(B.TECH) Semester-VII AY 2023-24
DL Lab Assignment No. 01

Student Name: _Rohit Saini_____ PRN No.: 1032200897

Date: 06-10-2023 Faculty: Prof. Anita Gunjal

Problem Statement: Implementation of Machine learning algorithm.

Objectives:

1. To understand and implement the functionalities of Machine learning algorithm

Theory: (describe the following and enlist the different techniques associated with it)

- Explain Supervised and unsupervised learning and reinforcement learning

Supervised Learning:

- Uses labeled data (input-output pairs) to train a model for making predictions or classifications.
- Examples: Image recognition, spam email detection.
- Key algorithms: Linear Regression, Decision Trees, Neural Networks.

Unsupervised Learning:

- Works with unlabeled data to find patterns, groupings, or reduce dimensionality.
- Examples: Customer segmentation, data compression.
- Key algorithms: K-Means Clustering, Principal Component Analysis (PCA).

Reinforcement Learning:

- Involves an agent interacting with an environment to learn optimal decision-making through trial and error.
- Examples: Game playing (e.g., chess), autonomous robotics.
- Key algorithms: Q-Learning, Deep Q Networks (DQN).

Operations to be performed

Build a model using any machine learning algorithm

Step 1: Importing the dataset

Step 2: Data clean up

Step 3: Splitting the test and train sets

Step 4: Fitting the linear regression model to the training set

Step 5: Predicting test results

Step 6: Visualizing the test results.

Program code: (paste your program code)

In [1]:
```
#Name: Rohit Saini
#Roll No.: PC41
#Panel: C
#ERP Id.: 1032200897
```

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="whitegrid")

import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
state_data = pd.read_csv("state.csv")
state_data.columns
```

Out[3]:
```
Index(['TimeStamp', 'Punjab', 'Haryana', 'Rajasthan', 'Delhi', 'UP',
       'Uttarakhand', 'HP', 'J&K', 'Chandigarh', 'Chhattisgarh', 'Gujarat',
       'MP', 'Maharashtra', 'Goa', 'DNH', 'Andhra Pradesh', 'Telangana',
       'Karnataka', 'Kerala', 'Tamil Nadu', 'Pondy', 'Bihar', 'Jharkhand',
       'Odisha', 'West Bengal', 'Sikkim', 'Arunachal Pradesh', 'Assam',
       'Manipur', 'Meghalaya', 'Mizoram', 'Nagaland', 'Tripura'],
      dtype='object')
```

In [4]:
```
#performing exploratory data analysis
state_data.head()
```

Out[4]:

	TimeStamp	Punjab	Haryana	Rajasthan	Delhi	UP	Uttarakhand	HP	J&K	Chandi
0	02/01/2019 00:00:00	119.9	130.3	234.1	85.8	313.9	40.7	30.0	52.5	
1	03/01/2019 00:00:00	121.9	133.5	240.2	85.5	311.8	39.3	30.1	54.1	
2	04/01/2019 00:00:00	118.8	128.2	239.8	83.5	320.7	38.1	30.1	53.2	
3	05/01/2019 00:00:00	121.0	127.5	239.1	79.2	299.0	39.2	30.2	51.5	
4	06/01/2019 00:00:00	121.4	132.6	240.4	76.6	286.8	39.2	31.0	53.2	

5 rows × 34 columns

In [5]:
```
#checking for null values
state_data.isnull().sum()
```

Out[5]:
```
TimeStamp           0
Punjab              0
Haryana             0
Rajasthan           0
Delhi               0
UP                  0
Uttarakhand         0
HP                  0
J&K                 0
Chandigarh          0
Chhattisgarh        0
Gujarat             0
MP                  0
Maharashtra         0
Goa                 0
DNH                 0
Andhra Pradesh      0
Telangana           0
Karnataka           0
Kerala              0
Tamil Nadu          0
Pondy               0
Bihar               0
Jharkhand           0
Odisha              0
West Bengal         0
Sikkim              0
Arunachal Pradesh   0
Assam               0
Manipur             0
Meghalaya           0
Mizoram             0
Nagaland            0
Tripura             0
dtype: int64
```

In [6]:
```
# checking for duplicate values
state_data.duplicated().sum()
```

Out[6]:
```
0
```

In [7]:
```
# checking for unique values
state_data.nunique()
```

Out[7]:
```
TimeStamp            474
Punjab               383
Haryana              374
Rajasthan            367
Delhi                328
UP                   426
Uttarakhand          201
HP                   138
J&K                  171
Chandigarh            46
Chhattisgarh         269
Gujarat              387
MP                   392
Maharashtra          406
Goa                   69
DNH                   81
Andhra Pradesh       313
Telangana            372
Karnataka            381
Kerala               227
Tamil Nadu           397
Pondy                 56
Bihar                288
Jharkhand             92
Odisha               278
West Bengal          354
Sikkim                22
Arunachal Pradesh     17
Assam                167
Manipur               16
Meghalaya             36
Mizoram               11
Nagaland              12
Tripura               38
dtype: int64
```

In [8]:
```
#checking for outliers
state_data.describe()
```

Out[8]:

	Punjab	Haryana	Rajasthan	Delhi	UP	Uttarakhand	H
count	479.000000	479.000000	479.000000	479.000000	479.000000	479.000000	479.00000
mean	139.283925	137.508559	218.463257	82.746764	312.864927	36.026514	26.48747
std	55.117346	37.161544	27.747140	25.437537	65.522780	6.766950	4.88624
min	56.100000	64.800000	105.800000	41.800000	186.800000	16.800000	11.80000
25%	103.800000	114.950000	205.800000	63.450000	263.750000	33.700000	25.60000
50%	117.900000	126.500000	223.200000	72.200000	289.200000	36.900000	28.00000
75%	157.900000	157.600000	238.100000	103.500000	368.400000	40.300000	29.65000
max	291.700000	224.500000	278.000000	139.100000	471.800000	53.200000	34.00000

8 rows × 33 columns

In [9]:
```
#checking for correlation
#removing the date column
state_data.drop(['TimeStamp'],axis=1,inplace=True)
state_data.corr()
```

Out[9]:

	Punjab	Haryana	Rajasthan	Delhi	UP	Uttarakhand	HP
Punjab	1.000000	0.954204	0.379866	0.895979	0.796746	0.646287	0.439537
Haryana	0.954204	1.000000	0.524308	0.938643	0.830256	0.762450	0.561598
Rajasthan	0.379866	0.524308	1.000000	0.447224	0.355965	0.758152	0.750326
Delhi	0.895979	0.938643	0.447224	1.000000	0.902487	0.727454	0.455119
UP	0.796746	0.830256	0.355965	0.902487	1.000000	0.638674	0.313157
Uttarakhand	0.646287	0.762450	0.758152	0.727454	0.638674	1.000000	0.862376
HP	0.439537	0.561598	0.750326	0.455119	0.313157	0.862376	1.000000
J&K	-0.168837	-0.126184	0.393304	-0.151198	-0.112345	0.218816	0.253572
Chandigarh	0.888155	0.920907	0.541708	0.930987	0.822472	0.809346	0.601955
Chhattisgarh	0.428080	0.466916	0.240842	0.521781	0.525068	0.444924	0.290917
Gujarat	0.201078	0.334278	0.643326	0.393591	0.368183	0.628358	0.531002
MP	-0.316029	-0.194271	0.597693	-0.232705	-0.194979	0.302104	0.445074
Maharashtra	-0.015554	0.079674	0.451837	0.171311	0.221102	0.409920	0.307486
Goa	0.231636	0.339210	0.432585	0.346624	0.233002	0.566494	0.607341
DNH	0.354388	0.475452	0.644641	0.373550	0.221084	0.783916	0.884969
Andhra Pradesh	0.126014	0.145515	0.335506	0.187162	0.243982	0.319716	0.189248
Telangana	-0.185938	-0.169950	0.172305	-0.274065	-0.280207	0.068063	0.225154
Karnataka	-0.357036	-0.343064	0.139472	-0.312579	-0.266288	-0.029090	-0.027616
Kerala	-0.193411	-0.123704	0.158097	-0.042318	0.020113	0.151824	0.071167
Tamil Nadu	0.334458	0.383550	0.415059	0.413933	0.409728	0.549271	0.397141
Pondy	0.528687	0.598985	0.534659	0.606499	0.570916	0.733926	0.594069
Bihar	0.664184	0.659901	0.156159	0.727994	0.869876	0.449103	0.175631
Jharkhand	0.274295	0.343025	0.357653	0.388228	0.455423	0.424625	0.310759
Odisha	0.628012	0.634383	0.150936	0.659277	0.629382	0.396205	0.248552
West Bengal	0.683583	0.688897	0.186185	0.766340	0.767062	0.501458	0.221982
Sikkim	-0.402998	-0.343130	0.303134	-0.381523	-0.425773	0.041556	0.210742
Arunachal Pradesh	0.372024	0.450933	0.531184	0.351417	0.242291	0.644008	0.708433
Assam	0.810056	0.805966	0.281023	0.752180	0.683119	0.547547	0.439183
Manipur	-0.000601	0.062689	0.390987	-0.035707	-0.086316	0.283900	0.381727

	Punjab	Haryana	Rajasthan	Delhi	UP	Uttarakhand	HP
Meghalaya	0.074794	0.183663	0.645495	0.029239	-0.103093	0.552688	0.751811
Mizoram	0.054273	0.115237	0.290207	0.067552	-0.010089	0.329881	0.419715
Nagaland	0.345971	0.344556	0.283352	0.290389	0.233587	0.338323	0.331626
Tripura	0.718911	0.713311	0.215746	0.746484	0.737230	0.495721	0.277179

33 rows × 33 columns

In [10]:
```
state_data
```

Out[10]:

	Punjab	Haryana	Rajasthan	Delhi	UP	Uttarakhand	HP	J&K	Chandigarh	Chha
0	119.9	130.3	234.1	85.8	313.9	40.7	30.0	52.5	5.0	
1	121.9	133.5	240.2	85.5	311.8	39.3	30.1	54.1	4.9	
2	118.8	128.2	239.8	83.5	320.7	38.1	30.1	53.2	4.8	
3	121.0	127.5	239.1	79.2	299.0	39.2	30.2	51.5	4.3	
4	121.4	132.6	240.4	76.6	286.8	39.2	31.0	53.2	4.3	
...	...	...	...	...	...	...	...	...	...	
474	108.5	109.0	219.8	59.3	242.3	32.1	26.4	45.4	3.1	
475	235.3	204.6	203.2	122.8	425.6	41.1	29.5	39.5	6.2	
476	242.9	215.7	212.6	129.2	434.0	43.9	30.7	41.2	6.5	
477	246.2	217.8	225.5	130.3	414.1	41.1	30.7	40.3	6.1	
478	214.3	208.0	225.7	128.7	439.6	42.8	29.9	38.6	5.7	

479 rows × 33 columns

In [11]:
```
#checking for skewness
state_data.skew()
```

Out[11]:
```
Punjab               1.142459
Haryana              0.616228
Rajasthan           -0.987449
Delhi                0.693088
UP                   0.651181
Uttarakhand         -0.937398
HP                  -1.702102
J&K                 -0.843650
Chandigarh           0.564693
Chhattisgarh         0.201572
Gujarat             -0.472148
MP                   0.072662
Maharashtra         -0.003594
Goa                 -0.839788
DNH                 -2.257284
Andhra Pradesh      -0.009078
Telangana            0.374846
Karnataka           -0.230668
Kerala               0.358679
Tamil Nadu          -0.308372
Pondy               -0.772761
Bihar                0.547492
Jharkhand           -0.319347
Odisha               0.292501
West Bengal          0.325922
Sikkim               0.590485
Arunachal Pradesh   -1.617015
Assam                0.707613
Manipur             -0.551150
Meghalaya           -1.131550
Mizoram             -0.223018
Nagaland            -0.029123
Tripura              0.490047
dtype: float64
```

In [12]:
```
state_data = state_data[['Goa', 'Gujarat', 'Maharashtra', 'DNH', 'Rajasthan']]
state_data
```

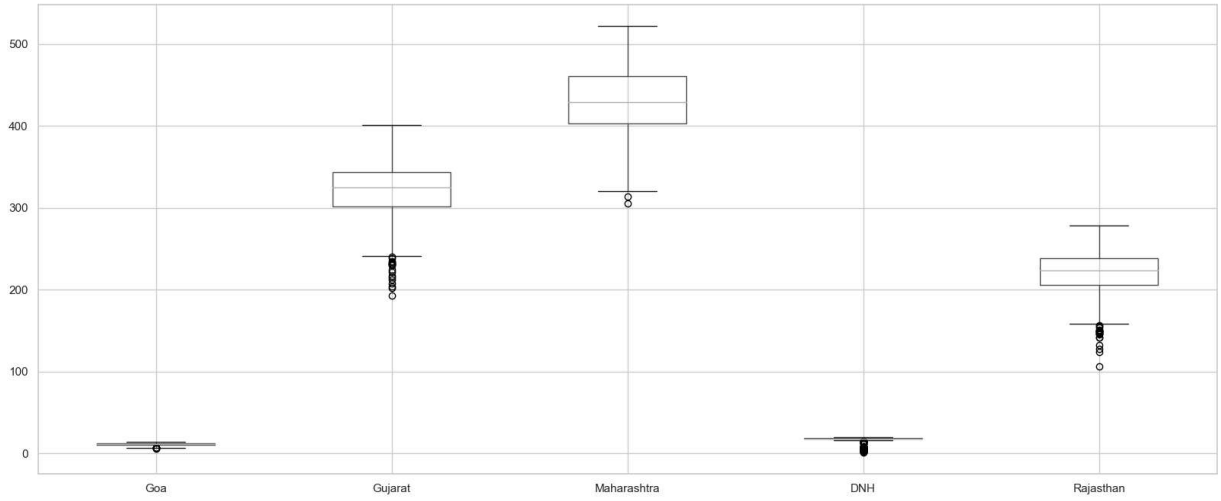
Out[12]:

	Goa	Gujarat	Maharashtra	DNH	Rajasthan
0	12.8	319.5	428.6	18.6	234.1
1	13.7	316.7	419.6	18.2	240.2
2	12.6	301.9	395.8	16.7	239.8
3	13.0	313.2	411.1	17.6	239.1
4	12.9	320.7	408.6	18.6	240.4
...	...	...	...	...	...
474	12.3	312.0	421.6	18.2	219.8
475	8.8	289.7	369.2	17.4	203.2
476	8.8	297.9	395.5	18.3	212.6
477	8.8	296.8	395.5	18.7	225.5
478	10.2	288.7	399.6	18.5	225.7

479 rows × 5 columns

In [13]:
```
# checking for outliers
from scipy.stats import zscore
state_data.boxplot(figsize=(20, 8))
```

Out[13]: <Axes: >



In [14]:
```
# removing outliers
z = np.abs(zscore(state_data))
z

threshold = 3
print(np.where(z > 3))

state_data = state_data[(z < 3).all(axis=1)]
state_data
```

```
(array([ 28,  28,  28,  29,  30,  31,  32,  33,  34,  35, 143, 144, 144,
       145, 145, 145, 146, 146, 146, 300, 419, 420, 421, 422, 423, 424,
       425], dtype=int64), array([1, 3, 4, 3, 3, 3, 3, 3, 3, 2, 0, 3, 4, 1, 3, 4, 1,
3, 4, 3, 3, 3,
       3, 3, 3, 3, 3], dtype=int64))
```

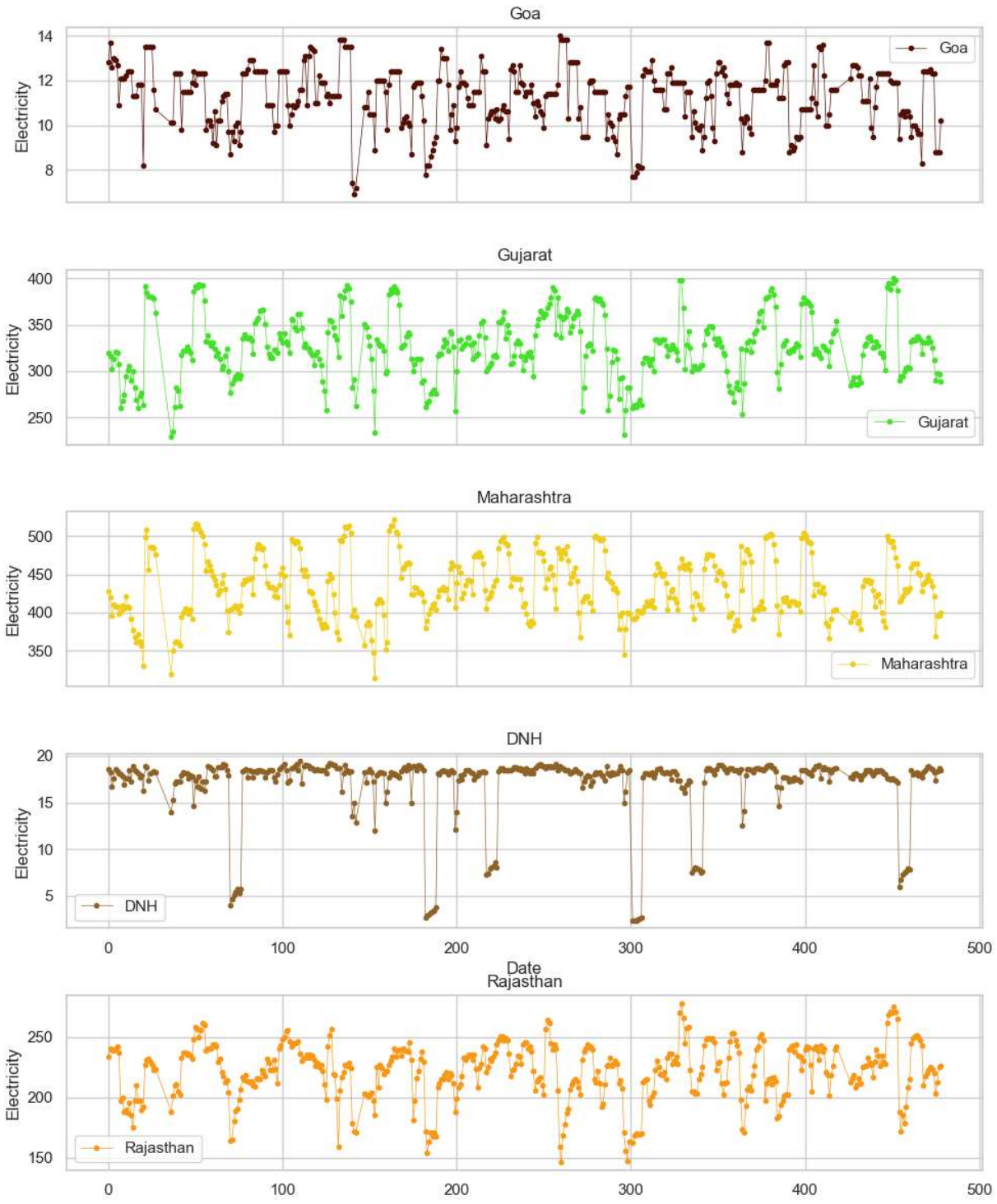
Out[14]:

	Goa	Gujarat	Maharashtra	DNH	Rajasthan
0	12.8	319.5	428.6	18.6	234.1
1	13.7	316.7	419.6	18.2	240.2
2	12.6	301.9	395.8	16.7	239.8
3	13.0	313.2	411.1	17.6	239.1
4	12.9	320.7	408.6	18.6	240.4
...	...	...	...	...	...
474	12.3	312.0	421.6	18.2	219.8
475	8.8	289.7	369.2	17.4	203.2
476	8.8	297.9	395.5	18.3	212.6
477	8.8	296.8	395.5	18.7	225.5
478	10.2	288.7	399.6	18.5	225.7

459 rows × 5 columns

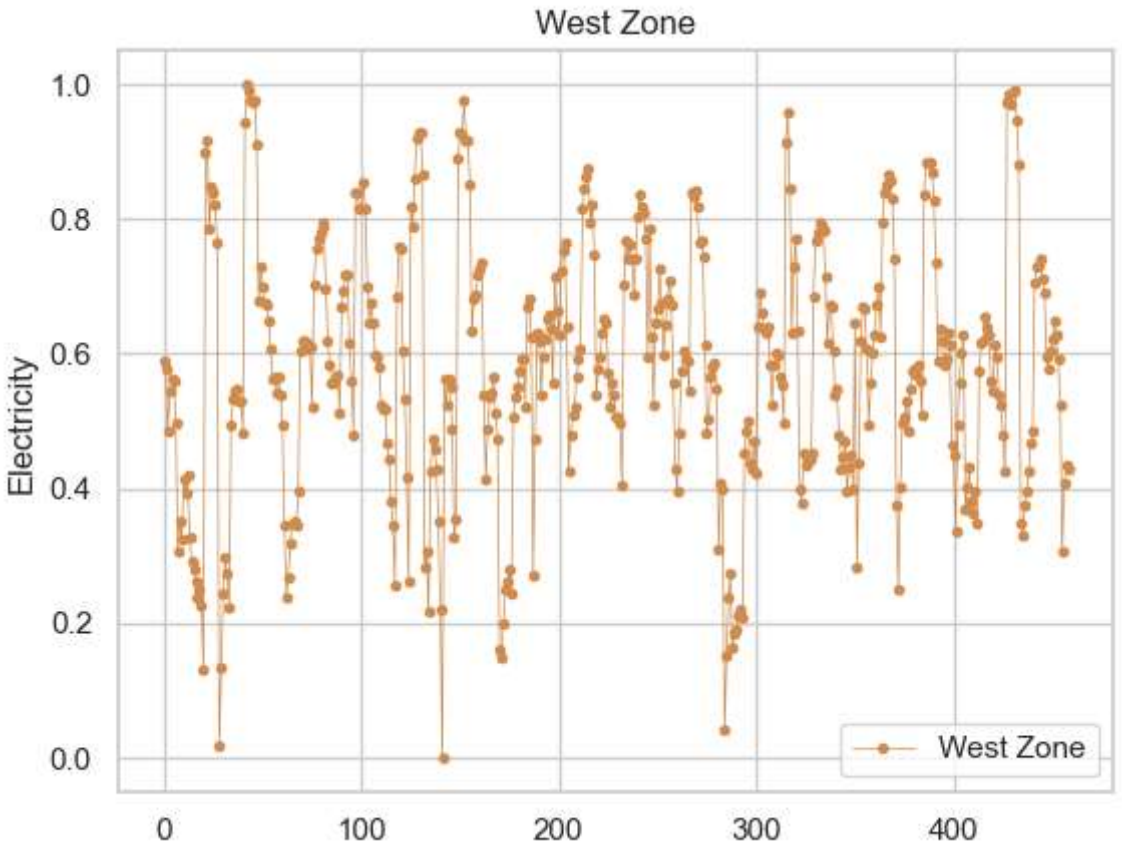
In [15]:

```
states = ['Goa', 'Gujarat', 'Maharashtra', 'DNH','Rajasthan']
fig, axes = plt.subplots(nrows=len(states), ncols=1, figsize=(10, 12))

# plot the data by day for each state with random color
for i, state in enumerate(states):
    axes[i].plot(state_data[state], marker='.', linestyle='-', linewidth=0.5, label
    axes[i].set_ylabel('Electricity')
    axes[i].set_title(state)
    axes[i].legend()
# set some figure-level properties
fig.tight_layout()

# set some axes-level properties
axes[0].set_xticklabels([])
axes[1].set_xticklabels([])
axes[2].set_xticklabels([])
axes[3].set_xlabel('Date')

axes[0].set_ylabel('Electricity')
axes[1].set_ylabel('Electricity')
axes[2].set_ylabel('Electricity')
axes[3].set_ylabel('Electricity')

plt.show()
```



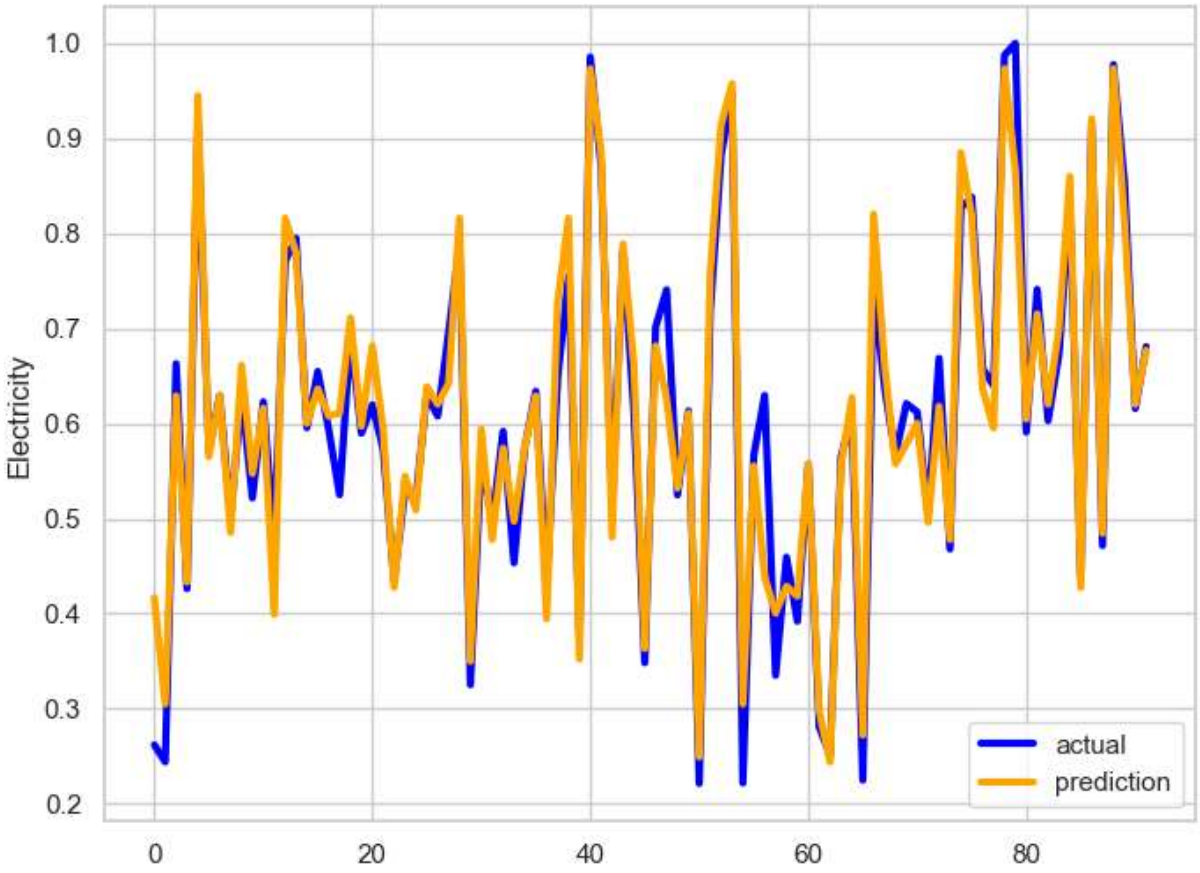
In [16]:
```
#add the goa , gujarat , maharashtra and dnh data to get the total electricity in w
west_zone = state_data['Goa'] + state_data['Gujarat'] + state_data['Maharashtra'] +
# west_zone
#scale the data
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
west_zone = scaler.fit_transform(np.array(west_zone).reshape(-1,1))
# west_zone
#data is of 503 days
# len(west_zone)
```

In [17]:
```
# plot the west zone
# plot the data by day for each state with random color
plt.plot(west_zone, marker='.', linestyle='-', linewidth=0.5, label='West Zone', co
plt.ylabel('Electricity')
plt.title('West Zone')
plt.legend()
plt.show()
```



In [18]:
```
#using decision tree regressor to predict the electricity in west zone
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split

#split the data into train and test
X_train, X_test, y_train, y_test = train_test_split(state_data, west_zone, test_siz

#fit the model
model = DecisionTreeRegressor()
model.fit(X_train, y_train)

#predict the electricity in west zone
y_pred = model.predict(X_test)

#plot the actual and predicted electricity in west zone
plt.figure(figsize=(8, 6))
plt.plot(y_test, label='actual', color='blue', linewidth=3)
plt.plot(y_pred, label='prediction', color='orange', linewidth=3)
plt.ylabel('Electricity')
```

```
plt.legend()
plt.show()
```



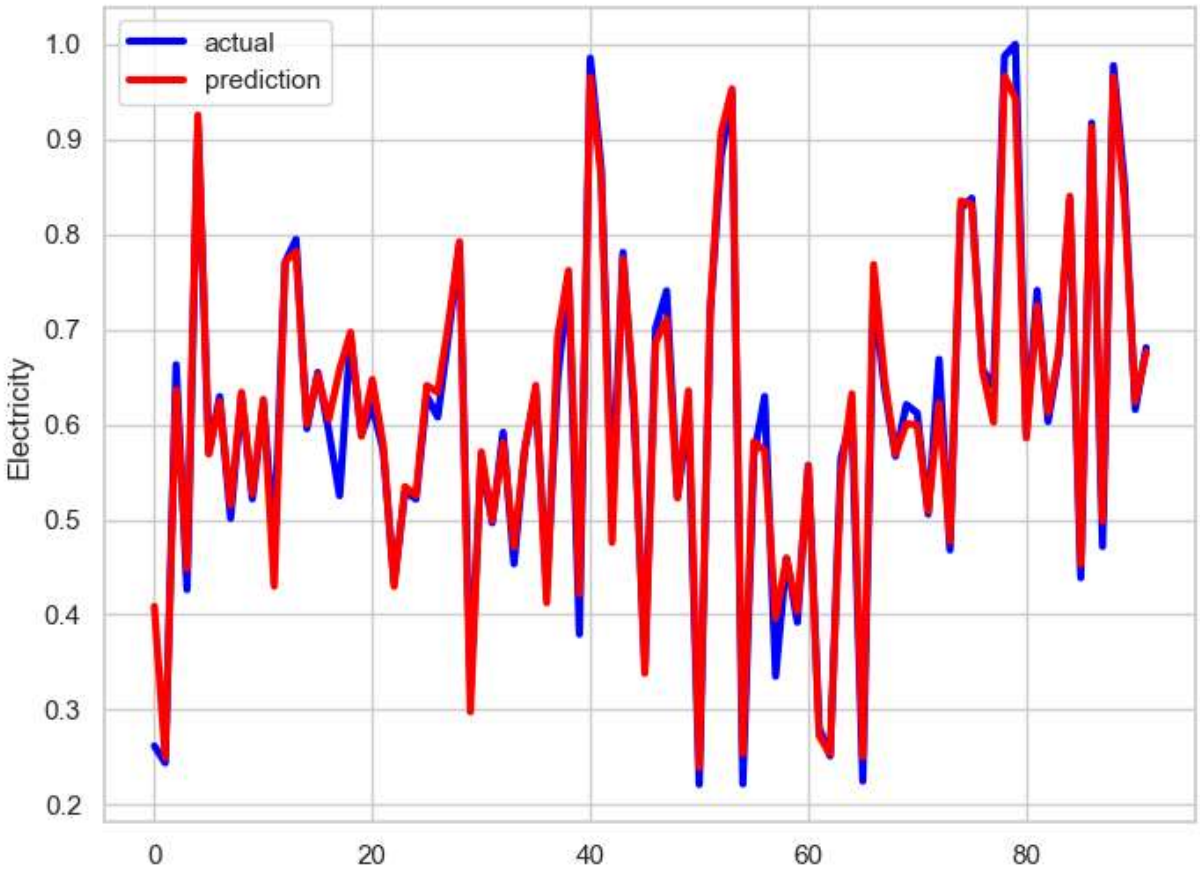
In [19]:
```
#print the predicted electricity consumption in west zone for next 7 days
# print("Predicted electricity consumption in west zone for next 7 days:")
# print(model.predict(state_data.tail(7)))

#rescale the data
y_pred = scaler.inverse_transform(y_pred.reshape(-1,1))
y_test = scaler.inverse_transform(y_test.reshape(-1,1))

#print the actual and predicted electricity consumption in west zone for next 7 day
print("Actual electricity consumption in west zone for next 7 days:")
print(y_test[-7:])
print("Predicted electricity consumption in west zone for next 7 days:")
print(y_pred[-7:])
```

```
Actual electricity consumption in west zone for next 7 days:
[[ 947. ]
 [1158.1]
 [ 961.5]
 [1184.7]
 [1130.7]
 [1025.1]
 [1053.8]]
Predicted electricity consumption in west zone for next 7 days:
[[ 942. ]
 [1159.6]
 [ 967.2]
 [1182.8]
 [1113.6]
 [1027. ]
 [1052.1]]
```

In [20]:
```python
#calculate the mean absolute error and accuracy of the model
from sklearn.metrics import mean_absolute_error, r2_score
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred))
print("Accuracy:", r2_score(y_test, y_pred))
#storing the accuracy in a variable
accuracy_decision_tree = r2_score(y_test, y_pred)
```

```
Mean Absolute Error: 13.573913043478287
Accuracy: 0.9389762032349372
```

In [21]:
```python
# using random forest regressor to predict the electricity in west zone
from sklearn.ensemble import RandomForestRegressor

# split the data into train and test
X_train, X_test, y_train, y_test = train_test_split(
    state_data, west_zone, test_size=0.2, random_state=42)

# fit the model
model = RandomForestRegressor()
model.fit(X_train, y_train)

# predict the electricity in west zone
y_pred = model.predict(X_test)

# plot the actual and predicted electricity in west zone
plt.figure(figsize=(8, 6))
plt.plot(y_test, label='actual', color='blue', linewidth=3)
plt.plot(y_pred, label='prediction', color='red', linewidth=3)
plt.ylabel('Electricity')
plt.legend()
plt.show()
```



In [22]:
```
y_pred = scaler.inverse_transform(y_pred.reshape(-1,1))
y_test = scaler.inverse_transform(y_test.reshape(-1,1))

#print the actual and predicted electricity consumption in west zone for next 7 day
print("Actual electricity consumption in west zone for next 7 days:")
print(y_test[-7:])
print("Predicted electricity consumption in west zone for next 7 days:")
print(y_pred[-7:])
```

```
Actual electricity consumption in west zone for next 7 days:
[[ 947. ]
 [1158.1]
 [ 961.5]
 [1184.7]
 [1130.7]
 [1025.1]
 [1053.8]]
Predicted electricity consumption in west zone for next 7 days:
[[ 953.434]
 [1156.141]
 [ 973.408]
 [1179.544]
 [1122.596]
 [1028.801]
 [1051.731]]
```

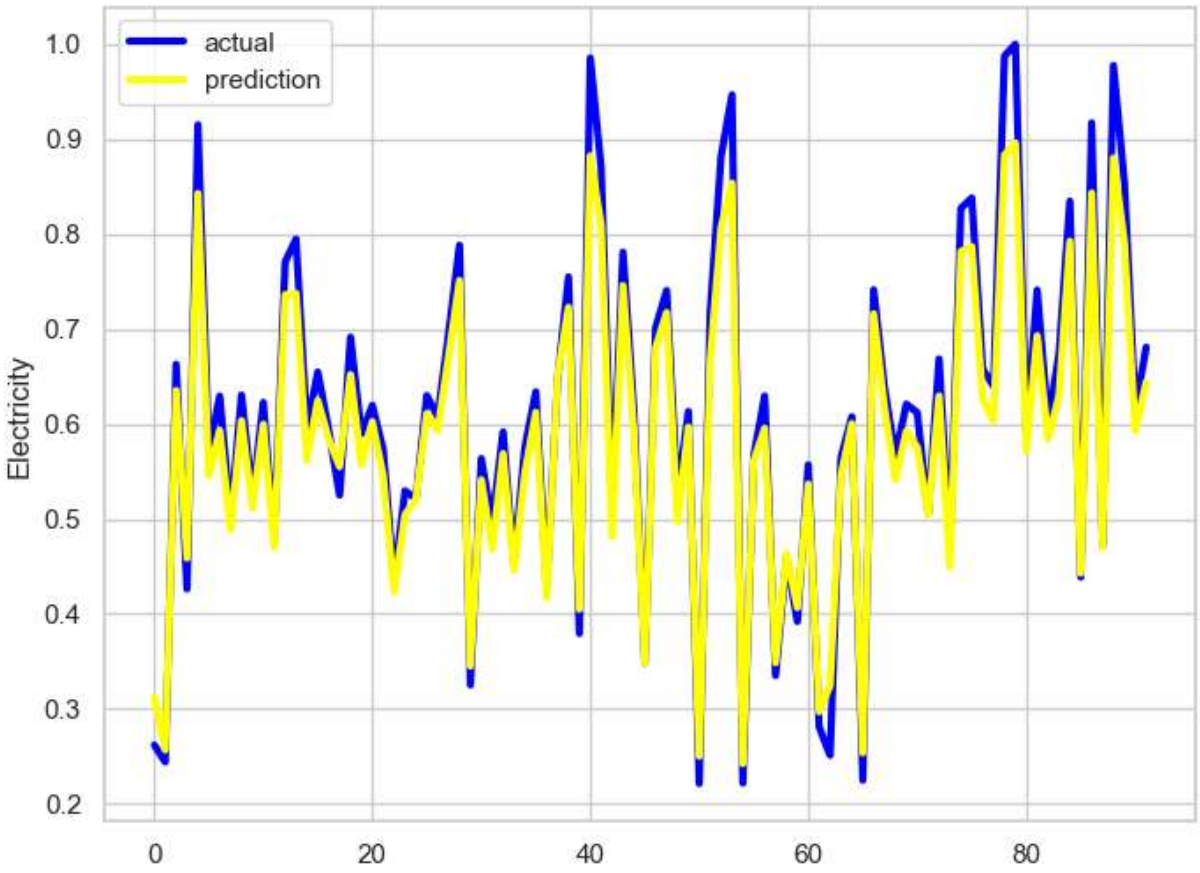
In [23]:
```
# print accuracy and mean absolute error of the model
from sklearn.metrics import mean_absolute_error, r2_score
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred))
```

```
print("Accuracy:", r2_score(y_test, y_pred))
#storing the accuracy in a variable
accuracy_random_forest = r2_score(y_test, y_pred)
```

```
Mean Absolute Error: 7.777293478260935
Accuracy: 0.9751096775733488
```

In [24]:

```
# using support vector regressor to predict the electricity in west zone
from sklearn.svm import SVR

# split the data into train and test
X_train, X_test, y_train, y_test = train_test_split(
    state_data, west_zone, test_size=0.2, random_state=42)

# fit the model
model = SVR()
model.fit(X_train, y_train)

# predict the electricity in west zone
y_pred = model.predict(X_test)

# plot the actual and predicted electricity in west zone
plt.figure(figsize=(8, 6))
plt.plot(y_test, label='actual', color='blue', linewidth=3)
plt.plot(y_pred, label='prediction', color='yellow', linewidth=3)
plt.ylabel('Electricity')
plt.legend()
plt.show()
```



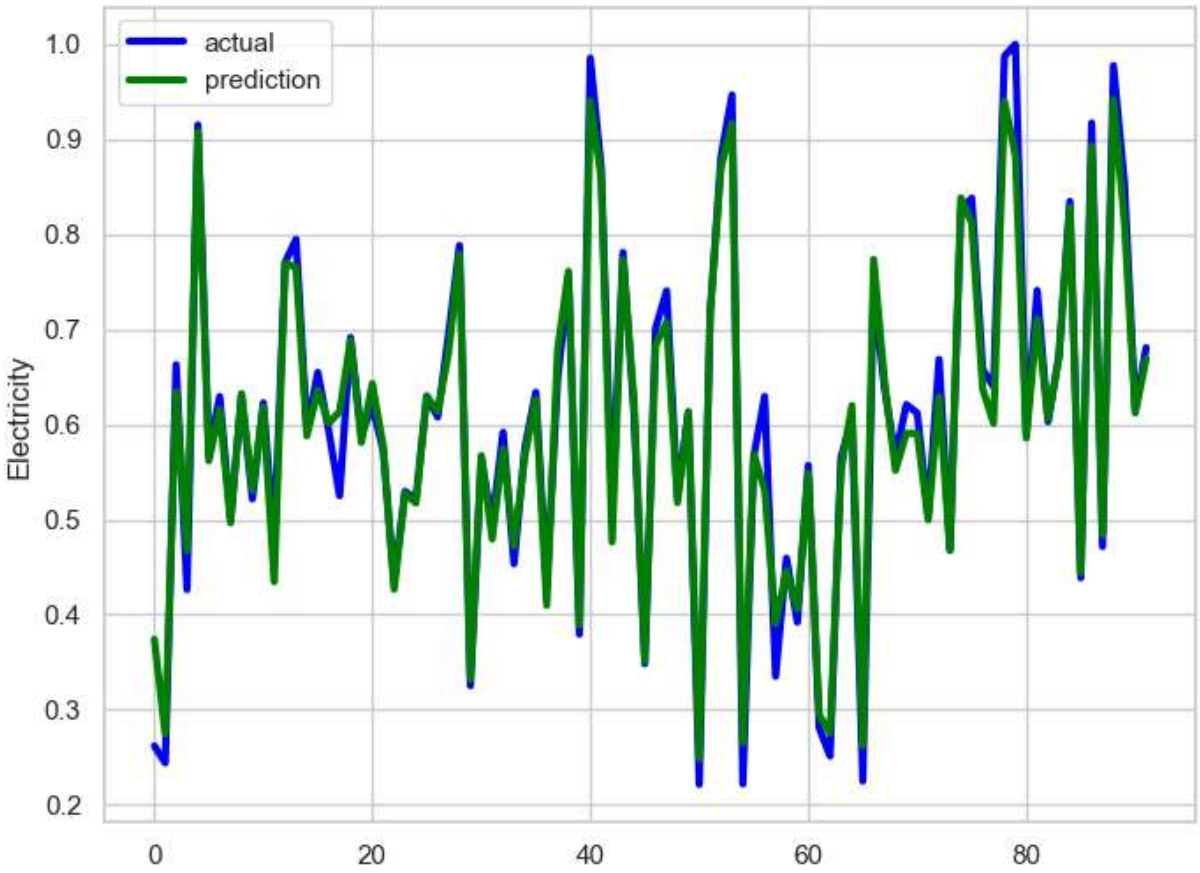
In [25]:
```
#print accuracy and mean absolute error of the model
from sklearn.metrics import mean_absolute_error, r2_score
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred))
print("Accuracy:", r2_score(y_test, y_pred))
#storing the accuracy in a variable
accuracy_support_vector = r2_score(y_test, y_pred)
```

```
Mean Absolute Error: 0.031400179490163375
Accuracy: 0.9540707769937778
```

In [26]:
```
#comparing the accuracy of all the models
print("Accuracy of Decision Tree Regressor:", accuracy_decision_tree)
print("Accuracy of Random Forest Regressor:", accuracy_random_forest)
print("Accuracy of Support Vector Regressor:", accuracy_support_vector)
```

```
Accuracy of Decision Tree Regressor: 0.9389762032349372
Accuracy of Random Forest Regressor: 0.9751096775733488
Accuracy of Support Vector Regressor: 0.9540707769937778
```

In [27]:
```
#ensemble learning
from sklearn.ensemble import VotingRegressor

# split the data into train and test
X_train, X_test, y_train, y_test = train_test_split(
    state_data, west_zone, test_size=0.2, random_state=42)

# fit the model
model = VotingRegressor([('dt', DecisionTreeRegressor()), ('rf', RandomForestRegres
model.fit(X_train, y_train)

# predict the electricity in west zone
y_pred = model.predict(X_test)

# plot the actual and predicted electricity in west zone
plt.figure(figsize=(8, 6))
plt.plot(y_test, label='actual', color='blue', linewidth=3)
plt.plot(y_pred, label='prediction', color='green', linewidth=3)
plt.ylabel('Electricity')
plt.legend()
plt.show()
```



In [28]:
```
#print accuracy and mean absolute error of the model
from sklearn.metrics import mean_absolute_error, r2_score
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred))
print("Accuracy:", r2_score(y_test, y_pred))
```

```
Mean Absolute Error: 0.020285334724298144
Accuracy: 0.9727213054565446
```

Output: **(paste output screen & graphs plotted)**

FAQs:

1) What is a Supervised learning?
2) Difference between Supervised and unsupervised learning.
3) List out any two algorithm of Supervised and unsupervised learning.
4) Give one example of reinforcement learning.

Conclusion:

Machine learning algorithm was studied and the implementation was performed using Colab laboratory.

Page No.
Date

DL Lab 1

FAQ

Q1 What is Supervised Learning?

Ans - Supervised learning is a type of machine learning where an algorithm learns to make predictions or decisions based on labeled data. In supervised learning, the algorithm is provided with dataset that includes input-output pairs. where the input data is associated with corresponding output labels or target values. The goal ~~of~~ is for the algorithm to learn a mapping from input data to the output labels so that it can make accurate predictions on new, unseen data.

Q2. Difference between Supervised and Unsupervised learning.

Ans - Supervised learning :-

- In supervised learning, the algorithm is provided with labelled training data, which means it has access to both input and output label corresponding to it.
- Supervised learning requires human intervention to label the training data, which can be time-consuming and costly.

Unsupervised learning:

- In unsupervised learning, the algorithm is given input data without explicit output labels or targets.
- The goal is to discover hidden patterns, structures, or relationships within data, such as clustering similar data points together or reducing the dimensionality of the data.

Q3. List out any two algorithm of Supervised and Unsupervised learning.

Ans - Supervised Learning Algorithms:

1. Linear Regression: Used for regression tasks to predict a continuous numerical output.
2. Random Forest: Used for both classification and regression tasks employing an ensemble of decision trees to make predictions.

- Unsupervised Learning Algorithms:
1. K-Means Clustering: Used for clustering data points into groups based on similarity.
2. Principal Component Analysis (PCA): Used for dimensionality reduction by finding the most important features in the data.

Q4. Give one example of reinforcement learning.

Ans- Training a computer program to play a game, such as chess.
- The program (agent) interacts with a game board (environment).
- It takes actions (moves) based on the current state of the game.
- The agent receives rewards or penalties based on its actions and outcome of the game (winning, losing or drawing).

4/2/2023