IBM NAAN MUDHALVAN ELECTRICITY PRICES PREDICTION

DOMAIN	APPLIED DATA SCIENCE
PROJECT TOPIC	ELECTRICITY PRICES PREDICTION
TEAM MEMBERS& REGISTER NUMBER	MOHAMMED ISMAIL.I [410621104064] SIVARAMAKRISHNAN.R [410621104097] SHAFI AHAMED.s [410621104091] SYED ASHRAF [410621104105]

PHASE_4: SUBMISSION DOCUMENT

Introduction:

- **Predicting electricity prices is a common task in energy economics, finance, and energy management. Accurate predictions can help energy companies, consumers, and policymakers make informed decisions. Here's a high-level overview of the steps involved in predicting electricity prices**

Import Libraries:

- Numpy
- Pandas
- Matplotlib
- Seaborn

Data Set Link:

https://www.kaggle.com/datasets/chakradharmattapalli/electricity-price-prediction

READING FILE:

```
df=pd.read_csv("Electricity.csv", low_memory=False)
df.head()
```

OUTPUT:

	DateTime	Holiday	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year	PeriodOfDay	ForecastWindProduction	SystemLoadEA	SMPEA	ORKTemperatu
0	01/11/2011 00:00	None	0	1	44	1	11	2011	0	315.31	3388.77	49.26	6.
1	01/11/2011 00:30	None	0	1	44	1	11	2011	1	321.80	3196.66	49.26	6.
2	01/11/2011 01:00	None	0	1	44	1	11	2011	2	328.57	3060.71	49.10	5.
3	01/11/2011 01:30	None	0	1	44	1	11	2011	3	335.60	2945.56	48.04	6.
4	01/11/2011 02:00	None	0	1	44	1	11	2011	4	342.90	2849.34	33.75	6.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38014 entries, 0 to 38013
Data columns (total 18 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   DateTime                38014 non-null  object
 1   Holiday                 38014 non-null  object
 2   HolidayFlag             38014 non-null  int64
 3   DayOfWeek               38014 non-null  int64
 4   WeekOfYear              38014 non-null  int64
 5   Day                     38014 non-null  int64
 6   Month                   38014 non-null  int64
 7   Year                    38014 non-null  int64
 8   PeriodOfDay             38014 non-null  int64
 9   ForecastWindProduction  38014 non-null  object
 10  SystemLoadEA            38014 non-null  object
 11  SMPEA                   38014 non-null  object
 12  ORKTemperature          38014 non-null  object
 13  ORKWindspeed            38014 non-null  object
 14  CO2Intensity            38014 non-null  object
 15  ActualWindProduction    38014 non-null  object
 16  SystemLoadEP2           38014 non-null  object
 17  SMPEP2                  38014 non-null  object
dtypes: int64(7), object(11)
memory usage: 5.2+ MB
```

```
df.describe()
```

	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year	PeriodOfDay
count	38014.000000	38014.000000	38014.000000	38014.000000	38014.000000	38014.000000	38014.000000
mean	0.040406	2.997317	28.124586	15.739412	6.904246	2012.383859	23.501105
std	0.196912	1.999959	15.587575	8.804247	3.573696	0.624956	13.853108
min	0.000000	0.000000	1.000000	1.000000	1.000000	2011.000000	0.000000
25%	0.000000	1.000000	15.000000	8.000000	4.000000	2012.000000	12.000000
50%	0.000000	3.000000	29.000000	16.000000	7.000000	2012.000000	24.000000
75%	0.000000	5.000000	43.000000	23.000000	10.000000	2013.000000	35.750000
max	1.000000	6.000000	52.000000	31.000000	12.000000	2013.000000	47.000000

MODEL BUILDING:

Building a model for a dataset involves a series of steps and considerations. Here's a general outline of the process:

- **Understand the Problem**
- **Data Collection**
- **Data Preprocessing**
- **Feature Engineering**
- **Model selection**
- **Model Training**
- **Model Evaluation**
- **Model Testing**
- **Model Interpretation**
- **Model Deployement**

PROGRAM:

```
x=df[['HolidayFlag','DayOfWeek','WeekOfYear','Day','Month','Year']]
y=df['PeriodOfDay']
```

+ Code + Mark

```
x_train, x_test, y_train, y_test=train_test_split(x,y, test_size=0.2, random_state=42)
```

```
x_train
```

	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year
15238	0	3	37	13	9	2012
20071	0	6	51	23	12	2012
14654	0	5	35	1	9	2012
3964	0	6	3	22	1	2012
2855	0	4	52	30	12	2011
...	...	...	...	...	...	...
16850	0	2	42	17	10	2012
6265	0	5	10	10	3	2012
11284	0	5	25	23	6	2012
860	0	4	46	18	11	2011
15795	0	1	39	25	9	2012

30411 rows × 6 columns

```
x_test
```

	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year
35833	0	5	46	16	11	2013
198	0	5	44	5	11	2011
36547	0	6	48	1	12	2013
26373	0	4	18	3	5	2013
21156	0	0	3	14	1	2013
...	...	...	...	...	...	...
13927	0	4	33	17	8	2012
16926	0	3	42	18	10	2012
24520	0	0	13	25	3	2013
9059	1	0	19	7	5	2012
9786	0	1	21	22	5	2012

7603 rows × 6 columns

```
y_train
```

```
15238    24
20071     9
14654    16
3964     28
2855     23
         ..
16850     4
6265     25
11284     6
860      44
15795     5
Name: PeriodOfDay, Length: 30411, dtype: int64
```

```
y_test
```

```
35833    27
198       6
36547    21
26373    23
21156    38
         ..
13927     9
16926    32
24520    42
9059     37
9786     44
Name: PeriodOfDay, Length: 7603, dtype: int64
```

MODEL EVALUATION:

- Model evaluation is a critical step in the machine learning and data analysis process. It involves assessing how well a trained model performs on a given dataset. The goal of model evaluation is to determine the model's effectiveness, generalization capability, and suitability for a specific task. Here are some common techniques and metrics used for model evaluation.

  - **Splitting the Data**
  - **Training the Model**
  - **Model Evaluation Metrics**
  - **Classification Problems**

- Accuracy
- Precision, Recall, F1-Score
- ROC AUC
- Confusion Matrix

➢ **Regression Problems**

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- R-squared (R2)

➢ **Error Analysis**

➢ **Deployment and Monitoring**

PROGRAM:

```
from sklearn.model_selection import cross_val_score
num_folds = 5
def perform_cross_validation(model, X, y, num_folds):
    mse_scores = -cross_val_score(model, X, y, cv=num_folds, scoring='neg_mean_squared_error')
    rmse_scores = np.sqrt(mse_scores)
    mae_scores = -cross_val_score(model, X, y, cv=num_folds, scoring='neg_mean_absolute_error')
    r2_scores = cross_val_score(model, X, y, cv=num_folds, scoring='r2')

    return mse_scores, rmse_scores, mae_scores, r2_scores
```

##LINE REGRESSION##

```
from sklearn.linear_model import LinearRegression, Ridge, Lasso

# Linear Regression
linear_model = LinearRegression()
linear_mse, linear_rmse, linear_mae, linear_r2 = perform_cross_validation(linear_model, x, y, num_folds)
print("Linear Regression:")
print(f"Average MSE: {np.mean(linear_mse) / np.mean(y) * 100:.2f}%")
print(f"Average RMSE: {np.mean(linear_rmse) / np.mean(y) * 100:.2f}%")
print(f"Average MAE: {np.mean(linear_mae) / np.mean(y) * 100:.2f}%")
print(f"Average R-squared: {np.mean(linear_r2) * 100:.2f}%")
print("\n")
```

```
Linear Regression:
Average MSE: 816.63%
Average RMSE: 58.95%
Average MAE: 51.06%
Average R-squared: -0.01%
```

##RIDGE REGRESSION##

```
# Ridge Regression
ridge_model = Ridge(alpha=1.0)  # You can adjust alpha as needed
ridge_mse, ridge_rmse, ridge_mae, ridge_r2 = perform_cross_validation(ridge_model, x, y, num_folds)
print("Ridge Regression:")
print(f"Average MSE: {np.mean(ridge_mse) / np.mean(y) * 100:.2f}%")
print(f"Average RMSE: {np.mean(ridge_rmse) / np.mean(y) * 100:.2f}%")
print(f"Average MAE: {np.mean(ridge_mae) / np.mean(y) * 100:.2f}%")
print(f"Average R-squared: {np.mean(ridge_r2) * 100:.2f}%")
print("\n")
```

```
Ridge Regression:
Average MSE: 816.63%
Average RMSE: 58.95%
Average MAE: 51.06%
Average R-squared: -0.01%
```

##LASSO REGRESSION##

```
# Lasso Regression
lasso_model = Lasso(alpha=1.0)  # You can adjust alpha as needed
lasso_mse, lasso_rmse, lasso_mae, lasso_r2 = perform_cross_validation(lasso_model, x, y, num_folds)
print("Lasso Regression:")
print(f"Average MSE: {np.mean(lasso_mse) / np.mean(y) * 100:.2f}%")
print(f"Average RMSE: {np.mean(lasso_rmse) / np.mean(y) * 100:.2f}%")
print(f"Average MAE: {np.mean(lasso_mae) / np.mean(y) * 100:.2f}%")
print(f"Average R-squared: {np.mean(lasso_r2) * 100:.2f}%")
print("\n")
```

```
Lasso Regression:
Average MSE: 816.57%
Average RMSE: 58.95%
Average MAE: 51.06%
Average R-squared: -0.00%
```

##DECISION TREE##

```
from sklearn.tree import DecisionTreeRegressor

# Decision Trees
tree_model = DecisionTreeRegressor(max_depth=None, random_state=0)  # You can adjust parameters as needed
tree_mse, tree_rmse, tree_mae, tree_r2 = perform_cross_validation(tree_model, x, y, num_folds)
print("Decision Trees:")
print(f"Average MSE: {np.mean(tree_mse) / np.mean(y) * 100:.2f}%")
print(f"Average RMSE: {np.mean(tree_rmse) / np.mean(y) * 100:.2f}%")
print(f"Average MAE: {np.mean(tree_mae) / np.mean(y) * 100:.2f}%")
print(f"Average R-squared: {np.mean(tree_r2) * 100:.2f}%")
print("\n")
```

```
Decision Trees:
Average MSE: 909.96%
Average RMSE: 62.04%
Average MAE: 53.06%
Average R-squared: -11.45%
```

##RANDOM FOREST##

```
from sklearn.ensemble import RandomForestRegressor

# Random Forest
forest_model = RandomForestRegressor(n_estimators=100, random_state=0)  # You can adjust parameters as needed
forest_mse, forest_rmse, forest_mae, forest_r2 = perform_cross_validation(forest_model, x, y, num_folds)
print("Random Forest:")
print(f"Average MSE: {np.mean(forest_mse) / np.mean(y) * 100:.2f}%")
print(f"Average RMSE: {np.mean(forest_rmse) / np.mean(y) * 100:.2f}%")
print(f"Average MAE: {np.mean(forest_mae) / np.mean(y) * 100:.2f}%")
print(f"Average R-squared: {np.mean(forest_r2) * 100:.2f}%")
```

```
Random Forest:
Average MSE: 834.38%
Average RMSE: 59.58%
Average MAE: 51.48%
Average R-squared: -2.18%
```

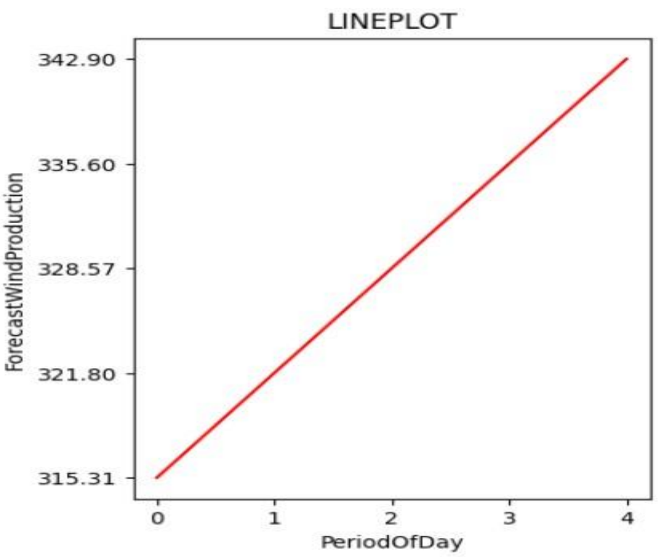
VISUALIZATION:

- Data visualization is a powerful way to represent and communicate information from data through visual elements like charts, graphs, and maps. Effective data visualization can make complex data more understandable and can help identify patterns, trends, and insights that might be hidden in raw data. Here are some key concepts and best practices for data visualization.

PROGRAM:

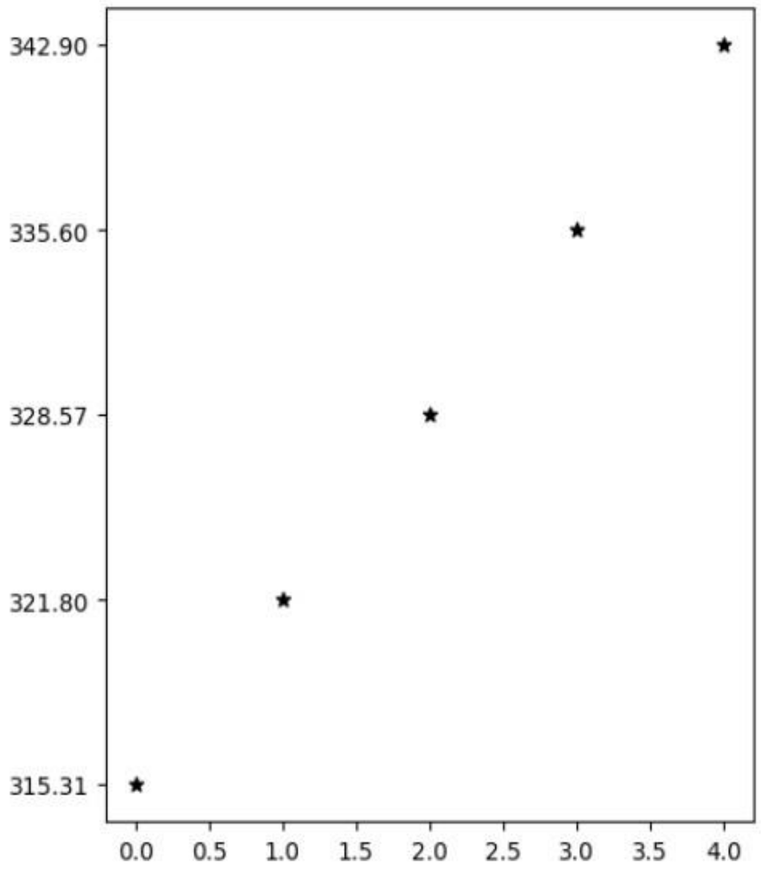
##LINE PLOT##

```
##Lineplot
a=df['PeriodOfDay'].head(5)
df1=df['ForecastWindProduction'].head(5)
fig = plt.figure(figsize =(4, 5))
plt.plot(a, df1,color='red')
plt.title("LINEPLOT")
plt.xlabel("PeriodOfDay")
plt.ylabel("ForecastWindProduction")
plt.show()
```

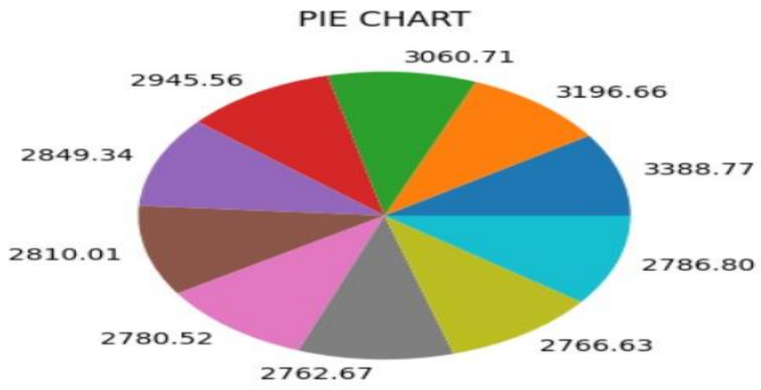


##SCATTER PLOT##

```
##Scatterplot
a=df['PeriodOfDay'].head()
df1=df['ForecastWindProduction'].head()
fig = plt.figure(figsize =(5, 6))
plt.scatter(a, df1,marker='*',color='black')
plt.show("SCATTERPLOT")
plt.show()
```

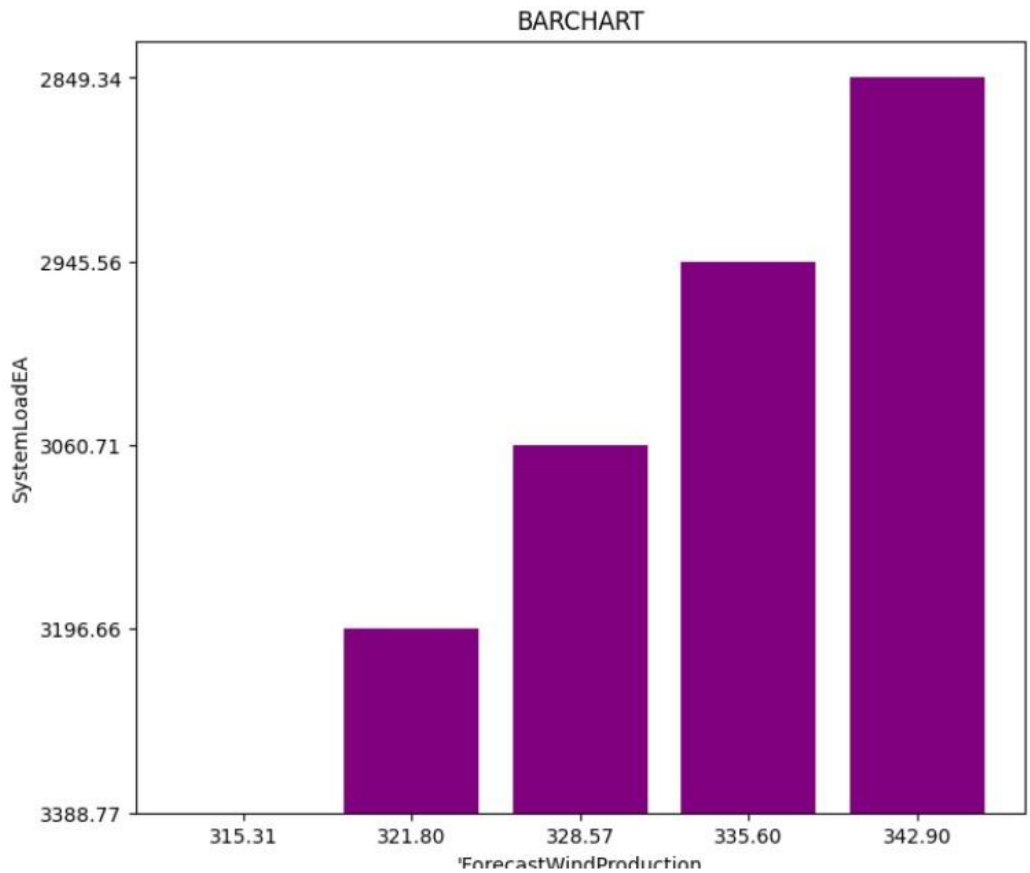


##PIE CHART##



##BAR CHART##

```
a=df['ForecastWindProduction'].head(5)
df1=df['SystemLoadEA'].head(5)
fig = plt.figure(figsize =(8, 7))
plt.bar(a, df1,color='purple')
plt.title("BARCHART")
plt.xlabel("'ForecastWindProduction")
plt.ylabel("SystemLoadEA")
plt.show()
```



Conclusion:

In this phase,The Model Building ,Model Evaluation and visualize the Dataset has been successfully verified and executed successfully.