# Project Report:
# Electricity Consumption Prediction & Billing Analysis

**Submitted to:**

Dr. Ramsha Karampuri sir

**Group Members:**

Tarun Gangadhar (BT20EEE107)

Dheeraj Reddy Pasham (BT20EEE083)

Tadepalli Sai Tanooj (BT20EEE106)

**Abstract:**

This project focuses on the prediction of electricity consumption and the subsequent calculation of electricity bills using machine learning models, specifically linear and polynomial regression. By analyzing historical data collected from approximately 190 quarters over 12 years at a college, the study aims to forecast monthly electricity usage and derive cost estimations. The data underwent rigorous preprocessing, including cleaning and formatting, to ensure accuracy and reliability of the models. The application of regression analysis facilitated the understanding of consumption patterns and contributed to more precise predictions. The outcomes not only demonstrate the potential of machine learning in resource management but also offer a scalable approach for similar settings. Ultimately, this project highlights how data-driven insights can lead to effective decision-making in energy management and financial planning.

**Introduction**

The project revolves around leveraging historical electricity consumption data to predict future usage and compute bills. This involves collecting data, preparing it through various preprocessing steps, and applying machine learning models to achieve accurate predictions.

**Data Collection:**

Data was meticulously gathered from the Maintenance Department of our college, encompassing detailed records of monthly electricity consumption across approximately 190 quarters spanning 12 years(2012 to 2023) monthly-wise. This extensive dataset forms the backbone of our predictive analysis, offering a comprehensive overview of energy usage patterns crucial for building robust predictive models.

**Data Formatting and Preparation:**

The raw data was sourced from 144 meticulously maintained Excel files, each representing a month's data over multiple years. The preparation involved structuring this data into a cohesive format suitable for analysis. Key attributes extracted and formatted include:

- **Quarter Number**: Identifier for the specific quarter.
- **Meter Number**: Unique identifier for the electricity meter.
- **Current Meter Reading**: Latest recorded reading of the meter.
- **Previous Meter Reading**: The meter reading from the previous month.
- **Units Consumed**: Calculated by subtracting the previous reading from the current reading, indicating the total electricity used in that quarter for the month.

The data was consolidated into a single dataset, ensuring consistency across various files to facilitate accurate analysis.

The Final Dataset was reduced to two columns: Quarter number and Units consumed.

| QtrNo | Total |
|---|---|
| C-28 | 95 |
| 42 | 149 |
| TA-30 | 0 |
| 54 | 147 |
| 62 | 145 |
| 29 | 0 |
| 43 | 145 |
| M-9 | 94 |
| M-10 | 62 |
| M-16 | 86 |
| M-25 | 80 |
| M-26 | 97 |
| 31 | 219 |
| NP-3 | 20713 |

**Fig1:** Arrangement of Excel Columns in our Dataset.

**Data Cleaning:**

The integrity and accuracy of our dataset are paramount. The following steps were taken to clean the data:

1. **Null Values Removal**: Any records missing essential data points were removed to maintain the quality and accuracy of our analyses.
2. **Duplicate Values**: Duplicate entries were identified and removed, retaining only the first instance to ensure each entry's uniqueness.
3. **Outliers**: Extreme values that deviated significantly from the norm were identified and removed to prevent them from skewing our results.

```python
import pandas as pd

# Load the dataset
df = pd.read_excel('sep2018.xlsx')

# Removing null values
df.dropna(inplace=True)

# Removing duplicate values
df.drop_duplicates(subset=['Quarter Number'], keep='first', inplace=True)

# Outlier detection and removal
Q1 = df['Units Consumed'].quantile(0.25)
Q3 = df['Units Consumed'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filtering out the outliers
df = df[(df['Units Consumed'] >= lower_bound) & (df['Units Consumed'] <= upper_bound)]
```

**Fig2:** Example code for Data cleaning using pandas for a month

The nomenclature of the excel files is in the format of mon2xxx.xlsx i.e month followed by the year.

## Methodology

### Machine Learning Overview and Regression Analysis

Machine learning is a subset of artificial intelligence that empowers systems to learn from data, identify patterns, and make decisions with minimal human intervention. In the realm of machine learning, models are trained to predict outcomes based on historical data, thereby enabling the automation of various decision-making processes. This approach is particularly potent in fields where patterns and relationships between variables need to be discovered and leveraged to predict future events or behaviors.

### Regression Analysis in Machine Learning

Regression analysis is a powerful statistical method used within machine learning to model the relationship between a dependent (target) variable and one or more independent (predictor) variables. The goal is to understand how the typical value of the dependent variable changes when any one of the independent variables are varied, while the other independent variables are heldfixed. Primarily, regression analysis is used for prediction and forecasting, where its use has substantial overlap with the field of machine learning.

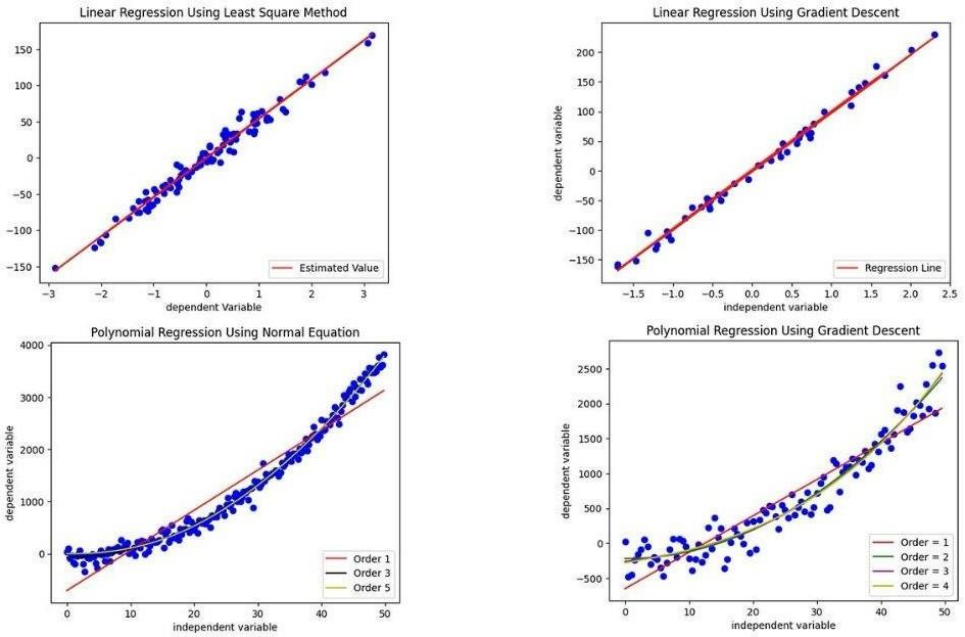



**Fig3:** Linear and Polynomial Regression

**1. Linear Regression:**

Linear regression is perhaps the simplest and most widely used statistical technique for predictive Modeling. It assumes a linear relationship between the dependent and independent variables, i.e., the line of best fit through the data points is a straight line. This method is not only straightforward to implement but also highly interpretable.

The primary advantage of linear regression is its simplicity and ease of understanding the output. The coefficients of the variables give the magnitude of impact each variable has on the target variable, assuming other variables remain constant. However, its major limitation is the assumption of linearity, which can be too simplistic for some real-world scenarios.

**2. Polynomial Regression:**

When data exhibits a non-linear relationship, polynomial regression becomes a useful technique. This type of regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y, and has been a substantial benefit in nonlinear analysis where the relationship between the independent and dependent variable is modelled as an nth degree polynomial.

Polynomial regression extends linear regression by adding extra predictors, obtained by raising each of the original predictors to a power. This method provides a good approximation of the relationship between the variables and allows the curve to bend accordingly to handle non-linear data more effectively.

**Practical Applications in Project**

In the context of predicting electricity consumption, linear regression can be utilized to establish a straightforward predictor of energy usage based on factors such as time of year, historical usage rates, and other relevant variables that have a direct, linear relationship with electricity consumption. On the other hand, polynomial regression can handle fluctuations and seasonal variations more effectively by capturing the non-linear patterns of electricity usage during different months or under varying weather conditions.

Both methods have their places in predictive analytics in energy management. Linear regression could serve as a starting point for predicting trends based on linear patterns, whereas polynomial regression could refine these predictions by accommodating seasonal shifts and other non-linear influences. By

leveraging both linear and polynomial regression models, this project aims to enhance prediction accuracy and provide more reliable decision-making tools for energy management strategies.

**Implementation**

Using Python and libraries like pandas and scikit-learn, the model was trained on the cleaned dataset. Google Colab was used to facilitate the development and testing of the model.

**Collab Link:**
**https://colab.research.google.com/drive/1g3GlgxGLW9j3E2TuRjtquVlO-iMJb4St?usp=sharing**

**Drive link for Data:**

**https://drive.google.com/drive/folders/1SlhUWyPduaJGGSdn2RD7lU1GzJ4hA4I3?usp=sharing**

**Instructions to Run the code:**

1. Copy the Data folder in the drive link given to the Google Drive.
2. Mount the Google Drive with Google collab
3. Run the code in snippets.
4. If any error occurs, restart the runtime.
5. Please check the quarter numbers and give the exact name of the quarter number as in the excel file.

**Code and Output:**

Following is the collab-GitHub gist where a sample quarter is being trained and the results are shown: -

Open in Colab

CONNECTING TO THE GOOGLE DRIVE

In [3]:
```
from google.colab import drive
drive.mount('/content/drive')

import os
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import numpy as np
import statsmodels.api as sm
import statsmodels.tsa.api as smt
import warnings
# Change this to the location of your data folder on Google Drive
data_folder_path = '/content/drive/My Drive/Data'
```

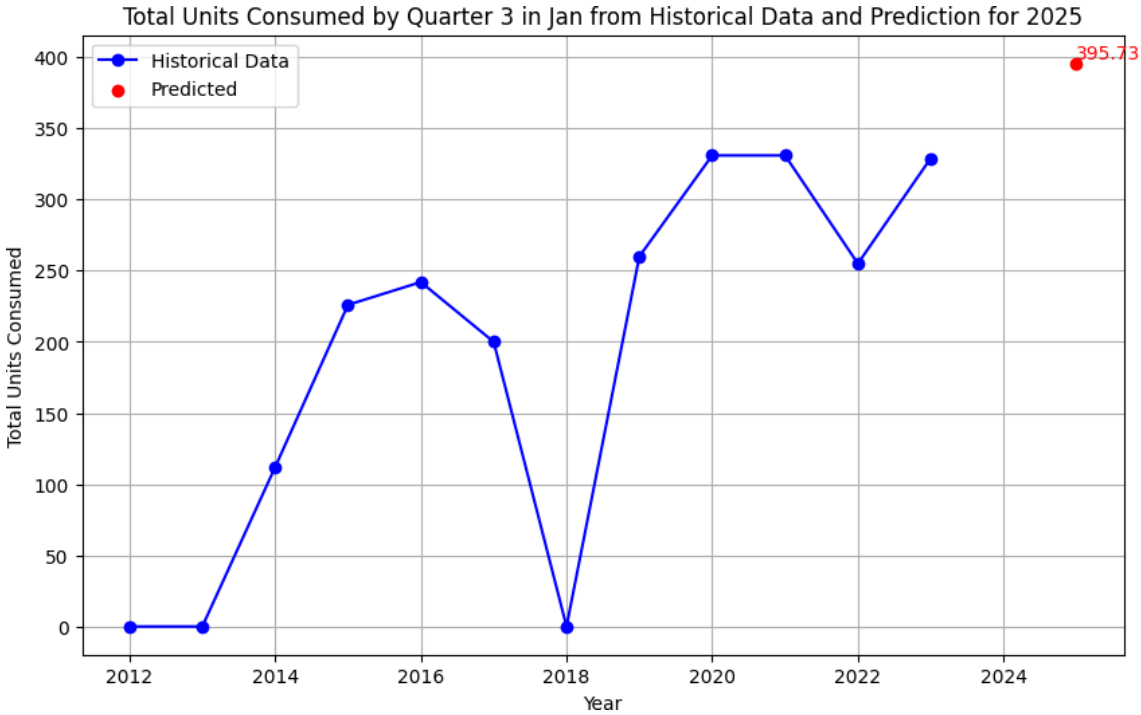
```
Mounted at /content/drive
```

In [4]:
```
import os
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import numpy as np

# Change this to the location of your data folder on Google Drive
data_folder_path = '/content/drive/My Drive/Data'

# User inputs
quarter_name = input("Enter the quarter name/number: ")
month = input("Enter the month (e.g., Jan, Feb, etc.): ")
year = int(input("Enter the year you want to predict for: "))

def prepare_dataset(data_folder_path, quarter_name, month):
    df_list = []
    for file_name in os.listdir(data_folder_path):
        if file_name.lower().startswith(month.lower()):
            year_part = file_name.rstrip('.xlsx')[-4:]
            try:
                year = int(year_part)
            except ValueError:
                print(f"Could not extract year from file name: {file_name}")
                continue

            file_path = os.path.join(data_folder_path, file_name)
            df = pd.read_excel(file_path)

            try:
                df_filtered = df[df['QtrNo'] == int(quarter_name)]
            except ValueError:
                df_filtered = df[df['QtrNo'].astype(str) == quarter_name]

            if not df_filtered.empty:
                df_filtered = df_filtered.copy()
                df_filtered.loc[:, 'Year'] = year
                df_list.append(df_filtered)
```

```
    if df_list:
        combined_df = pd.concat(df_list)
        combined_df.reset_index(drop=True, inplace=True)
        return combined_df
    else:
        return pd.DataFrame()

dataset = prepare_dataset(data_folder_path, quarter_name, month)

def train_and_predict(dataset, prediction_year):
    if not dataset.empty:
        X = dataset[['Year']]
        y = dataset['Total']
        model = LinearRegression()
        model.fit(X, y)
        prediction_input_df = pd.DataFrame([[prediction_year]], columns=['Yea
        prediction = model.predict(prediction_input_df)[0]
        return prediction
    else:
        return "No data available for the specified quarter and month."

# Training the model and predicting for the user-specified year
prediction = train_and_predict(dataset, year)
print(f"Predicted total units consumed for quarter {quarter_name} in {month}



import matplotlib.pyplot as plt

def plot_data_with_prediction(dataset, prediction_year, prediction_value, qua
    # Ensure the dataset is sorted by Year for proper plotting
    dataset = dataset.sort_values(by='Year')

    plt.figure(figsize=(10, 6))

    # Plot
    plt.plot(dataset['Year'], dataset['Total'], label='Historical Data', mark

    plt.scatter(prediction_year, prediction_value, color='red', label='Predic
    plt.text(prediction_year, prediction_value, f'{prediction_value:.2f}', co

    plt.title(f"Total Units Consumed by Quarter {quarter_name} in {month} fro
    plt.xlabel("Year")
    plt.ylabel("Total Units Consumed")
    plt.legend()
    plt.grid(True)
    plt.show()


# Assuming 'dataset' and 'prediction' from previous steps
if isinstance(prediction, (int, float)):
    plot_data_with_prediction(dataset, year, prediction, quarter_name, month)
else:
    print("Prediction was not generated due to lack of data.")



def calculate_electricity_bill(units):
    if units <= 100:
        unit_charge = 2.90
        fixed_charge = 70
```

```
    elif units <= 300:
        unit_charge = 4.85
        fixed_charge = 110
    elif units <= 500:
        unit_charge = 6.65
        fixed_charge = 110
    else:
        unit_charge = 7.80
        fixed_charge = 135

    energy_charges = units * unit_charge
    wheeling_charges = units * 1.57
    consumer_charges = energy_charges + fixed_charge + wheeling_charges
    gov_ed_charges = (16 * consumer_charges) / 100
    gov_tax = units * 0.2604

    total_bill = consumer_charges + gov_ed_charges + gov_tax
    return total_bill


# Example usage with the predicted units
total_bill = calculate_electricity_bill(prediction)
print(f"Bill: Rs. {total_bill:.2f}")
```

```
Enter the quarter name/number: 3
Enter the month (e.g., Jan, Feb, etc.): Jan
Enter the year you want to predict for: 2025
Predicted total units consumed for quarter 3 in Jan 2025: 395.72727272727207
```




```
Bill: Rs. 4003.99
```

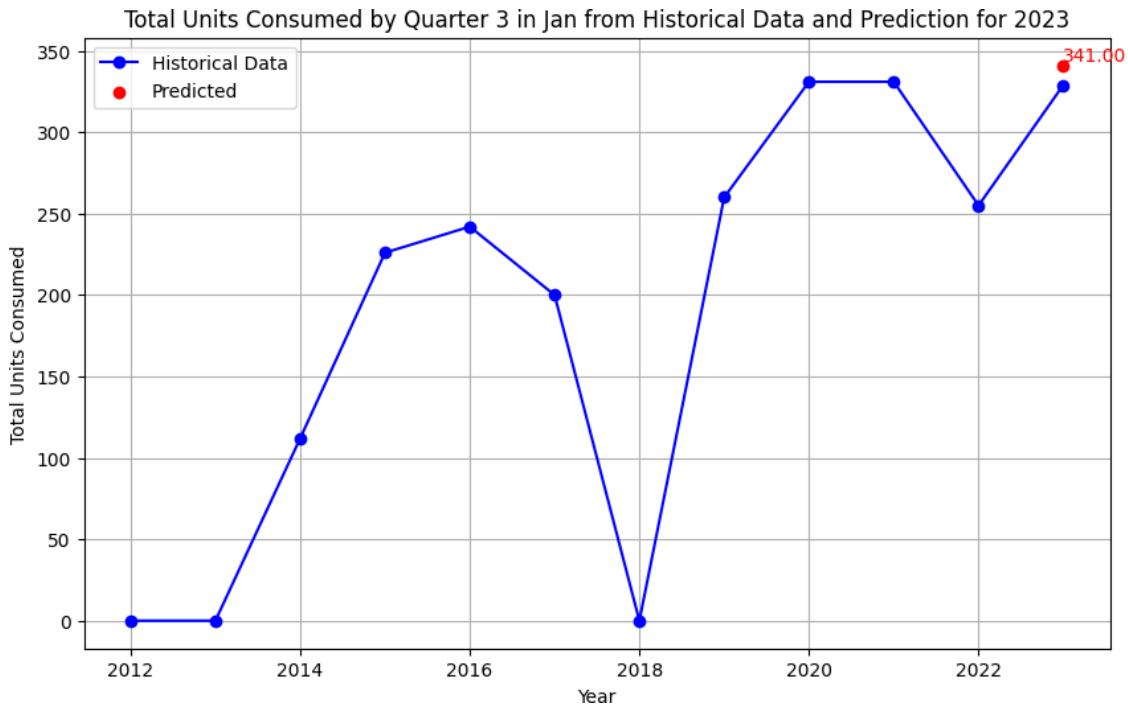
In [7]:

```
import os
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

```
import numpy as np

# Change this to the location of your data folder on Google Drive
data_folder_path = '/content/drive/My Drive/Data'

# User inputs
quarter_name = input("Enter the quarter name/number: ")
month = input("Enter the month (e.g., Jan, Feb, etc.): ")
year = int(input("Enter the year you want to predict for: "))

def prepare_dataset(data_folder_path, quarter_name, month):
    df_list = []
    for file_name in os.listdir(data_folder_path):
        if file_name.lower().startswith(month.lower()):
            year_part = file_name.rstrip('.xlsx')[-4:]
            try:
                year = int(year_part)
            except ValueError:
                print(f"Could not extract year from file name: {file_name}")
                continue

            file_path = os.path.join(data_folder_path, file_name)
            df = pd.read_excel(file_path)

            try:
                df_filtered = df[df['QtrNo'] == int(quarter_name)]
            except ValueError:
                df_filtered = df[df['QtrNo'].astype(str) == quarter_name]

            if not df_filtered.empty:
                df_filtered = df_filtered.copy()
                df_filtered.loc[:, 'Year'] = year
                df_list.append(df_filtered)

    if df_list:
        combined_df = pd.concat(df_list)
        combined_df.reset_index(drop=True, inplace=True)
        return combined_df
    else:
        return pd.DataFrame()

dataset = prepare_dataset(data_folder_path, quarter_name, month)

def train_and_predict(dataset, prediction_year):
    if not dataset.empty:
        X = dataset[['Year']]
        y = dataset['Total']
        model = LinearRegression()
        model.fit(X, y)
        prediction_input_df = pd.DataFrame([[prediction_year]], columns=['Yea
        prediction = model.predict(prediction_input_df)[0]
        return prediction
    else:
        return "No data available for the specified quarter and month."

# Training the model and predicting for the user-specified year
prediction = train_and_predict(dataset, year)
print(f"Predicted total units consumed for quarter {quarter_name} in {month}



import matplotlib pyplot as plt
```

```
def plot_data_with_prediction(dataset, prediction_year, prediction_value, qua
    # Ensure the dataset is sorted by Year for proper plotting
    dataset = dataset.sort_values(by='Year')

    plt.figure(figsize=(10, 6))

    # Plot
    plt.plot(dataset['Year'], dataset['Total'], label='Historical Data', mark

    plt.scatter(prediction_year, prediction_value, color='red', label='Predic
    plt.text(prediction_year, prediction_value, f'{prediction_value:.2f}', co

    plt.title(f"Total Units Consumed by Quarter {quarter_name} in {month} fro
    plt.xlabel("Year")
    plt.ylabel("Total Units Consumed")
    plt.legend()
    plt.grid(True)
    plt.show()


# Assuming 'dataset' and 'prediction' from previous steps
if isinstance(prediction, (int, float)):
    plot_data_with_prediction(dataset, year, prediction, quarter_name, month)
else:
    print("Prediction was not generated due to lack of data.")
```

```
Enter the quarter name/number: 3
Enter the month (e.g., Jan, Feb, etc.): Jan
Enter the year you want to predict for: 2023
Predicted total units consumed for quarter 3 in Jan 2023: 341.0
```



Actual: 329 units, predicted: 341 units Accuracy: 96.4 percent

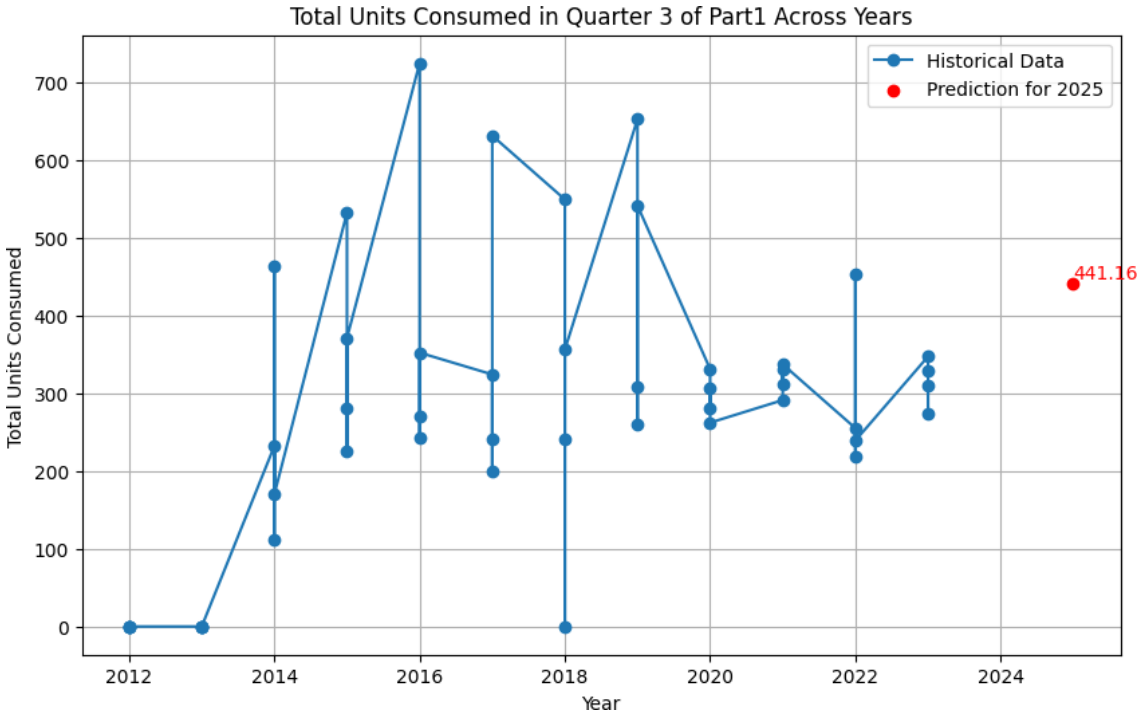
In [ ]:

```
import os
import pandas as pd
from sklearn.linear_model import LinearRegression
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm

# Change this to the location of your data folder on Google Drive
data_folder_path = '/content/drive/My Drive/Data'

# User inputs
part_name = input("Enter the part name (Part1, Part2, Part3): ")
quarter_number = int(input("Enter the quarter number : "))
year = int(input("Enter the year you want to predict for: "))

# Mapping parts to months
part_to_months = {
    'Part1': ['Jan', 'Feb', 'Mar', 'Apr'],
    'Part2': ['May', 'Jun', 'Jul', 'Aug'],
    'Part3': ['Sep', 'Oct', 'Nov', 'Dec']
}

def prepare_dataset(data_folder_path, part_name, quarter_number):
    df_list = []
    for file_name in os.listdir(data_folder_path):
        # Check if file name contains any month from the selected part
        if any(month.lower() in file_name.lower() for month in part_to_months
            year_part = file_name.rstrip('.xlsx')[-4:]
            try:
                year = int(year_part)
            except ValueError:
                print(f"Could not extract year from file name: {file_name}")
                continue

            file_path = os.path.join(data_folder_path, file_name)
            df = pd.read_excel(file_path)
            df = df[df['QtrNo'] == quarter_number]  # Filter data by the spec
            if not df.empty:
                df['Year'] = year
                df_list.append(df)

    if df_list:
        combined_df = pd.concat(df_list)
        combined_df.reset_index(drop=True, inplace=True)
        return combined_df
    else:
        return pd.DataFrame()

dataset = prepare_dataset(data_folder_path, part_name, quarter_number)

def train_and_predict(dataset, prediction_year):
    if not dataset.empty:
        X = dataset[['Year']]
        y = dataset['Total']
        model = LinearRegression()
        model.fit(X, y)
        prediction_input_df = pd.DataFrame([[prediction_year]], columns=['Yea
        prediction = model.predict(prediction_input_df)[0]
```

```
        return prediction
    else:
        return "No data available for the specified quarter and part."

prediction = train_and_predict(dataset, year)
print(f"Predicted total units consumed for quarter {quarter_number} in {part_

def calculate_electricity_bill(units):
    if units <= 100:
        unit_charge = 2.90
        fixed_charge = 70
    elif units <= 300:
        unit_charge = 4.85
        fixed_charge = 110
    elif units <= 500:
        unit_charge = 6.65
        fixed_charge = 110
    else:
        unit_charge = 7.80
        fixed_charge = 135

    energy_charges = units * unit_charge
    wheeling_charges = units * 1.57
    consumer_charges = energy_charges + fixed_charge + wheeling_charges
    gov_ed_charges = (16 * consumer_charges) / 100
    gov_tax = units * 0.2604

    total_bill = consumer_charges + gov_ed_charges + gov_tax
    return total_bill


# Example usage with the predicted units
total_bill = calculate_electricity_bill(prediction)
print(f"units: Rs. {total_bill:.2f}")


def plot_data_with_prediction(dataset, prediction_year, prediction_value, qua
    dataset = dataset.sort_values(by='Year')
    plt.figure(figsize=(10, 6))
    plt.plot(dataset['Year'], dataset['Total'], label='Historical Data', mark
    plt.scatter(prediction_year, prediction_value, color='red', label='Predic
    plt.text(prediction_year, prediction_value, f'{prediction_value:.2f}', co
    plt.title(f"Total Units Consumed in Quarter {quarter_number} of {part_nam
    plt.xlabel("Year")
    plt.ylabel("Total Units Consumed")
    plt.legend()
    plt.grid(True)
    plt.show()

if isinstance(prediction, (int, float)):
    plot_data_with_prediction(dataset, year, prediction, quarter_number, part
else:
    print("Prediction was not generated due to lack of data.")
```

```
Enter the part name (Part1, Part2, Part3): Part1
Enter the quarter number (1-4): 3
Enter the year you want to predict for: 2025
```

```
Predicted total units consumed for quarter 3 in Part1 of 2025: 441.162296037298
8
units: Rs. 4449.05
```



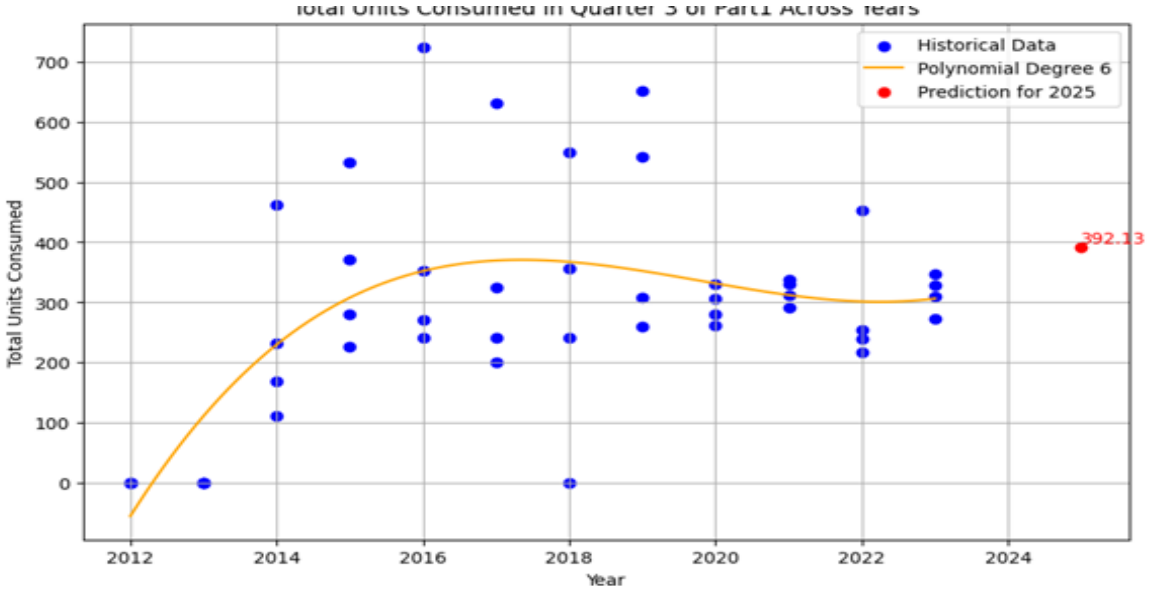

In [6]:

```python
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm

# Change this to the location of your data folder on Google Drive
data_folder_path = '/content/drive/My Drive/Data'

# User inputs
part_name = input("Enter the part name (Part1, Part2, Part3): ")
quarter_number = int(input("Enter the quarter number: "))
year = int(input("Enter the year you want to predict for: "))

# Mapping parts to months
part_to_months = {
    'Part1': ['Jan', 'Feb', 'Mar', 'Apr'],
    'Part2': ['May', 'Jun', 'Jul', 'Aug'],
    'Part3': ['Sep', 'Oct', 'Nov', 'Dec']
}

def prepare_dataset(data_folder_path, part_name, quarter_number):
    df_list = []
    for file_name in os.listdir(data_folder_path):
        if any(month.lower() in file_name.lower() for month in part_to_months
            year_part = file_name.rstrip('.xlsx')[-4:]
            try:
                year = int(year_part)
            except ValueError:
                print(f"Could not extract year from file name: {file_name}")
                continue

            file_path = os.path.join(data_folder_path, file_name)
```

```
            df = pd.read_excel(file_path)
            df = df[df['QtrNo'] == quarter_number]
            if not df.empty:
                df['Year'] = year
                df_list.append(df)

    if df_list:
        combined_df = pd.concat(df_list)
        combined_df.reset_index(drop=True, inplace=True)
        return combined_df
    else:
        return pd.DataFrame()

dataset = prepare_dataset(data_folder_path, part_name, quarter_number)

def train_and_predict_polynomial(dataset, prediction_year, degree=6):
    if not dataset.empty:
        X = dataset[['Year']].values
        y = dataset['Total'].values
        polynomial_features = PolynomialFeatures(degree=degree)
        polynomial_model = make_pipeline(polynomial_features, LinearRegressio
        polynomial_model.fit(X, y)
        prediction = polynomial_model.predict(np.array([[prediction_year]]))[
        return polynomial_model, prediction
    else:
        return None, "No data available for the specified quarter and part."

polynomial_model, prediction = train_and_predict_polynomial(dataset, year)

if polynomial_model:
    print(f"Predicted total units consumed for quarter {quarter_number} in {p

    def plot_data_with_polynomial_prediction(dataset, model, prediction_year,
        X = dataset['Year'].values.reshape(-1, 1)
        y = dataset['Total'].values
        plt.figure(figsize=(10, 6))
        plt.scatter(dataset['Year'], dataset['Total'], label='Historical Data
        # Generate a range of years for plotting the polynomial curve
        X_plot = np.linspace(X.min(), X.max(), 100).reshape(-1, 1)
        plt.plot(X_plot, model.predict(X_plot), color='orange', label=f'Polyn
        plt.scatter(prediction_year, prediction_value, color='red', label='Pr
        plt.text(prediction_year, prediction_value, f'{prediction_value:.2f}'
        plt.title(f"Total Units Consumed in Quarter {quarter_number} of {part
        plt.xlabel("Year")
        plt.ylabel("Total Units Consumed")
        plt.legend()
        plt.grid(True)
        plt.show()

    plot_data_with_polynomial_prediction(dataset, polynomial_model, year, pre
else:
    print(prediction)
```

```
Enter the part name (Part1, Part2, Part3): Part1
Enter the quarter number (1-4): 3
Enter the year you want to predict for: 2025
Predicted total units consumed for quarter 3 in Part1 of 2025: 392.133089721202
85
```

Total Units Consumed in Quarter 3 of Part1 Across Years




**Discussion**

The models were able to predict the electricity consumption with reasonable accuracy above 90 percentage on average. Predictions were visualized using matplotlib, highlighting trends and the effectiveness of the models.

The project demonstrated the potential of machine learning in predicting electricity consumption and effectively calculating bills, which could help in better energy management and cost estimation in real-world scenarios.

**Future Scope**

Monitoring the day wise data for a whole year and then predicting using this model would lead to high accurate results.

**Acknowledgements**

Sincere Thanks to Dr. Ramsha Karampuri Sir for providing us this opportunity to work on this innovative project.

Thanks to the college's Maintenance Department for providing the data necessary for this analysis.

**References**

1. Gross,G., Galiana, F.D, Short-Term Load Forecasting, Proceeding of IEEE, December 1987.
2. Bowerman, B.L., O'Connel, R.T., Forecasting and Time Series an Applied Approach, Duxbury, 3rd Ed, 1993.