Practical 1:

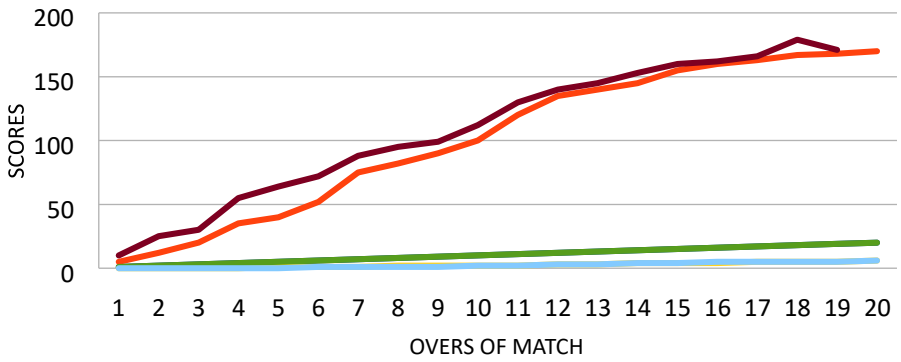
Title: Predict the performance of cricket team using historical day in MS-Excel.

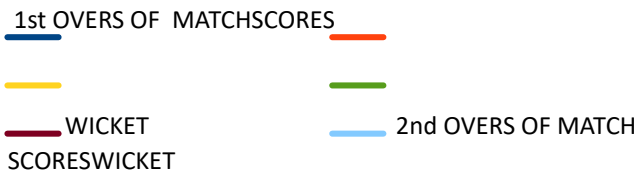
Name : Kalpesh Patil

Roll.no : 03

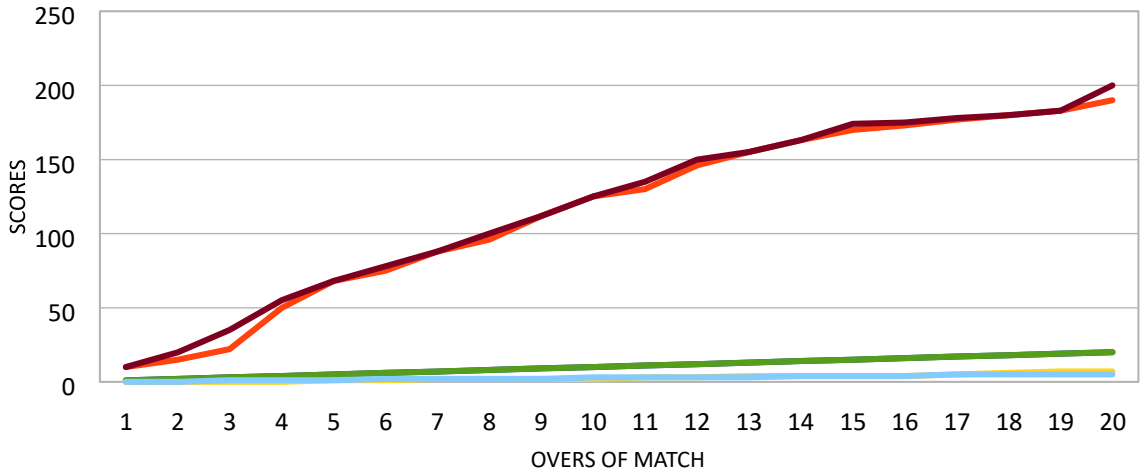
Batch :T1

1st OVERS OF MATCH	SCORES	WICKET		2nd OVERS OF MATCH	SCORES	WICKET
1	5	0	1	10	0	
2	12	0	2	25	0	
3	20	0	3	30	0	
4	35	0	4	55	0	
5	40	1	5	64	0	
6	52	1	6	72	1	
7	75	1	7	88	1	
8	82	2	8	95	1	
9	90	2	9	99	1	
10	100	2	10	112	2	
11	120	2	11	130	2	
12	135	3	12	140	3	
13	140	3	13	145	3	
14	145	4	14	153	4	
15	155	4	15	160	4	
16	160	4	16	162	5	
17	163	5	17	166	5	
18	167	5	18	179	5	
19	168	5	19	171	5	
20	170	6	20		6	





1ST OVERS OF MATCH	SCORES	SCORES	2nd OVERS OF MATCH	WICKET	WICKET
1	10	0	1	10	0
2	15	0	2	20	0
3	22	0	3	35	1
4	50	0	4	55	1
5	68	1	5	68	1
6	75	1	6	78	2
7	88	2	7	88	2
8	96	2	8	100	2
9	112	2	9	112	2
10	125	2	10	125	3
11	130	3	11	135	3
12	146	3	12	150	3
13	155	4	13	155	3
14	163	4	14	163	4
15	170	4	15	174	4
16	173	4	16	175	4
17	177	5	17	178	5
18	180	6	18	180	5
19	183	7	19	183	5
20	190	7	20	200	5



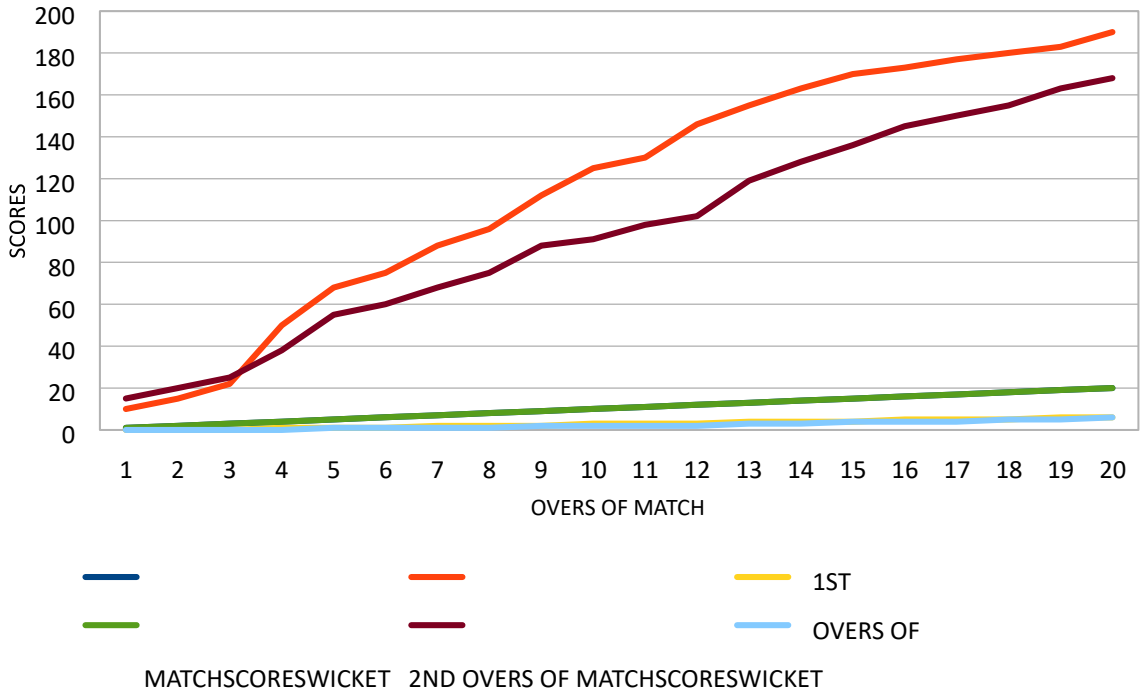
1ST OVERS OF MATCH SCORESSCORES

WICKET SCORESWICKET

2nd OVERS OF MATCH

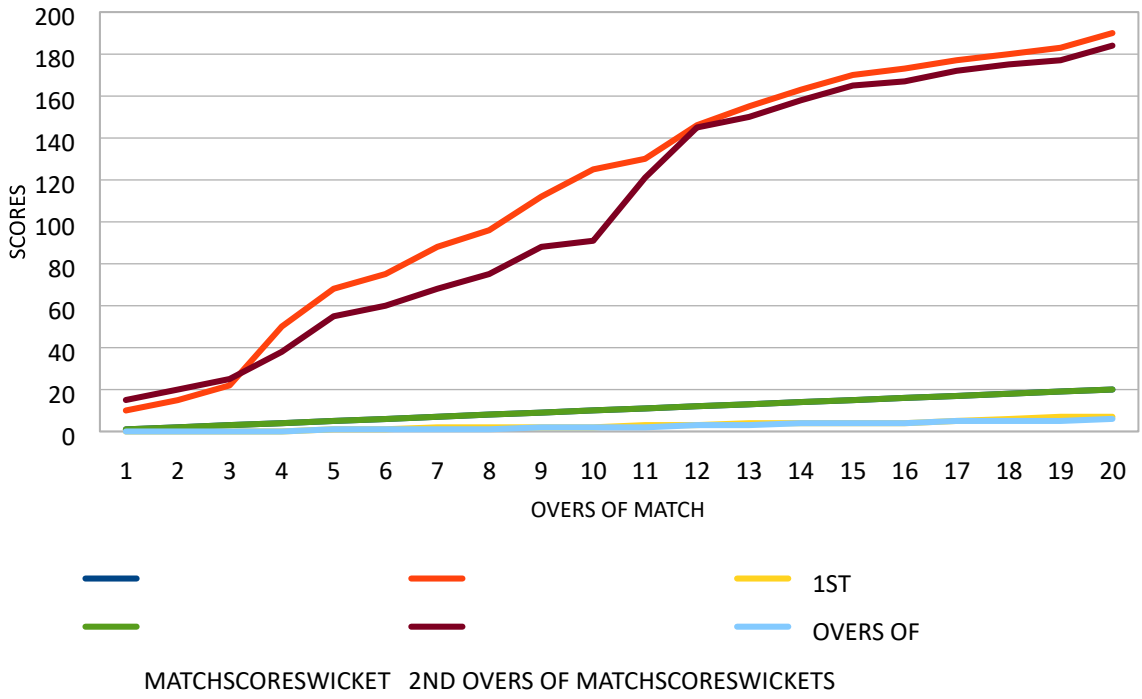
1ST OVERS OF MATCH	SCORES	WICKET	2ND OVERS OF MATCH	SCORES	WICKET
1	10	0	1	15	0
2	15	0	2	20	0
3	22	0	3	25	0
4	50	1	4	38	0
5	68	1	5	55	1
6	75	1	6	60	1
7	88	2	7	68	1
8	96	2	8	75	1
9	112	2	9	88	2
10	125	3	10	91	2
11	130	3	11	98	2
12	146	3	12	102	2
13	155	4	13	119	3
14	163	4	14	128	3
15	170	4	15	136	4
16	173	5	16	145	4
17	177	5	17	150	4

18	180	5	18	155	5
19	183	6	19	163	5
20	190	6	20	168	6



1ST OVERS OF MATCH		SCORES		WICKET		2ND OVERS OF MATCH		SCORES	WICKETS
1	10	0	1	15	0				
2	15	0	2	20	0				
3	22	0	3	25	0				
4	50	0	4	38	0				
5	68	1	5	55	1				
6	75	1	6	60	1				
7	88	2	7	68	1				
8	96	2	8	75	1				
9	112	2	9	88	2				
10	125	2	10	91	2				
11	130	3	11	121	2				
12	146	3	12	145	3				

13	155	4	13	150	3
14	163	4	14	158	4
15	170	4	15	165	4
16	173	4	16	167	4
17	177	5	17	172	5
18	180	6	18	175	5
19	183	7	19	177	5
20	190	7	20	184	6



Practical 3

Title:Using linear regression predict the value of continuos data.

Name : Kalpesh Patil

Roll.no : 03

Batch :T1

```
import numpy as np import
matplotlib.pyplot as plt #
Sample data points
x = np.array([2, 4, 6, 8])  # Independent variable (Hours Studied) y
= np.array([3, 7, 5, 10])  # Dependent variable (Exam Score)
# Perform linear regression using NumPy's polyfit (degree 1 for linear
regression) m, c = np.polyfit(x, y, 1) # Print the results print(f"Slope
(m): {m}") print(f"Intercept (c): {c}")
# Predict y values using the regression line y_pred
= m * x + c
# Print the original data points and the predicted values
print("\nOriginal and Predicted Values:") for xi, yi,
y_pred_val in zip(x, y, y_pred):
    print(f"x: {xi}, y (original): {yi}, y (predicted): {y_pred_val}")
# Plotting the data points and the regression line
plt.scatter(x, y, color='blue', label='Original Data')  # Plot original data points
plt.plot(x, y_pred, color='red', label='Regression Line')  # Plot the regression line
plt.xlabel('X values (Hours Studied)') plt.ylabel('Y
values (Exam Score)')
plt.title('Linear Regression: Predicted vs Original')
plt.legend() plt.grid(True) plt.show() output:
```

Slope (m): 0.9500000000000001 Intercept

(c): 1.4999999999999998 Original and

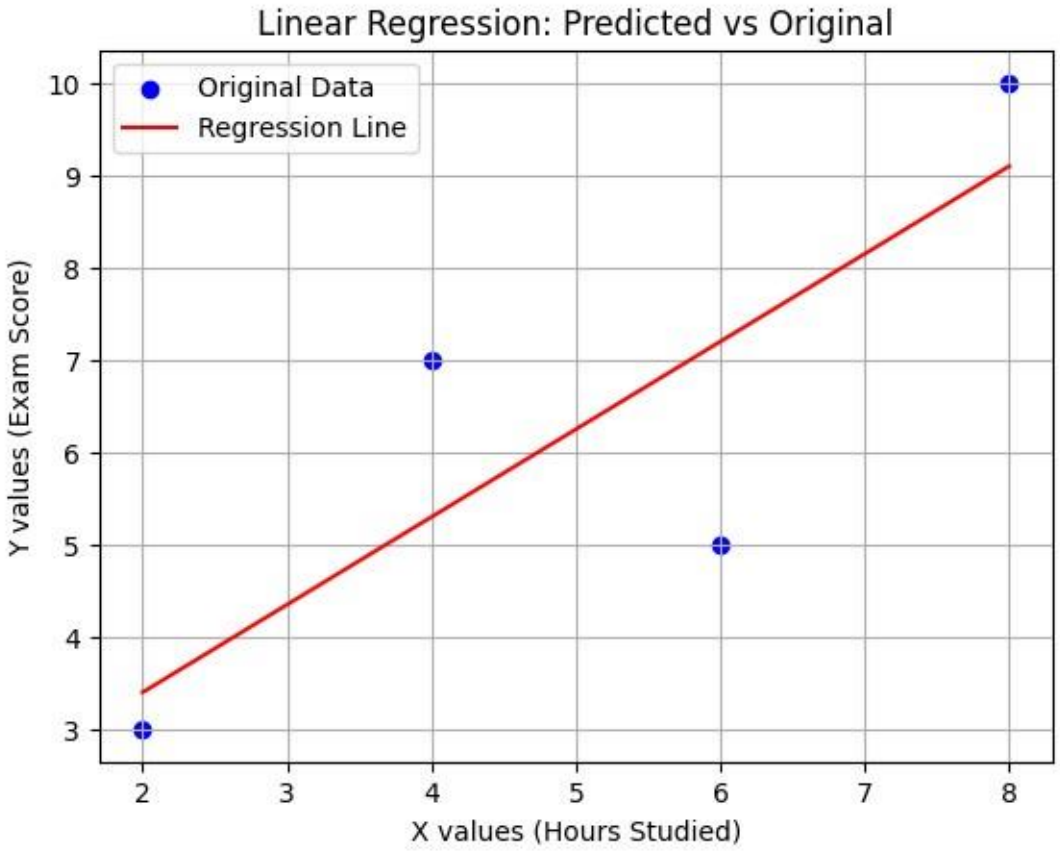
Predicted Values:

x: 2, y (original): 3, y (predicted): 3.4 x:

4, y (original): 7, y (predicted): 5.3 x:

6, y (original): 5, y (predicted): 7.2 x:

8, y (original): 10, y (predicted): 9.1



Practical 4

Title: Using Chi-Square analysis to predict the hypothesis.

Name : Kalpesh Patil

Roll.no : 03

Batch :T1

```
import numpy as np # Import necessary
library from scipy.stats import
chi2_contingency # Observed data in
contingency table format observed = [[20,
15, 25],  # Male preferences
        [10, 25, 15]]  # Female preferences #
Perform the chi-square test
chi2_stat, p_val, dof, expected = chi2_contingency(observed)

# Output the results
print("Chi-Square Statistic:", chi2_stat)
print("p-value:", p_val) print("Degrees
of Freedom:", dof) print("Expected
Frequencies:") print(expected)

# Set significance level alpha
= 0.05

# Interpretation of the result
if p_val < alpha:
   print("Reject the null hypothesis: There is a significant association.") else:
   print("Fail to reject the null hypothesis: No significant association.")
```

OUTPUT:

Chi-Square Statistic: 7.48611111111111 p-value:

0.023681631925797347

Degrees of Freedom: 2 Expected

Frequencies:

[[16.36363636 21.81818182 21.81818182]

[13.63636364 18.18181818 18.18181818]]

Reject the null hypothesis: There is a significant association.

Practical 5

Title: Predict the email spam or ham (not spam) using classification algorithm.

Name : Kalpesh Patil

Roll.no : 03

Batch :T1

```
from sklearn.model_selection import train_test_split from
sklearn.feature_extraction.text import CountVectorizer from
sklearn.naive_bayes import MultinomialNB from
sklearn.metrics import accuracy_score, confusion_matrix,
classification_report import numpy as np
# Sample email data (increase dataset size for better training) emails
= [
    "Free money win now", "Meeting at 10am", "Win a lottery now!",
    "Normal message without spam", "Get cash prizes", "Important project
update",
    "Win big prizes here", "Lunch meeting tomorrow", "You won a gift", "Weekly
report due"
]
labels = [1, 0, 1, 0, 1, 0, 1, 0, 1, 0]  # 1 = spam, 0 = ham
# Convert text data to feature vectors
vectorizer = CountVectorizer() X =
vectorizer.fit_transform(emails) y =
labels
# Split the data into training and testing sets (use a larger test size if possible)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
random_state=42)
```

```
# Train the Naive Bayes model
model = MultinomialNB()
model.fit(X_train, y_train) #
Make predictions y_pred =
model.predict(X_test) #
Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred)) print("Confusion
Matrix:\n", confusion_matrix(y_test, y_pred)) print("Classification
Report:\n", classification_report(y_test, y_pred, zero_division=0))
```

OUTPUT:

Accuracy: 0.75

Confusion Matrix:

[[2 0]

[1 1]]

Classification Report:

	precision	recall	f1-score	support
0	0.67	1.00	0.80	2
1	1.00	0.50	0.67	2
accuracy			0.75	4
macro avg	0.83	0.75	0.73	4
weighted avg	0.83	0.75	0.73	4

Practical 6

Title: Predict the credit worthiness of customer/credit

card fraud detection. (Use Kaggle credit card data)

Name : Kalpesh Patil

Roll.no : 03

Batch :T1

```
# Step 1: Unzip the dataset (if not already done)
!unzip 'archive (2).zip'  # Replace with your correct zip file name if different
# Step 2: Load the dataset using Pandas import
pandas as pd
# Replace 'credit_card_data.csv' with the actual name of your extracted file
data = pd.read_csv('credit_card_data.csv') print("First few rows of the
dataset:") print(data.head())
# Step 3: Basic data inspection
print("\nColumn names:") print(data.columns)
# Sample 10% of the data for faster testing (adjust fraction as needed)
data_sampled = data.sample(frac=0.1, random_state=42)
# Separate features and target X =
data_sampled.drop('Class', axis=1) y =
data_sampled['Class']
# Split the data into training and testing sets from
sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)


# Train a Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train) # Make predictions on the
test set y_pred = model.predict(X_test) # Evaluate the
model
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report print("\nModel Evaluation:")
print("Accuracy:", accuracy_score(y_test, y_pred)) print("Confusion
Matrix:\n", confusion_matrix(y_test, y_pred)) print("Classification
Report:\n", classification_report(y_test, y_pred))
```

OUTPUT:

```
Archive:  archive (2).zip
inflating: credit_card_data.csv
First few rows of the dataset:
   Time       V1       V2       V3       V4       V5       V6       V7  \
0   0.0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599
1   0.0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803
2   1.0 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461
3   1.0 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609
4   2.0 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941
        V8       V9  ...       V21       V22       V23       V24       V25  \
0  0.098698  0.363787  ... -0.018307  0.277838 -0.110474  0.066928  0.128539
```

```
1  0.085102 -0.255425  ... -0.225775 -0.638672  0.101288 -0.339846  0.167170    2
   0.247676 -1.514654  ...  0.247998  0.771679  0.909412 -0.689281 -0.327642
3  0.377436 -1.387024  ... -0.108300  0.005274 -0.190321 -1.175575  0.647376    4
-0.270533  0.817739  ... -0.009431  0.798278 -0.137458  0.141267 -0.206010


       V26       V27       V28  Amount  Class
0  -0.189115  0.133558 -0.021053  149.62      0
1  0.125895 -0.008983  0.014724    2.69      0
2  -0.139097 -0.055353 -0.059752  378.66      0
3  -0.221929  0.062723  0.061458  123.50      0
4  0.502292  0.219422  0.215153   69.99      0
[5 rows x 31 columns]
Column names:
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
      dtype='object')
Model Evaluation:
Accuracy: 0.9990637799882972
Confusion Matrix:
 [[8532    6]
 [   2    5]]
Classification Report:
                   precision    recall  f1-score   support


         0                1.00      1.00      1.00      8538
1                 0.45      0.71      0.56         7
accuracy                                   1.00      8545
```

macro avg	0.73	0.86	0.78	8545
weighted avg	1.00	1.00	1.00	8545

Practical 7

Title: Predict the sentiment from social media or customer review.

Name : Kalpesh Patil

Roll.no : 03

Batch :T1

```
# Install textblob if it's not installed yet
!pip install textblob # Importing
TextBlob library from textblob
import TextBlob # Function to
predict sentiment def
predict_sentiment(text):
    """
    Predict sentiment of a given text using TextBlob.
    Parameters:
    text (str): The input text or review to analyze.
    Returns:
    str: The predicted sentiment ('Positive', 'Negative', or 'Neutral').
    """
     analysis = TextBlob(text)     polarity =
analysis.sentiment.polarity     if
polarity > 0:
        return "Positive"
elif polarity < 0:
return "Negative"
else:
        return "Neutral"
```

```
# Example reviews reviews
= [
    "I love this product! It's amazing.",
    "The service was terrible and very disappointing.",
    "It was okay, nothing special but not bad either."]
# Predict sentiment for each review for
review in reviews:
    sentiment = predict_sentiment(review)
    print(f"Review: '{review}' \nPredicted Sentiment: {sentiment}\n")
```

OUTPUT:

```
Requirement already satisfied: textblob in
/usr/local/lib/python3.10/distpackages (0.17.1)
Requirement already satisfied: nltk>=3.1 in
/usr/local/lib/python3.10/distpackages (from textblob) (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages
(from nltk>=3.1->textblob) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages
(from nltk>=3.1->textblob) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in
/usr/local/lib/python3.10/dist-packages (from nltk>=3.1->textblob) (2024.9.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from nltk>=3.1->textblob) (4.66.6)
Review: 'I love this product! It's amazing.'
Predicted Sentiment: Positive
Review: 'The service was terrible and very disappointing.'
Predicted Sentiment: Negative
Review: 'It was okay, nothing special but not bad either.'
Predicted Sentiment: Positive
```

Practical 8

Title: Predict the future energy consumption for household or industry based on past data.

Name : Kalpesh Patil

Roll.no : 03

Batch :T1

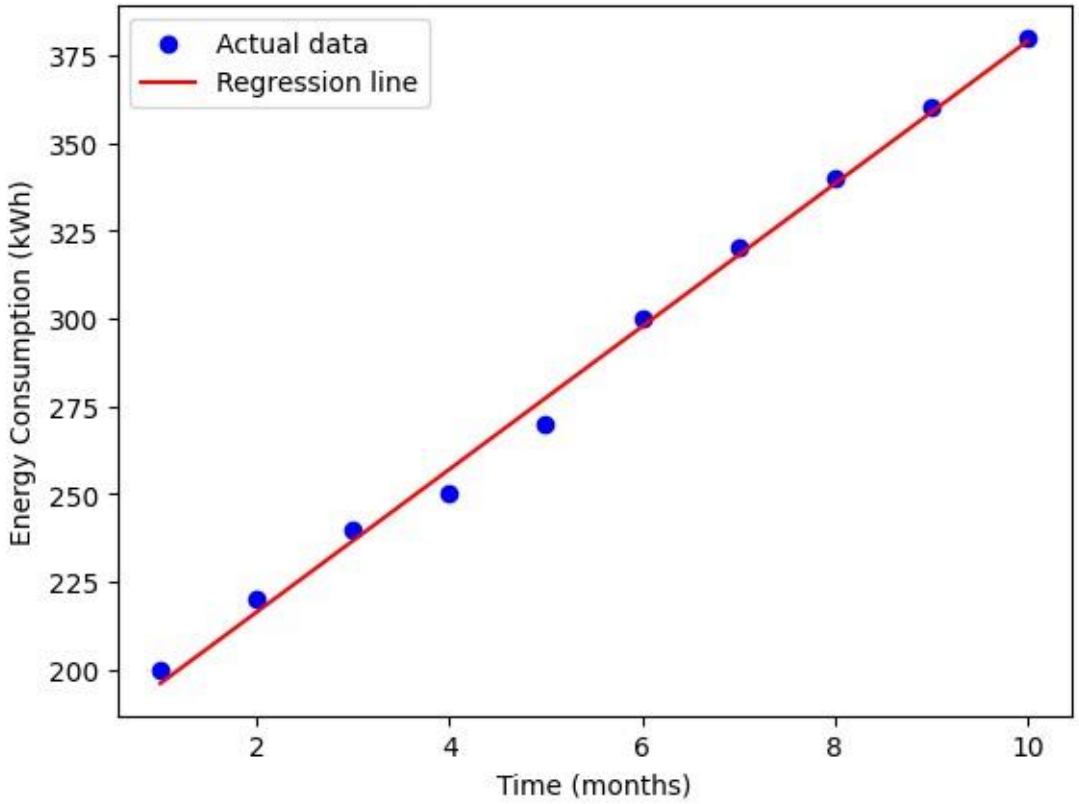
```
# Importing required libraries

import numpy as np import

pandas as pd

from sklearn.model_selection import train_test_split from

sklearn.linear_model import LinearRegression import

matplotlib.pyplot as plt

# Sample data for energy consumption (past data)

# Here, 'time' could represent months or years, and 'consumption' is the

energy consumed in kWh data = {

   'time': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],  # Time (e.g., months or years)

   'consumption': [200, 220, 240, 250, 270, 300, 320, 340, 360, 380]  # Energy
Consumption in kWh

}

# Converting to DataFrame df

= pd.DataFrame(data)

# Define feature (X) and target (y) X

= df[['time']]  # Time as the feature

y = df['consumption']  # Energy consumption as the target


# Split 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)

# Create and train the linear regression model

model = LinearRegression() model.fit(X_train,

y_train)


# Predict future consumption (for example, predicting for the next 2 months)

future_months = np.array([11, 12]).reshape(-1, 1) predictions =

model.predict(future_months)


# Output the predictions for month, prediction in

zip(future_months.flatten(), predictions):

   print(f"Predicted energy consumption for month {month}: {prediction:.2f}
kWh")


# Plotting the data and the regression line plt.scatter(X,

y, color='blue', label='Actual data')

plt.plot(X, model.predict(X), color='red', label='Regression line')

plt.xlabel('Time (months)') plt.ylabel('Energy Consumption

(kWh)') plt.legend() plt.show()
```

OUTPUT:

Predicted energy consumption for month 11: 399.40 kWh Predicted

energy consumption for month 12: 419.74 kWh


Actual data
Regression line
Energy Consumption (kWh)
375
350
325
300
275
250
225
200
2
4
6
8
10
Time (months)