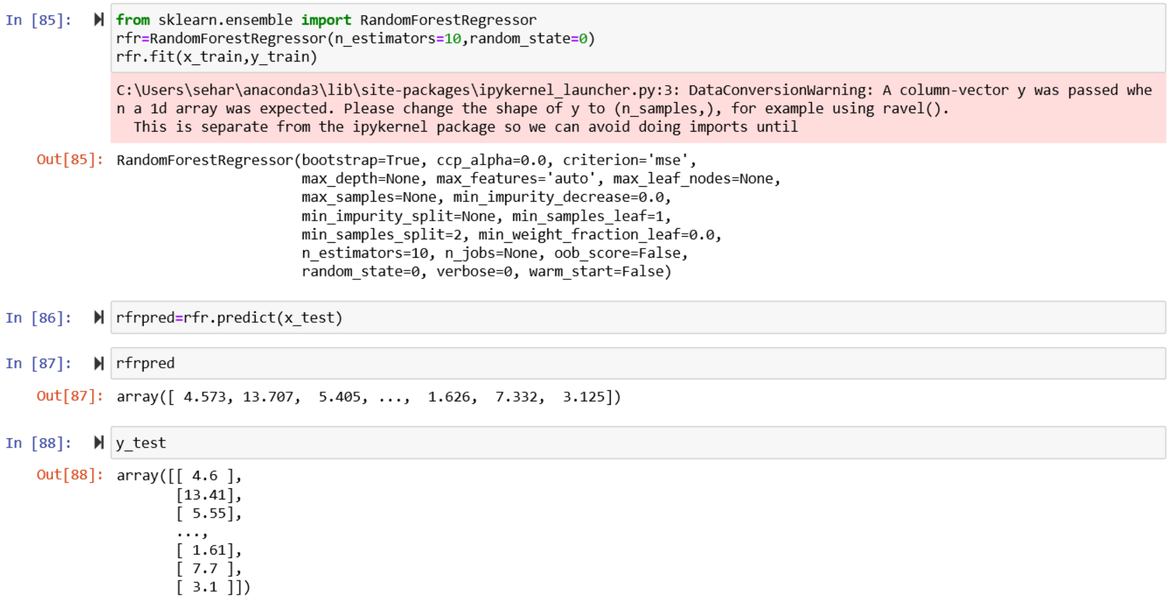


Fig: Random Forest Regression



```
In [85]:  from sklearn.ensemble import RandomForestRegressor
          rfr=RandomForestRegressor(n_estimators=10,random_state=0)
          rfr.fit(x_train,y_train)

          C:\Users\sehar\anaconda3\lib\site-packages\ipykernel_launcher.py:3: DataConversionWarning: A column-vector y was passed whe
          n a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
            This is separate from the ipykernel package so we can avoid doing imports until

Out[85]:  RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                                max_depth=None, max_features='auto', max_leaf_nodes=None,
                                max_samples=None, min_impurity_decrease=0.0,
                                min_impurity_split=None, min_samples_leaf=1,
                                min_samples_split=2, min_weight_fraction_leaf=0.0,
                                n_estimators=10, n_jobs=None, oob_score=False,
                                random_state=0, verbose=0, warm_start=False)

In [86]:  rfrpred=rfr.predict(x_test)

In [87]:  rfrpred

Out[87]:  array([ 4.573, 13.707,  5.405, ...,  1.626,  7.332,  3.125])

In [88]:  y_test

Out[88]:  array([[ 4.6 ],
                 [13.41],
                 [ 5.55],
                 ...,
                 [ 1.61],
                 [ 7.7 ],
                 [ 3.1 ]])
```

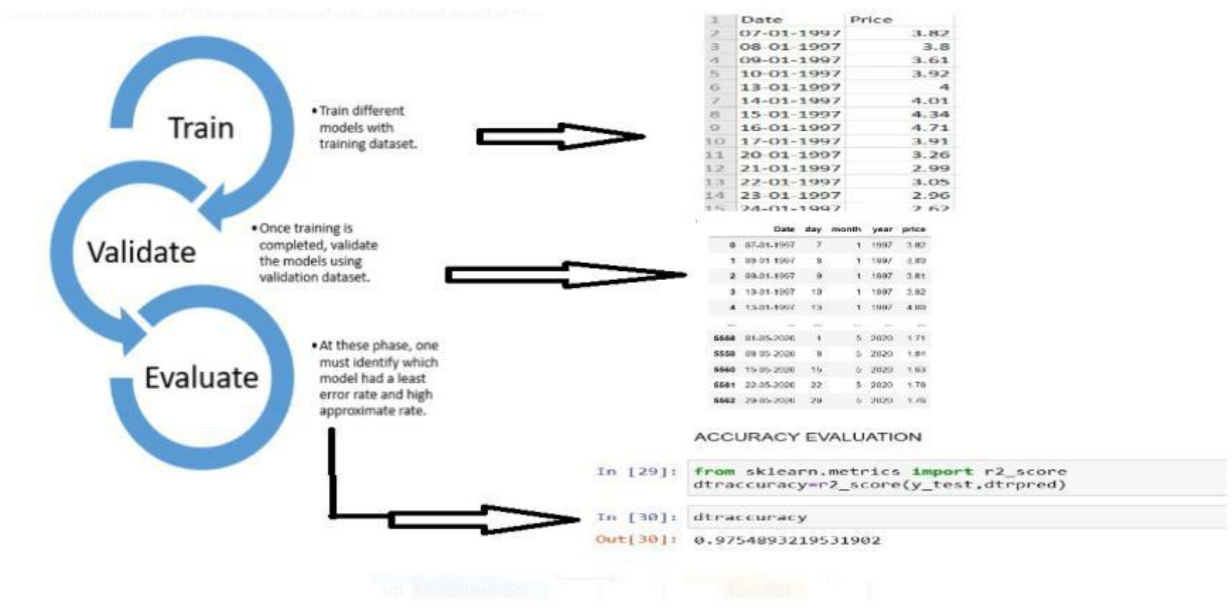
Fig: Implementation of Random Forest Regression

Testing and Evaluation:

When talking about the strategy of Machine Learning testing, think accuracy and efficiency as the main goal. Advantages such as detecting redundant unsuccessful tests, and keeping untested code out of production, prediction, and prevention ultimately reduce much of the risk in the deployment phase. Some of the critical contributions quality assurance include –

- Defect alerts
- Enhanced analytics
- Faster predictions
- Improved optimization
- Cleaner traceability
- Real-time feedback.

The sooner we can implement an in-house AI platform to assist in application testing; we will discover a more accurate and efficient deployment with reduced effort. Using defined test methods and analytics will launch application development to new heights.


Train
•Train different models with training dataset.
Validate
•Once training is completed, validate the models using validation dataset.
Evaluate
•At these phase, one must identify which model had a least error rate and high approximate rate.

1	Date	Price
2	07-01-1997	3.82
3	08-01-1997	3.8
4	09-01-1997	3.61
5	10-01-1997	3.92
6	13-01-1997	4
7	14-01-1997	4.01
8	15-01-1997	4.34
9	16-01-1997	4.71
10	17-01-1997	3.91
11	20-01-1997	3.26
12	21-01-1997	2.99
13	22-01-1997	3.05
14	23-01-1997	2.96
15	24-01-1997	2.62

	Date	day	month	year	price
0	07-01-1997	7	1	1997	3.82
1	08-01-1997	8	1	1997	3.80
2	09-01-1997	9	1	1997	3.61
3	10-01-1997	10	1	1997	3.92
4	13-01-1997	13	1	1997	4.00
...	...	...	...	...	...
5558	01-05-2020	1	5	2020	1.71
5559	08-05-2020	8	5	2020	1.84
5560	15-05-2020	15	5	2020	1.63
5561	22-05-2020	22	5	2020	1.70
5562	29-05-2020	29	5	2020	1.76

ACCURACY EVALUATION

```
In [29]: from sklearn.metrics import r2_score
         dtraccuracy=r2_score(y_test,dtrpred)

In [30]: dtraccuracy

Out[30]: 0.9754893219531902
```

Fig: Steps to evaluate

Based on historical data, create a training data set. Train predictive model using the regression model as it is nominal.

Regression model consists of the following algorithms -

1.Simple Linear Regression

2.MultiLinear Regression

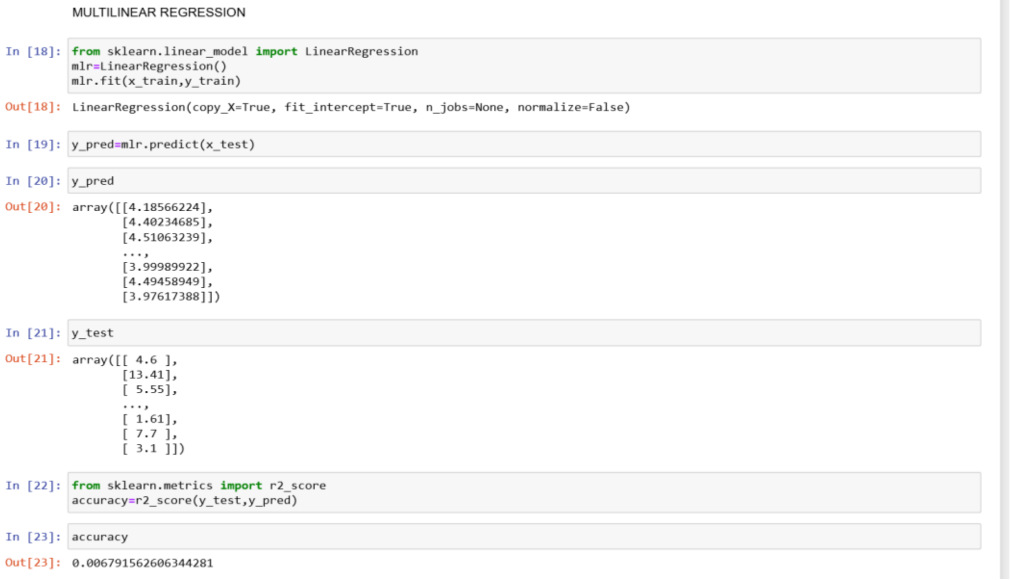
3.Polynomial Regression

4.Decision Tree - Regression

5.Random Forest Regression

The algorithms used for the dataset Natural gas price prediction are Multilinear Regression and Decision Tree Regression. The dataset is trained for these algorithms and then the efficiency is checked for each of them, then the algorithm with the highest accuracy is finalized and is taken under consideration

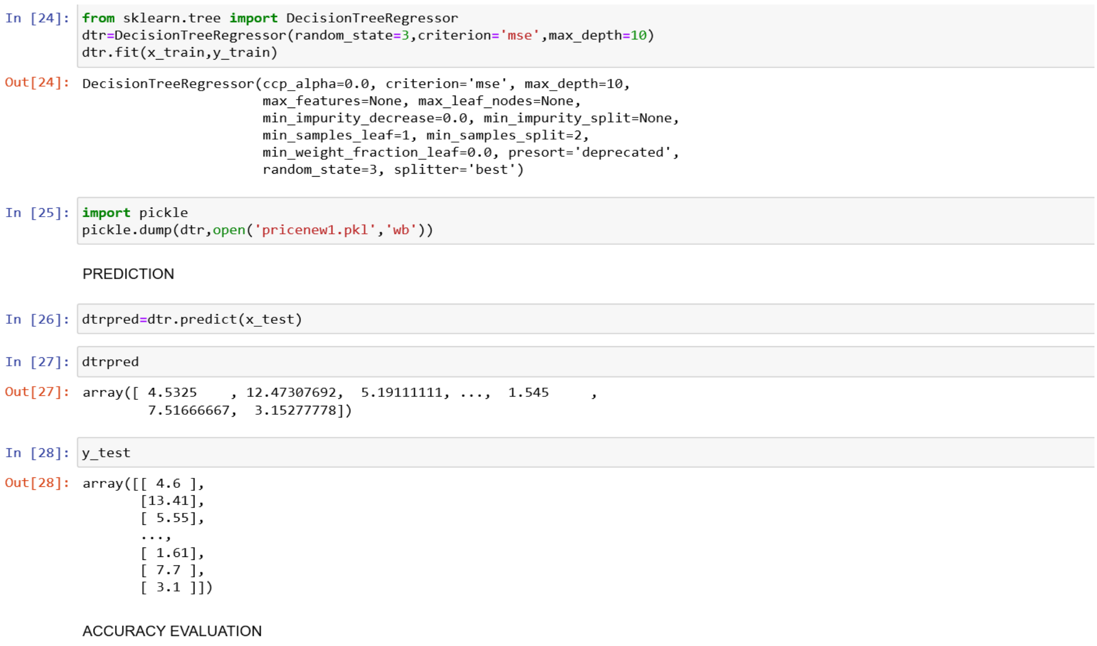
For training, the data and other algorithms are ignored.



MULTILINEAR REGRESSION

```
In [18]: from sklearn.linear_model import LinearRegression
         mlr=LinearRegression()
         mlr.fit(x_train,y_train)

Out[18]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

In [19]: y_pred=mlr.predict(x_test)

In [20]: y_pred

Out[20]: array([[4.18566224],
                [4.40234685],
                [4.51063239],
                ...,
                [3.99989922],
                [4.49458949],
                [3.97617388]])

In [21]: y_test

Out[21]: array([[ 4.6 ],
                [13.41],
                [ 5.55],
                ...,
                [ 1.61],
                [ 7.7 ],
                [ 3.1 ]])

In [22]: from sklearn.metrics import r2_score
         accuracy=r2_score(y_test,y_pred)

In [23]: accuracy

Out[23]: 0.006791562606344281
```

Fig: MultilinearRegression



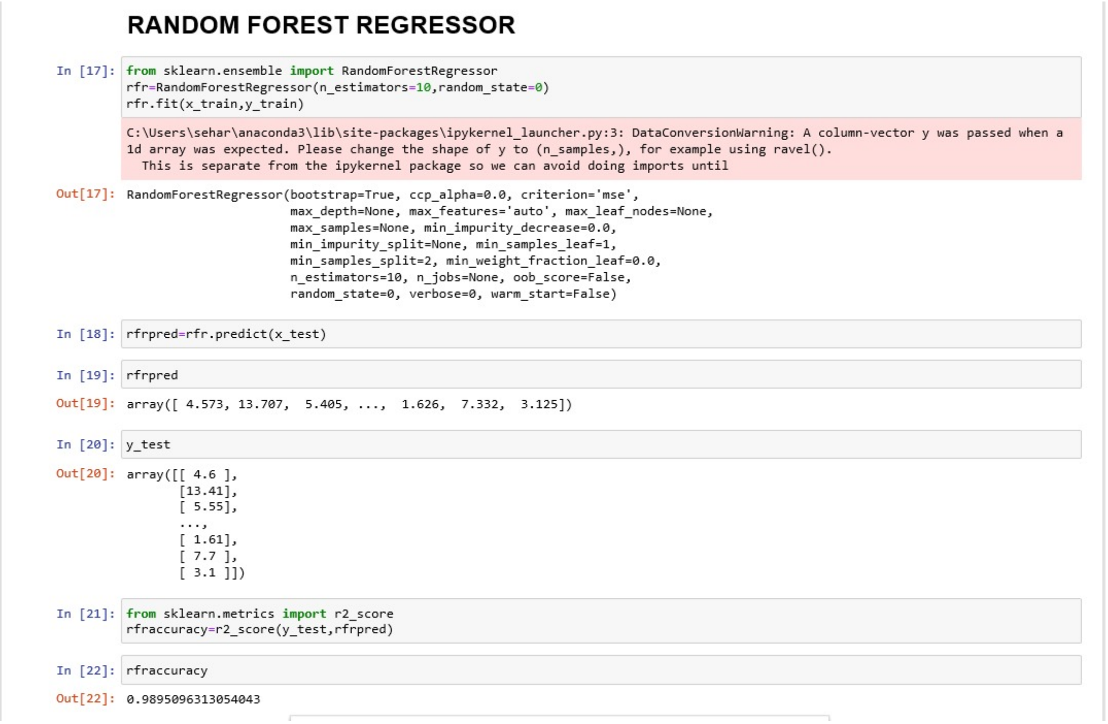
```
In [24]: from sklearn.tree import DecisionTreeRegressor
         dtr=DecisionTreeRegressor(random_state=3,criterion='mse',max_depth=10)
         dtr.fit(x_train,y_train)

Out[24]: DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=10,
                               max_features=None, max_leaf_nodes=None,
                               min_impurity_decrease=0.0, min_impurity_split=None,
                               min_samples_leaf=1, min_samples_split=2,
                               min_weight_fraction_leaf=0.0, presort='deprecated',
                               random_state=3, splitter='best')

In [25]: import pickle
         pickle.dump(dtr,open('pricenew1.pkl','wb'))
```

PREDICTION

```
In [26]: dtrpred=dtr.predict(x_test)

In [27]: dtrpred

Out[27]: array([ 4.5325    , 12.47307692,  5.19111111, ...,  1.545     ,
                 7.51666667,  3.15277778])

In [28]: y_test

Out[28]: array([[ 4.6 ],
                [13.41],
                [ 5.55],
                ...,
                [ 1.61],
                [ 7.7 ],
                [ 3.1 ]])
```

ACCURACY EVALUATION

Fig: Decision Tree Regression



RANDOM FOREST REGRESSOR

```
In [17]: from sklearn.ensemble import RandomForestRegressor
         rfr=RandomForestRegressor(n_estimators=10,random_state=0)
         rfr.fit(x_train,y_train)

         C:\Users\sehar\anaconda3\lib\site-packages\ipykernel_launcher.py:3: DataConversionWarning: A column-vector y was passed when a
         1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
           This is separate from the ipykernel package so we can avoid doing imports until

Out[17]: RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                               max_depth=None, max_features='auto', max_leaf_nodes=None,
                               max_samples=None, min_impurity_decrease=0.0,
                               min_impurity_split=None, min_samples_leaf=1,
                               min_samples_split=2, min_weight_fraction_leaf=0.0,
                               n_estimators=10, n_jobs=None, oob_score=False,
                               random_state=0, verbose=0, warm_start=False)

In [18]: rfrpred=rfr.predict(x_test)

In [19]: rfrpred

Out[19]: array([ 4.573, 13.707,  5.405, ...,  1.626,  7.332,  3.125])

In [20]: y_test

Out[20]: array([[ 4.6 ],
                [13.41],
                [ 5.55],
                ...,
                [ 1.61],
                [ 7.7 ],
                [ 3.1 ]])

In [21]: from sklearn.metrics import r2_score
         rfraccuracy=r2_score(y_test,rfrpred)

In [22]: rfraccuracy

Out[22]: 0.9895096313054043
```

Fig: Random Forest Regressor

Now to check the accuracy for the Algorithm we import r2_score from sklearn.metrics. The metrics module applies functions estimate prediction error for specific purposes. These metrics are detailed in area on Classification metrics, Multilabel ranking metrics, Regression metrics, and Clustering metrics.

R2_score: R2 is a statistical measure of how close the data are to the fitted with the regression line. It is also known as the coefficient of determination or can be the coefficient of multiple determination for multiple regression. It is basically used to check the accuracy between tested value and the value predicted by the machine. In general, a model fits the data well if the differences between the observed values and the predicted values are small. R-squared is always between 0 and 100%:

- 0 percent specifies that the model explains none of the variability of the response data around its mean.
- 100 percent specifies that the model explains all the variability of the response data around its mean.

Graphical representation of R-squared:

The R-squared for the regression model on the left is 80%, and for the model on the right, it is 75%. When a regression model accounts for more of the variance, the data points are closer to the regression line. In practice, you'll never see a regression model with an R2 of 100%. In that case, the fitted values equal the data values and, consequently, all of the observations fall exactly on the regression line.

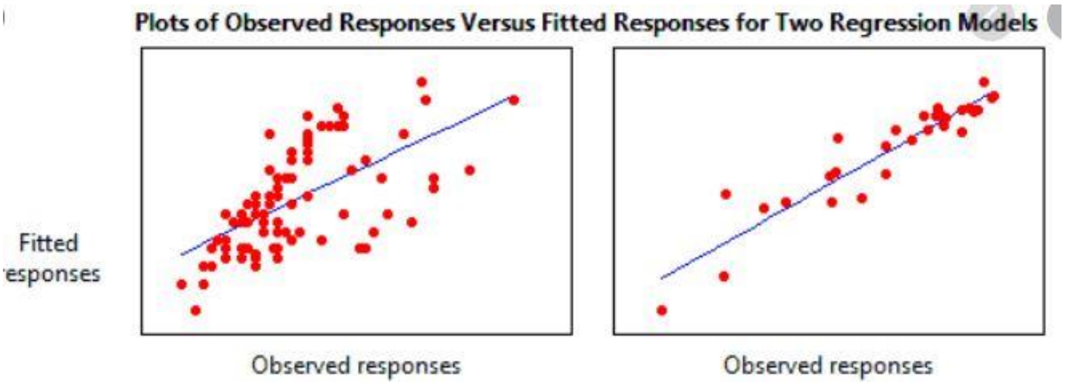
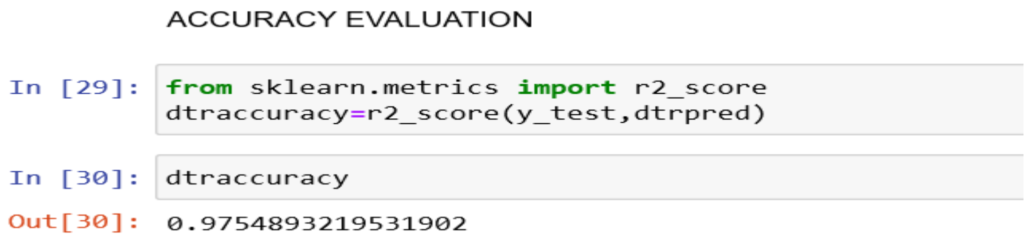


Fig: Visual Representation of R-squared

Accuracy Evaluation: A model must not only predict correctly but do so because it is algorithmically correct, not lucky. For example, if 99% of emails aren't spammed, then classifying all emails as not spam gets 99% accuracy through chance. Therefore, you need to check your model for algorithmic correctness. Follow these steps: Train your model for a few iterations and verify that the loss decreases. Train your algorithm without regularization. If your model is compound enough, it will memorize the training data and your training loss will be close to 0.Test specific sub computations of your algorithm. For example, you can test that a part of your RNN runs once per element of the input data.



ACCURACY EVALUATION

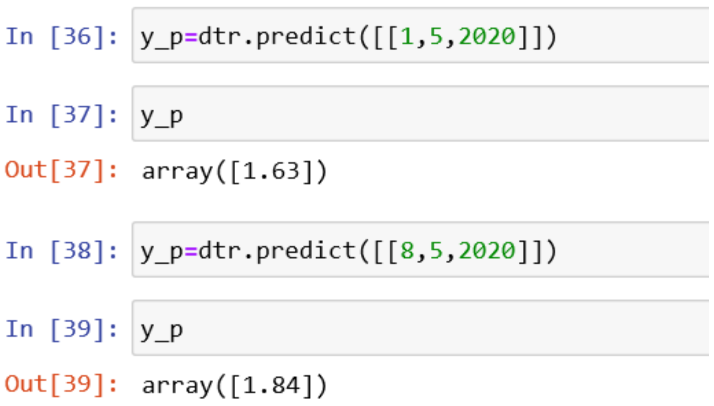
```
In [29]: from sklearn.metrics import r2_score
         dtraccuracy=r2_score(y_test,dtrpred)

In [30]: dtraccuracy

Out[30]: 0.9754893219531902
```

Fig: Decision Tree Regression Accuracy

Sno.	Algorithms	Accuracy	Selected Yes/No	Reasons
1	Multilenear Regression	0.06567	No	Very low accuracy
2	Desision Tree Regression	0.9754	Yes	Accuracy can be used as it is not too low or high
3	Random Forest Regressor	0.9895	No	Machine over trained

Table: Accuracy Analysis

From the above predictions, we can conclude that the decision tree regression algorithm has a maximum accuracy of 97%. Now the next step is model saving using a pickle file. Now let’s test the model by predicting values to check whether the model built is working accordingly



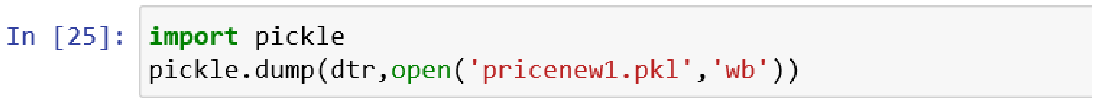
```
In [36]: y_p=dtr.predict([[1,5,2020]])

In [37]: y_p
Out[37]: array([1.63])

In [38]: y_p=dtr.predict([[8,5,2020]])

In [39]: y_p
Out[39]: array([1.84])
```

Fig: Predict values

Saving the Model:

We use sets of data in the form of dictionaries, DataFrames, or any other data types. When working with those, you might want to save them to a file, so you can use them later on or send them to someone else. This is what pickle module is for: it serializes objects so they can be saved to a file. Then can be loaded in a program again later on when needed. Pickling is helpful for applications where you need some degree of persistence in your dataset. As Program's state data can be saved to disk, so you can continue working on it later on. It can also be used to send data over a Transmission Control Protocol or socket connection, or to store python objects in a database. Pickle file is very useful for when you're working with machine learning algorithms, where you want to save them to be able to make new predictions at a later time, without having to rewrite everything or train the model all over again in future when we need it in future.

We can pickle objects with the following data types: Booleans, integers, floats, complex numbers, Strings, Tuples, Lists, Sets, Dictionaries which contain picklable objects. All these can be pickled, but you can also do an equivalent for classes and functions, for instance, if they are defined at the highest level of a module. Not everything is often pickled easily, though: samples of this are generators, inner classes, lambda functions and default dictionaries. With default dictionaries, you need to create them with a module-level function.



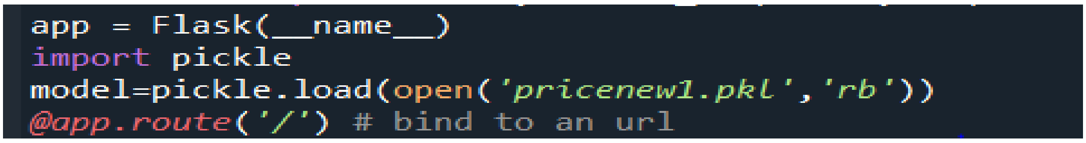
```
In [25]: import pickle
         pickle.dump(dtr,open('pricenew1.pkl','wb'))
```

Fig: To dump the complete model in pickle file

Pickling in Python - The Very Basics

- To save a pickle, use pickle. dump.

- A convention is to name pickle files *. pickle, but you can name it whatever you would like.
- Loading the pickled file from your disk drive is as simple as pickle. Load and specify the file path.
- Save the dataframe to a pickle file called my_df.pickle in the current working directory.



```
app = Flask(__name__)
import pickle
model=pickle.load(open('pricenew1.pkl','rb'))
@app.route('/') # bind to an url
```

Fig: Opening pickle file in Flask

5. FLOWCHART

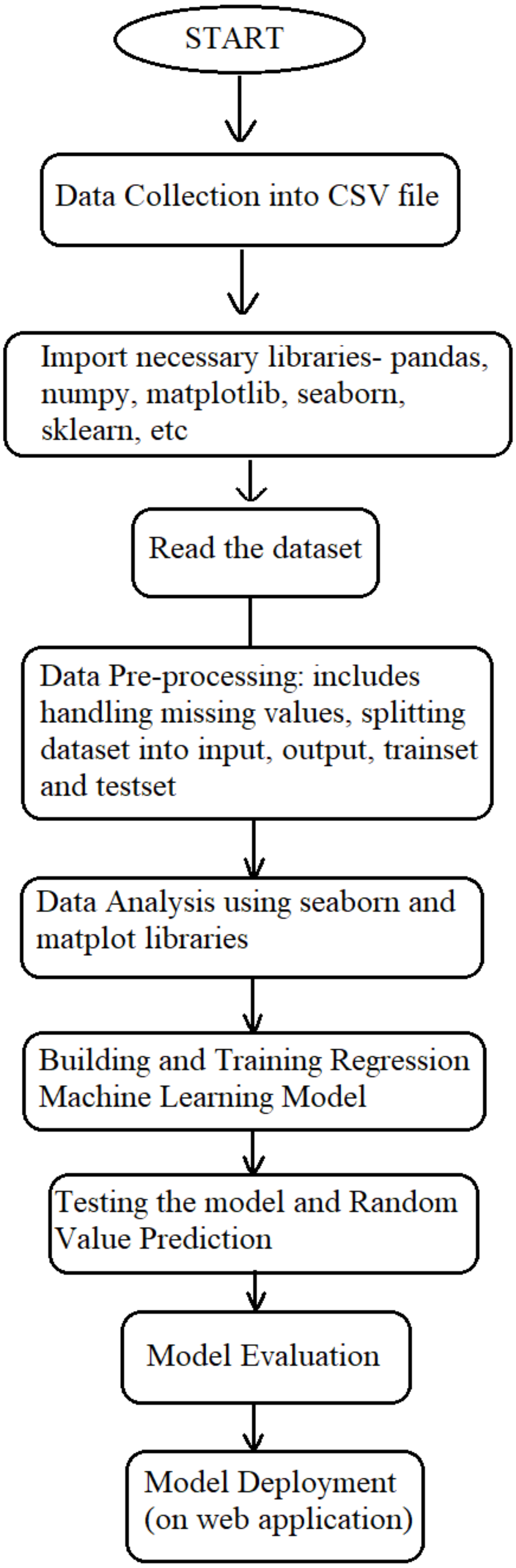

START
Data Collection into CSV file
Import necessary libraries- pandas, numpy, matplotlib, seaborn, sklearn, etc
Read the dataset
Data Pre-processing: includes handling missing values, splitting dataset into input, output, trainset and testset
Data Analysis using seaborn and matplot libraries
Building and Training Regression Machine Learning Model
Testing the model and Random Value Prediction
Model Evaluation
Model Deployment (on web application)

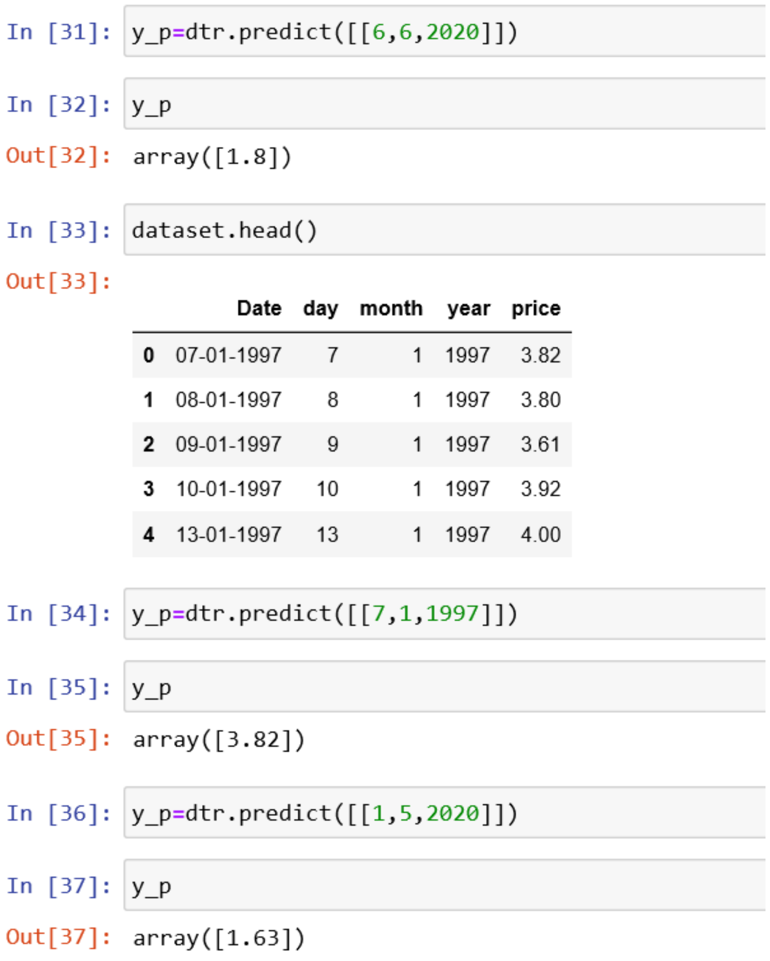
Fig: Flowchart for the Project

6. RESULT

In this the main aim is to predict the Natural Gas Price. The aim of this project is to debate data-driven models for gas price forecasting, which utilize machine learning approaches such as Logistic Regression etc. Finally, the model is integrated to a web application.

As the complete model is saved and stored in a pickle file we then import the complete pickle file in the flask app where we open it so that can be used for building a complete UI consists of the machine learning model and can test any random value to check whether it is predicting correct value as result it values predicted are close to the tested value.

RANDOM VALUE PREDICTION



```
In [31]: y_p=dtr.predict([[6,6,2020]])

In [32]: y_p

Out[32]: array([1.8])

In [33]: dataset.head()

Out[33]:
```

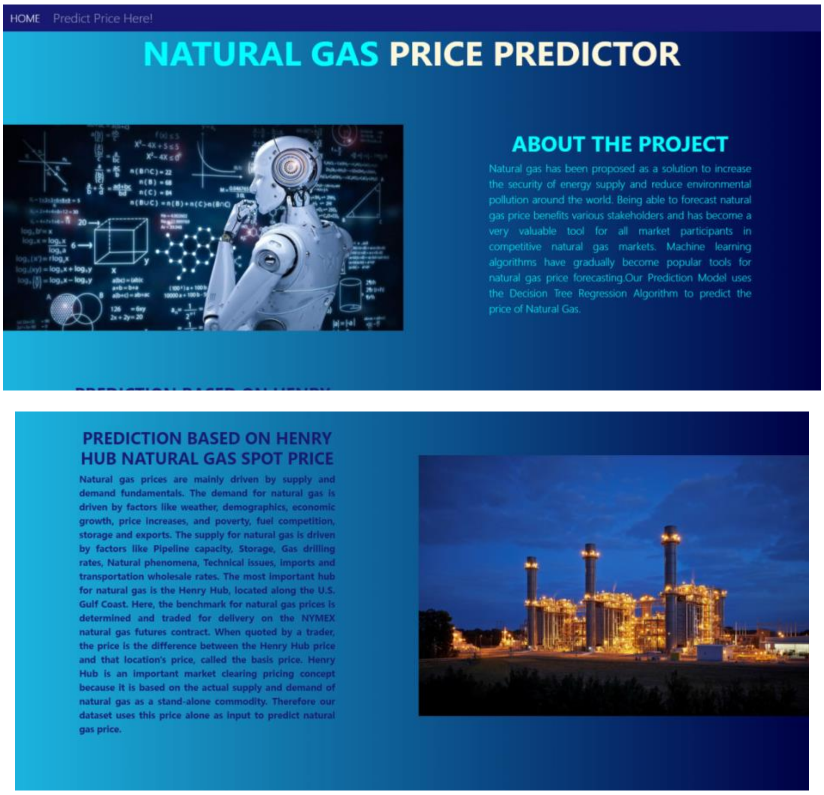
	Date	day	month	year	price
0	07-01-1997	7	1	1997	3.82
1	08-01-1997	8	1	1997	3.80
2	09-01-1997	9	1	1997	3.61
3	10-01-1997	10	1	1997	3.92
4	13-01-1997	13	1	1997	4.00

```
In [34]: y_p=dtr.predict([[7,1,1997]])

In [35]: y_p

Out[35]: array([3.82])

In [36]: y_p=dtr.predict([[1,5,2020]])

In [37]: y_p

Out[37]: array([1.63])
```

Fig: Random value check

The interface (UI) is that the point of human-computer interaction and communication during a device. This can include display screens, keyboards, a mouse and therefore the appearance of a desktop. It is also the way through which a user interacts with an application or a website. As the application is created using Flask app.

Flask (source code) is a Python web framework built with a a little core and easy-to-extend philosophy. Flask is taken into account more Pythonic than the Django web framework because in common situations the equivalent Flask web application is more explicit. This is developed with HTML5, CSS3, and bootstrap. The navbar consists of two links one of the existing page and the other is to link to the next page where the price is predicted. The page consists of the abstract of project and in the next page is completely for prediction.



NATURAL GAS PRICE PREDICTOR

ABOUT THE PROJECT

Natural gas has been proposed as a solution to increase the security of energy supply and reduce environmental pollution around the world. Being able to forecast natural gas price benefits various stakeholders and has become a very valuable tool for all market participants in competitive natural gas markets. Machine learning algorithms have gradually become popular tools for natural gas price forecasting.Our Prediction Model uses the Decision Tree Regression Algorithm to predict the price of Natural Gas.

PREDICTION BASED ON HENRY HUB NATURAL GAS SPOT PRICE

Natural gas prices are mainly driven by supply and demand fundamentals. The demand for natural gas is driven by factors like weather, demographics, economic growth, price increases, and poverty, fuel competition, storage and exports. The supply for natural gas is driven by factors like Pipeline capacity, Storage, Gas drilling rates, Natural phenomena, Technical issues, imports and transportation wholesale rates. The most important hub for natural gas is the Henry Hub, located along the U.S. Gulf Coast. Here, the benchmark for natural gas prices is determined and traded for delivery on the NYMEX natural gas futures contract. When quoted by a trader, the price is the difference between the Henry Hub price and that location's price, called the basis price. Henry Hub is an important market clearing pricing concept because it is based on the actual supply and demand of natural gas as a stand-alone commodity. Therefore our dataset uses this price alone as input to predict natural gas price.

Fig: Home page

The Homepage consists of nothing much its just the introduction part to the project that is the abstract and henry hub spot price the theory is specified using paragraph in html and images are stored by using image tag the page is divided using two columns. In order to make the page attractive we use gradient colours as background the gradient colour used in the home page background: #000046; background: -webkit-linear-gradient(to right, #1CB5E0, #000046); background: linear-gradient(to right, #1CB5E0, #000046), it is the shade of light and dark blue.

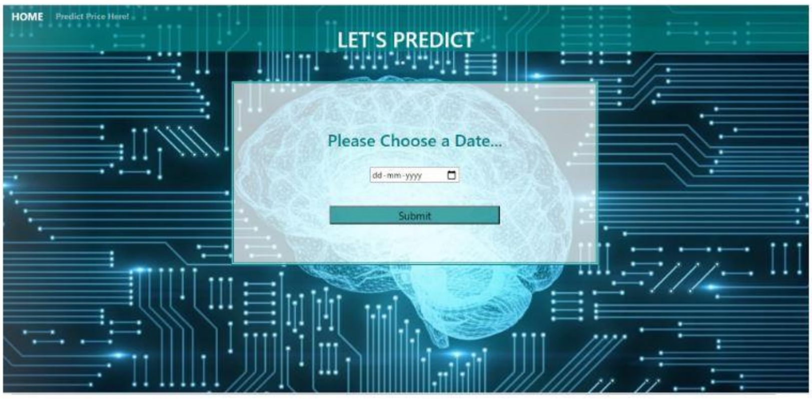
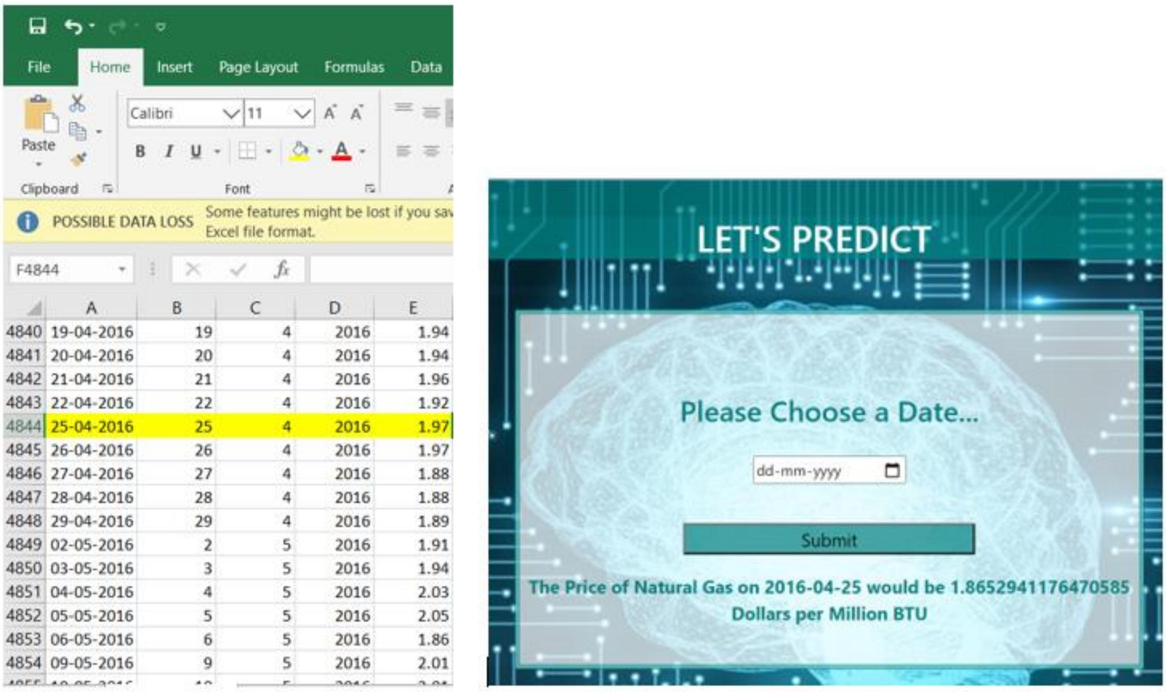


Fig: Prediction Page

The Prediction Page also consists of a navbar which have to href links, one for the previous page that is home page and other is the prediction page. It consists of a form which has a calendar to choose a date of all years you can select a date from any year. It also consists a today button in calendar which keeps the present date. The form consists of a submit button, which on clicking will post the predicted value.

The figure 12 consists of both the dates, to check whether the UI works properly as the model built in jupyter notebook which clearly shows yes. The date chosen is 25-04-2016 and we obtain the predicting price as 1.86 while in dataset it is 1.97 which is close to each other. Therefore, it works well.



POSSIBLE DATA LOSS Some features might be lost if you sav Excel file format.

	A	B	C	D	E
4840	19-04-2016	19	4	2016	1.94
4841	20-04-2016	20	4	2016	1.94
4842	21-04-2016	21	4	2016	1.96
4843	22-04-2016	22	4	2016	1.92
4844	25-04-2016	25	4	2016	1.97
4845	26-04-2016	26	4	2016	1.97
4846	27-04-2016	27	4	2016	1.88
4847	28-04-2016	28	4	2016	1.88
4848	29-04-2016	29	4	2016	1.89
4849	02-05-2016	2	5	2016	1.91
4850	03-05-2016	3	5	2016	1.94
4851	04-05-2016	4	5	2016	2.03
4852	05-05-2016	5	5	2016	2.05
4853	06-05-2016	6	5	2016	1.86
4854	09-05-2016	9	5	2016	2.01

The Price of Natural Gas on 2016-04-25 would be 1.8652941176470585 Dollars per Million BTU

Fig: Comparison of price on date 25-04-2016 in the dataset and the UI

7. MERITS AND DEMERITS

Merits:

- Being able to forecast natural gas price benefits various stakeholders and has become a very valuable tool for all market participants in competitive natural gas markets.
- Machine learning algorithms have gradually become popular tools for natural gas price forecasting.
- We can know the values without actually having to go the sites and search for them.
- We can link the prices of the past years and predict an approximate value for the coming years

Demerits:

- Due to sudden situations or natural calamities, the values we predicted will be wrong.
- High error-susceptibility.
- Another major challenge is the ability to accurately interpret results generated by the algorithms as the condition may not be same always.

8. APPLICATIONS

Natural Gas Price Prediction is helpful for businessmen to make decisions on investing in project and also to the customers to plan to make purchases when a suitable rate is predicted. Natural Gas has been proposed as a solution to increase the security of energy supply and reduce environmental pollution around the world. Being able to forecast natural gas price benefits various stakeholders and has become a very valuable tool for all market participants in competitive natural gas markets.

9. CONCLUSION

The model is trained to take the date as an input from the user, check the price as per the dataset we trained it on, and give the predicted price as the output. The webpage designed gives an introduction about Natural Gas and Machine Learning. It then navigates us to another page, where the date is asked from the user. When the date is entered in the given box, it predicts and displays the price in Million Dollars Per BTU.

The entire process of taking the inputs, evaluating the variables and then predicting the output works on the Decision Tree Regression algorithm under Machine Learning.

10. FUTURE SCOPE

Since this is a simple model, predicting based on dates, we intend to include the location as an input and predict values for it as the price changes depending on the location.

This prediction gives values in Million Dollars per BTU, which is the standard currency for US. Change in currencies can be added in this, to ease the predictions according to the countries where it is accessed.

Since the future prediction is not a part of this model, we can add in features like past record, set aside the natural calamities, add the increase or decrease in the demand for automobiles etc. And design the model to be able to predict the approximate price for future dates, so that investments and business can be thought of and planned well.

11. BIBILOGRAPHY

[1]https://github.com/datasets/natural-gas

[2] https://fred.stlouisfed.org/series/WHHNGSP

[3] https://en.wikipedia.org/wiki/Natural_gas_prices

[4] https://www.investopedia.com/terms/n/nymex.asp

[5]https://www.investopedia.com/terms/h/henry_hub.asp#:~:text=Gas%20producers%20can%20rely%20on,easily%20obtain%20this%20pricing%20data.

[6]https://pandas.pydata.org/pandasdocs/stable/reference/api/pandas.read_csv.html

[7]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[8] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.describe.html

[9]https://www.investopedia.com/terms/m/mlr.asp#:~:text=Multiple%20linear%20regression%20(MLR)%2C%20also%20known%20simply%20as%20multiple,uses%20just%20one%20explanatory%20variable.

[10]https://saedsayad.com/decision_tree_reg.htm#:~:text=Decision%20Tree%20%2D%20Regression,decision%20nodes%20and%20leaf%20nodes.

[11]https://medium.com/datadriveninvestor/random-forest-regression-9871bc9a25eb

[12]https://scikit-learn.org/stable/modules/model_evaluation.html

[13] https://statisticsbyjim.com/regression/interpret-r-squared-regression/

[14] https://blog.minitab.com/blog/adventures-in-statistics-2/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit#:~:text=R%2Dsquared%20is%20a%20statistical,multiple%20determination%20for%20multiple%20regression.&text=0%25%20indicates%20that%20the%20model,response%20data%20around%20its%20mean.

[15] https://docs.w3cub.com/scikit_learn/modules/generated/sklearn.metrics.r2_score/

[16]https://developers.google.com/machine-learning/testing-debugging/pipeline/deploying

[17] https://scikit-learn.org/stable/supervised_learning.html

[18]https://scikit-learn.org/stable/modules/neural_networks_supervised.html#regression

[19] https://www.xenonstack.com/insights/machine-learning-model-testing/

[20]https://www.datacamp.com/community/tutorials/pickle-python-tutorial?utm_source=adwords_ppc&utm_campaignid=1455363063&utm_adgroupid=65083631748&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adpostion=&utm_creative=332602034364&utm_targetid=dsa-429603003980&utm_loc_interest_ms=&utm_loc_physical_ms=9062152&gclid=Cj0KCQjww_f2BRC-ARIsAP3zarG1bRn-lDn1EeoB6LAyHkFAkRHWD961hFdRLijJ2C8zzjeJKYaKAUsaAnRSEALw_wcB

[21]https://realpython.com/the-ultimate-flask-front-end/

[22]https://medium.com/app-affairs/9-applications-of-machine-learning-from-day-to-day-life-112a47a429d0

[23]https://www.fullstackpython.com/flask.html

[24]https://en.wikipedia.org/wiki/Natural_gas

[25] https://www.geeksforgeeks.org/

APPENDIX

A. CODE

Flask code:

```
from flask import Flask , render_template , request
app = Flask(__name__)
import pickle
model=pickle.load(open('pricenew1.pkl','rb'))
@app.route('/') # bind to an url
def helloworld():
    return render_template("demo1.html")
@app.route('/login2')#url
def user():
    return render_template("pro.html")
@app.route('/login3',methods = ['POST']) # bind to an url
def admin():
    from datetime import datetime
    date_time_str = request.form["preddate"]
    date_time_obj = datetime.strptime(date_time_str, '%Y-%m-%d')
    #print ("The type of the date is now",  type(date_time_obj))
    #print ("The date is", date_time_obj)
    year_inp=date_time_obj.year
    month_inp=date_time_obj.month
    day_inp=date_time_obj.day
    t=[[int(day_inp),int(month_inp),int(year_inp)]]
    y=model.predict(t)
    return render_template("pro.html", y ="The Price of Natural Gas on "+date_time_str+" would be "+str(y[0])+" Dollars per Million BTU")
```

```
if __name__ == '__main__' :
    app.run(debug = True)
```

Homepage HTML, CSS, Bootstrap Code:

```
<html>
<head>
<head>
<!-- Required meta tags -->
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">


<!-- Google Font -->
<link href="https://fonts.googleapis.com/css?family=Nunito:200,300,400,700" rel="stylesheet">


<!-- Bootstrap CSS -->
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
            crossorigin="anonymous">


<!-- Custom CSS -->
<link rel="stylesheet" href="/static/demo1.css">


<title>gas prediction</title>


</head>


</head>
<body  style=" background: #000046;  /* fallback for old browsers */ background: -webkit-linear-gradient(to right, #1CB5E0, #000046);  /* Chrome 10-25, Safari 5.1-6 */ background: linear-gradient(to right, #1CB5E0, #000046); /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */;">
```

```html
<nav id="mainnavbar" class="navbar navbar-dark navbar-expand-md py-0 " style="background-color:rgb(25, 25, 112)">
<a href="#" class="navbar-brand" style="color:white;">HOME</a>
<button class="navbar-toggler" data-toggle="collapse" data-target="#navLinks" aria-label="Toggle navigation">
<span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navLinks">
<ul class="navbar-nav">
<li class="nav-item">
<a  href="/login2" class="nav-link">Predict Price Here!</a>
</li>
</ul>
</div>
</nav>
<h1 style="text-align:center;color:rgb(0,255,255);font-weight:bold;">NATURAL GAS <span style="color:rgb(255,248,220);font-weight:bold;">PRICE PREDICTOR</span></h1>
<section class="container-fluid px-0 ">
<div class="row align-items-center content">
<div class="col-md-6 order-2 order-md-1">
<img class="img-fluid" src="/static/mac2.jpg">
</div>
<div class="col-md-6 text-center order-1 order-md-2">
<div class="row justify-content-center">
<div class="col-10 col-lg-8 blurb mb-5 mb-md-0">
<h2 style="color:rgb(0, 255, 255);font-weight:bold;font-size:40px;">ABOUT THE PROJECT</h2>
<p class="lead text-justify" style="color:rgb(0, 255, 255);font-size:20px;">Natural gas has been proposed as a solution to increase the security of energy supply and reduce environmental pollution around the world. Being able to forecast natural gas price benefits various stakeholders and has become a very valuable tool for all market participants in competitive natural gas markets. Machine learning algorithms have gradually become popular tools for natural gas price forecasting.Our Prediction Model uses the Decision Tree Regression Algorithm to predict the price of Natural Gas.</p>
```

```
</div>
</div>
</div>
</div>
<div class="row align-items-center content">
<div class="col-md-6 text-center">
<div class="row justify-content-center">
<div class="col-10 col-lg-8 blurb mb-5 mb-md-0">
<h2 style="color:rgb(25, 25, 112);font-weight:bold;">PREDICTION BASED ON HENRY HUB NATURAL GAS SPOT PRICE</h2>
<p class="lead text-justify" style="color:rgb(25, 25, 112);font-weight:bold;">Natural gas prices are mainly driven by supply and demand fundamentals.
                                        The demand for natural gas is driven by factors like weather, demographics, economic growth, price increases, and poverty, fuel competition, storage and exports.
                                        The supply for natural gas is driven by factors like Pipeline capacity, Storage, Gas drilling rates, Natural phenomena, Technical issues, imports and transportation wholesale rates.
                                        The most important hub for natural gas is the Henry Hub, located along the U.S. Gulf Coast. Here, the benchmark for natural gas prices is determined and traded for delivery on the NYMEX natural gas futures contract.
                                        When quoted by a trader, the price is the difference between the Henry Hub price and that location's price, called the basis price.
                                        Henry Hub is an important market clearing pricing concept because it is based on the actual supply and demand of natural gas as a stand-alone commodity.
                                        Therefore our dataset uses this price alone as input to predict natural gas price.</p>
</div>
</div>
</div>
<div class="col-md-6">
<img class="img-fluid" src="/static/gas1.jpeg">
</div>
</div>
```

```
</section>

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"

        crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js" integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"

        crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js" integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy"

        crossorigin="anonymous"></script>

</script>

</body>

</html>
```

Prediction Page:

```
<!DOCTYPE html>

<html>

<head>

<!-- Required meta tags -->

<style>

                body{background-image:url('/static/bg9.jpg');background-color:Ivory;background-size: cover;height: 100%;}

</style>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">


<!-- Google Font -->

<link href="https://fonts.googleapis.com/css?family=Nunito:200,300,400,700" rel="stylesheet">


<!-- Bootstrap CSS -->
```

```
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
integrity="sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLP
MO"
        crossorigin="anonymous">


     <title></title>




</head>
<body>
     <nav id="mainnavbar" class="navbar navbar-dark navbar-expand-md py-0 "
style="background-color:rgba(0, 128, 128,0.7);">
<a href="/" class="navbar-brand" style="font-weight:bold;">HOME</a>
<button class="navbar-toggler" data-toggle="collapse" data-target="#navLinks" aria-
label="Toggle navigation" >
<span class="navbar-toggler-icon"></span>
</button>

<div class="collapse navbar-collapse" id="navLinks">
<ul class="navbar-nav">
<li class="nav-item">
<a  href="/login2" class="nav-link" style="font-weight:bold;">Predict Price Here!</a>
</li>
</ul>
</div>
</nav>
<!--<div class="jumbotron jumbotron-fluid bg ">
<div class="container">
<h1 class="display-4" style="text-align:center;color:black;font-weight:bold;">LET'S
PREDICT!!!</h1>
<p class="lead">T</p>
```

```
</div>

</div>-->


        <h1 style="color:white;text-align:center;background-color:rgba(0, 139,
139,0.7);">LET'S PREDICT</h1>

        <section class="container-fluid px-0">

<div class="row align-items-center">

<div>

<div id="headinggroup" class="text-white text-center d-none d-lg-block mt-5"
style="margin:100px 450px">

<form action="/login3" method = "post" style="width:700px;height:400px;background-
color:rgba(255,255,255,0.7); border:5px double rgb(0, 139, 139);" >

<p><br><br><br>

<h2 style="color:rgb(0, 128, 128);">Please Choose a Date... </h2><br>

<input style="" type="date" id="start" name="preddate"value="predate"

                  min="07-01-1997" max="29-05-2020">

<br><br>

<center><p style="margin:20px;"><input class="a" type = "submit" value = "Submit"
style="background-color:rgba(0, 128, 128, 0.7);font-
size:20px;width:50%;"/></p></center>

<b style="color:rgb(0,128,128);font-size:20px;">{{y}}</b>

</form>

</p>


</div>

</div>

</div>

</section>

        <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"

      crossorigin="anonymous"></script>

</body>

</html>
```