# Combined Cycle Power Plant Performance Prediction

## Introduction

A Combined Cycle Power Plant (CCPP) integrates both gas and steam turbines to generate electricity efficiently. The waste heat from the gas turbine is used to power a steam turbine, improving overall energy conversion. To optimize performance, it is essential to predict power output based on operational parameters.  
  
Machine Learning (ML) techniques such as Linear Regression, Decision Trees, and Random Forests can analyze plant conditions and accurately predict power output. This project focuses on developing an ML-based system to estimate power generation using key features like Ambient Temperature (AT), Exhaust Vacuum (V), Ambient Pressure (AP), and Relative Humidity (RH).

## Objectives

By the end of this project, you will:

- Understand the impact of operational parameters on CCPP performance.

- Apply Machine Learning techniques to build predictive models.

- Train and evaluate models using performance metrics.

- Develop a Flask web application to provide real-time predictions.

- Optimize and compare different ML models to enhance accuracy.

## Project Workflow

1. Data Collection: Use publicly available datasets on CCPP performance.

1. Data Preprocessing: Handle missing values, normalize data, and split into training and testing sets.

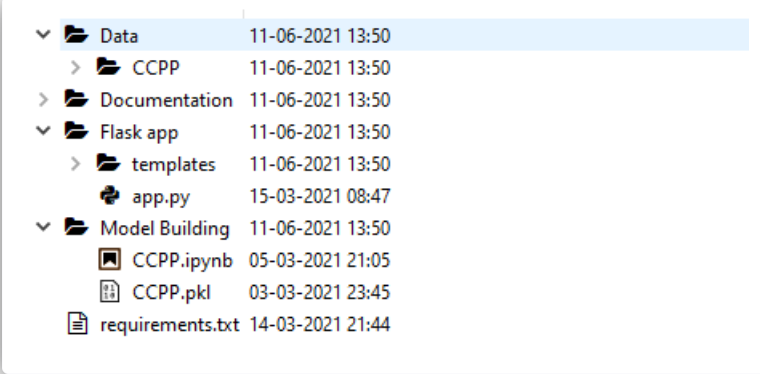
1. Model Building: Train Linear Regression, Decision Tree, and Random Forest models.

1. Model Evaluation: Measure accuracy using metrics like RMSE and R-squared.

1. Application Development: Deploy the model via a Flask web app.

1. Deployment: Integrate a user-friendly web interface for predictions.

## Project Structure



## Technical Architecture

- Frontend: HTML, CSS, JavaScript for user interaction.

- Backend: Flask handles HTTP requests and serves predictions.

- Machine Learning Models: Implemented using scikit-learn.

- Database: CSV dataset for training and testing models.

# Data Collection

* **Dataset Source:** The dataset is collected from publicly available sources and contains operational data from Combined Cycle Power Plants.
* **Dataset Features:**
  + **AT** (Ambient Temperature in °C)
  + **V** (Exhaust Vacuum in cm Hg)
  + **AP** (Ambient Pressure in millibar)
  + **RH** (Relative Humidity in %)
  + **PE** (Power Output in MW - target variable)
* **Dataset Structure:**
  + The dataset consists of multiple instances of power plant operating conditions, with each record representing a specific timestamp.
  + The data is stored in CSV format and split into **training (80%)** and **testing (20%)** sets.

## Data Preprocessing

Handling Missing Values:

Check for missing values and apply imputation techniques if necessary.

Normalization:

Scale features using Min-Max scaling or Standardization to improve model performance.

Feature Engineering:

Analyze feature correlations to remove irrelevant or redundant data.

Generate new features if needed to enhance model prediction accuracy.

Data Splitting:

The dataset is divided into training (80%) and testing (20%) sets to evaluate model performance.

# Model Building

- Linear Regression: Predicts PE using a linear relationship with input features.

- Decision Tree: Splits data into decision nodes based on feature importance.

- Random Forest: An ensemble method using multiple decision trees to reduce overfitting.

## Model Evaluation

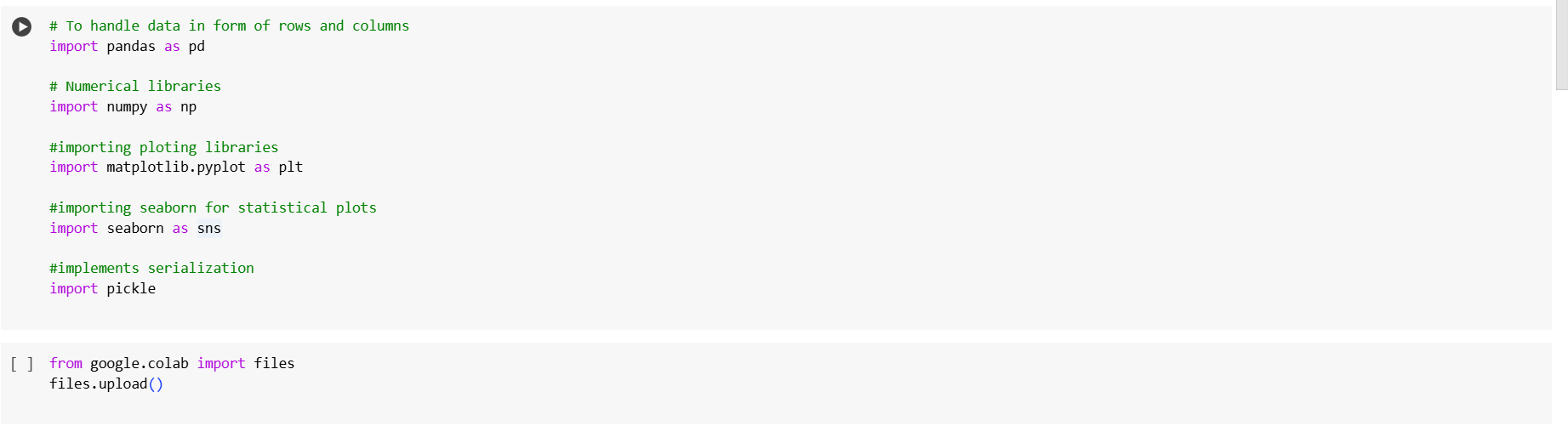
Metrics used for evaluation:  
- RMSE (Root Mean Squared Error): Measures prediction error.  
- R² Score: Indicates model fit.  
- Comparison: Random Forest is expected to perform better due to its ensemble approach.

# Model Training

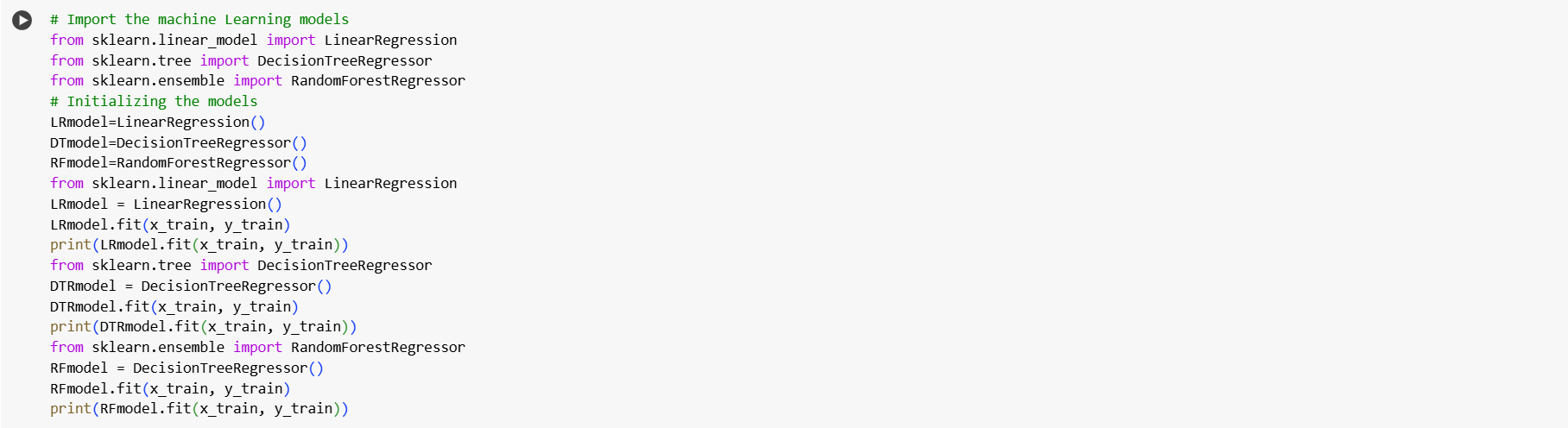
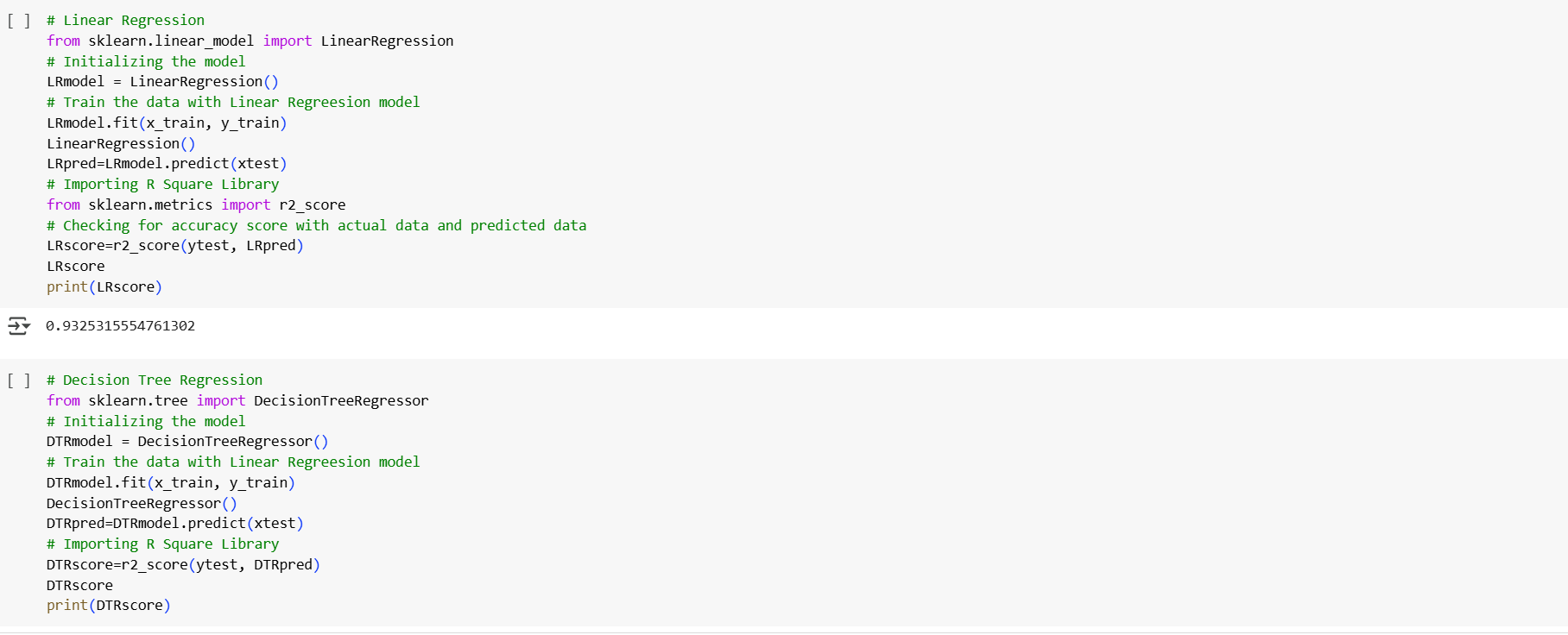
**Training Process;**

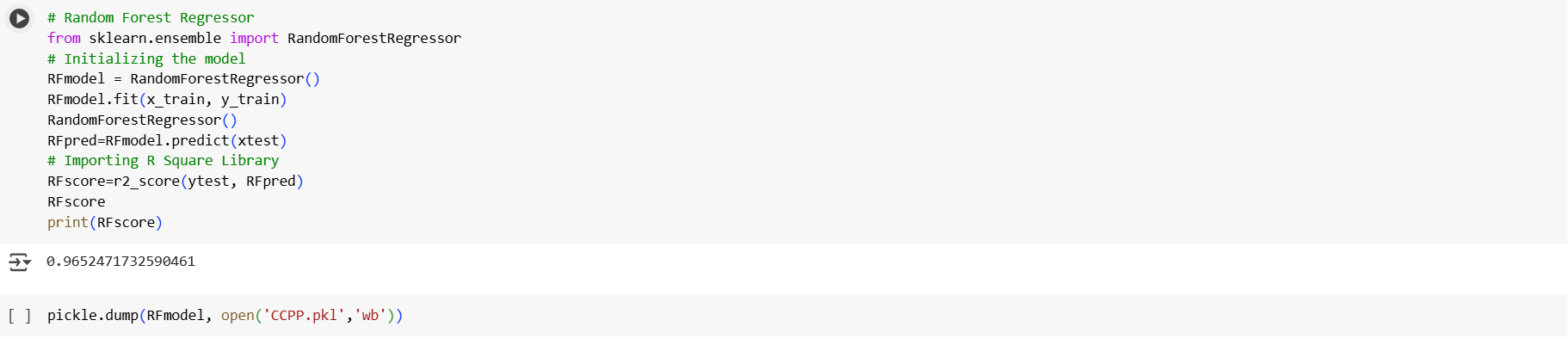
1. **Load the dataset:** Read the XLSX file using pandas.
2. **Preprocess the data:** Handle missing values, normalize features, and split into training and testing sets.
3. **Train different models:** Implement Linear Regression, Decision Tree, and Random Forest models.
4. **Optimize hyperparameters:** Use GridSearchCV or RandomizedSearchCV to fine-tune model parameters.
5. **Save the best model:** Store the trained model as CCPP.pkl using pickle.

**Training Script (train\_model)**









# Flask Web Application

**Backend (app.py)**

* Handles HTTP requests and serves predictions.
* Loads the trained model (CCPP.pkl).
* Accepts user input (AT, V, AP, RH) via an API.
* Returns predicted **power output (PE)** as JSON.



## Frontend (index1.html & home.html)

Users enter operational parameters.

The system predicts power output in real-time.

HOME.HTML



INDEX.HTML



# ****Running the Code****

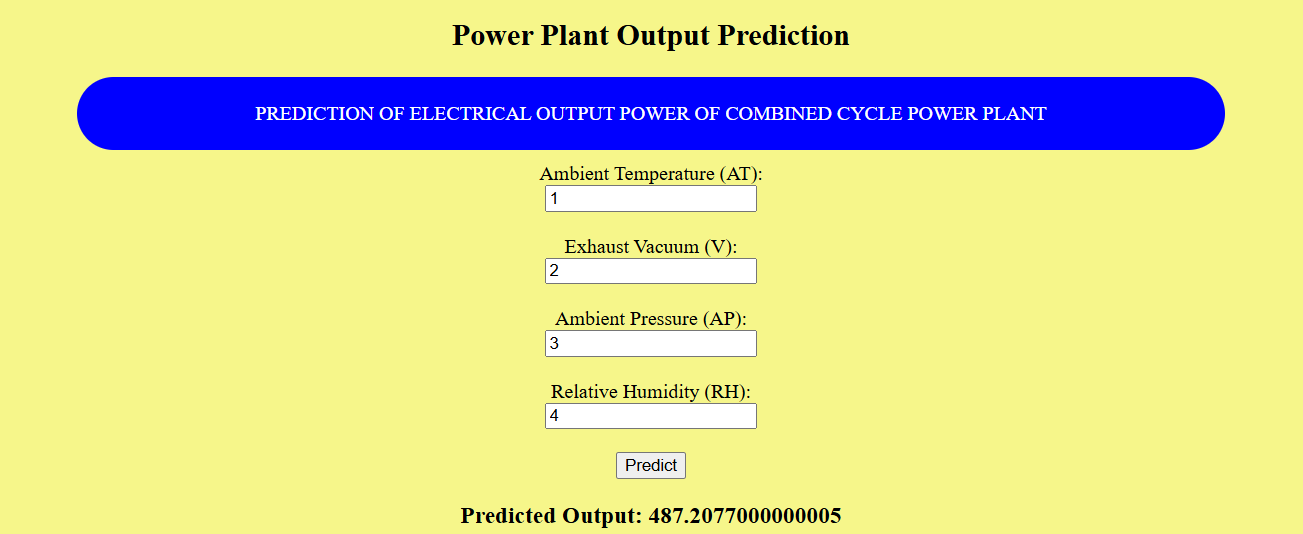
The execution of the project begins with training the deep learning model and then deploying it via a Flask web application. The process starts by running the train\_model.py script, which loads the dataset, preprocesses the images, and trains a convolutional neural network using transfer learning techniques . Once training is completed, the model is saved as CCPP.pkl, which serves as the core of the prediction system.

After the model is trained and saved, app.py is executed to launch the Flask web server. The Flask application initializes by loading CCPP.pkl, setting up routes, and rendering HTML templates. When a user accesses the web application through http://127.0.0.1:5000/, they are directed to the homepage, index.html.



From the homepage (home.html), users can navigate to the prediction page (index1.html), where they can enter power plant operational parameters (AT, V, AP, RH). When the user submits the form, the input values are sent to the Flask backend, which processes the data using the trained model and predicts the power output. The result is then displayed on the same page. Users can enter new values for additional predictions or navigate back to the homepage. This setup ensures seamless interaction between the backend and frontend components, making the system user-friendly and efficient.

## 



## Conclusion

This project demonstrates the potential of Machine Learning in power plant performance optimization. The trained model enables power plants to predict output efficiently, aiding in decision-making and improving energy utilization.  
  
Future Enhancements:  
- Incorporate real-time sensor data.  
- Implement deep learning models for enhanced accuracy.  
- Deploy on a cloud platform for scalability.  
  
By leveraging ML and Flask, this system provides a practical approach to improving Combined Cycle Power Plant efficiency. 🚀