Model Development Phase Template

Date	15 july 2024
Team ID	740032
Project Title	Price prediction of natural gas using machine learning approach.
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:

```
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)
```

+ Code + Text

training the model with decision tree

```
[ ] import numpy as np
    import pandas as pd  # Use pandas as pd, not np
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.model_selection import train_test_split
    from sklearn.tree import DecisionTreeRegressor

    # #loading the datset
    data=pd.read_csv('/content/daily_csv.csv')

    # Convert 'Date' column to datetime objects
    data['Date'] = pd.to_datetime(data['Date'])

    # Extract numerical features from the datetime column if needed
    data['Year'] = data['Date'].dt.year
    data['Month'] = data['Date'].dt.month
    data['Day'] = data['Date'].dt.day
    data=data.drop(['Date'],axis=1)
```

```
[ ] pd.get_dummies(data)
```

	Price	Year	Month	Day
0	3.82	1997	1	7
1	3.80	1997	1	8
2	3.61	1997	1	9
3	3.92	1997	1	10
4	4.00	1997	1	13
...	...	...	...	...
5933	2.23	2020	8	5
5934	2.26	2020	8	6
5935	2.15	2020	8	7
5936	2.18	2020	8	10
5937	2.19	2020	8	11

5938 rows × 4 columns

```
[ ] x_train=data.drop(['Price'],axis=1)
    y_train=data['Price']
    x_test=data.drop(['Price'],axis=1)
    y_test=data['Price']
```

```
[ ] data['Price'].fillna(data['Price'].mean(),inplace=True)
```

```
[ ] model=DecisionTreeRegressor()
    model.fit(x_train,y_train)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

```
[ ] y_pred=model.predict(x_test)
    y_pred
```

```
array([3.82, 3.8 , 3.61, ..., 2.15, 2.18, 2.19])
```

```
[ ] model.predict([[2023,7,26]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionT
  warnings.warn(
array([1.85])
```

```
[ ] from sklearn.metrics import r2_score
    accuracy=r2_score(y_test,y_pred)
    accuracy
```

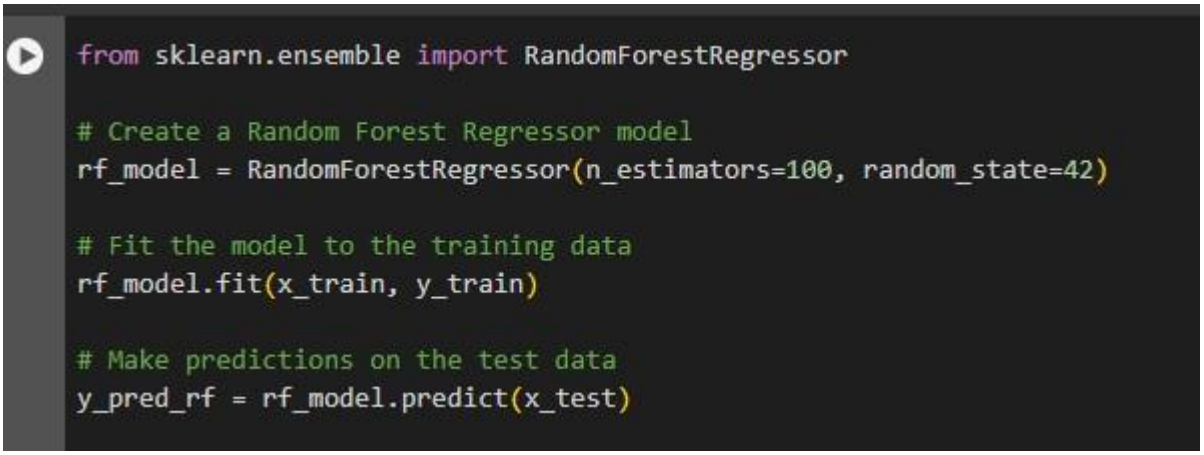
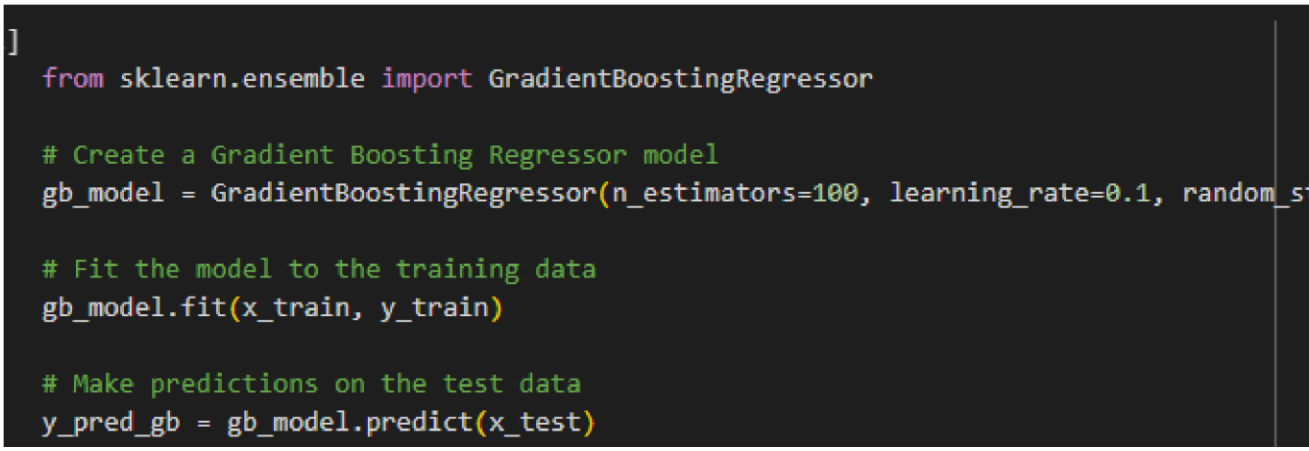
```
1.0
```

```
[ ] train_predictions = model.predict(x_train)
    test_predictions = model.predict(x_test)

    train_r2 = r2_score(y_train, train_predictions)
    test_r2 = r2_score(y_test, test_predictions)

    print(f'Training R² score: {train_r2}')
    print(f'Test R² score: {test_r2}')
```

```
Training R² score: 1.0
Test R² score: 1.0
```

Model Validation and Evaluation Report:

Decision Tree	``` import numpy as np import pandas as pd # Use pandas as pd, not np import matplotlib.pyplot as plt import seaborn as sns from sklearn.model_selection import train_test_split from sklearn.tree import DecisionTreeRegressor # #loading the datset data=pd.read_csv('/content/daily_csv.csv') # Convert 'Date' column to datetime objects data['Date'] = pd.to_datetime(data['Date']) # Extract numerical features from the datetime column if needed data['Year'] = data['Date'].dt.year data['Month'] = data['Date'].dt.month data['Day'] = data['Date'].dt.day data=data.drop(['Date'],axis=1) ```

Random forest	
Gradient Boosting	

```
from sklearn.ensemble import RandomForestRegressor

# Create a Random Forest Regressor model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)

# Fit the model to the training data
rf_model.fit(x_train, y_train)

# Make predictions on the test data
y_pred_rf = rf_model.predict(x_test)
```

```
]
from sklearn.ensemble import GradientBoostingRegressor

# Create a Gradient Boosting Regressor model
gb_model = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, random_s

# Fit the model to the training data
gb_model.fit(x_train, y_train)

# Make predictions on the test data
y_pred_gb = gb_model.predict(x_test)
```