Overview

- Introduction
- Abstract
- Methodology
- Steps Performed
- Data Extraction
- Data Visualization
- Decomposition of Data
- Models:- MAE ,MSE ,RMSE , Decision Tree ,Random Forest ,Linear Regression,
- Conclusion

Introduction

- Combined-cycle power plants represent an efficient approach to energy generation by combining gas turbines, steam turbines, and heat recovery steam generators within a single cycle.
- This system capitalizes on the high-efficiency potential of using exhaust gases from gas turbines to power steam turbines, thereby increasing overall energy production.
- The primary objective of this project is to develop a predictive model for energy output as a function of specific ambient conditions and exhaust vacuum levels.
- By analyzing these factors, we aim to provide actionable insights for optimizing plant operations and maximizing energy efficiency.

Abstract

This project develops a predictive model for energy output in a combined-cycle power plant, utilizing ambient and operational data to optimize performance. Combined-cycle power plants generate electricity efficiently by harnessing both gas and steam turbines, where exhaust heat from gas turbines powers steam turbines in a continuous cycle.

The model leverages 9,568 observations over six years, focusing on key variables: temperature, exhaust vacuum, ambient pressure, and relative humidity. By quantifying how these factors influence energy production, this model provides insights that allow operators to make real-time adjustments for improved efficiency.

Goal: Enable data-driven decisions that maximize energy output and optimize resource management, enhancing overall plant productivity.

Methodology

Training and Testing were both done in Jupyter Notebook, Google Collab which provides a Jupyter notebook environment and free RAM and GPU to train and test the models.

Programming Language:- Python

Libraries:- Pandas, Numpy, Seaborn, Sklearn etc

Algorithm:- MAE, MSE, RMSE, Decision Tree, Random Forest, Linear Regression

Steps performed

1) Importing Data
2) Split the Data into Training & Test sets
3) Creating & Training the Model
4) Making Predictions
5) Evaluating & Improving Predictions
6) Streamlit App Development

Dataset extraction

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

```
#reading the file
file_path = '/Altamash/Excelr code/PROJECTS/PROJECT_ ENERGY_PRODUCTION/Copy of Regrerssion_energy_produ
df = pd.read_csv(file_path, delimiter=';')
```

```
#In Exploratory Data Analysis (EDA), the head() function is used to display the first few rows of a dat
df.head()
```

	temperature	exhaust_vacuum	amb_pressure	r_humidity	energy_production
0	9.59	38.56	1017.01	60.10	481.30
1	12.04	42.34	1019.72	94.67	465.36
2	13.87	45.08	1024.42	81.69	465.48
3	13.72	54.30	1017.89	79.08	467.05
4	15.14	49.64	1023.78	75.00	463.58

```
#In Exploratory Data Analysis (EDA), the tail() function is used to display the last few rows of a data
df.tail()
```

Continue...

	temperature	exhaust_vacuum	amb_pressure	r_humidity	energy_production
0	9.59	38.56	1017.01	60.10	481.30
1	12.04	42.34	1019.72	94.67	465.36
2	13.87	45.08	1024.42	81.69	465.48
3	13.72	54.30	1017.89	79.08	467.05
4	15.14	49.64	1023.78	75.00	463.58

```
#to get no.of rows and columns
df.shape
```

```
(9568, 5)
```

```
#to get basic info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9568 entries, 0 to 9567
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   temperature        9568 non-null   float64
 1   exhaust_vacuum     9568 non-null   float64
 2   amb_pressure       9568 non-null   float64
 3   r_humidity         9568 non-null   float64
 4   energy_production  9568 non-null   float64
dtypes: float64(5)
memory usage: 373.9 KB
```

Removing duplicate values

```
#checking for missing values
df.isnull().sum()
```

```
temperature          0
exhaust_vacuum       0
amb_pressure         0
r_humidity           0
energy_production    0
dtype: int64
```

No missing values in the dataset

```
#checking for duplicates values
df.duplicated().sum()
```

```
41
```

41 duplicates in the dataset

```
#remove duplicated from the dataset
df.drop_duplicates(inplace=True)
```

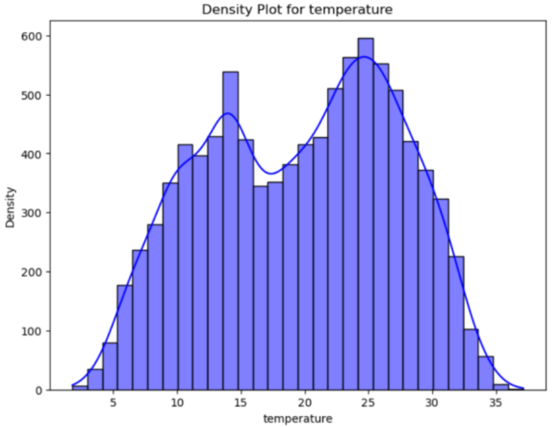
```
#recheck for duplicates in the dataset
df.duplicated().sum()
```

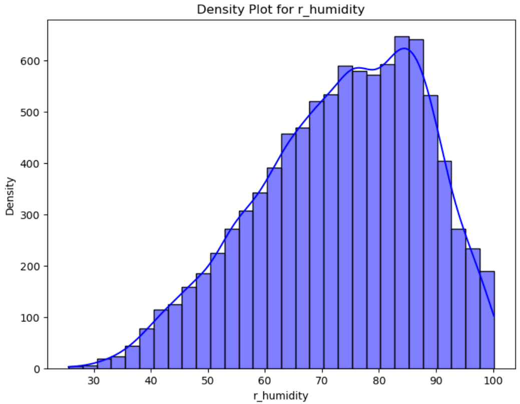
```
0
```

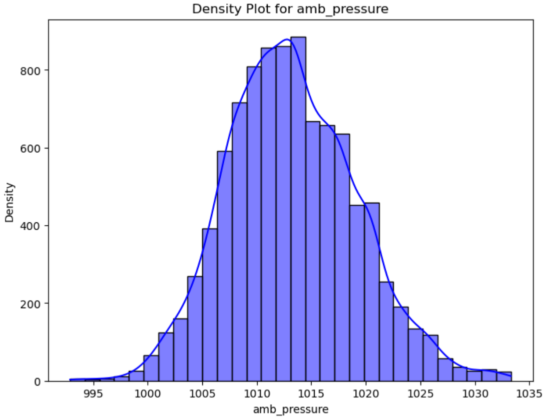
Data visualization

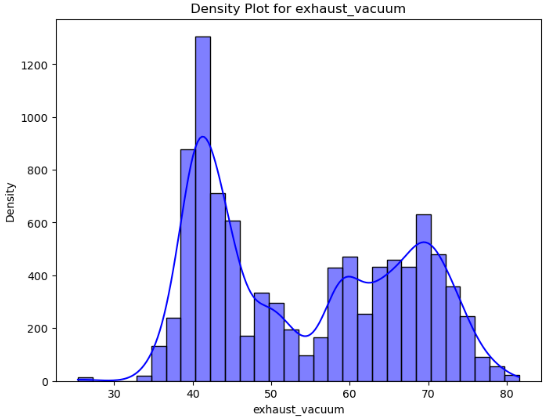
- Histogram: Shows data distribution by frequency across intervals.
- Box Plot: Displays data spread, center, and outliers.
- Pairplot: Visualizes relationships between pairs of variables.
- Correlation Matrix: Shows correlation values between variables.
- Scatter Plot: Plots relationships between two continuous variables.

Histogram

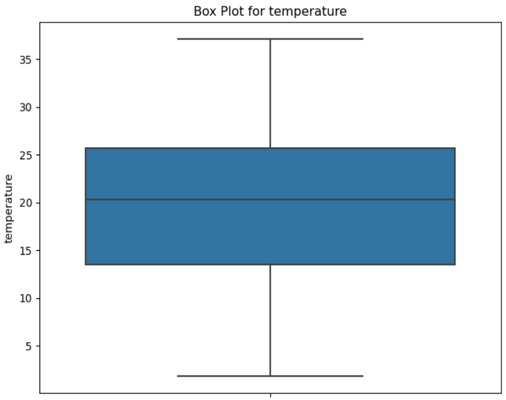

Density Plot for temperature
Density
temperature
0
100
200
300
400
500
600
5
10
15
20
25
30
35

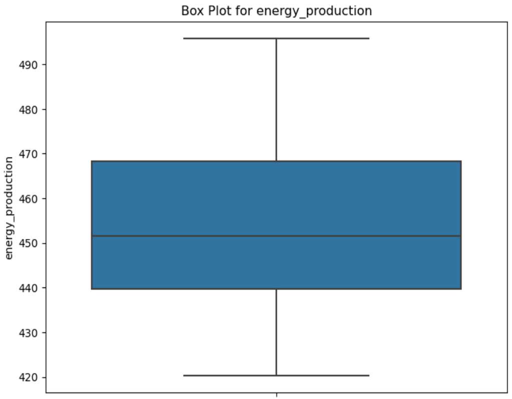

Density Plot for r_humidity
Density
r_humidity
0
100
200
300
400
500
600
30
40
50
60
70
80
90
100

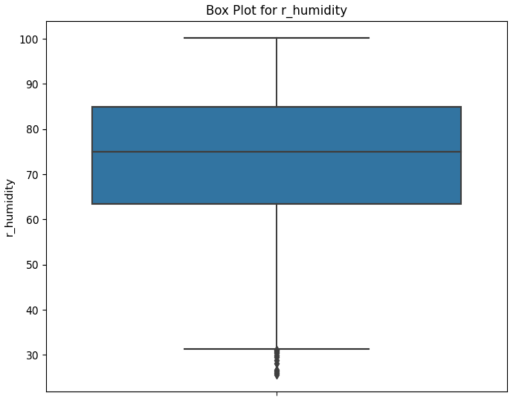

Density Plot for amb_pressure
Density
amb_pressure
0
200
400
600
800
995
1000
1005
1010
1015
1020
1025
1030
1035

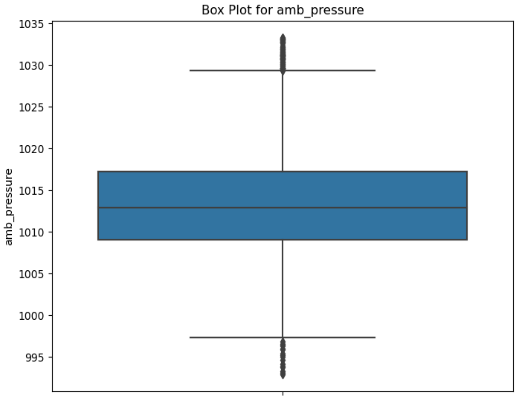

Density Plot for exhaust_vacuum
Density
exhaust_vacuum
0
200
400
600
800
1000
1200
30
40
50
60
70
80

Box plot


Box Plot for temperature
temperature
35
30
25
20
15
10
5

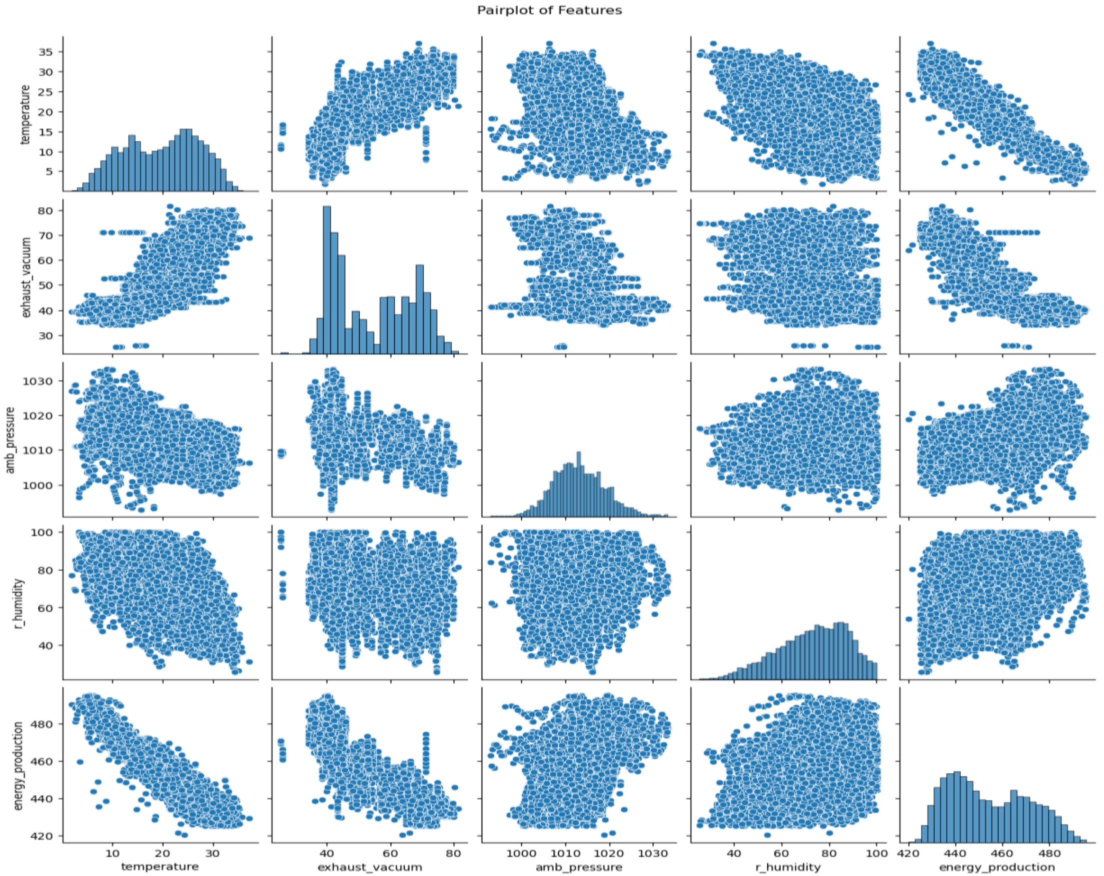

Box Plot for energy_production
energy_production
490
480
470
460
450
440
430
420


Box Plot for r_humidity
r_humidity
100
90
80
70
60
50
40
30


Box Plot for amb_pressure
amb_pressure
1035
1030
1025
1020
1015
1010
1005
1000
995

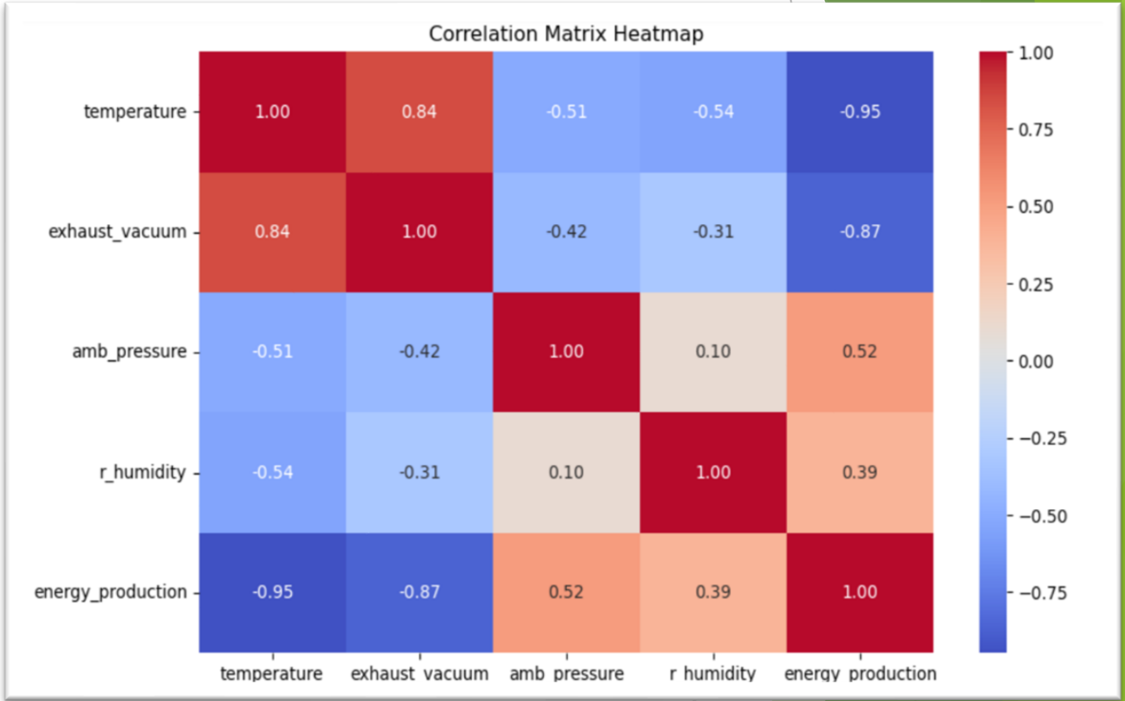
Pairplots

- This pairplot shows the relationships between several variables in a dataset.
- Diagonal plots show the distribution of each variable, while off-diagonal scatter plots reveal interactions between pairs of variables.
- Notably, temperature and energy production have a negative correlation, and there's a pattern between exhaust vacuum and energy production, suggesting these may be important relationships for further analysis.
- Temperature and energy production go in opposite directions: when one goes up, the other goes down.
- Exhaust vacuum and energy production seem linked in a clear pattern.



Correlation Matrix

- Temperature and energy production: Strong negative (-0.95), meaning higher temperatures reduce energy production.
- Exhaust vacuum and energy production: Strong negative (-0.87), higher vacuum reduces energy production.
- Ambient pressure and energy production: Moderate positive (0.52), higher pressure increases energy production.
- Temperature and exhaust vacuum: Strong positive (0.84), both increase together.
- Relative humidity and energy production: Weak positive (0.39), slight increase in energy with higher humidity.



Train and Evaluate Different Regression Models

The models listed are:

1. Linear Regression
2. Ridge Regression
3. Lasso Regression
4. Elastic Net
5. Random Forest
6. Decision Tree
7. Bagging Regressor
8. XGBoost
9. AdaBoost
10. Gradient Boosting
11. K-Nearest Neighbors

Continue….

```
Elastic Net
====================================
Model performance for Training set
- r2_score: 0.9279
- mean_squared_error: 4.5722
- mean_absolute_error: 3.6418
-----------------------------------
Model performance for Test set
- r2_score: 0.9278
- mean_squared_error: 4.5850
- mean_absolute_error: 3.6635
====================================
```

```
Linear Regression
====================================
Model performance for Training set
- r2_score: 0.9284
- mean_squared_error: 4.5561
- mean_absolute_error: 3.6213
-----------------------------------
Model performance for Test set
- r2_score: 0.9283
- mean_squared_error: 4.5693
- mean_absolute_error: 3.6441
====================================
```

```
Random Forest
====================================
Model performance for Training set
- r2_score: 0.9945
- mean_squared_error: 1.2590
- mean_absolute_error: 0.8852
-----------------------------------
Model performance for Test set
- r2_score: 0.9617
- mean_squared_error: 3.3396
- mean_absolute_error: 2.3406
====================================
```

```
Decision Tree
====================================
Model performance for Training set
- r2_score: 1.0000
- mean_squared_error: 0.0000
- mean_absolute_error: 0.0000
-----------------------------------
Model performance for Test set
- r2_score: 0.9327
- mean_squared_error: 4.4272
- mean_absolute_error: 2.9695
====================================
```

```
Bagging Regressor
====================================
Model performance for Training set
- r2_score: 0.9923
- mean_squared_error: 1.4974
- mean_absolute_error: 0.9987
-----------------------------------
Model performance for Test set
- r2_score: 0.9588
- mean_squared_error: 3.4648
- mean_absolute_error: 2.4380
====================================
```

```
XGBoost
====================================
Model performance for Training set
- r2_score: 0.9878
- mean_squared_error: 1.8781
- mean_absolute_error: 1.3643
-----------------------------------
Model performance for Test set
- r2_score: 0.9647
- mean_squared_error: 3.2042
- mean_absolute_error: 2.2316
====================================
```

Continue….

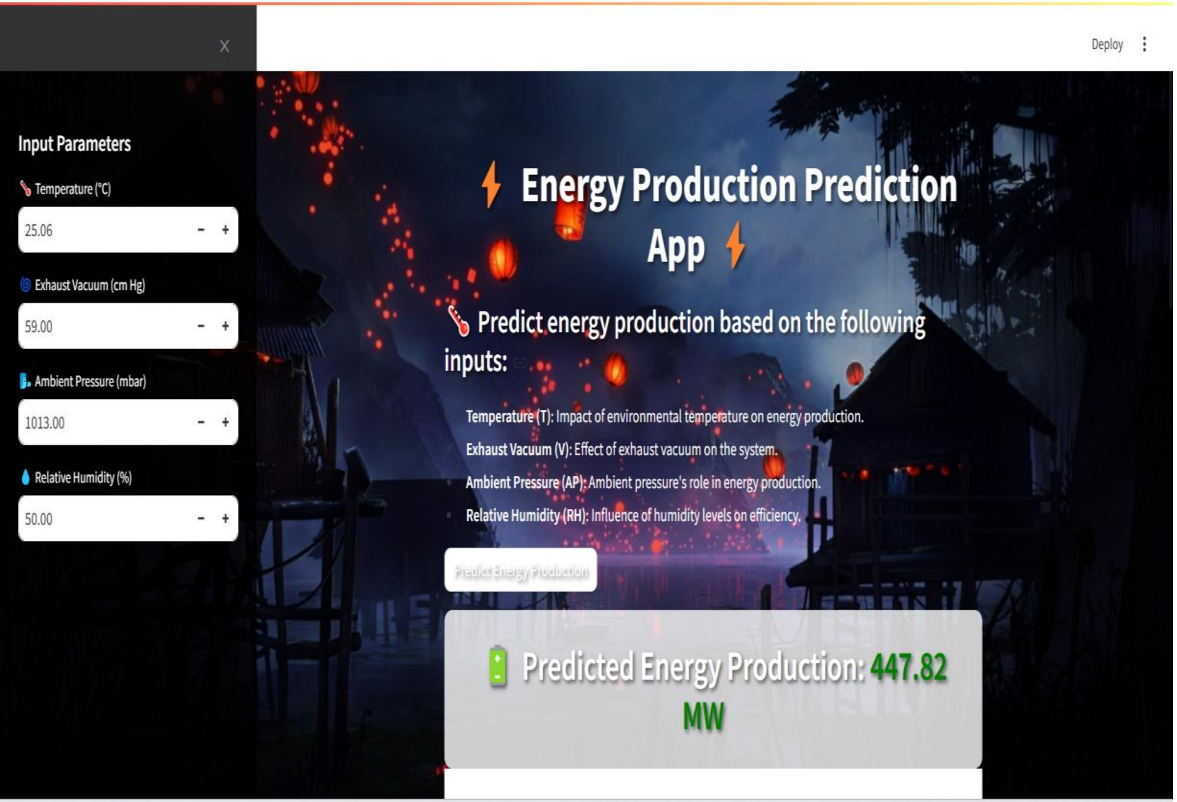
```
Ridge Regression
===================================
Model performance for Training set
- r2_score: 0.9284
- mean_squared_error: 4.5561
- mean_absolute_error: 3.6213
----------------------------------
Model performance for Test set
- r2_score: 0.9283
- mean_squared_error: 4.5693
- mean_absolute_error: 3.6441
===================================
```

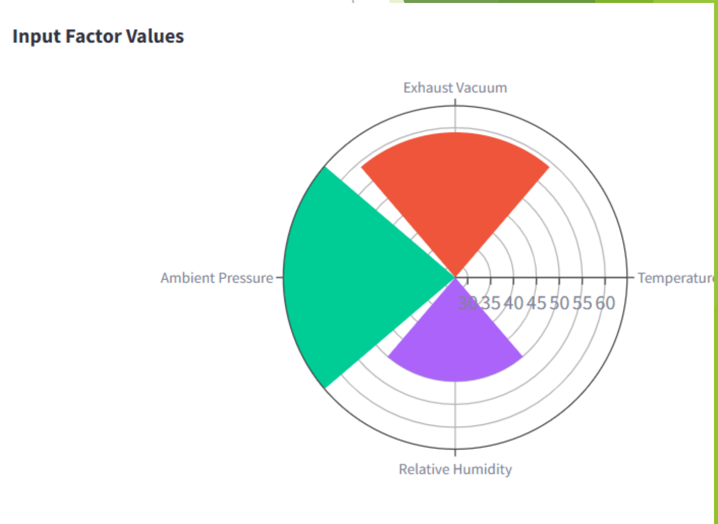
```
K-Nearest Neighbors
===================================
Model performance for Training set
- r2_score: 0.9647
- mean_squared_error: 3.2003
- mean_absolute_error: 2.3378
----------------------------------
Model performance for Test set
- r2_score: 0.9468
- mean_squared_error: 3.9368
- mean_absolute_error: 2.8719
===================================
```

```
Gradient Boosting
===================================
Model performance for Training set
- r2_score: 0.9539
- mean_squared_error: 3.6556
- mean_absolute_error: 2.8113
----------------------------------
Model performance for Test set
- r2_score: 0.9461
- mean_squared_error: 3.9614
- mean_absolute_error: 2.9774
===================================
```

```
AdaBoost
===================================
Model performance for Training set
- r2_score: 0.8977
- mean_squared_error: 5.4464
- mean_absolute_error: 4.4668
----------------------------------
Model performance for Test set
- r2_score: 0.8892
- mean_squared_error: 5.6799
- mean_absolute_error: 4.5934
===================================
```

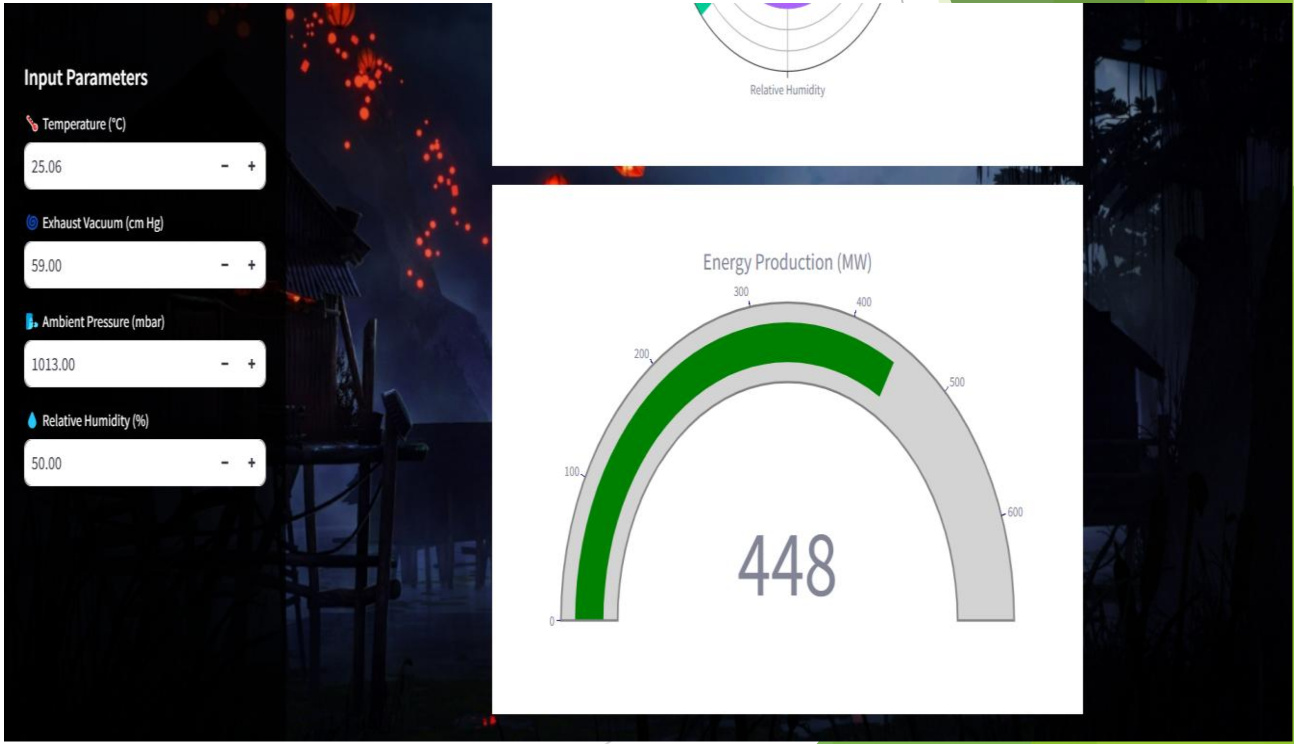
Result


Deploy
Input Parameters
Temperature (°C)
25.06
Exhaust Vacuum (cm Hg)
59.00
Ambient Pressure (mbar)
1013.00
Relative Humidity (%)
50.00
Energy Production Prediction App
Predict energy production based on the following inputs:
Temperature (T): Impact of environmental temperature on energy production.
Exhaust Vacuum (V): Effect of exhaust vacuum on the system.
Ambient Pressure (AP): Ambient pressure's role in energy production.
Relative Humidity (RH): Influence of humidity levels on efficiency.
Predict Energy Production
Predicted Energy Production: 447.82 MW


Input Factor Values
Exhaust Vacuum
Ambient Pressure
Temperature
30 35 40 45 50 55 60
Relative Humidity

Final output

The Energy Production Prediction App built with Streamlit leverages a Random Forest model to accurately forecast energy production based on environmental factors like temperature, exhaust vacuum, ambient pressure, and relative humidity. This user-friendly and interactive tool provides valuable insights for optimizing energy systems, helping industries enhance efficiency and adapt to varying conditions effectively.



Conclusion:

- Based on the analysis, Random Forest, XGBoost, and GradientBoosting appear to be the top-performing models. They consistently achieve high R2-scores and low error metrics on both training and testing sets, suggesting they are well-suited for the given task.
- R^2 Score: Measures how well the model explains the variance in the data. A higher value (closer to 1) means better performance. Mean Absolute Error (MAE): The average of absolute differences between predicted and actual values. Lower is better. Mean Squared Error (MSE) and Root Mean Squared Error (RMSE): Penalizes larger errors more. Lower values indicate better predictions. Performance Comparison:
- Linear Regression Training set: R^2: 0.9284 (Explains ~92.8% of the variance in the training data) MAE: 3.6213 Test set: R^2: 0.9283 (Good generalization, close to training set performance) MAE: 3.6441 Linear Regression performs consistently well on both sets, showing no signs of overfitting.
- Random Forest Training set: R^2: 0.9945 (Very high accuracy on training data) MAE: 0.8852 Test set: R^2: 0.9617 (Still performs well but slightly drops on test set) MAE: 2.3406 Random Forest shows excellent performance but may slightly overfit the training data as the test performance is lower.

- DecisionTreeRegressor Training set: R^2: 1.0000 (Perfect fit, likely overfitting) MAE: 0.0000 (No errors on the training set) Test set: R^2: 0.9327 (Significant drop in test performance) MAE: 2.9695 This model is likely overfitting, as it performs perfectly on the training set but much worse on the test set.
- BaggingRegressor Training set: R^2: 0.9923 (High accuracy on training set) MAE: 0.9987 Test set: R^2: 0.9588 (Good generalization but slightly lower than training) MAE: 2.4380 Bagging performs similarly to Random Forest, with slightly less test accuracy, but good overall results.
- XGBoost Training set: R^2: 0.9878 (Very high accuracy) MAE: 1.3643 Test set: R^2: 0.9647 (Great generalization, slight drop compared to training) MAE: 2.2316 XGBoost performs very well on both sets and generalizes better than Decision Trees.
- AdaBoost Training set: R^2: 0.8977 (Lower accuracy than other models on training) MAE: 4.4668 Test set: R^2: 0.8892 (Similar test performance, but not as strong as others) MAE: 4.5934 AdaBoost performs consistently, but its R^2 and MAE values suggest it may not be the best choice for this dataset.
- GradientBoosting Training set: R^2: 0.9539 (Strong performance on the training set) MAE: 2.8113 Test set: R^2: 0.9461 (Good generalization with only a slight drop) MAE: 2.9774 GradientBoosting performs well, slightly below XGBoost but still solid on both sets.