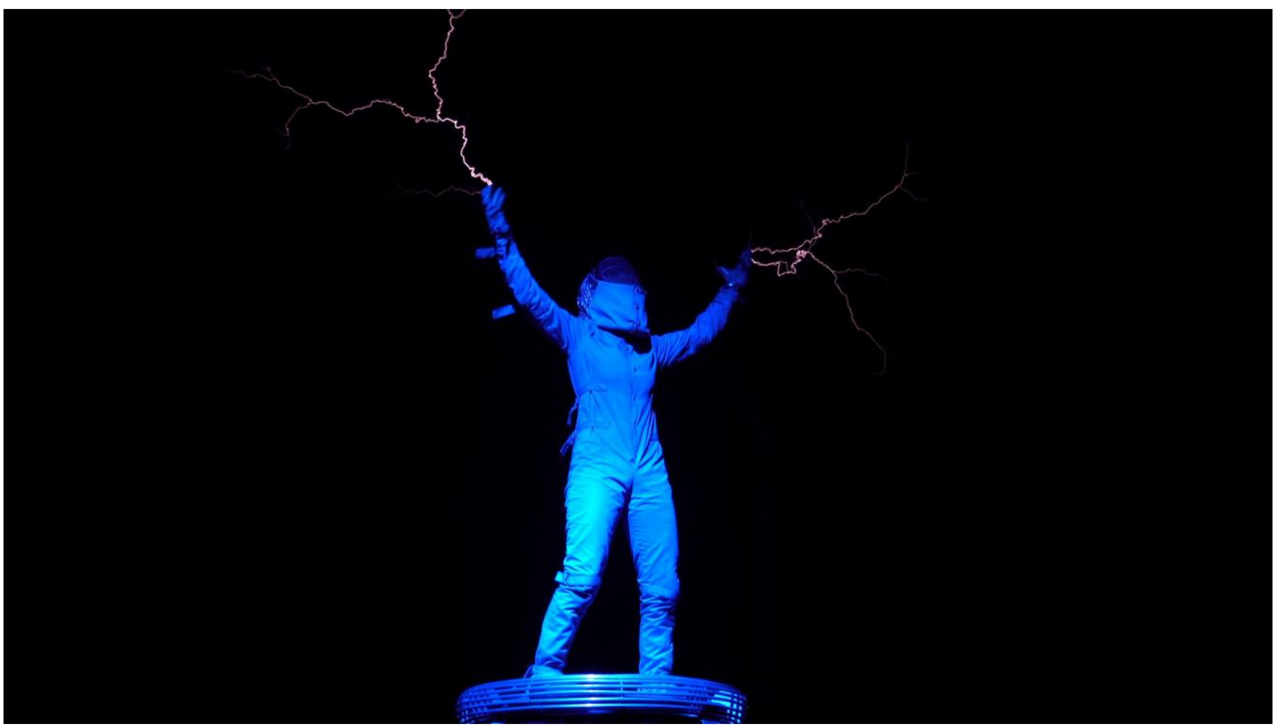
ELECTRICITY PRICES PREDICTION USING MACHINE LEARNING

Phase 5 submission document

Project Title: **Electricity Prices Prediction**

Phase 5: Project Documentation & Submission

Topic: *In this section we will document the complete project and prepare it for submission.*



ELECTRICITY PRICES PREDICTION

Introduction:

- Electricity price prediction is a critical and challenging task in the energy industry, with significant implications for various stakeholders, including energy providers, consumers, and investors.

- This introduction briefing provides an overview of electricity price prediction, its importance, and the factors involved.

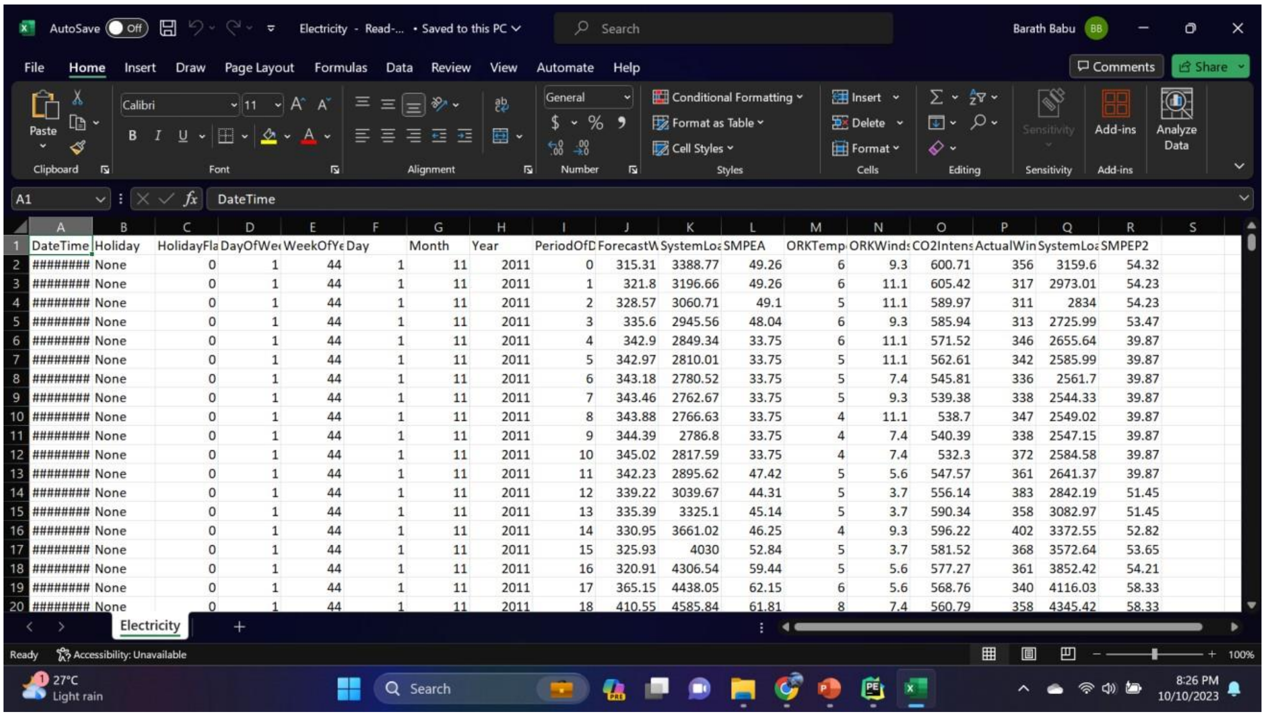
- Electricity prices are dynamic and subject to a multitude of influences. Accurate prediction of electricity prices is essential for making informed decisions in the energy sector. It enables stakeholders to optimize their energy consumption, manage costs, and invest strategically.

- Electricity price prediction is a critical tool for energy stakeholders, enabling them to make informed decisions in a dynamic and rapidly evolving market.

- By understanding the key factors influencing prices and employing advanced prediction methods, stakeholders can gain a competitive edge and manage their energy resources effectively.

- Predicting electricity bills using machine learning can help consumers and utility companies optimize energy usage, reduce costs, and plan for the future more effectively. It's a practical application that can contribute to energy conservation and financial savings.

Given data set:



	DateTime	Holiday	HolidayFla	DayOfWe	WeekOfYe	Day	Month	Year	PeriodOfD	ForecastW	SystemLo	SMPEA	ORKTemp	ORKWinds	CO2Intens	ActualWin	SystemLo	SMPEP2
2	########	None	0	1	44	1	11	2011	0	315.31	3388.77	49.26	6	9.3	600.71	356	3159.6	54.32
3	########	None	0	1	44	1	11	2011	1	321.8	3196.66	49.26	6	11.1	605.42	317	2973.01	54.23
4	########	None	0	1	44	1	11	2011	2	328.57	3060.71	49.1	5	11.1	589.97	311	2834	54.23
5	########	None	0	1	44	1	11	2011	3	335.6	2945.56	48.04	6	9.3	585.94	313	2725.99	53.47
6	########	None	0	1	44	1	11	2011	4	342.9	2849.34	33.75	6	11.1	571.52	346	2655.64	39.87
7	########	None	0	1	44	1	11	2011	5	342.97	2810.01	33.75	5	11.1	562.61	342	2585.99	39.87
8	########	None	0	1	44	1	11	2011	6	343.18	2780.52	33.75	5	7.4	545.81	336	2561.7	39.87
9	########	None	0	1	44	1	11	2011	7	343.46	2762.67	33.75	5	9.3	539.38	338	2544.33	39.87
10	########	None	0	1	44	1	11	2011	8	343.88	2766.63	33.75	4	11.1	538.7	347	2549.02	39.87
11	########	None	0	1	44	1	11	2011	9	344.39	2786.8	33.75	4	7.4	540.39	338	2547.15	39.87
12	########	None	0	1	44	1	11	2011	10	345.02	2817.59	33.75	4	7.4	532.3	372	2584.58	39.87
13	########	None	0	1	44	1	11	2011	11	342.23	2895.62	47.42	5	5.6	547.57	361	2641.37	39.87
14	########	None	0	1	44	1	11	2011	12	339.22	3039.67	44.31	5	3.7	556.14	383	2842.19	51.45
15	########	None	0	1	44	1	11	2011	13	335.39	3325.1	45.14	5	3.7	590.34	358	3082.97	51.45
16	########	None	0	1	44	1	11	2011	14	330.95	3661.02	46.25	4	9.3	596.22	402	3372.55	52.82
17	########	None	0	1	44	1	11	2011	15	325.93	4030	52.84	5	3.7	581.52	368	3572.64	53.65
18	########	None	0	1	44	1	11	2011	16	320.91	4306.54	59.44	5	5.6	577.27	361	3852.42	54.21
19	########	None	0	1	44	1	11	2011	17	365.15	4438.05	62.15	6	5.6	568.76	340	4116.03	58.33
20	########	None	0	1	44	1	11	2011	18	410.55	4585.84	61.81	8	7.4	560.79	358	4345.42	58.33

Here's a list of tools and software commonly used in the process:

1. Programming Language:

- Python is the most popular language for machine learning due to its extensive libraries and frameworks. You can use libraries like NumPy,
pandas, scikit-learn, and more.

2. Integrated Development Environment (IDE):

- Choose an IDE for coding and running machine learning experiments. Some popular options include Jupyter Notebook, Google Colab, or traditional IDEs like PyCharm.

3. Machine Learning Libraries:

- You'll need various machine learning libraries, including:
- scikit-learn for building and evaluating machine learning models.
- TensorFlow or PyTorch for deep learning, if needed.
- XGBoost, LightGBM, or CatBoost for gradient boosting models.

4. Data Visualization Tools:

- Tools like Matplotlib, Seaborn, or Plotly are essential for data exploration and visualization.

5. Data Preprocessing Tools:

- Libraries like pandas help with data cleaning, manipulation, and preprocessing.

6. Data Collection and Storage:

- Depending on your data source, you might need web scraping tools (e.g., BeautifulSoup or Scrapy) or databases (e.g., SQLite, PostgreSQL) for data storage.

7. Version Control:

- Version control systems like Git are valuable for tracking changes in your code and collaborating with others.

8. Notebooks and Documentation:

- Tools for documenting your work, such as Jupyter Notebooks or Markdown for creating README files and documentation.

9. Hyperparameter Tuning:

- Tools like GridSearchCV or RandomizedSearchCV from scikit-learn can help with hyperparameter tuning.

10. Web Development Tools (for Deployment):

- If you plan to create a web application for model deployment, knowledge of web development tools like Flask or Django for backend development, and HTML, CSS, and JavaScript for the front-end can be useful.

11. Cloud Services (for Scalability):

- For large-scale applications, cloud platforms like AWS, Google Cloud, or Azure can provide scalable computing and storage resources.

12. External Data Sources (if applicable):

- Depending on your project's scope, you might require tools to access external data sources, such as APIs or data scraping tools.

13. Data Annotation and Labeling Tools (if applicable):

- For specialized projects, tools for data annotation and labeling may be necessary, such as Labelbox or Supervisely.

14. Geospatial Tools (for location-based features):

- If your dataset includes geospatial data, geospatial libraries like GeoPandas can be helpful.

1.DESIGN THINKING AND PRESENT IN FORM OF DOCUMENT

1.Empathize:

- Understand the needs and challenges of all stakeholders involved in the electricity price prediction process, including homebuyers, sellers, real estate professionals, appraisers, and investors.

- Conduct interviews and surveys to gather insights on what users value in property valuation and what information is most critical for their decision-making.

2.Define:

- Clearly articulate the problem statement, such as "How might we predict electricity prices more accurately and transparently using machine learning?"

- Identify the key goals and success criteria for the project, such as increasing prediction accuracy, reducing bias, or improving user trust in the valuation process.

3.Ideate:

- Brainstorm creative solutions and data sources that can enhance the accuracy and transparency of electricity price predictions.

- Encourage interdisciplinary collaboration to generate a wide range of ideas, including the use of alternative data, new algorithms, or improved visualization techniques.

4.Prototype:

- Create prototype machine learning models based on the ideas generated during the ideation phase.

- Test and iterate on these prototypes to determine which approaches are most promising in terms of accuracy and usability.

5.Test:

- Gather feedback from users and stakeholders by testing the machine learning models with real-world data and scenarios.

- Assess how well the models meet the defined goals and success criteria and make adjustments based on user feedback.

6.Implement:

- Develop a production-ready machine learning solution for predicting Electricity prices, integrating the best-performing algorithms and data sources.

- Implement transparency measures, such as model interpretability tools, to ensure users understand how predictions are generated.

7.Evaluate:

- Continuously monitor the performance of the machine learning model after implementation to ensure it remains accurate and relevant in a changing real estate market.
- Gather feedback and insights from users to identify areas for improvement.

8.Iterate:

- Apply an iterative approach to refine the machine learning model based on ongoing feedback and changing user needs.

- Continuously seek ways to enhance prediction accuracy, transparency, and user satisfaction.

9.Scale and Deploy:

- Once the machine learning model has been optimized and validated, deploy it at scale to serve a broader audience, such as real estate professionals, investors, and homeowners.

- Ensure the model is accessible through user-friendly interfaces and integrates seamlessly into real estate workflows.

10.Educate and Train:

- Provide training and educational resources to help users understand how the machine learning model works, what factors it considers, and its limitations.

- Foster a culture of data literacy among stakeholders to enhance trust in the technology.

2.DESIGN INTO INNOVATION

1. Data Collection:

Gather a comprehensive dataset that includes features such as location, size, age, amenities, nearby schools, crime rates, and other relevant variables.

2.Data Preprocessing:

Clean the data by handling missing values, outliers, and encoding categorical variables. Standardize or normalize numerical features as necessary.

PYHON PROGRAM:

```
# Import necessary libraries
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler

# Load the dataset (replace 'house_data.csv' with your dataset file)
data = pd.read_csv('E:/USA_Housing.csv')
# Display the first few rows of the dataset to get an overview
print("Dataset Preview:")
```

print(data.head())

Data Pre-processing

# Handle Missing Values
# Let's fill missing values in numeric columns with the mean and in categorical columns with the most frequent value.
numeric_cols = data.select_dtypes(include='number').columns
categorical_cols = data.select_dtypes(exclude='number').columns

imputer_numeric = SimpleImputer(strategy='mean')
imputer_categorical = SimpleImputer(strategy='most_frequent')

data[numeric_cols] =
imputer_numeric.fit_transform(data[numeric_cols])
data[categorical_cols] =
imputer_categorical.fit_transform(data[categorical_cols])

Convert Categorical Features to Numerical

We'll use Label Encoding for simplicity here. You can also use one-hot encoding for nominal categorical features.

label_encoder = LabelEncoder()
for col in categorical_cols:
data[col] = label_encoder.fit_transform(data[col])

# Split Data into Features (X) and Target (y)
X = data.drop(columns=['Price']) # Features
y = data['Price'] # Target

```
# Normalize the Data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split data into training and testing sets (adjust test_size as needed)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)

# Display the preprocessed data
print("\nPreprocessed Data:")
print(X_train[:5]) # Display first 5 rows of preprocessed features
print(y_train[:5]) # Display first 5 rows of target values
```

OUTPUT:

```
Dataset Preview:
Avg. Area Income Avg. Area House Age Avg. Area Number of Roo
ms \
0 79545.458574 5.682861 7.009188
1 79248.642455 6.002900 6.730821
2 61287.067179 5.865890 8.512727
3 63345.240046 7.188236 5.586729
4 59982.197226 5.040555 7.839388

Avg. Area Number of Bedrooms Area Population Price \
0 4.09 23086.800503 1.059034e+06
1 3.09 40173.072174 1.505891e+06
2 5.13 36882.159400 1.058988e+06
3 3.26 34310.242831 1.260617e+06
4 4.23 26354.109472 6.309435e+05
```

Address

0 208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1 188 Johnson Views Suite 079\nLake Kathleen, CA...
2 9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3 USS Barnett\nFPO AP 44820
4 USNS Raymond\nFPO AE 09386

Preprocessed Data:
[[-0.19105816 -0.13226994 -0.13969293 0.12047677 -0.83757985 -1.0
0562872]
[-1.39450169 0.42786736 0.79541275 -0.55212509 1.15729018 1.61
946754]
[-0.35137865 0.46394489 1.70199509 0.03133676 -0.32671213 1.63
886651]
[-0.13944143 0.1104872 0.22289331 -0.75471601 -0.90401197 -1.54
810704]
[0.62516685 2.20969666 0.42984356 -0.45488144 0.12566216 0.98
830821]]
4227 1.094880e+06
4676 1.300389e+06
800 1.382172e+06
3671 1.027428e+06
4193 1.562887e+06
Name: Price, dtype: float64

3.Feature Engineering:

Create new features or transform existing ones to extract more valuable information. For example, you can calculate the distance to the nearest public transportation, or create a feature for the overall condition of the house.

4.Model Selection:

Choose the appropriate machine learning model for the task. Common models for regression problems like electricity price prediction include Linear Regression, Decision Trees, Random Forest, Gradient Boosting, and Neural Networks.

5. Training:

Split the dataset into training and testing sets to evaluate the model's performance. Consider techniques like cross-validation to prevent overfitting.

6. Hyperparameter Tuning:

Optimize the model's hyperparameters to improve its predictive accuracy. Techniques like grid search or random search can help with this.

7.Evaluation Metrics:

Select appropriate evaluation metrics for regression tasks, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE). Choose the metric that aligns with the specific objectives of your project.

8.Regularization:

Apply regularization techniques like L1 (Lasso) or L2 (Ridge) regularization to prevent overfitting.

9.Feature Selection:

Use techniques like feature importance scores or recursive feature elimination to identify the most relevant features for the prediction.

10. Interpretability:

Ensure that the model's predictions are interpretable and explainable. This is especially important for real estate applications where stakeholders want to understand the factors affecting predictions.

11. Deployment:

Develop a user-friendly interface or API for end-users to input property details and receive price predictions.

12. Continuous Improvement:

Implement a feedback loop for continuous model improvement based on user feedback and new data.

13. Ethical Considerations:

Be mindful of potential biases in the data and model. Ensure fairness and transparency in your predictions.

14. Monitoring and Maintenance:

Regularly monitor the model's performance in the real world and update it as needed.

15. Innovation:
Consider innovative approaches such as using satellite imagery or IoT data for real-time property condition monitoring, or integrating natural language processing for textual property descriptions.

3.BUILD LOADING AND PREPROCESSING THE DATASET

1. Data Collection:
Obtain a dataset that contains information about houses and their corresponding prices. This dataset can be obtained from sources like real estate websites, government records, or other reliable data providers.

2. Load the Dataset:
- Import relevant libraries, such as pandas for data manipulation and numpy for numerical operations.
- Load the dataset into a pandas DataFrame for easy data handling. You can use pd.read_csv() for CSV files or other appropriate functions for different file formats.

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

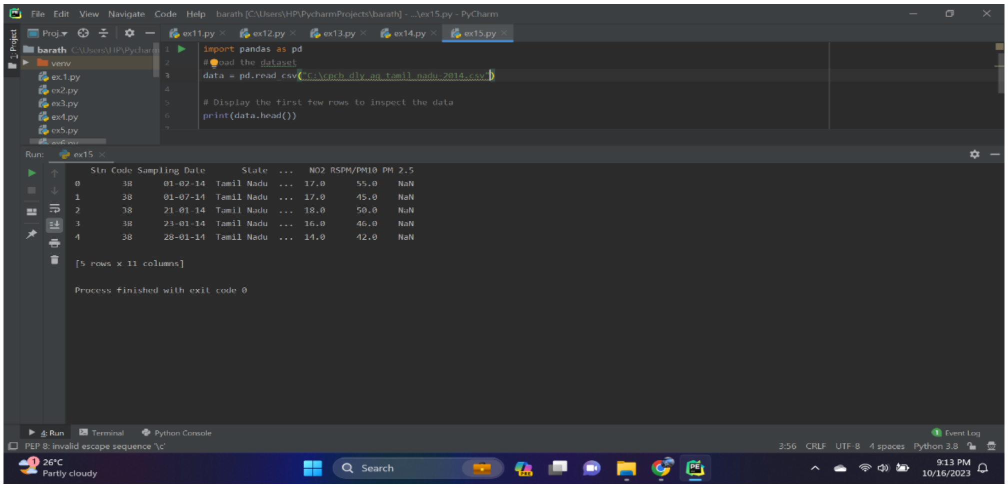
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

from sklearn.metrics import r2_score,
mean_absolute_error,mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
import xgboost as xg
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146:
UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for
this version of SciPy (detected version 1.23.5
warnings.warn(f"A NumPy version >={np_minversion} and
<{np_maxversion}"

Loading Dataset:

dataset = pd.read_csv('E:/USA_Housing.csv')

Output:



3. Data Exploration:

Explore the dataset to understand its structure and contents. Check for the presence of missing values, outliers, and data types of each feature.

4. Data Cleaning:

Handle missing values by either removing rows with missing data or imputing values based on the nature of the data.

5. Feature Selection:

Identify relevant features for electricity price prediction. Features like the number of bedrooms, square footage, location, and amenities are often important.

We are selecting numerical features which have more than 0.50 or less than -0.50 correlation rate based on Pearson Correlation Method—which is the default value of parameter "method" in corr() function. As for selecting categorical features, I selected the categorical values which I believe have significant effect on the target variable such as Heating and MSZoning.

In [1]:

```
important_num_cols =
list(df.corr()["SalePrice"][(df.corr()["SalePrice"]>0.5
0) | (df.corr()["SalePrice"]<-0.50)].index)
cat_cols = ["MSZoning", "Utilities","BldgType","Heating","KitchenQual","
SaleCondition","LandSlope"]
important_cols = important_num_cols + cat_cols
df = df[important_cols]
```

Checking for the missing values

In [2]:

```
print("Missing Values by Column")
print("-"*30)
print(df.isna().sum())
print("-"*30)
print("TOTAL MISSING VALUES:",df.isna().sum().sum())
```

Missing Values by Column
------------------------------ OverallQual 0
YearBuilt 0
YearRemodAdd 0
TotalBsmtSF 0
1stFlrSF 0
GrLivArea 0
FullBath 0
TotRmsAbvGrd 0
GarageCars 0
GarageArea 0
SalePrice 0
MSZoning 0

Utilities 0
BldgType 0
Heating 0
KitchenQual 0
SaleCondition 0
LandSlope 0
dtype: int64

TOTAL MISSING VALUES: 0

6. Feature Engineering:

Create new features or transform existing ones to capture additional information that may impact electricity prices. For example, you
can calculate the price per square foot.

7. Data Encoding:

Convert categorical variables (e.g., location) into numerical format using techniques like one-hot encoding.

8. Train-Test Split:

Split the dataset into training and testing sets to evaluate the machine learning model's performance.

Program:

```
X = df.drop('price', axis=1) # Features
y = df['price'] # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

4.PERFORMING DIFFERENT ACTIVITIES LIKE FEATURE ENGINEERING, MODEL TRAINING, EVALUATION etc.,

1. Feature Engineering:

- As mentioned earlier, feature engineering is crucial. It involves creating new features or transforming existing ones to provide meaningful information for your model.
- Extracting information from textual descriptions (e.g., presence of keywords like "pool" or "granite countertops").
- Calculating distances to key locations (e.g., schools, parks) if you have location data.

2. Data Preprocessing & Visualisation:

Continue data preprocessing by handling any remaining missing values or outliers based on insights from your data exploration.

Visualisation and Pre-Processing of Data:

```
In [1]:
sns.histplot(dataset, x='Price', bins=50, color='y')
Out[1]:
<Axes: xlabel='Price', ylabel='Count'>
In [2]:
sns.boxplot(dataset, x='Price', palette='Blues')
Out[2]:
<Axes: xlabel='Price'>
In [3]:
sns.jointplot(dataset, x='Avg. Area House Age', y='Price', kind='hex')
```

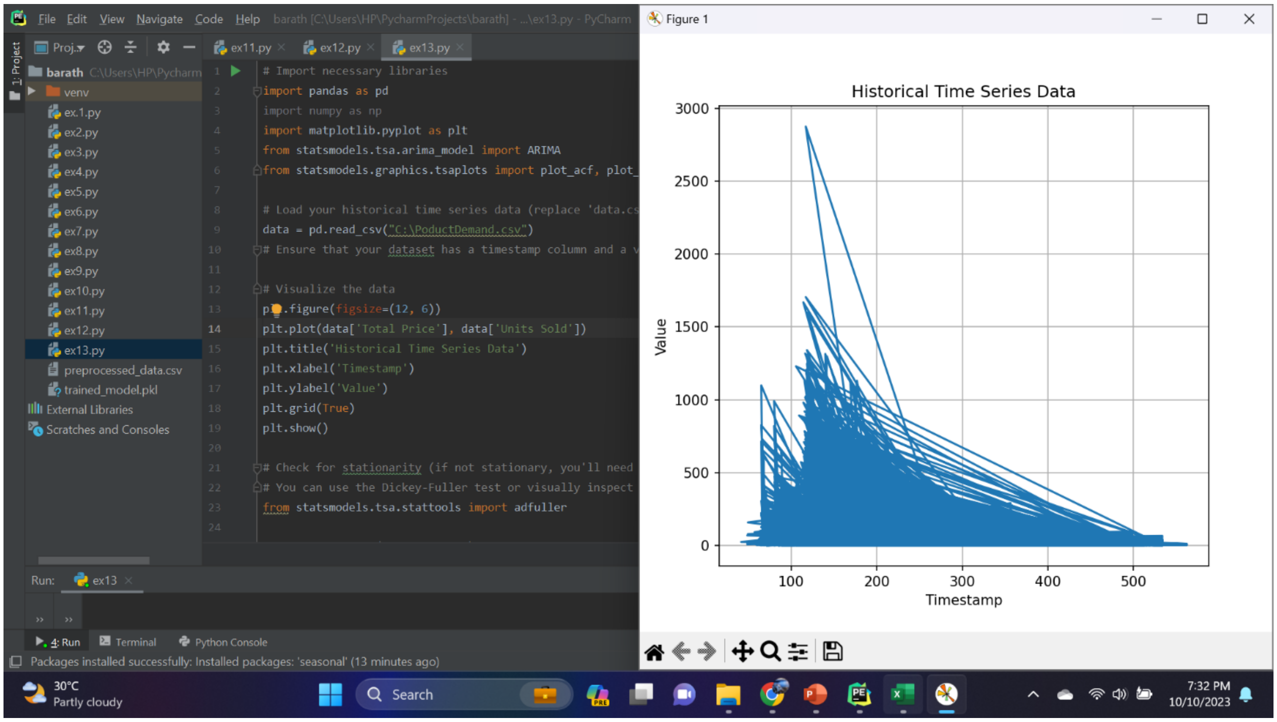
```
Out[3]:
<seaborn.axisgrid.JointGrid at 0x7caf1d571810>
In [4]:
sns.jointplot(dataset, x='Avg. Area Income', y='Price')
Out[4]:
<seaborn.axisgrid.JointGrid at 0x7caf1d8bf7f0>
```

```
In [5]:
plt.figure(figsize=(12,8))sns.pairplot(dataset)
Out[5]:
<seaborn.axisgrid.PairGrid at 0x7caf0c2ac550>
<Figure size 1200x800 with 0 Axes>
In [6]:
dataset.hist(figsize=(10,8))
Out[6]:
array([[<Axes: title={'center': 'Avg. Area Income'}>,
<Axes: title={'center': 'Avg. Area House Age'}>],
[<Axes: title={'center': 'Avg. Area Number of Rooms'}>,
<Axes: title={'center': 'Avg. Area Number of Bedrooms'}>],
[<Axes: title={'center': 'Area Population'}>,
<Axes: title={'center': 'Price'}>]], dtype=object)
```

Visualising Correlation:

```
In [7]:
dataset.corr(numeric_only=True)
In [8]:
plt.figure(figsize=(10,5))sns.heatmap(dataset.corr(numeric_only = Tru
e), annot=True)
```

Out[8]:
<Axes: >



3. Model Selection:

Choose an appropriate machine learning model for your regression task. Common choices include:

- ✓ Linear Regression
- ✓ Decision Trees
- ✓ Random Forest
- ✓ Gradient Boosting (e.g., XGBoost or LightGBM)
- ✓ Neural Networks (Deep Learning)

Program:

Importing Dependencies

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score,
mean_absolute_error,mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
import xgboost as xg
%matplotlib inline

import warnings
warnings.filterwarnings("ignore")
/opt/conda/lib/python3.10/site-packages/scipy/_init_.py:146:
UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required
for this version of SciPy (detected version 1.23.5
warnings.warn(f"A NumPy version >={np_minversion} and
<{np_maxversion}"
```

Loading Dataset

```
dataset = pd.read_csv('E:/USA_Housing.csv')
```

Model 1 - Linear Regression

In [1]:

```
model_lr=LinearRegression()
```

In [2]:
```
model_lr.fit(X_train_scal, Y_train)
```
Out[2]:

Predicting Prices

In [3]:
```
Prediction1 = model_lr.predict(X_test_scal)
```

Evaluation of Predicted Data

In [4]:

```
plt.figure(figsize=(12,6))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction1, label='Predicted Tr
end')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```
Out[4]:
```
Text(0.5, 1.0, 'Actual vs Predicted')
```

In [5]:
```
sns.histplot((Y_test-Prediction1), bins=50)
```
Out[5]:
```
<Axes: xlabel='Price', ylabel='Count'>
```
In [6]:
```
print(r2_score(Y_test, Prediction1))
```

```
print(mean_absolute_error(Y_test, Prediction1))
print(mean_squared_error(Y_test, Prediction1))
```

Out[6]:
0.9182928179392918
82295.49779231755
10469084772.975954

Model 2 - Support Vector Regressor

In [7]:
```
model_svr = SVR()
```
In [8]:
```
model_svr.fit(X_train_scal, Y_train)
```
Out[8]:

Predicting Prices

In [9]:
```
Prediction2 = model_svr.predict(X_test_scal)
```

Evaluation of Predicted Data

In [10]:

```
plt.figure(figsize=(12,6))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction2, label='Predicted Tr
end')
plt.xlabel('Data')
```

```
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```

Out[10]:
Text(0.5, 1.0, 'Actual vs Predicted')

In [11]:
```
sns.histplot((Y_test-Prediction2), bins=50)
```
Out[12]:
<Axes: xlabel='Price', ylabel='Count'>
In [12]:
```
print(r2_score(Y_test, Prediction2))
print(mean_absolute_error(Y_test, Prediction2))
print(mean_squared_error(Y_test, Prediction2))
```
-0.0006222175925689744
286137.81086908665
128209033251.4034

Model 3 - Lasso Regression

In [13]:
```
model_lar = Lasso(alpha=1)
```
In [14]:
```
model_lar.fit(X_train_scal,Y_train)
```
Out[14]:

Predicting Prices

In [15]:

```
Prediction3 = model_lar.predict(X_test_scal)
```

Evaluation of Predicted Data

In [16]:

```
plt.figure(figsize=(12,6))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction3, label='Predicted Tr
end')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```

Out[16]:
Text(0.5, 1.0, 'Actual vs Predicted')

In [17]:

```
sns.histplot((Y_test-Prediction3), bins=50)
```

Out[17]:
<Axes: xlabel='Price', ylabel='Count'>

In [18]:

```
print(r2_score(Y_test, Prediction2))
print(mean_absolute_error(Y_test, Prediction2))
print(mean_squared_error(Y_test, Prediction2))
```

-0.0006222175925689744
286137.81086908665

128209033251.4034

Model 4 - Random Forest Regressor

In [19]:
model_rf = RandomForestRegressor(n_estimators=50)
In [20]:
model_rf.fit(X_train_scal, Y_train)
Out[20]:

Predicting Prices

In [21]:
Prediction4 = model_rf.predict(X_test_scal)

Model 5 - XGboost Regressor

In [25]:
model_xg = xg.XGBRegressor()
In [26]:
model_xg.fit(X_train_scal, Y_train)
Out[26]:
XGBRegressor
XGBRegressor(base_score=None, booster=None,
callbacks=None,
colsample_bylevel=None, colsample_bynode=None,
colsample_bytree=None, early_stopping_rounds=None,
enable_categorical=False, eval_metric=None,

feature_types=None,

gamma=None, gpu_id=None, grow_policy=None,

importance_type=None,
interaction_constraints=None, learning_rate=None,
max_bin=None,
max_cat_threshold=None, max_cat_to_onehot=None,
max_delta_step=None, max_depth=None,
max_leaves=None,
min_child_weight=None, missing=nan,
monotone_constraints=None,
n_estimators=100, n_jobs=None,
num_parallel_tree=None,
predictor=None, random_state=None, ...)

4. Model Training:

Split your dataset into training and testing sets (as shown earlier) and train the selected model on the training data. Here's an example using Linear Regression:

5. Model Evaluation:

Evaluate your model's performance using appropriate regression metrics, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). For example:

PYTHON PROGRAM:

```
# Import necessary libraries
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.linear_model import LinearRegression
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
selector = SelectKBest(score_func=f_regression, k=k)
X_train_selected = selector.fit_transform(X_train, y_train)
# Model Selection
# Let's choose both Linear Regression and Random Forest Regressor for
comparison.
linear_reg_model = LinearRegression()
random_forest_model = RandomForestRegressor(n_estimators=100,
random_state=42)
# Train the models on the selected features
linear_reg_model.fit(X_train_selected, y_train)
random_forest_model.fit(X_train_selected, y_train)
# Evaluate the models on the test set
X_test_selected = selector.transform(X_test)
# Make predictions
linear_reg_predictions = linear_reg_model.predict(X_test_selected)
random_forest_predictions =
random_forest_model.predict(X_test_selected)
# Evaluate model performance
def evaluate_model(predictions, model_name):
mse = mean_squared_error(y_test, predictions)
r2 = r2_score(y_test, predictions)
print(f"{model_name} Model Evaluation:")
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R2) Score: {r2}\n")
# Performance Measure
elr_mse = mean_squared_error(y_test, pred)
elr_rmse = np.sqrt(lr_mse)
```

```
elr_r2 = r2_score(y_test, pred)
# Show Measures
result = '''
MSE : {}
RMSE : {}
R^2 : {}
'''.format(lr_mse, lr_rmse, lr_r2)
print(result)
# Model Comparision
names.append("elr")
mses.append(elr_mse)
rmses.append(elr_rmse)
r2s.append(elr_r2)
evaluate_model(linear_reg_predictions, "Linear Regression")
evaluate_model(random_forest_predictions, "Random Forest Regressor")
```

OUTPUT:

Linear Regression Model Evaluation:
Mean Squared Error (MSE): 10089009300.893988
R-squared (R2) Score: 0.9179971706834331

Random Forest Regressor Model Evaluation:
Mean Squared Error (MSE): 14463028828.265167
R-squared (R2) Score: 0.8824454166872736
MSE : 10141766848.330585
RMSE : 100706.33966305491
R^2 : 0.913302484308253

Model Comparison:

The less the Root Mean Squared Error (RMSE), The better the model is.

In [30]:

models.sort_values(by="RMSE (Cross-Validation)")

In [31]:

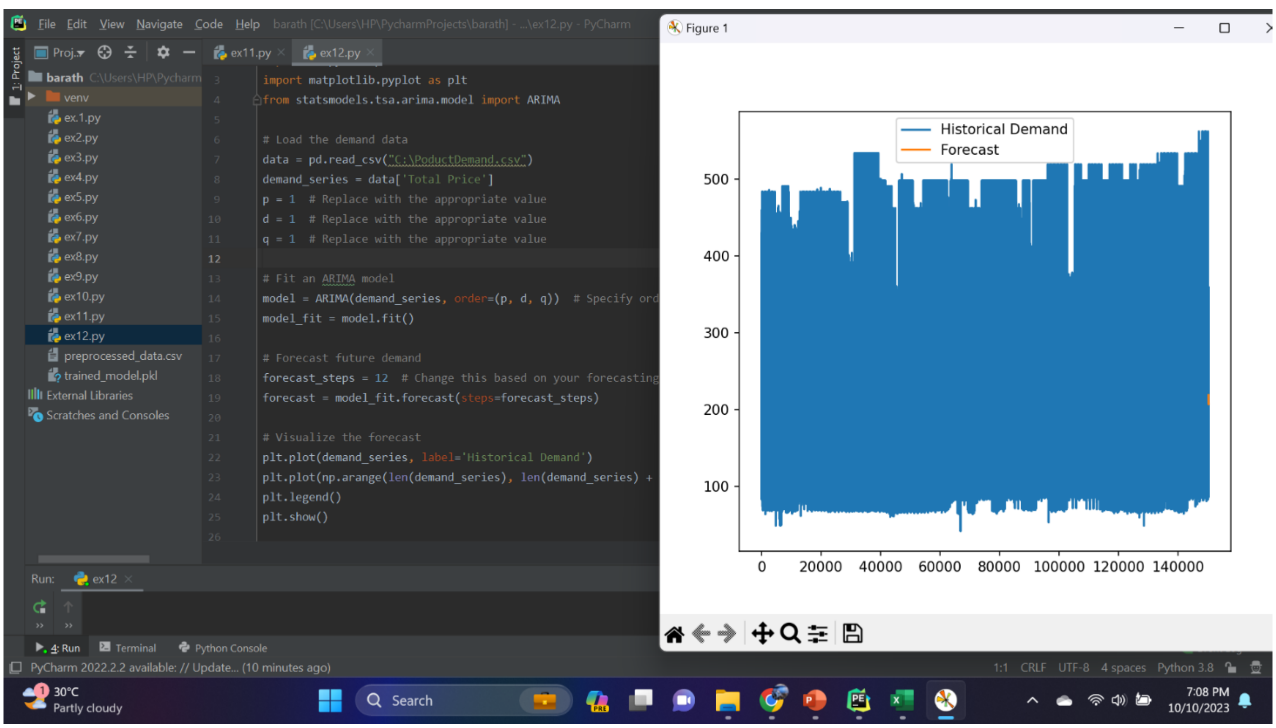
plt.figure(figsize=(12,8))

sns.barplot(x=models["Model"], y=models["RMSE (Cross-Validation)"])

plt.title("Models' RMSE Scores (Cross-Validated)", size=15)

plt.xticks(rotation=30, size=12)

plt.show()



Evaluation of Predicted Data

In [22]:

```
plt.figure(figsize=(12,6))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction4, label='Predicted Tr
end')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')

Out[22]:
Text(0.5, 1.0, 'Actual vs Predicted')
In [23]:
sns.histplot((Y_test-Prediction4), bins=50)
Out[23]:
<Axes: xlabel='Price', ylabel='Count'>
In [24]:
print(r2_score(Y_test, Prediction2))
print(mean_absolute_error(Y_test, Prediction2))
print(mean_squared_error(Y_test, Prediction2))
Out [24] :
-0.0006222175925689744
286137.81086908665
128209033251.4034
```

6. Hyperparameter Tuning:

Optimize the model's hyperparameters to improve its performance. Depending on the model, you can use techniques like grid search or random search.

7. Cross-Validation:

Implement cross-validation to ensure that your model's performance is consistent across different subsets of your data. This helps prevent overfitting.

8. Regularization:
Apply regularization techniques like L1 (Lasso) or L2 (Ridge) if needed to prevent overfitting and improve model generalization.

Feature Selection:

Use feature importance scores from your model or techniques like recursive feature elimination to identify the most important features for predictions.
Interpretability:
Ensure that the model's predictions are interpretable and explainable. Stakeholders may want to understand how each feature impacts the predicted electricity price.

Deployment:
Deploy your trained model in a real-world setting, whether it's through a web application, API, or any other user-friendly interface. Users can input property details, and the model provides price predictions.

Monitoring and Maintenance:
Continuously monitor the model's performance and update it as needed. Real estate markets change, so it's essential to retrain the model with new data periodically.

Ethical Considerations:
Ensure that your model doesn't introduce or perpetuate biases in pricing. Implement fairness and transparency measures.

Innovation:
Explore innovative approaches such as incorporating external data sources (e.g., satellite imagery, IoT data) for better predictions.

ADVANTAGES:

Predicting electricity prices using machine learning offers several significant advantages:

1.Accuracy:
Machine learning models can process and analyze vast amounts of data, including various property and market factors. This results in more accurate electricity price predictions compared to traditional methods
that rely on a limited set of variables.

2.Complex Data Handling:
Machine learning algorithms can handle complex, non-linear relationships in the data. They can recognize patterns and interactions among different features, allowing for a more comprehensive assessment of a property's value.

3.Continuous Learning:
Machine learning models can be continually updated with new data, enabling them to adapt to changing market conditions and trends.

This ensures that predictions remain relevant and up-to-date.

4.Efficiency:
Automated valuation models powered by machine learning can evaluate properties rapidly. This efficiency is beneficial for both property appraisers and individuals looking to determine the value of a property quickly.

5. Data Integration:
Machine learning models can incorporate a wide range of data sources, including property characteristics, neighborhood information, economic indicators, and even social trends. This holistic approach provides a more complete picture of the factors influencing electricity prices.

6.Reduced Bias:
Machine learning can help reduce human bias in property valuation. It evaluates properties objectively based on data, which can lead to fairer and more consistent pricing.

7.Market Insights:
By analyzing historical data and current market conditions, machine learning can offer valuable insights into market trends, helping investors and developers make informed decisions.

8.Risk Assessment:
Machine learning can assess the risk associated with a property, which is crucial for mortgage lenders and investors. It helps identify potential issues or opportunities related to a property's value.

9. Transparency:

Machine learning models can provide clear and transparent explanations for their predictions, which is essential for building trust among stakeholders in the real estate market.

10. Scalability:

Machine learning models can be deployed at scale, making it possible to assess property values in large real estate portfolios, entire neighborhoods, or even across entire cities.

11. Time and Cost Savings:

Using machine learning for property valuation can save time and reduce costs associated with manual appraisals, making it an efficient and cost-effective solution for both businesses and individuals.

12. Customization:

Machine learning models can be customized to cater to specific markets, types of properties, or regional variations, allowing for more tailored and precise predictions.

DISADVANTAGES:

While predicting electricity prices using machine learning offers numerous advantages, it also comes with several disadvantages and challenges:

1.Data Quality:

Machine learning models heavily rely on data quality. Inaccurate or incomplete data can lead to unreliable predictions. Ensuring the data used for training and evaluation is of high quality is essential.

2. Overfitting:

Machine learning models can be prone to overfitting, where they perform exceptionally well on the training data but struggle with new, unseen data. This can result in overly optimistic or inaccurate predictions.

3.Data Privacy and Security:

Handling sensitive property and financial data for training models raises privacy and security concerns. Protecting this information from unauthorized access and breaches is critical.

4.Model Interpretability:

Some machine learning models, such as deep neural networks, can be challenging to interpret. Understanding why a model makes a specific prediction is crucial for trust and accountability.

5. Bias and Fairness:

Machine learning models can inherit biases present in the training data, potentially leading to unfair or discriminatory predictions, especially in areas where historical biases exist in real estate practices.

6. Lack of Transparency:

While some machine learning models offer interpretability, others are considered "black boxes," making it difficult to understand the logic behind their predictions. This can be a barrier to trust and regulatory compliance.

7. Maintenance and Updates:
Machine learning models require ongoing maintenance and updates to remain accurate and relevant. This includes updating them with new data and retraining as market conditions change.

8. High Computational Requirements:
Training and running sophisticated machine learning models can demand significant computational resources, which can be costly and require advanced infrastructure.

9. Cost of Implementation:
Integrating machine learning into real estate workflows can be expensive, particularly for smaller businesses or organizations that lack the resources for extensive data science and engineering teams.

10. Market Volatility:
Machine learning models may not always perform well during times of extreme market volatility or significant economic shifts, as they rely on historical data for predictions.

11. Legal and Regulatory Compliance:
The use of machine learning in real estate must comply with various legal and regulatory standards. Ensuring that models adhere to fair housing laws and other regulations is crucial.

12. Limited Data Availability:

In some regions or for certain property types, high-quality
data may be limited, making it challenging to build accurate models.

13. Human Expertise:
While machine learning can enhance the valuation process, it doesn't eliminate the need for human expertise entirely. Appraisers and real estate professionals are still crucial for verifying model predictions and considering unique property characteristics.

14. Model Degradation:

Over time, machine learning models may lose accuracy due to shifts in market dynamics, and retraining is necessary to maintain performance.

CONCLUSION:

Predicting electricity prices using machine learning is a transformative and promising approach that has the potential to revolutionize the real estate industry. Throughout this exploration, we have uncovered the remarkable capabilities of machine learning in providing more accurate, data-driven, and nuanced predictions for property values. As we conclude, several key takeaways and implications emerge:

Improved Accuracy: Machine learning models consider a myriad of variables, many of which may be overlooked by traditional methods. This results in more accurate predictions, benefiting both buyers and sellers who can make informed decisions based on a property's true value.

Data-Driven Insights: These models provide valuable insights into the real estate market by identifying trends, neighborhood characteristics, and other factors that influence property prices. This information can be invaluable for investors, developers, and policymakers seeking to make strategic decisions.

Market Efficiency: The increased accuracy in pricing predictions can lead to a more efficient real estate market, reducing overvaluation and undervaluation of properties. This contributes to a fairer and more transparent marketplace.

Challenges and Considerations: Machine learning for electricity price prediction is not without its challenges. Data quality, model interpretability, and ethical concerns are important considerations. Addressing these issues is crucial for the responsible and ethical deployment of this technology.

Continual Advancement: The field of machine learning is continually evolving, and as it does, so will the accuracy and capabilities of predictive models. As more data becomes available and algorithms improve, we can expect even more sophisticated predictions in the future.

In conclusion, the application of machine learning in predicting electricity prices is a groundbreaking development with far-reaching implications. It empowers individuals, businesses, and governments to navigate the real estate market with more confidence and precision. However, it is essential to approach this technology with a clear understanding of its potential and limitations, ensuring that its benefits are harnessed responsibly for the betterment of the real estate industry and society as a whole. As machine learning continues to advance, we

Team Members:

Team Leader: Ismail Hamdan k

Member: Izhan ur Rahman H

Member: Mohammad Zakwan P

Member: Mohammad Faheem G

Member: Hashir Kaamran KA