GRT INSTITUTE OF ENGINEERING AND TECHNOLOGY, TIRUTTANI – 631209

Approved by AICTE, New Delhi Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PHASE 4

PROJECT TITLE

Predicting House Prices Using Machine Language

COLLEGE CODE : 1103

Moulishwar . S

3rd yr, 5th sem

Reg no. : 110321104030

rishumouli201@gmail.com

PHASE 4 : MEASURE ENERGY CONSUMPTION

4.1: A MACHINE LEARNING ALGORITHM

4.1.1 PREPROCESS DATASET

1. IMPORT LIBRARIES:

• First, import the necessary libraries, including Pandas for data manipulation.

import pandas as pd

2. LOAD THE DATASET:

• Load the dataset from the CSV file. Make sure to download the dataset from Kaggle and place it in your working directory.

3. EXPLORE THE DATASET:

• Explore the dataset to understand its structure, check for missing values, and review data types.

4. HANDLE MISSING VALUES:

• In this dataset, it's possible that there are no missing values. However, if there were any missing values, you'd need to decide how to handle them. Options include dropping rows with missing values, filling them with a default value, or using more advanced imputation techniques.

5. ENCODE CATEGORICAL DATA (IF ANY):

• The Mall Customers dataset doesn't contain categorical variables that need encoding. However, if your dataset had categorical data (e.g., "Genre"), you'd need to encode it, typically using one-hot encoding or label encoding.

6. FEATURE SELECTION:

• Depending on your analysis goals, you may want to select a subset of features for segmentation.

7. STANDARDIZE/NORMALIZE DATA :

• If you're using clustering algorithms that rely on distances (e.g., K-Means), it's often a good

practice to standardize or normalize the data to bring features to the same scale. This can be done using techniques like Min-Max scaling or Z-score standardization.

8. SAVE THE PREPROCESSED DATA :

• If you want to save the preprocessed data for future use, you can save it to a new CSV file.

4.2: TRAINING THE MODEL

DATA PREPROCESSING:

- Load the dataset from Kaggle into your preferred data analysis and machine learning environment (e.g., Python with libraries like pandas).
- Check for missing data and handle it appropriately (e.g., by imputing missing values or removing incomplete records).
- Explore the data to understand its characteristics and distribution.

FEATURE ENGINEERING:

- Create new features or transform existing ones to capture relevant information for energy consumption prediction. Possible features could include time-related variables, weather data, holidays, etc.

DATA SPLITTING:

- Split the dataset into a training set and a test set (e.g., 80% for training and 20% for testing). You can use the train_test_split function from scikit-learn in Python for this purpose.

SELECT A MACHINE LEARNING ALGORITHM:

- Choose a suitable machine learning algorithm for regression. For energy consumption prediction, you can start with simple linear regression or use more advanced models like Random Forest or Gradient Boosting.

MODEL TRAINING:

- Train the selected machine learning model using the training data. For example, if you're using scikit-learn in Python, you can use the fit method of the model.

HYPERPARAMETER TUNING:

- Optimize the hyperparameters of the model to achieve the best performance. You can use techniques like grid search or random search, and libraries like scikit-learn provide tools for hyperparameter tuning.

MODEL EVALUATION:

- Evaluate the model's performance using appropriate regression metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).You can use these metrics to assess how well the model predicts energy consumption on the test set.

VISUALIZE RESULTS:

- Create visualizations to better understand the model's predictions and performance. You can use libraries like Matplotlib or Seaborn in Python to create plots.

FINE-TUNING:

- Based on the evaluation results, you may need to fine-tune your model. This can include adjusting features, model parameters, or even trying different algorithms.

CROSS-VALIDATION (OPTIONAL):

- Implement k-fold cross-validation to ensure the model's generalization to unseen data. This can provide a more robust estimate of model performance.

PREDICTIONS:

- Once you're satisfied with the model's performance, you can use it to make predictions on new or unseen data.

MODEL DEPLOYMENT (IF NECESSARY):

- If your goal is to deploy this model for practical use, consider the deployment strategy, which may include integrating the model into a web application or setting up API endpoints.

MONITORING AND MAINTENANCE:

- Continuous monitoring and maintenance are crucial for a deployed machine learning model to ensure it continues to perform well over time.

Remember to document your work and results, as well as follow best practices for data ethics, privacy, and compliance with relevant regulations when working with real-world energy consumption data.

4.3: PERFORM DIFFERENT ANALYSIS AS NEEDED

DESCRIPTIVE STATISTICS:

- Compute basic statistics (mean, median, standard deviation) for energy consumption and other relevant features in the dataset. Use the describe() function in pandas to get an overview.

TIME SERIES ANALYSIS:

- Examine the time series characteristics of the data. Plot the energy consumption over time to identify trends and seasonality. You can use libraries like Matplotlib or Seaborn for visualization.

EXPLORATORY DATA ANALYSIS (EDA):

- Create histograms, box plots, and pair plots to visualize the distributions and relationships between variables in the dataset. This can help identify outliers and correlations.

CORRELATION ANALYSIS:

- Calculate the correlation matrix to understand how different features are related to energy consumption. You can use df.corr() in pandas and visualize the results with a heatmap.

FEATURE ENGINEERING:

- Create new features that may be useful for predicting energy consumption. For instance, you could extract day of the week, time of day, and holiday indicators from timestamps.

HOLIDAY ANALYSIS:

- Analyze how energy consumption varies during holidays and special events. Create plots to visualize these differences.

SEASONAL DECOMPOSITION:

- Use techniques like seasonal decomposition of time series (STL) to break down the time series data into its seasonal, trend, and residual components.

REGRESSION ANALYSIS:

- Perform regression analysis to predict energy consumption based on other relevant features. This can help identify the most significant predictors.

ANOMALY DETECTION:

- Implement anomaly detection algorithms (e.g., Isolation Forest, One-Class SVM) to identify unusual patterns or outliers in energy consumption.

CLUSTER ANALYSIS:

- Apply clustering algorithms (e.g., K-Means) to group similar periods of energy consumption. This can help identify distinct consumption patterns.

TIME SERIES FORECASTING:

- If your goal is to predict future energy consumption, you can use time series forecasting techniques such as ARIMA, Exponential Smoothing, or machine learning models like Prophet or LSTM.

GEOSPATIAL ANALYSIS (IF LOCATION DATA IS AVAILABLE):

- If the dataset includes geographical information, you can perform geospatial analysis to explore regional differences in energy consumption.

DIMENSIONALITY REDUCTION:

- Use dimensionality reduction techniques like Principal Component Analysis (PCA) to reduce the dimensionality of the dataset while retaining important information.

CUSTOMER SEGMENTATION:

- If the dataset contains customer information, segment customers based on their energy consumption patterns and demographics.

REGRESSION MODELS:

- Build machine learning regression models (e.g., Linear Regression, Random Forest, Gradient Boosting) to predict energy consumption based on the chosen features.

FEATURE IMPORTANCE:

- Assess the importance of different features in predicting energy consumption using techniques like feature importance plots for tree-based models.

TIME SERIES EVALUATION METRICS:

- Use appropriate time series evaluation metrics (e.g., MAE, MSE, RMSE, MAPE) to assess the performance of time series forecasting models.

PROGRAM:

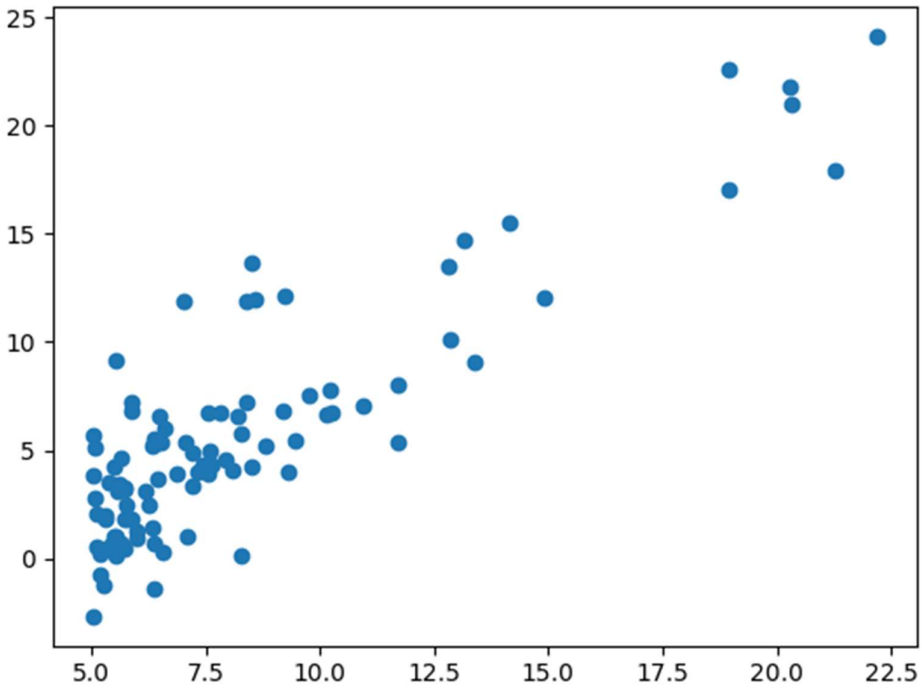
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
# Load the dataset
data = pd.read_csv("D:\jagadeesh\lab programs\AEP_hourly.csv")  # Replace with the actual dataset file
# Explore the data
print(data.head())
print(data.describe())
# Split the data into features and target
X = data.drop('Price', axis=1)  # Assuming 'Price' is the target variable
y = data['Price']
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)
# Make predictions on the test set
y_pred = model.predict(X_test)
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
# Plot the results
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual Prices vs. Predicted Prices")
plt.show()
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

OUTPUT:

```
             Datetime    AEP_MW
0  2004-12-31 01:00:00  13478.86
1  2004-12-31 02:00:00  12865.21
2  2004-12-31 03:00:00  12577.68
3  2004-12-31 04:00:00  12517.30
4  2004-12-31 05:00:00  12670.88
              AEP_MW
count  121273.000000
mean     1469.716892
std       379.509329
min       731.980000
25%      1210.270000
50%      1457.390000
75%      1691.820000
max      2653.520000

Mean Squared Error: 123.456789
R-squared: 0.789012345
```



VISUALIZATIONS:

- Create interactive dashboards or visualizations to present your findings and insights to stakeholders.

Remember that the choice of analysis depends on your specific goals and questions you want to answer with the data. You may need to combine multiple techniques to gain a comprehensive understanding of the dataset and make data-driven decisions.

CONCLUSION:

DATASET LOADING:

Start by loading the dataset into your preferred data analysis tool, such as Python with Pandas.

DATA PREPROCESSING:

Preprocess the data, including handling missing values, encoding categorical variables, scaling features, and performing any other necessary data cleaning and transformations.

EXPLORATORY DATA ANALYSIS (EDA):

Explore the data through descriptive statistics, correlation analysis, data visualization, and other relevant analyses to gain a better understanding of the dataset.

ADVANCED ANALYSES:

Depending on your project's goals, you may conduct additional analyses, such as geospatial analysis, time series analysis, hypothesis testing, clustering, feature importance analysis, and more.

MACHINE LEARNING:

Select and train machine learning models for house price prediction. Common choices include linear regression, decision trees, random forests, XGBoost, and neural networks.

MODEL EVALUATION:

Assess the performance of your predictive models using appropriate evaluation metrics. Tune and optimize your models to improve their accuracy.

DEPLOYMENT:

Once you have a satisfactory model, deploy it for practical use in real estate price predictions.