

Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

Name: GOWTHAM.J

TITLE : (DATA PREPROCESSING)

1.Data preprocessing is a crucial step before building a machine learning model, as it ensures the data is clean, well-structured, and ready for analysis.

DESCRIPTION :

Data preprocessing is a fundamental step in the machine learning pipeline that involves transforming raw data into a clean, structured format suitable for building models. The quality and relevance of the data you feed into your model directly impact its performance, making data preprocessing crucial for achieving accurate and reliable results.

Key Steps in Data Preprocessing

1. Data Cleaning:

- **Handling Missing Values:** Real-world data often has missing values, which can lead to incorrect model predictions. Common techniques include:
 - **Removing Missing Data:** Dropping rows or columns with missing values if they are not significant.
 - **Imputing Missing Data:** Replacing missing values with a mean, median, mode, or using advanced methods like forward/backward filling.
- **Removing Duplicates:** Ensuring that no duplicate rows exist in the dataset, as they can bias the model.

2. Data Transformation:

- **Normalization/Standardization:** Scaling features to a similar range or distribution is crucial when the model assumes that all input features have the same scale (e.g., in algorithms like KNN, SVM).
 - **Normalization:** Scaling data to a range of [0, 1].
 - **Standardization:** Rescaling data to have a mean of 0 and a standard deviation of 1.
- **Encoding Categorical Variables:** Converting categorical data into numerical format, which is essential for algorithms that require numerical input.
 - **Label Encoding:** Assigning a unique integer to each category.
 - **One-Hot Encoding:** Creating binary columns for each category.

3. Feature Engineering:

Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

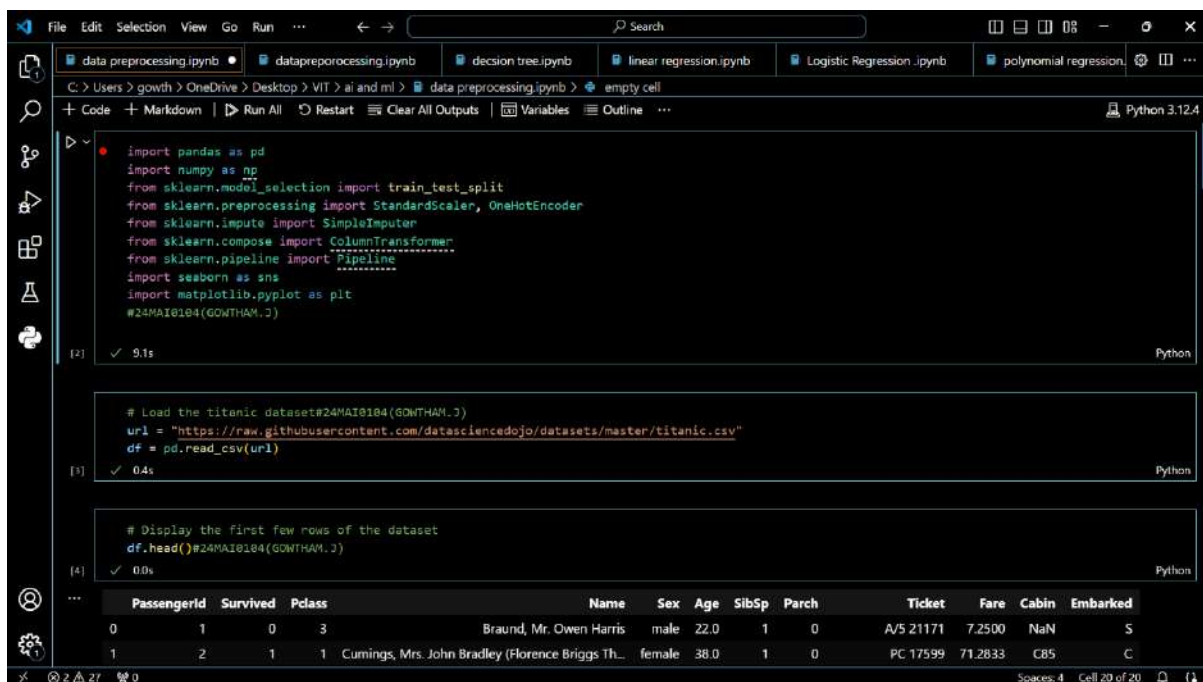
Name: GOWTHAM.J

- **Creating New Features:** Generating new features that might help the model better capture the underlying patterns in the data. For example, creating interaction terms, polynomial features, or aggregating data over time periods.
- **Feature Selection:** Choosing the most relevant features to improve model performance and reduce overfitting. Techniques include:
 - **Correlation Analysis:** Identifying and removing highly correlated features.
 - **Feature Importance:** Using models like Random Forest to identify the most important features.

4. Splitting the Dataset:

- **Training and Testing Split:** Dividing the dataset into training and testing sets to evaluate the model's performance. This step helps in understanding how the model generalizes to unseen data.
- **Cross-Validation:** A technique to ensure that the model's performance is robust and not overly dependent on a single train-test split.

SOURCE CODE AND RESULT :



```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
import seaborn as sns
import matplotlib.pyplot as plt
#24MAI0104(GOWTHAM.J)

# Load the titanic dataset#24MAI0104(GOWTHAM.J)
url = "https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv"
df = pd.read_csv(url)

# Display the first few rows of the dataset
df.head()
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

Name: GOWTHAM.J

```
File Edit Selection View Go Run ... Search
data preprocessing.ipynb | data preprocessing.ipynb | decision tree.ipynb | linear regression.ipynb | Logistic Regression .ipynb | polynomial regression...
C:\Users\gowth\OneDrive\Desktop\VIIT> ai and ml > data preprocessing.ipynb > empty cell
+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ... Python 3.12.4

[6] age_imputer=SimpleImputer(strategy='median')
     embarked_imputer=SimpleImputer(strategy='most_frequent')#24MAI0104(GOWTHAM.J)
✓ 0.0s Python

[7] df['Age']=age_imputer.fit_transform(df[['Age']])
     df['Embarked']=embarked_imputer.fit_transform(df[['Embarked']]).ravel()#24MAI0104(GOWTHAM.J)
✓ 0.0s Python

[8] df.drop(columns=['Cabin'],inplace=True)#24MAI0104(GOWTHAM.J)
✓ 0.0s Python

[9] print(df.isnull().sum())#24MAI0104(GOWTHAM.J)
✓ 0.0s Python

... PassengerId    0
     Survived      0
     Pclass       0
     Name         0
     Sex          0
     Age          0
     SibSp        0
     Parch        0
     Ticket       0
     Ticket       0
```

```
File Edit Selection View Go Run ... Search
data preprocessing.ipynb | data preprocessing.ipynb | decision tree.ipynb | linear regression.ipynb | Logistic Regression .ipynb | polynomial regression...
C:\Users\gowth\OneDrive\Desktop\VIIT> ai and ml > data preprocessing.ipynb > empty cell
+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ... Python 3.12.4

[4] df.isnull().sum()#24MAI0104(GOWTHAM.J)
✓ 0.0s Python

...
  PassengerId  Survived  Pclass
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3
   Name      Sex  Age  SibSp  Parch    Ticket   Fare Cabin Embarked
0  Braund, Mr. Owen Harris  male   22.0    1    0   A/5 21171   7.2500   NaN        S
1  Cumings, Mrs. John Bradley (Florence Briggs Th... female   38.0    1    0   PC 17599  71.2833   C85        C
2    Heikkinen, Miss. Laina  female   26.0    0    0  STON/O2. 3101282   7.9250   NaN        S
3  Futrelle, Mrs. Jacques Heath (Lily May Peel)  female   35.0    1    0   113803  53.1000  C123        S
4    Allen, Mr. William Henry  male   35.0    0    0   373450   8.0500   NaN        S

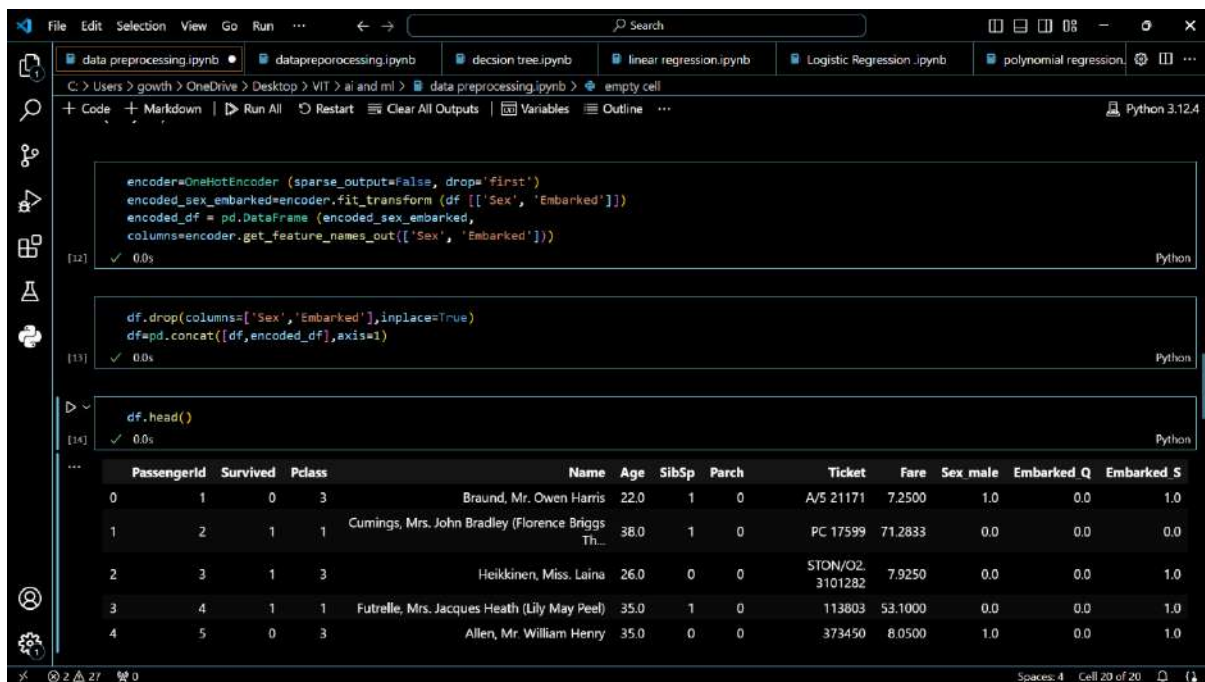
[5] df.isnull().sum()#24MAI0104(GOWTHAM.J)
✓ 0.0s Python

...
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age           177
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin         687
Embarked       2
dtype: int64
```

Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

Name: GOWTHAM.J



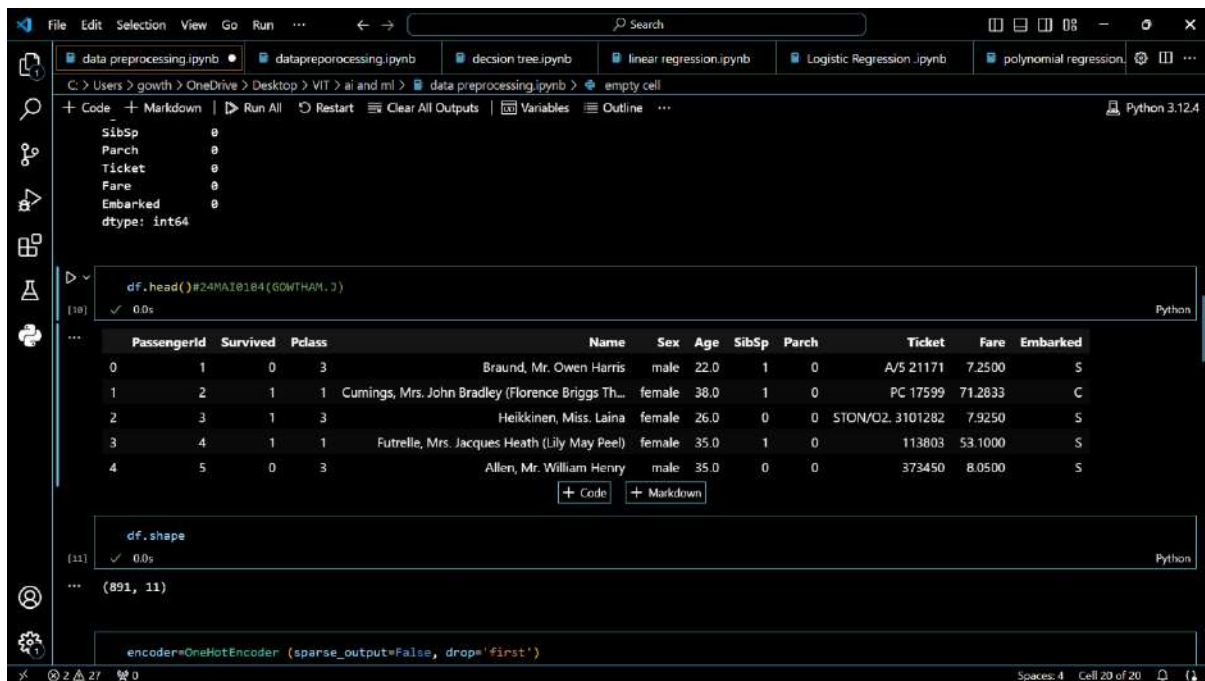
The screenshot shows a Jupyter Notebook with three cells. The first cell imports OneHotEncoder and performs a fit_transform on the 'Sex' and 'Embarked' columns. The second cell drops the original columns and concatenates the encoded features. The third cell displays the head of the resulting DataFrame.

```
encoder=OneHotEncoder (sparse_output=False, drop='first')
encoded_sex_embarked=encoder.fit_transform (df [['Sex', 'Embarked']])
encoded_df = pd.DataFrame (encoded_sex_embarked,
columns=encoder.get_feature_names_out(['Sex', 'Embarked']))
```

```
df.drop(columns=['Sex','Embarked'],inplace=True)
df=pd.concat([df,encoded_df],axis=1)
```

```
df.head()
```

PassengerId	Survived	Pclass	Name	Age	SibSp	Parch	Ticket	Fare	Sex male	Embarked Q	Embarked S	
0	1	0	3	Braund, Mr. Owen Harris	22.0	1	0	A/5 21171	7.2500	1.0	0.0	1.0
1	2	1	1	Cummings, Mrs. John Bradley (Florence Briggs Th...	38.0	1	0	PC 17599	71.2833	0.0	0.0	0.0
2	3	1	3	Heikkinen, Miss. Laina	26.0	0	0	STON/O2. 3101282	7.9250	0.0	0.0	1.0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	35.0	1	0	113803	53.1000	0.0	0.0	1.0
4	5	0	3	Allen, Mr. William Henry	35.0	0	0	373450	8.0500	1.0	0.0	1.0



The screenshot shows a Jupyter Notebook with two cells. The first cell prints the dtypes of the variables. The second cell displays the head of the DataFrame and its shape.

```
SibSp      0
Parch      0
Ticket     0
Fare       0
Embarked   0
dtype: int64
```

```
df.head()
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S
1	2	1	1	Cummings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S

```
df.shape
```

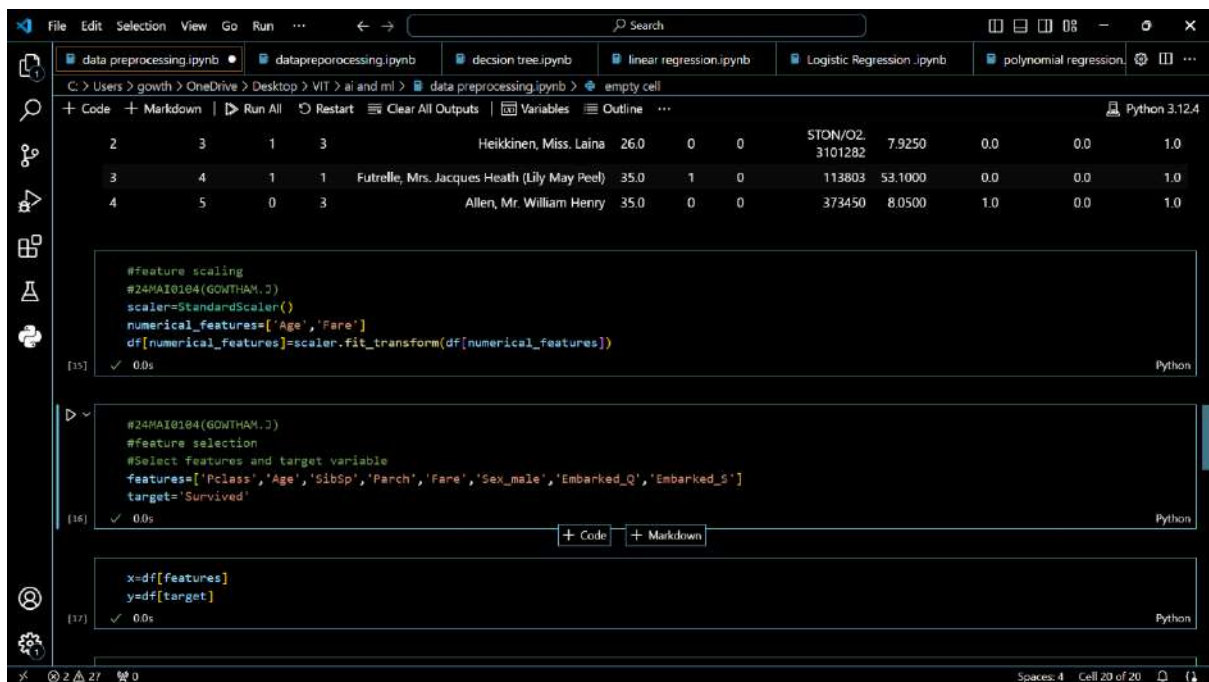
(891, 11)

```
encoder=OneHotEncoder (sparse_output=False, drop='first')
```

Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

Name: GOWTHAM.J



The image shows a Jupyter Notebook interface with the following content:

- File Explorer:** data preprocessing.ipynb, datapreprocessing.ipynb, decision tree.ipynb, linear regression.ipynb, Logistic Regression .ipynb, polynomial regression.
- Code Cell [15]:**

```
#feature scaling
#24MAI0104(GOWTHAM.J)
scaler=StandardScaler()
numerical_features=['Age','Fare']
df[numerical_features]=scaler.fit_transform(df[numerical_features])
```

Output: ✓ 0.0s
- Code Cell [16]:**

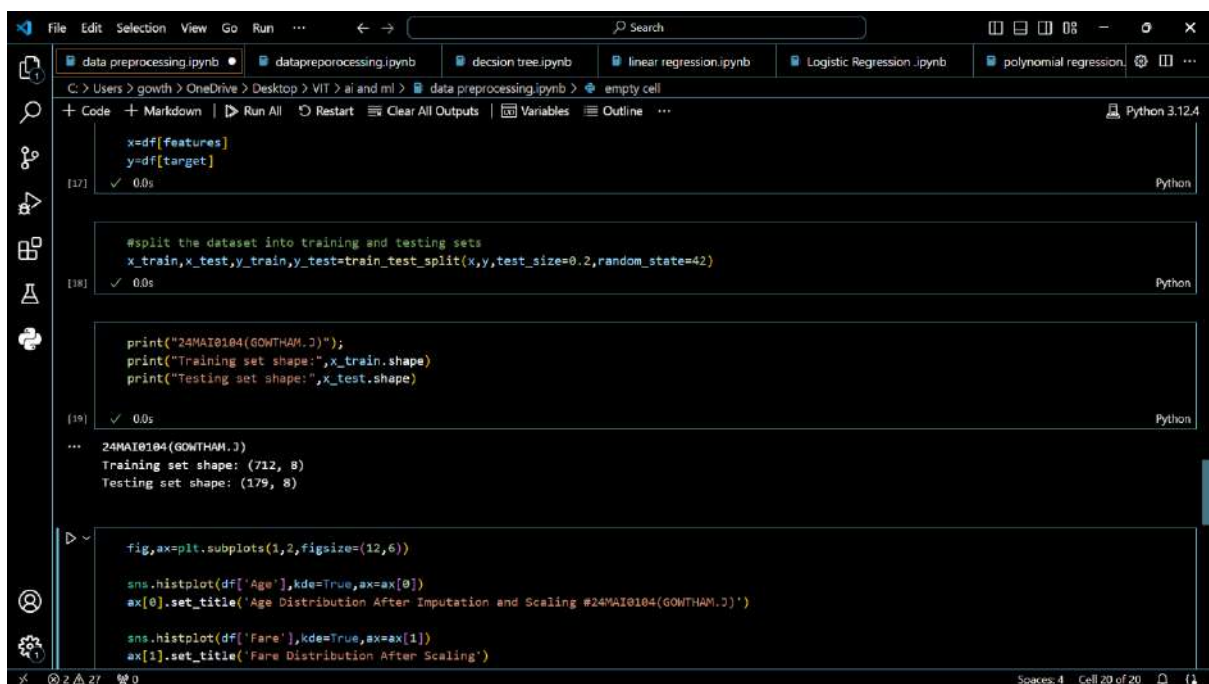
```
#24MAI0104(GOWTHAM.J)
#feature selection
#Select features and target variable
features=['Pclass','Age','SibSp','Parch','Fare','Sex_male','Embarked_Q','Embarked_S']
target='Survived'
```

Output: ✓ 0.0s
- Code Cell [17]:**

```
x=df[features]
y=df[target]
```

Output: ✓ 0.0s

The notebook is running on Python 3.12.4. The status bar at the bottom indicates 27 lines of code, 0 errors, and 4 spaces.



The image shows a Jupyter Notebook interface with the following content:

- Code Cell [17]:**

```
x=df[features]
y=df[target]
```

Output: ✓ 0.0s
- Code Cell [18]:**

```
#split the dataset 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)
```

Output: ✓ 0.0s
- Code Cell [19]:**

```
print("24MAI0104(GOWTHAM.J)");
print("Training set shape:",x_train.shape)
print("Testing set shape:",x_test.shape)
```

Output: ✓ 0.0s
- Code Cell [20]:**

```
fig,ax=plt.subplots(1,2,figsize=(12,6))

sns.histplot(df['Age'],kde=True,ax=ax[0])
ax[0].set_title('Age Distribution After Imputation and Scaling #24MAI0104(GOWTHAM.J)')

sns.histplot(df['Fare'],kde=True,ax=ax[1])
ax[1].set_title('Fare Distribution After Scaling')
```

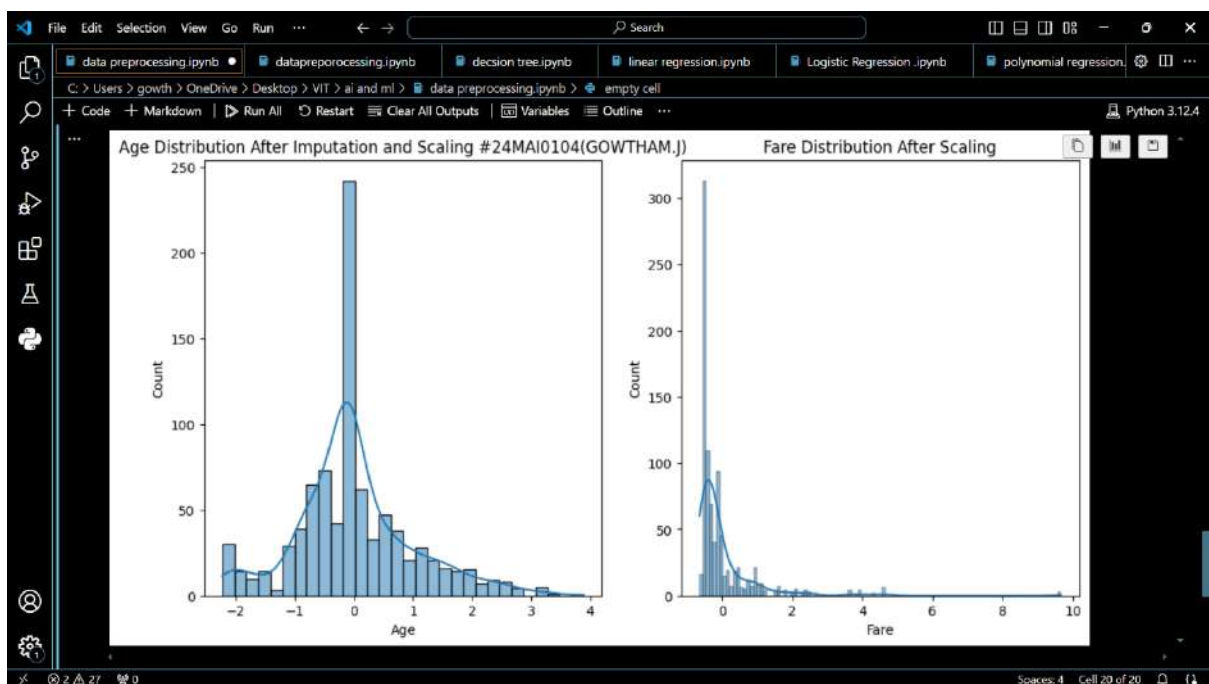
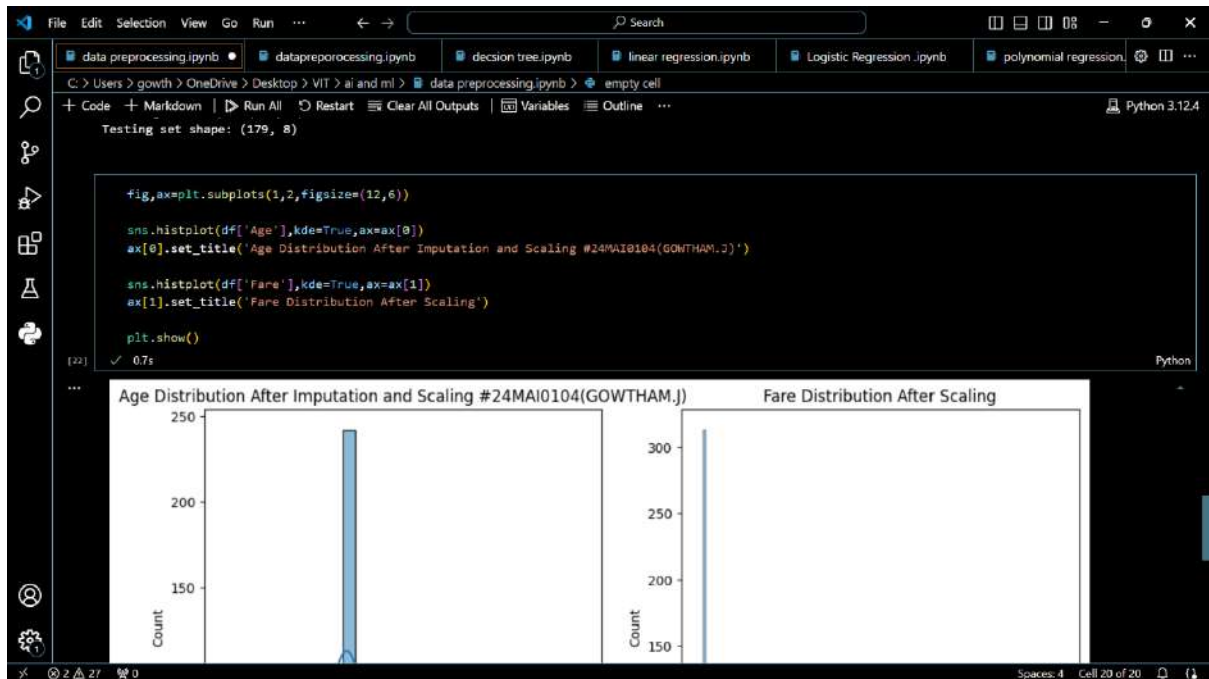
Output: 24MAI0104(GOWTHAM.J)
Training set shape: (712, 8)
Testing set shape: (179, 8)

The notebook is running on Python 3.12.4. The status bar at the bottom indicates 27 lines of code, 0 errors, and 4 spaces.

Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

Name: GOWTHAM.J



Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

Name: GOWTHAM.J

CONCLUSION :

Data preprocessing is essential for preparing data for machine learning:

1. Clean data by handling missing values, duplicates, and outliers.
2. Transform data by normalizing, encoding categorical variables, and engineering features.
3. Integrate data sources and reduce dimensionality.
4. Split data for training and testing to ensure model performance.
5. It improves model accuracy and reliability by optimizing data quality and structure.

Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

Name: GOWTHAM.J

TITLE (LINEAR REGRESSION MODEL) :

2. Predicting gold prices using the **LINEAR REGRESSION MODEL** involves data collection, preprocessing, model training, and evaluation. Below is a detailed example using Python and Scikit-learn to predict gold prices.

DESCRIPTION :

Linear Regression is one of the most fundamental and widely used algorithms in machine learning and statistics. It is a method for modeling the relationship between a dependent variable (often called the target or response variable) and one or more independent variables (also known as features or predictors).

In this project, we aim to predict the future prices of gold using a Linear Regression model. The main objective is to build a model that can effectively learn the relationship between historical gold prices and other related features to forecast the price of gold for the next day or a specified future date.

Steps Involved:

1. **Data Collection:** We will begin by gathering historical gold price data, which typically includes daily prices over a specific period. This data can be sourced from financial platforms like Yahoo Finance .
2. **Data Preprocessing:** The collected data may contain missing values, noise, or irrelevant information. We will clean the data by handling missing values, converting date formats, and selecting relevant features for the model.
3. **Feature Selection:** To predict the future price of gold, we will choose features that may influence gold prices. In a simple case, we might use the previous day's price as a predictor. For a more complex model, additional features like moving averages, economic indicators, or commodity prices may be considered.
4. **Splitting the Data:** The dataset will be split into training and testing sets. The training set will be used to train the model, while the testing set will help evaluate its performance.
5. **Model Training:** We will train a Linear Regression model on the training data. The model will learn the relationship between the selected features and the target variable (future gold price).
6. **Model Evaluation:** After training, the model's performance will be evaluated on the test set using metrics like Mean Squared Error (MSE) and R^2 score. These metrics will help us understand how well the model predicts future gold prices.

Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

Name: GOWTHAM.J

7. **Prediction and Visualization:** Finally, we will use the trained model to make predictions on unseen data. The actual vs. predicted prices will be visualized to assess the model's accuracy.

SOURCE CODE AND RESULT :

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import yfinance as yf #24MAI0104(GOWTHAM.J)

#Download historical gold price data #24MAI0104(GOWTHAM.J)
gold_data=yf.download('GC=F',start='2018-01-01', end='2024-06-01')

# display the few rows of the dataset
print(gold_data.tail()) #24MAI0104(GOWTHAM.J)
```

Date	Open	High	Low	Close	Adj Close
2024-05-24	2342.600098	2345.399902	2332.500000	2332.500000	2332.500000
2024-05-28	2336.899902	2359.699951	2336.899902	2355.199951	2355.199951
2024-05-29	2340.300049	2340.300049	2340.300049	2340.300049	2340.300049
2024-05-30	2336.899902	2349.500000	2320.800049	2342.899902	2342.899902
2024-05-31	2344.100098	2354.000000	2319.800000	2322.899902	2322.899902

```
pip install yfinance
```

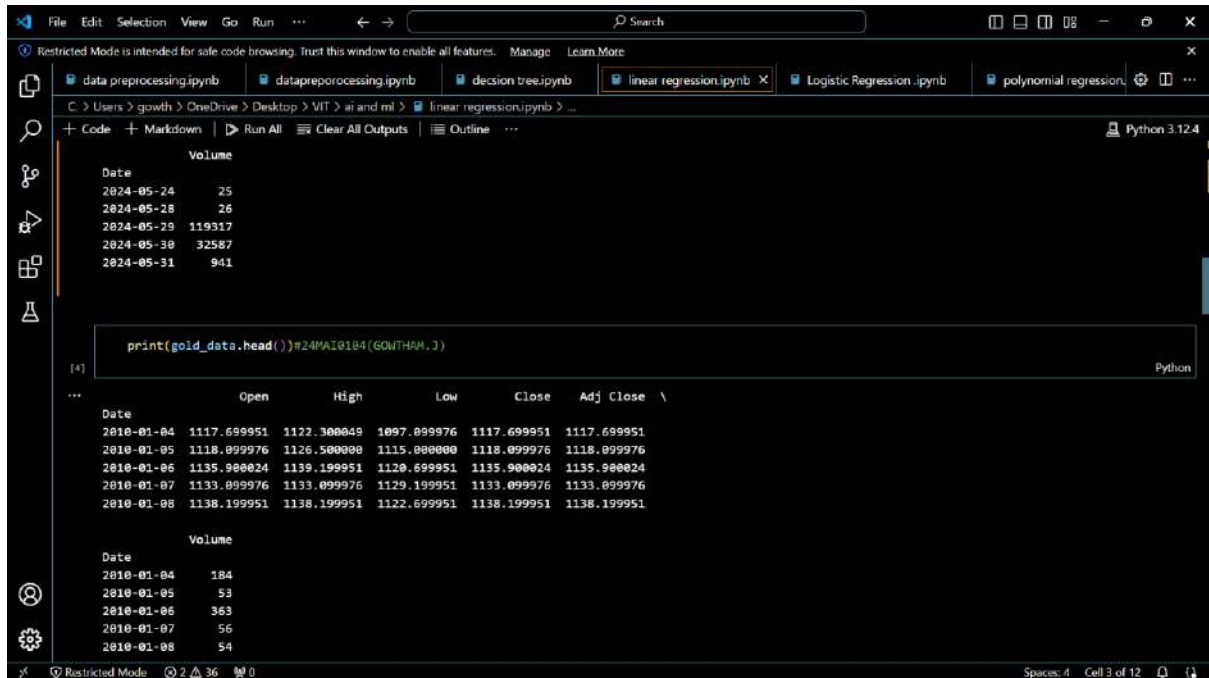
Requirement already satisfied: yfinance in c:\users\gowth\appdata\local\programs\python\python312\lib\site-packages (0.2.41)
Requirement already satisfied: pandas<=1.3.0 in c:\users\gowth\appdata\local\programs\python\python312\lib\site-packages (from yfinance) (2.2.2)
Requirement already satisfied: numpy<=1.16.5 in c:\users\gowth\appdata\local\programs\python\python312\lib\site-packages (from yfinance) (2.0.0)
Requirement already satisfied: requests>=2.31 in c:\users\gowth\appdata\local\programs\python\python312\lib\site-packages (from yfinance) (2.32.3)
Requirement already satisfied: multitasking>=0.0.7 in c:\users\gowth\appdata\local\programs\python\python312\lib\site-packages (from yfinance) (0.0.11)
Requirement already satisfied: lxml>=4.9.1 in c:\users\gowth\appdata\local\programs\python\python312\lib\site-packages (from yfinance) (5.2.2)
Requirement already satisfied: platformdirs>=2.0.0 in c:\users\gowth\appdata\local\programs\python\python312\lib\site-packages (from yfinance) (4.2.2)
Requirement already satisfied: pytz>=2022.5 in c:\users\gowth\appdata\local\programs\python\python312\lib\site-packages (from yfinance) (2024.1)
Requirement already satisfied: frozendict>=2.3.4 in c:\users\gowth\appdata\local\programs\python\python312\lib\site-packages (from yfinance) (2.4.4)
Requirement already satisfied: peewee>=3.16.2 in c:\users\gowth\appdata\local\programs\python\python312\lib\site-packages (from yfinance) (3.17.6)
Requirement already satisfied: beautifulsoup4>=4.11.1 in c:\users\gowth\appdata\local\programs\python\python312\lib\site-packages (from yfinance) (4.12)
Requirement already satisfied: html5lib>=1.1 in c:\users\gowth\appdata\local\programs\python\python312\lib\site-packages (from yfinance) (1.1)
Requirement already satisfied: soupsieve>=1.2 in c:\users\gowth\appdata\local\programs\python\python312\lib\site-packages (from beautifulsoup4>=4.11.1->yfinance) (1.1)
Requirement already satisfied: six>=1.9 in c:\users\gowth\appdata\local\programs\python\python312\lib\site-packages (from html5lib>=1.1->yfinance) (1.1)
Requirement already satisfied: webencodings in c:\users\gowth\appdata\local\programs\python\python312\lib\site-packages (from html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\gowth\appdata\local\programs\python\python312\lib\site-packages (from pandas>=1.3.0->yfinance) (2.8.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\gowth\appdata\local\programs\python\python312\lib\site-packages (from pandas>=1.3.0->yfinance) (2022.7)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\gowth\appdata\local\programs\python\python312\lib\site-packages (from requests>=2.31->yfinance) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\gowth\appdata\local\programs\python\python312\lib\site-packages (from requests>=2.31->yfinance) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\gowth\appdata\local\programs\python\python312\lib\site-packages (from requests>=2.31->yfinance) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\gowth\appdata\local\programs\python\python312\lib\site-packages (from requests>=2.31->yfinance) (2024.6.20)
Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 24.0 -> 24.2

Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

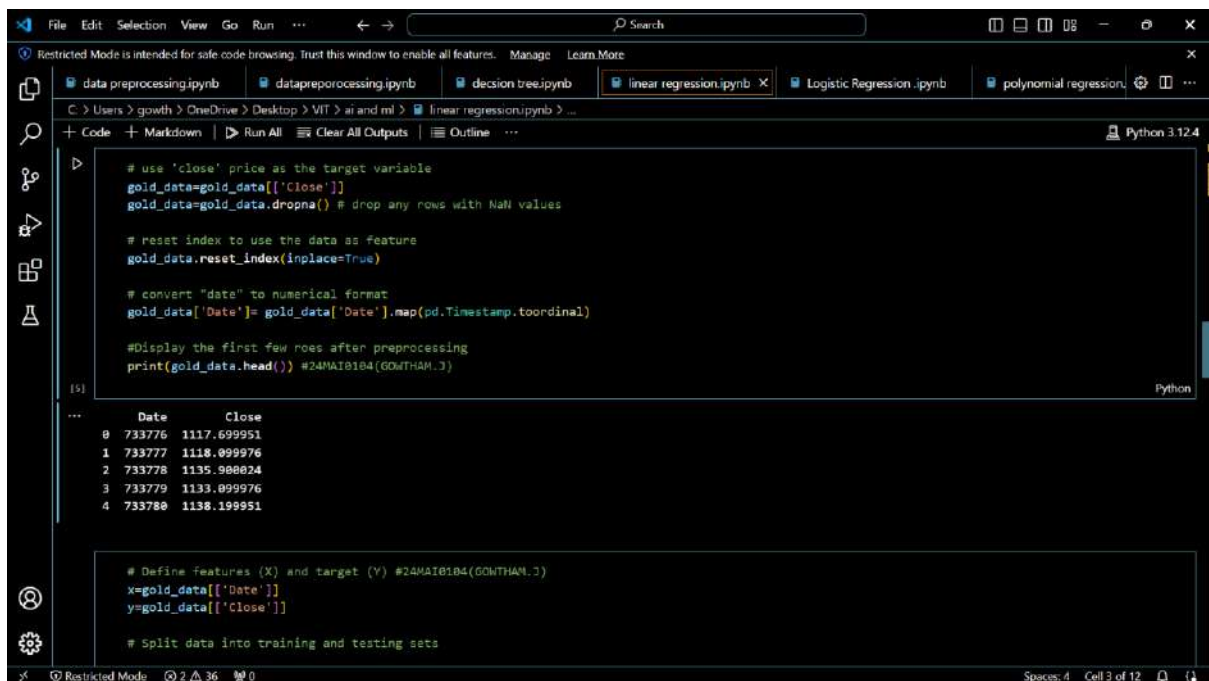
Name: GOWTHAM.J



The screenshot shows a Jupyter Notebook window with the title bar 'Artificial Intelligence and Machine Learning Laboratory'. The notebook has several tabs open: 'data preprocessing.ipynb', 'datapreporocessing.ipynb', 'decision tree.ipynb', 'linear regression.ipynb' (active), 'Logistic Regression .ipynb', and 'polynomial regression.ipynb'. The active cell contains the code `print(gold_data.head())#24MAI0104(GOWTHAM.J)`. The output displays the first five rows of a dataset with columns 'Date' and 'Volume'.

```
print(gold_data.head())#24MAI0104(GOWTHAM.J)
```

Date	Volume
2024-05-24	25
2024-05-28	26
2024-05-29	119317
2024-05-30	32587
2024-05-31	941



The screenshot shows the same Jupyter Notebook window. The active cell contains code for data preprocessing: `# use 'close' price as the target variable`, `gold_data=gold_data[['Close']]`, `gold_data=gold_data.dropna()`, `# reset index to use the data as feature`, `gold_data.reset_index(inplace=True)`, `# convert "date" to numerical format`, `gold_data['Date']= gold_data['Date'].map(pd.Timestamp.toordinal)`, and `#Display the first few rows after preprocessing`. The output shows the first five rows of the dataset with columns 'Date' and 'Close'.

```
# use 'close' price as the target variable
gold_data=gold_data[['Close']]
gold_data=gold_data.dropna() # drop any rows with NaN values

# reset index to use the data as feature
gold_data.reset_index(inplace=True)

# convert "date" to numerical format
gold_data['Date']= gold_data['Date'].map(pd.Timestamp.toordinal)

#Display the first few rows after preprocessing
print(gold_data.head()) #24MAI0104(GOWTHAM.J)
```

Date	Close
733776	1117.699951
733777	1118.099976
733778	1135.900024
733779	1133.099976
733780	1138.199951

Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

Name: GOWTHAM.J

```
File Edit Selection View Go Run ... Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
data preprocessing.ipynb datapreprocessing.ipynb decision tree.ipynb linear regression.ipynb X Logistic Regression .ipynb polynomial regression
C:\Users\gowth> OneDrive\Desktop> VIT> ai and ml> linear regression.ipynb > ...
+ Code + Markdown Run All Clear All Outputs Outline Python 3.12.4

# create the model #24MAI0104(GOWTHAM.J)
model=LinearRegression()

# train the model
model.fit(x_train,y_train)

[7] Python

# Make predictions #24MAI0104(GOWTHAM.J)
y_pred=model.predict(x_test)

[8] Python

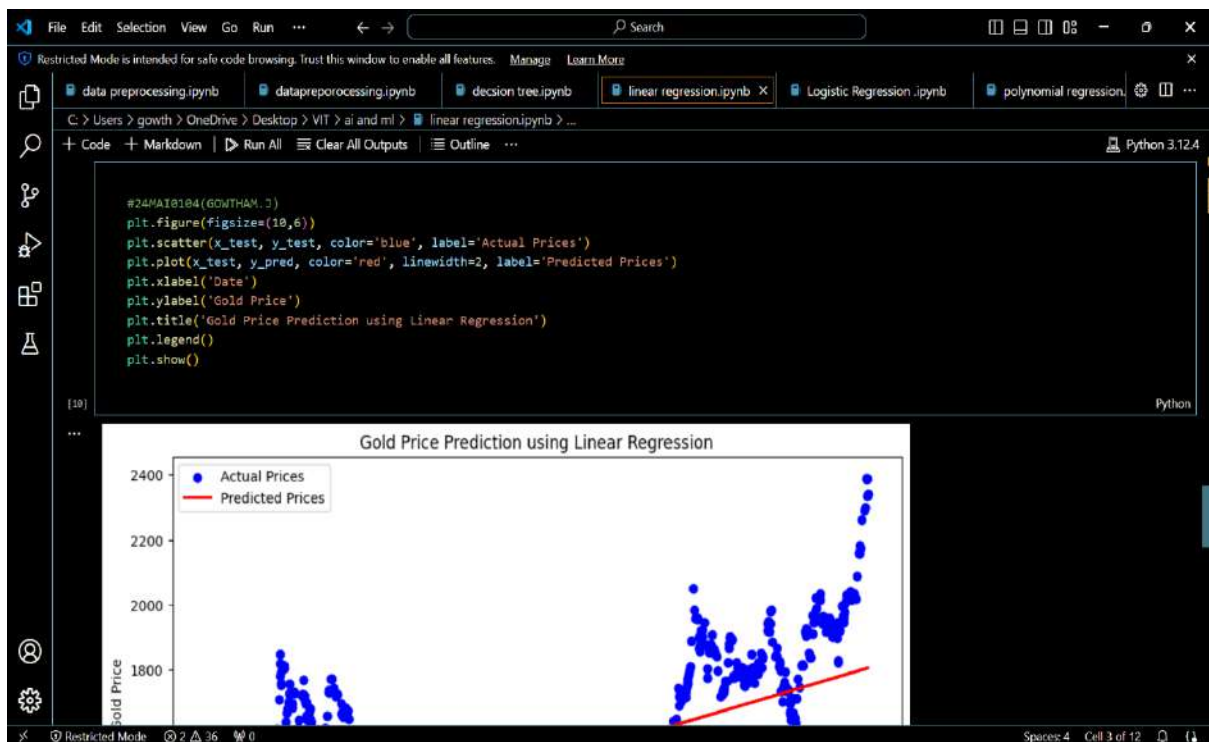
# Calculate performance metrics #24MAI0104(GOWTHAM.J)
mse=mean_squared_error(y_test,y_pred)
r2=r2_score(y_test,y_pred)

print("#24MAI0104(GOWTHAM.J) Mean Squared Error(MSE):",mse)
print("#24MAI0104(GOWTHAM.J) R^2 Score:",r2)

[9] Python

...
#24MAI0104(GOWTHAM.J) Mean Squared Error(MSE): 56054.31721283448
#24MAI0104(GOWTHAM.J) R^2 Score: 0.3331372311208952

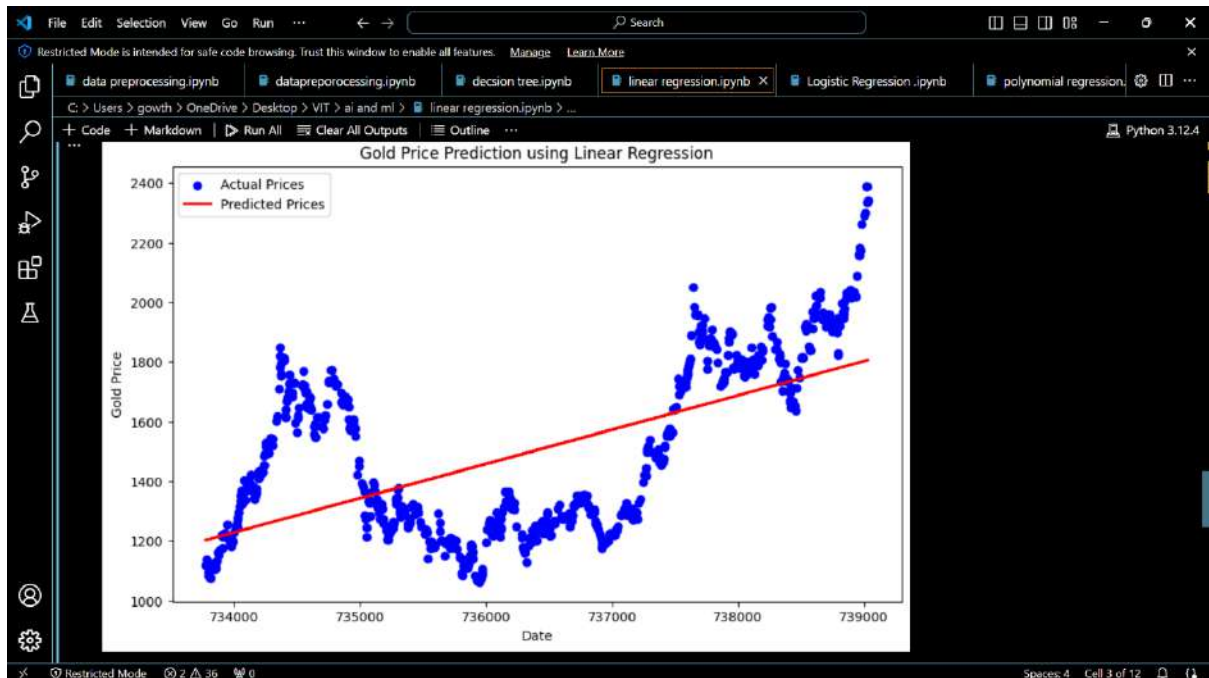
X Restricted Mode 2 36 0 Spaces: 4 Cell 3 of 12
```



Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

Name: GOWTHAM.J



```
File Edit Selection View Go Run ... Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
data preprocessing.ipynb datapreprocessing.ipynb decision tree.ipynb linear regression.ipynb x Logistic Regression .ipynb polynomial regression
C:\Users\gowth\OneDrive\Desktop\VIIT\ai and ml> linear regression.ipynb > ...
+ Code + Markdown + Run All + Clear All Outputs + Outline ... Python 3.12.4

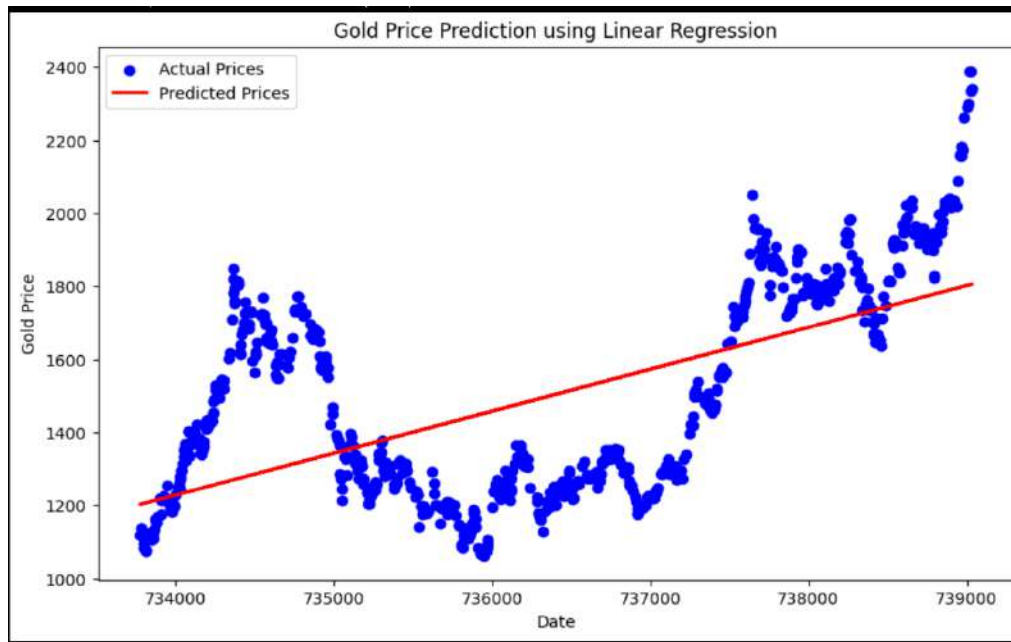
print(y_pred)

[11]:
...
[[1302.9184487 ]
[1442.79975233]
[1665.71404653]
[1434.18637156]
[1478.28688108]
[1798.36011032]
[1232.86295181]
[1334.15630961]
[1207.5970349 ]
[1285.34715194]
[1255.48743196]
[1574.5270555 ]
[1308.8903927 ]
[1390.08586205]
[1404.2118065 ]
[1697.75582298]
[1453.93972479]
[1675.24618791]
[1384.11391805]
[1781.93726433]
[1566.6027452 ]
[1248.48188227]
[1379.17557974]
[1705.90982344]
[1560.05657581]
...
```


Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

Name: GOWTHAM.J



CONCLUSION :

In this project, we successfully developed a Linear Regression model to predict gold prices based on historical data. The steps involved collecting and preprocessing the data, selecting relevant features, training the model, and evaluating its performance.

Key Takeaways:

- **Model Performance:** The Linear Regression model provided a basic but effective method for predicting gold prices. The evaluation metrics, such as Mean Squared Error (MSE) and R^2 score, helped us understand the accuracy and reliability of our predictions.
- **Simplicity and Interpretability:** Linear Regression is a straightforward algorithm that offers easy interpretability.

Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

Name: GOWTHAM.J

TITLE : (POLYNOMIAL REGRESSION)

3. Predicting house prices using polynomial regression can be accomplished using publicly available datasets. One of the most popular datasets for this purpose is the **Ames Housing dataset**.

This dataset is comprehensive and provides detailed information about houses in Ames, Iowa, which can be used for various regression tasks, including polynomial regression.

DESCRIPTION :

Polynomial regression is a form of regression analysis that models the relationship between the independent variable x and the dependent variable Y as an N -th degree polynomial. Here's a brief overview:

Polynomial Regression Overview

- **Objective:** To model the relationship between a feature and the target variable as a polynomial function, which allows for capturing non-linear relationships.
- **Mathematical Form:**

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \dots + \beta_n x^n + \epsilon$$

Where y is the dependent variable, x is the independent variable, β_i are the coefficients, and ϵ is the error term.

- **Advantages:**
 - Can model non-linear relationships between features and target variables.
 - Flexible in fitting the data, making it useful for capturing complex patterns.
- **Disadvantages:**
 - Risk of overfitting, especially with high-degree polynomials.
 - Can become computationally expensive and harder to interpret as the degree increases.
- **Applications:**
 - Useful for regression tasks where the relationship between variables is not linear.

Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

Name: GOWTHAM.J

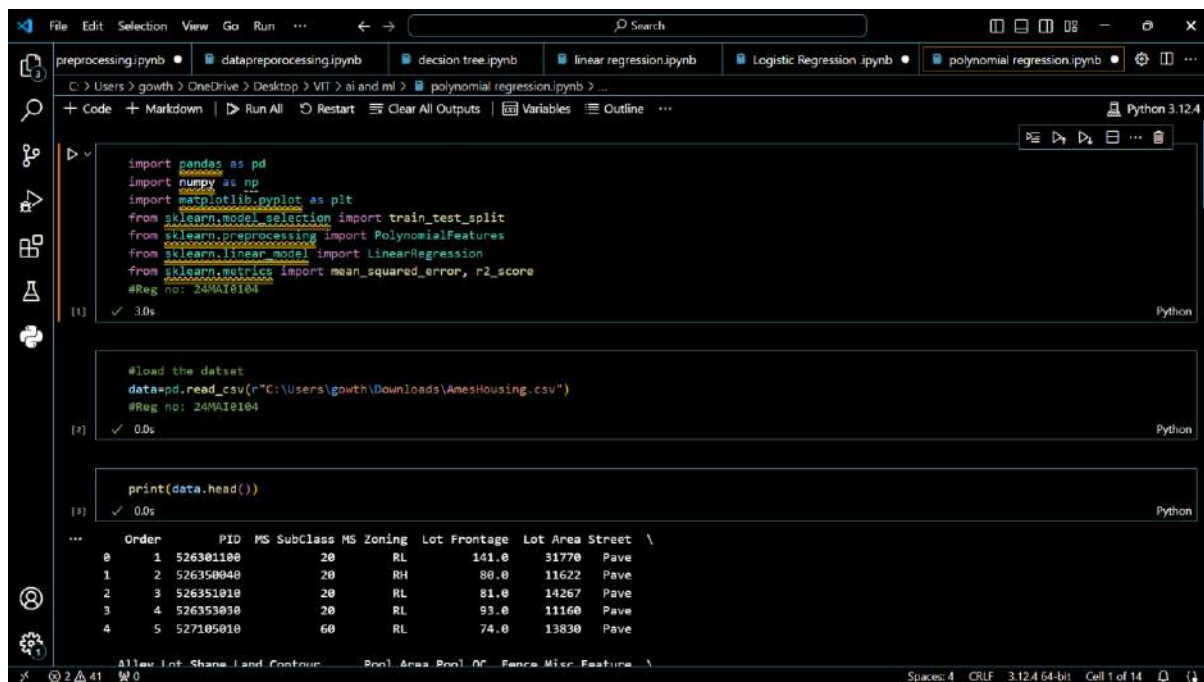
- Commonly applied in scenarios like predicting house prices, where relationships between features (e.g., square footage, number of rooms) and target variables (e.g., price) can be complex.

Example with the Ames Housing Dataset:

1. **Feature Selection:** Choose features relevant to house prices (e.g., square footage, number of bedrooms).
2. **Polynomial Features:** Generate polynomial features from selected features (e.g., square footage squared, cubic terms).
3. **Model Training:** Fit a polynomial regression model using these features.
4. **Evaluation:** Assess model performance using metrics like Mean Squared Error (MSE) and adjust the polynomial degree to balance bias and variance.

Polynomial regression can effectively capture intricate relationships in datasets like the Ames Housing dataset, providing insights into how various features impact house prices.

SOURCE CODE AND RESULT:



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

#Reg no: 24MAI0104

#load the dataset
data=pd.read_csv(r"C:\Users\gowth\Downloads\AmesHousing.csv")

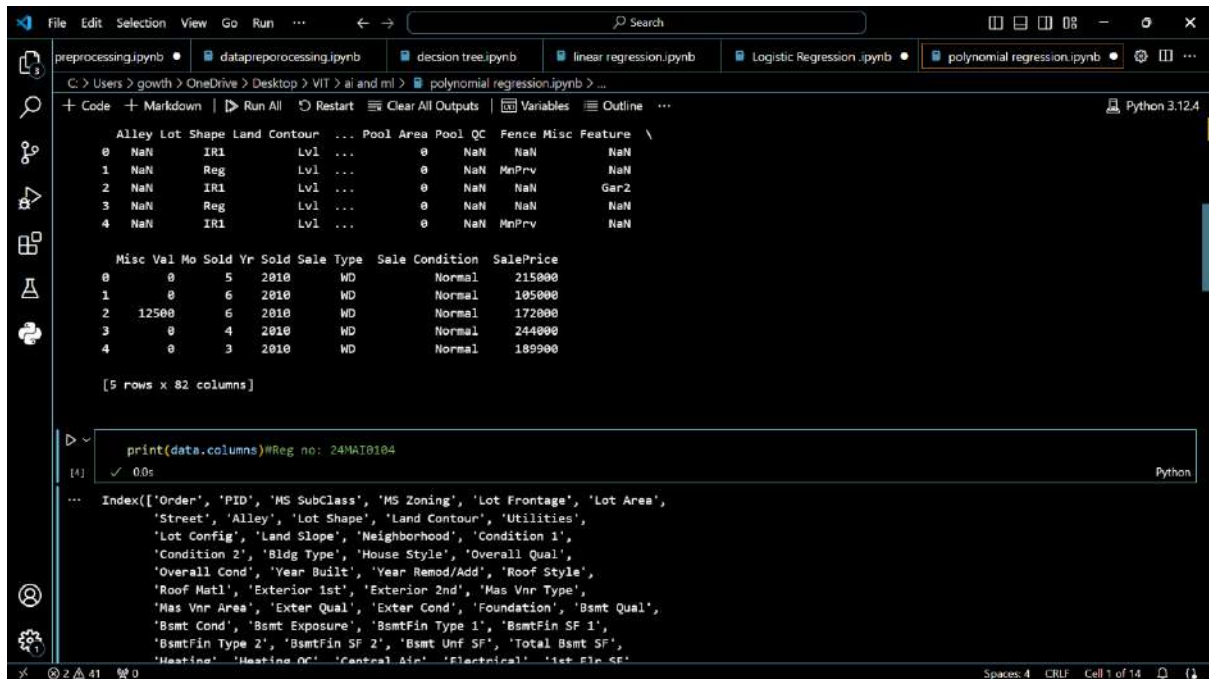
print(data.head())
```

Order	PID	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street
0	1	526301100	20	RL	141.0	31770 Pave
1	2	526350040	20	RH	80.0	11622 Pave
2	3	526351010	20	RL	81.0	14267 Pave
3	4	526353030	20	RL	93.0	11160 Pave
4	5	527105010	60	RL	74.0	13830 Pave

Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

Name: GOWTHAM.J



```
preprocessing.ipynb • datapreprocessing.ipynb • decision tree.ipynb • linear regression.ipynb • Logistic Regression .ipynb • polynomial regression.ipynb •
```

C:\Users\gowtham> OneDrive\Desktop> VIT> ai and ml> polynomial regression.ipynb> ...

+ Code + Markdown | ▶ Run All | ⌂ Restart | 🗑 Clear All Outputs | 📄 Variables | 📖 Outline ...

Python 3.12.4

	Alley	Lot Shape	Land Contour	Pool Area	Pool QC	Fence Misc Feature	Sale Price
0	NaN	IR1	Lvl ...	0	NaN	NaN	NaN
1	NaN	Reg	Lvl ...	0	NaN	MnPrv	NaN
2	NaN	IR1	Lvl ...	0	NaN	NaN	Gar2
3	NaN	Reg	Lvl ...	0	NaN	NaN	NaN
4	NaN	IR1	Lvl ...	0	NaN	MnPrv	NaN

	Misc Val	Mo Sold	Yr Sold	Sale Type	Sale Condition	Sale Price
0	0	5	2010	WD	Normal	215000
1	0	6	2010	WD	Normal	105000
2	12500	6	2010	WD	Normal	172000
3	0	4	2010	WD	Normal	244000
4	0	3	2010	WD	Normal	189900

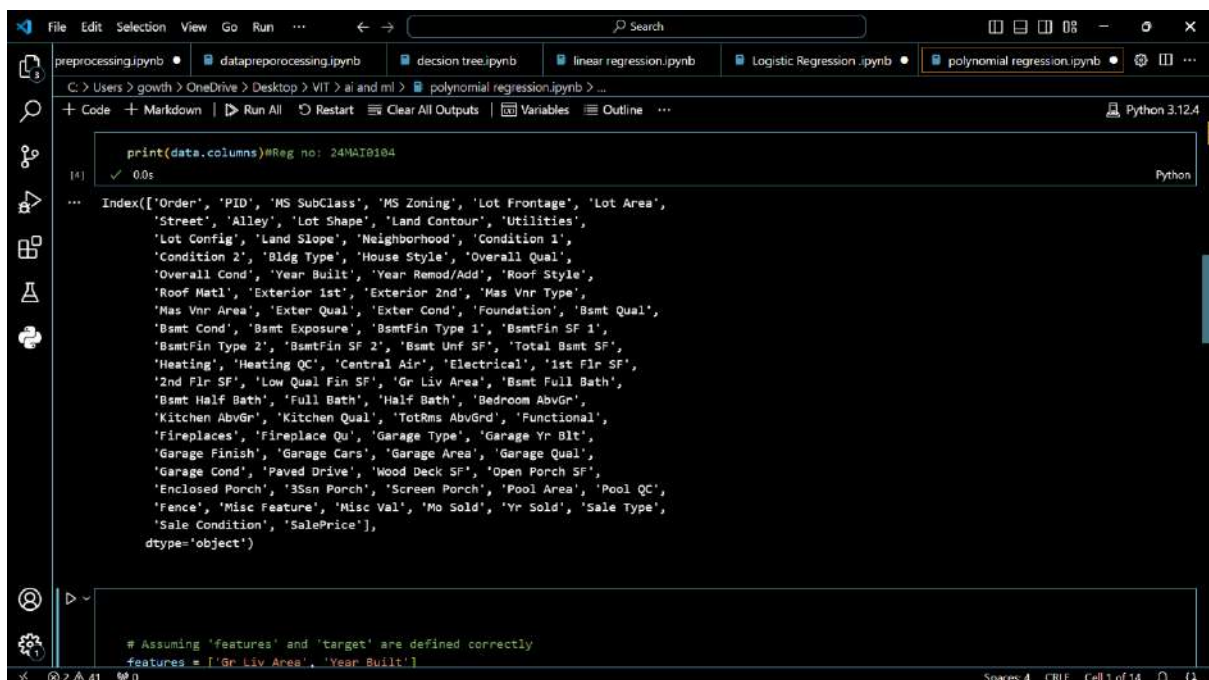
[5 rows x 82 columns]

```
print(data.columns)#Reg no: 24MAI0104
```

[4] ✓ 0.0s Python

```
Index(['Order', 'PID', 'MS SubClass', 'MS Zoning', 'Lot Frontage', 'Lot Area',
      'Street', 'Alley', 'Lot Shape', 'Land Contour', 'Utilities',
      'Lot Config', 'Land Slope', 'Neighborhood', 'Condition 1',
      'Condition 2', 'Bldg Type', 'House Style', 'Overall Qual',
      'Overall Cond', 'Year Built', 'Year Remod/Add', 'Roof Style',
      'Roof Matl', 'Exterior 1st', 'Exterior 2nd', 'Mas Vnr Type',
      'Mas Vnr Area', 'Exter Qual', 'Exter Cond', 'Foundation', 'Bsmt Qual',
      'Bsmt Cond', 'Bsmt Exposure', 'BsmtFin Type 1', 'BsmtFin SF 1',
      'BsmtFin Type 2', 'BsmtFin SF 2', 'Bsmt Unf SF', 'Total Bsmt SF',
      'Heating', 'Heating QC', 'Central Air', 'Electrical', '1st Flr SF',
      '2nd Flr SF', 'Low Qual Fin SF', 'Gr Liv Area', 'Bsmt Full Bath',
      'Bsmt Half Bath', 'Full Bath', 'Half Bath', 'Bedroom AbvGr',
      'Kitchen AbvGr', 'Kitchen Qual', 'TotRms AbvGrd', 'Functional',
      'Fireplaces', 'Fireplace Qu', 'Garage Type', 'Garage Yr Blt',
      'Garage Finish', 'Garage Cars', 'Garage Area', 'Garage Qual',
      'Garage Cond', 'Paved Drive', 'Wood Deck SF', 'Open Porch SF',
      'Enclosed Porch', '3Ssn Porch', 'Screen Porch', 'Pool Area', 'Pool QC',
      'Fence', 'Misc Feature', 'Misc Val', 'Mo Sold', 'Yr Sold', 'Sale Type',
      'Sale Condition', 'SalePrice'],
      dtype='object')
```

Spaces: 4 CRLF Cell 1 of 14



```
preprocessing.ipynb • datapreprocessing.ipynb • decision tree.ipynb • linear regression.ipynb • Logistic Regression .ipynb • polynomial regression.ipynb •
```

C:\Users\gowtham> OneDrive\Desktop> VIT> ai and ml> polynomial regression.ipynb> ...

+ Code + Markdown | ▶ Run All | ⌂ Restart | 🗑 Clear All Outputs | 📄 Variables | 📖 Outline ...

Python 3.12.4

```
print(data.columns)#Reg no: 24MAI0104
```

[4] ✓ 0.0s Python

```
Index(['Order', 'PID', 'MS SubClass', 'MS Zoning', 'Lot Frontage', 'Lot Area',
      'Street', 'Alley', 'Lot Shape', 'Land Contour', 'Utilities',
      'Lot Config', 'Land Slope', 'Neighborhood', 'Condition 1',
      'Condition 2', 'Bldg Type', 'House Style', 'Overall Qual',
      'Overall Cond', 'Year Built', 'Year Remod/Add', 'Roof Style',
      'Roof Matl', 'Exterior 1st', 'Exterior 2nd', 'Mas Vnr Type',
      'Mas Vnr Area', 'Exter Qual', 'Exter Cond', 'Foundation', 'Bsmt Qual',
      'Bsmt Cond', 'Bsmt Exposure', 'BsmtFin Type 1', 'BsmtFin SF 1',
      'BsmtFin Type 2', 'BsmtFin SF 2', 'Bsmt Unf SF', 'Total Bsmt SF',
      'Heating', 'Heating QC', 'Central Air', 'Electrical', '1st Flr SF',
      '2nd Flr SF', 'Low Qual Fin SF', 'Gr Liv Area', 'Bsmt Full Bath',
      'Bsmt Half Bath', 'Full Bath', 'Half Bath', 'Bedroom AbvGr',
      'Kitchen AbvGr', 'Kitchen Qual', 'TotRms AbvGrd', 'Functional',
      'Fireplaces', 'Fireplace Qu', 'Garage Type', 'Garage Yr Blt',
      'Garage Finish', 'Garage Cars', 'Garage Area', 'Garage Qual',
      'Garage Cond', 'Paved Drive', 'Wood Deck SF', 'Open Porch SF',
      'Enclosed Porch', '3Ssn Porch', 'Screen Porch', 'Pool Area', 'Pool QC',
      'Fence', 'Misc Feature', 'Misc Val', 'Mo Sold', 'Yr Sold', 'Sale Type',
      'Sale Condition', 'SalePrice'],
      dtype='object')
```

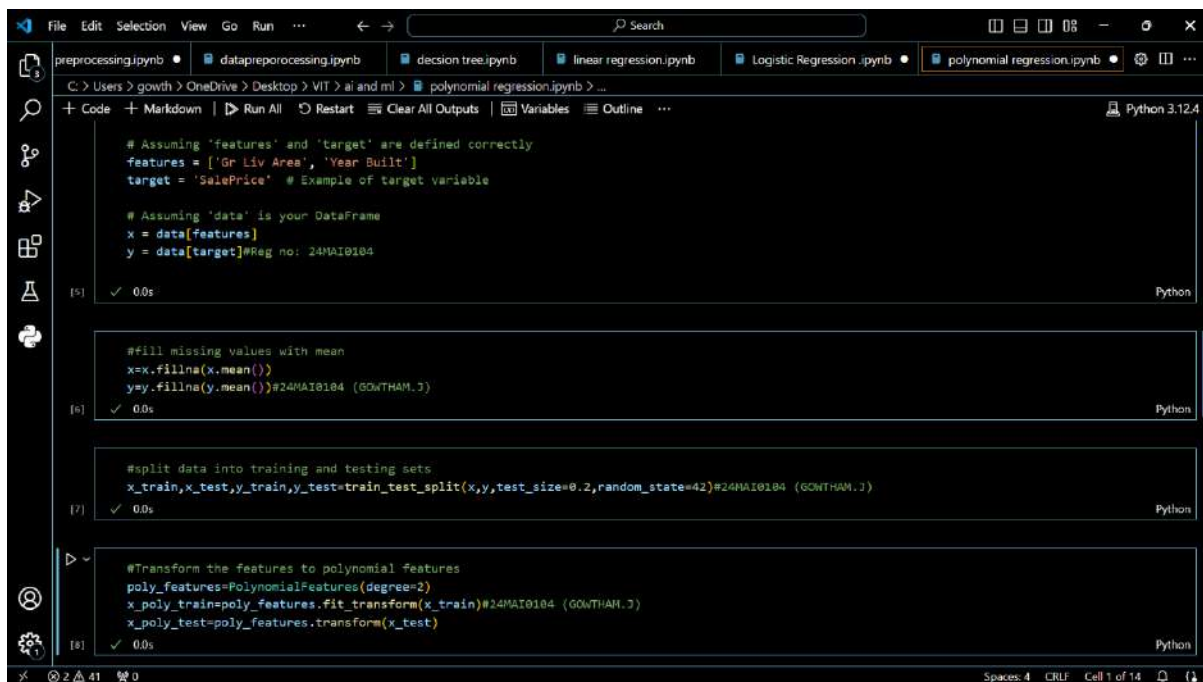
```
# Assuming 'features' and 'target' are defined correctly
features = ['Gr Liv Area', 'Year Built']
```

Spaces: 4 CRLF Cell 1 of 14

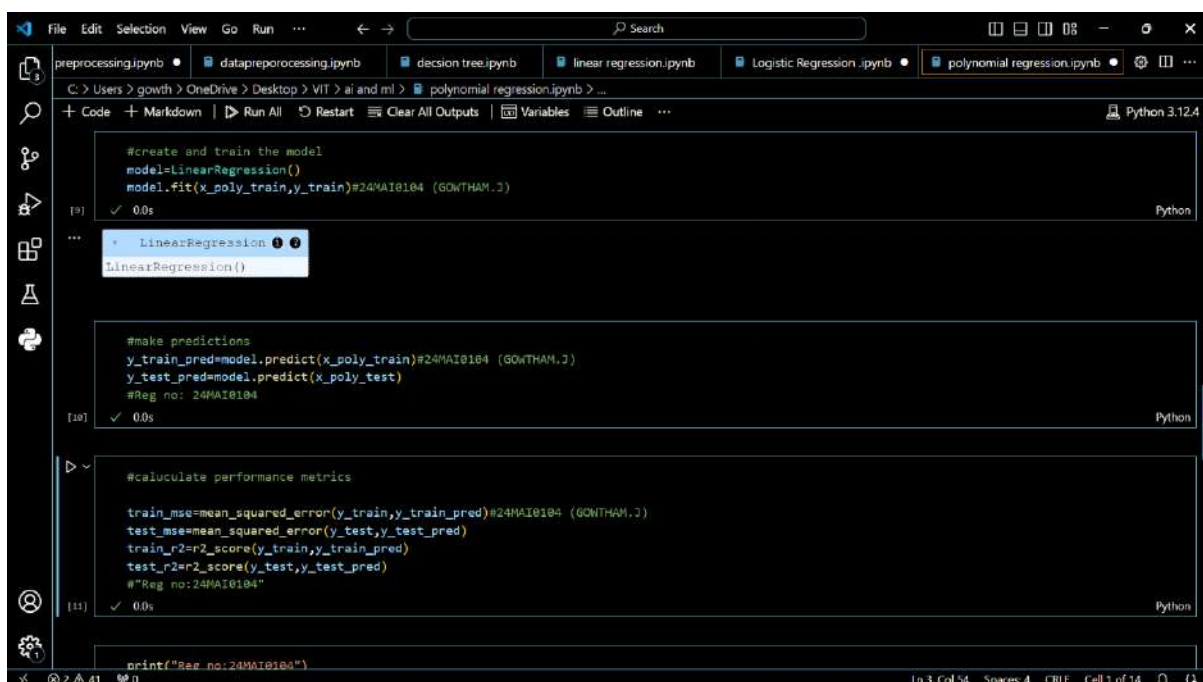
Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

Name: GOWTHAM.J



```
preprocessing.ipynb • datapreprocessing.ipynb • decision tree.ipynb • linear regression.ipynb • Logistic Regression .ipynb • polynomial regression.ipynb •  
C:\> Users > gowth > OneDrive > Desktop > VIT > ai and ml > polynomial regression.ipynb > ...  
+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ... Python 3.12.4  
[5] ✓ 0.0s Python  
# Assuming 'features' and 'target' are defined correctly  
features = ['Gr Liv Area', 'Year Built']  
target = 'SalePrice' # Example of target variable  
  
# Assuming 'data' is your DataFrame  
x = data[features]  
y = data[target] #Reg no: 24MAI0104  
  
[6] ✓ 0.0s Python  
#fill missing values with mean  
x=x.fillna(x.mean())  
y=y.fillna(y.mean())#24MAI0104 (GOWTHAM.J)  
  
[7] ✓ 0.0s Python  
#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)#24MAI0104 (GOWTHAM.J)  
  
[8] ✓ 0.0s Python  
#Transform the features to polynomial features  
poly_features=PolynomialFeatures(degree=2)  
x_poly_train=poly_features.fit_transform(x_train)#24MAI0104 (GOWTHAM.J)  
x_poly_test=poly_features.transform(x_test)
```



```
preprocessing.ipynb • datapreprocessing.ipynb • decision tree.ipynb • linear regression.ipynb • Logistic Regression .ipynb • polynomial regression.ipynb •  
C:\> Users > gowth > OneDrive > Desktop > VIT > ai and ml > polynomial regression.ipynb > ...  
+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ... Python 3.12.4  
[9] ✓ 0.0s Python  
#create and train the model  
model=LinearRegression()  
model.fit(x_poly_train,y_train)#24MAI0104 (GOWTHAM.J)  
...  
LinearRegression()  
LinearRegression()  
  
[10] ✓ 0.0s Python  
#make predictions  
y_train_pred=model.predict(x_poly_train)#24MAI0104 (GOWTHAM.J)  
y_test_pred=model.predict(x_poly_test)  
#Reg no: 24MAI0104  
  
[11] ✓ 0.0s Python  
#caluculate performance metrics  
train_mse=mean_squared_error(y_train,y_train_pred)#24MAI0104 (GOWTHAM.J)  
test_mse=mean_squared_error(y_test,y_test_pred)  
train_r2=r2_score(y_train,y_train_pred)  
test_r2=r2_score(y_test,y_test_pred)  
#""Reg no:24MAI0104"  
  
print("Reg no:24MAI0104")  
Ln 3, Col 54 Spaces: 4 CRLF Cell 1 of 14
```

Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

Name: GOWTHAM.J

```
File Edit Selection View Go Run ... Search
preprocessing.ipynb • datapreprocessing.ipynb • decision tree.ipynb • linear regression.ipynb • Logistic Regression .ipynb • polynomial regression.ipynb •
C:\Users\gowth> OneDrive\Desktop> VIT> ai and ml> polynomial regression.ipynb> ...
+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ... Python 3.12.4

print("Reg no:24MAI0104")
print("Training Mean Squared Error(MSE):",train_mse)
print("Training Mean Squared Error(MSE):",test_mse)#24MAI0104 (GOWTHAM.J)
print("Training R^2 Score :",train_r2)
print("Testing R^2 Score:",test_r2)

[12] ✓ 0.0s Python

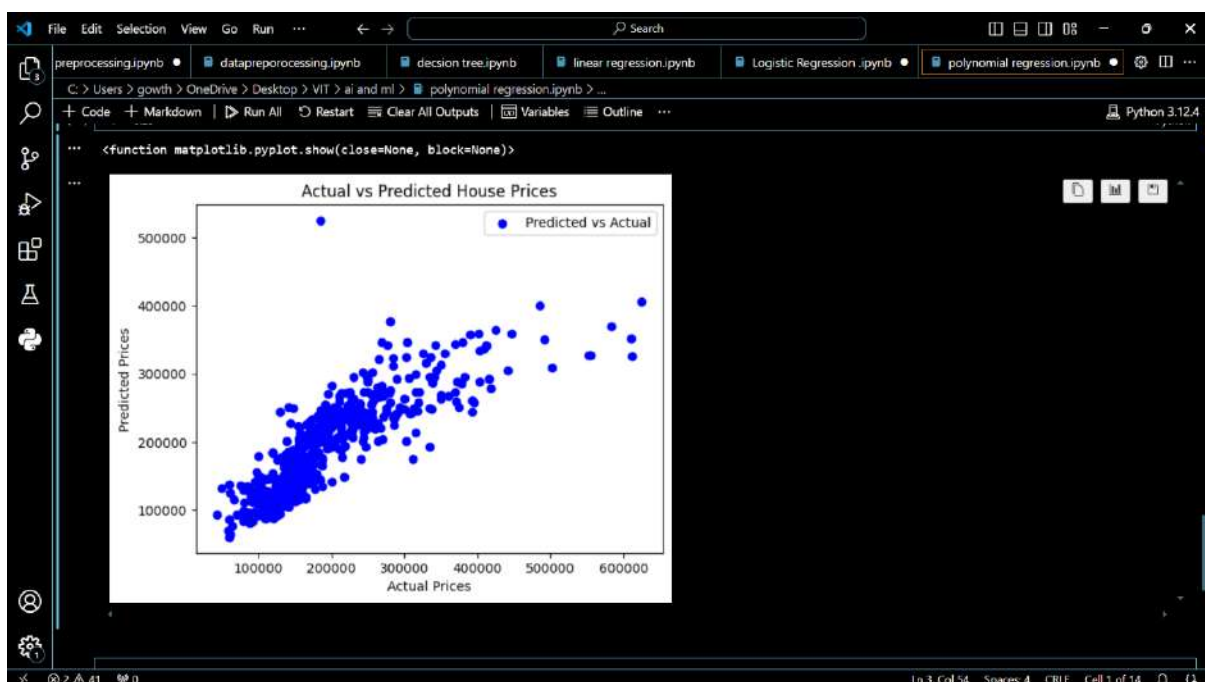
...
Reg no:24MAI0104
Training Mean Squared Error(MSE): 1889928769.0922892
Training Mean Squared Error(MSE): 2407047040.306275
Training R^2 Score : 0.6821372899447561
Testing R^2 Score: 0.6997776550824131

#Reg no:24MAI0104
#visualize the actual vs predicted prices
plt.scatter(y_test,y_test_pred,color='blue',label='Predicted vs Actual')
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted House Prices")
plt.legend()
plt.show

[13] ✓ 0.2s Python

<function matplotlib.pyplot.show(close=None, block=None)>

...
Actual vs Predicted House Prices
Predicted vs Actual
```



Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

Name: GOWTHAM.J

CONCULSION :

When using polynomial regression for predicting house prices, here's a concise consultation guide:

1. Data Preparation

- **Feature Selection:** Identify relevant features such as square footage, number of bedrooms, and age of the house.
- **Preprocessing:** Handle missing values, normalize or standardize features, and create polynomial features.

2. Polynomial Degree

- **Choosing Degree:** Start with a low-degree polynomial (e.g., quadratic) and incrementally increase the degree to capture more complexity.
- **Avoid Overfitting:** Use cross-validation to find the optimal polynomial degree that balances model complexity and generalization.

3. Model Training

- **Train-Test Split:** Divide the dataset into training and testing sets to evaluate model performance.
- **Fit Model:** Train the polynomial regression model on the training set.

4. Evaluation

- **Performance Metrics:** Evaluate using metrics like Mean Squared Error (MSE) or R^2 score.
- **Visual Inspection:** Plot predictions versus actual values to visually inspect fit quality.

5. Refinement

- **Feature Engineering:** Experiment with different polynomial terms and interaction features.
- **Regularization:** Consider using regularization techniques (e.g., Ridge or Lasso) to mitigate overfitting.

6. Deployment

- **Scalability:** Ensure the model performs well on new, unseen data before deploying.
- **Interpretability:** While polynomial models can be complex, aim to interpret the impact of key features on predictions.

Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

Name: GOWTHAM.J

TITLE: (Logistic Regression)

4. Logistic Regression is a popular algorithm for binary classification tasks. A simple example of classifying images using Logistic Regression.

use the MNIST dataset, which contains images of handwritten digits and classify whether the digit is a '0' or a '1'.

DESCRIPTION :

Logistic Regression is widely used for binary classification tasks, where the goal is to predict one of two possible outcomes. In this example, we'll use Logistic Regression to classify images from the MNIST dataset, which consists of handwritten digits. Specifically, we'll focus on classifying whether a given digit is a '0' or a '1'.

Dataset:

The MNIST dataset is a well-known dataset in machine learning, comprising 70,000 images of handwritten digits (0-9). Each image is 28x28 pixels, and each pixel has a grayscale value between 0 and 255. For this binary classification task, we'll filter the dataset to include only the images labeled as '0' or '1'.

Problem Statement:

Given an image of a handwritten digit, the task is to determine whether the digit is a '0' or a '1'. Logistic Regression will be used to model this binary classification problem.

Steps:

1. Data Preprocessing:

- Data Selection: Filter the MNIST dataset to include only images labeled as '0' or '1'.
- Flattening the Images: Each 28x28 image will be flattened into a 784-dimensional vector (since $28 \times 28 = 784$). This will serve as the input feature vector for Logistic Regression.

2. Model Training:

- Logistic Regression Model: Initialize and train a Logistic Regression model using the preprocessed feature vectors.
- Training Process: The model will learn the optimal coefficients by minimizing the difference between the predicted probabilities and the actual labels using a technique called Maximum Likelihood Estimation (MLE).

Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

Name: GOWTHAM.J

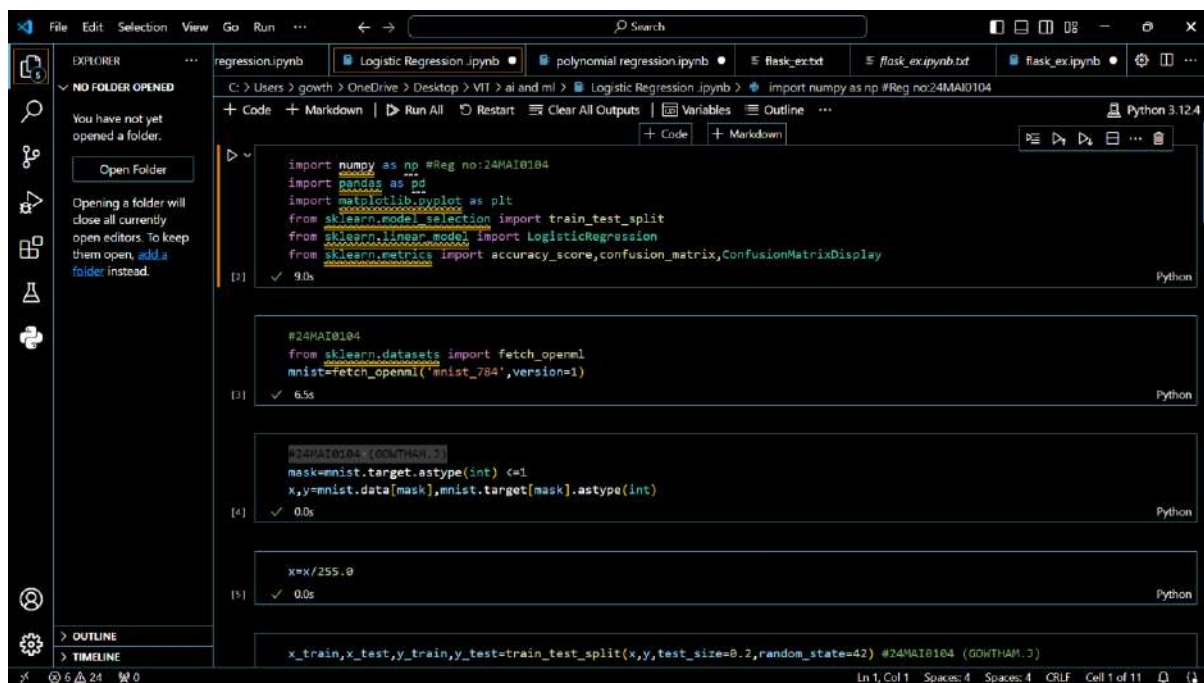
3. Prediction:

- Probability Output: For a given input image, the model will output a probability indicating the likelihood that the image represents a '1'.
- Decision Rule: If the probability is greater than 0.5, the model predicts '1'; otherwise, it predicts '0'.

4. Evaluation:

- Accuracy: The accuracy of the model can be evaluated on a separate test set of '0' and '1' images to determine how well the model generalizes to unseen data.
- Confusion Matrix: This will give insights into how many '0's and '1's were correctly or incorrectly classified.

SOURCE CODE AND RESULT :



```
File Edit Selection View Go Run ... Search
regression.ipynb Logistic Regression.ipynb polynomial regression.ipynb flask_ext.txt flask_ext.py flask_ext.py flask_ext.py
C:\Users\gowtham> OneDrive > Desktop > VIT > ai and ml > Logistic Regression.ipynb > import numpy as np #Reg no:24MAI0104
+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ... Python 3.12.4

import numpy as np #Reg no:24MAI0104
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay

[2] ✓ 9.0s Python

#24MAI0104
from sklearn.datasets import fetch_openml
mnist=fetch_openml('mnist_784',version=1)

[3] ✓ 6.5s Python

#24MAI0104 (GOWTHAM.J)
mask=mnist.target.astype(int) <=1
x,y=mnist.data[mask],mnist.target[mask].astype(int)

[4] ✓ 0.0s Python

x=x/255.0

[5] ✓ 0.0s Python

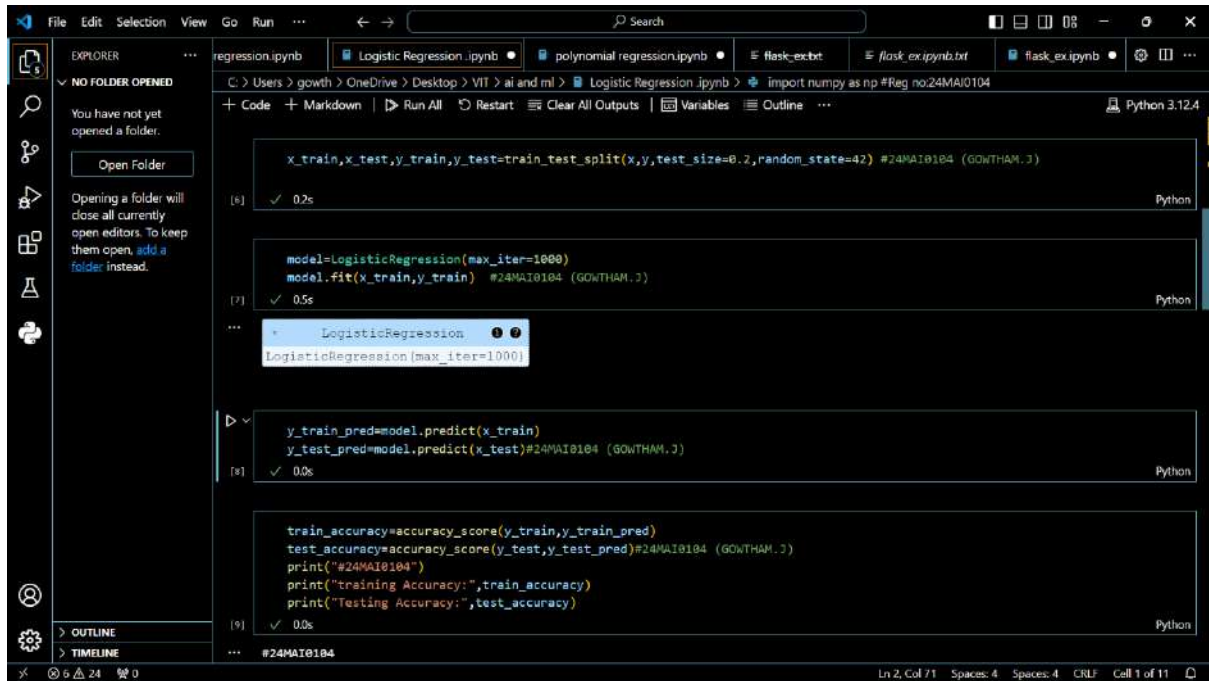
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42) #24MAI0104 (GOWTHAM.J)

Ln 1, Col 1 Spaces: 4 Spaces: 4 ORLF Cell 1 of 11
```

Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

Name: GOWTHAM.J



The screenshot shows a Jupyter Notebook with the following code and output:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42) #24MAI0104 (GOWTHAM.J)
```

Output: 0.2s

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression(max_iter=1000)
model.fit(x_train,y_train) #24MAI0104 (GOWTHAM.J)
```

Output: 0.5s

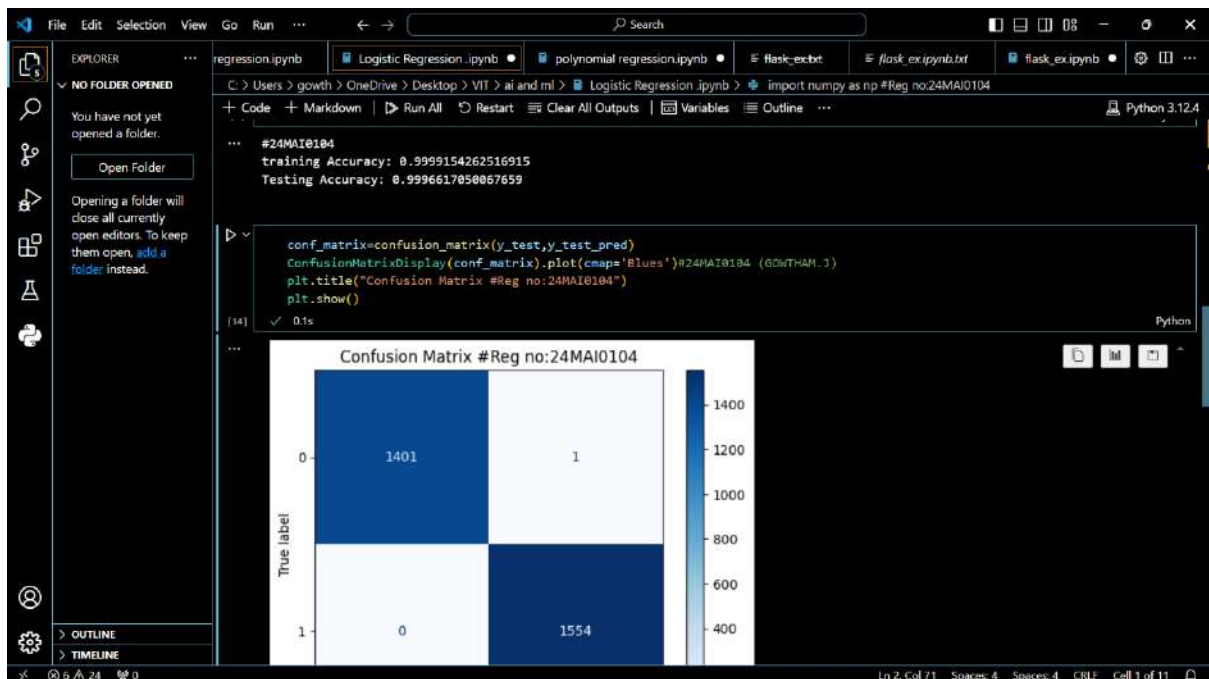
```
LogisticRegression(max_iter=1000)
```

```
y_train_pred=model.predict(x_train)
y_test_pred=model.predict(x_test)#24MAI0104 (GOWTHAM.J)
```

Output: 0.0s

```
train_accuracy=accuracy_score(y_train,y_train_pred)
test_accuracy=accuracy_score(y_test,y_test_pred)#24MAI0104 (GOWTHAM.J)
print("#24MAI0104")
print("training Accuracy:",train_accuracy)
print("Testing Accuracy:",test_accuracy)
```

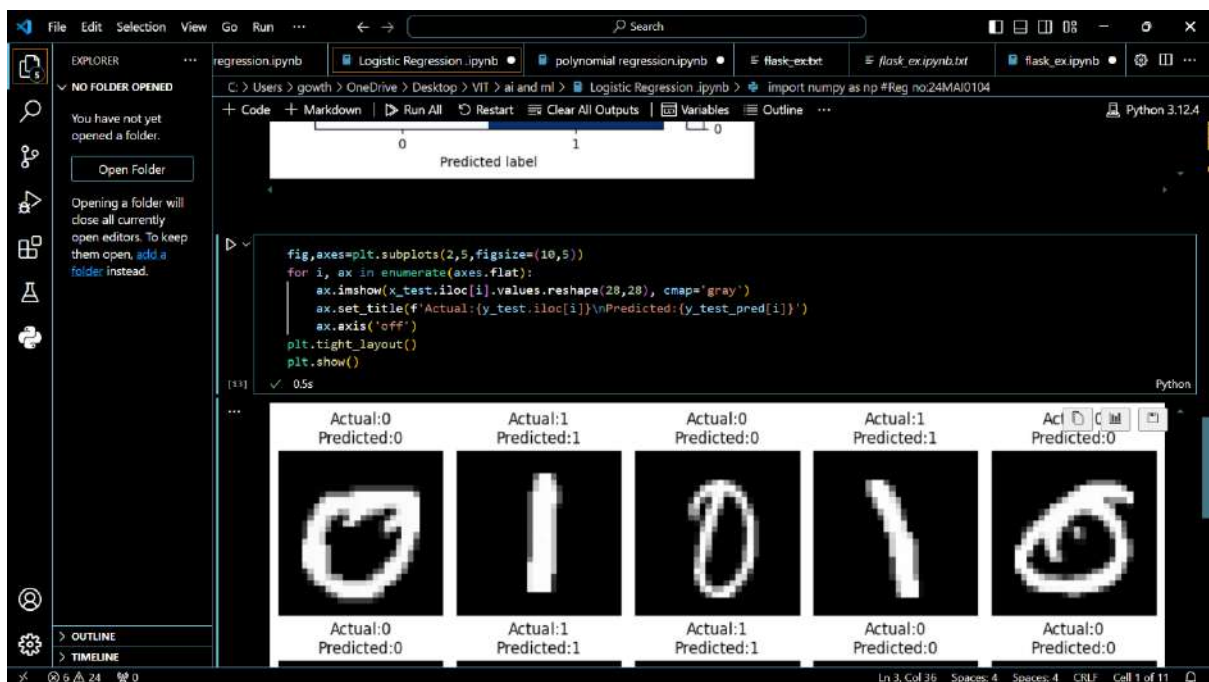
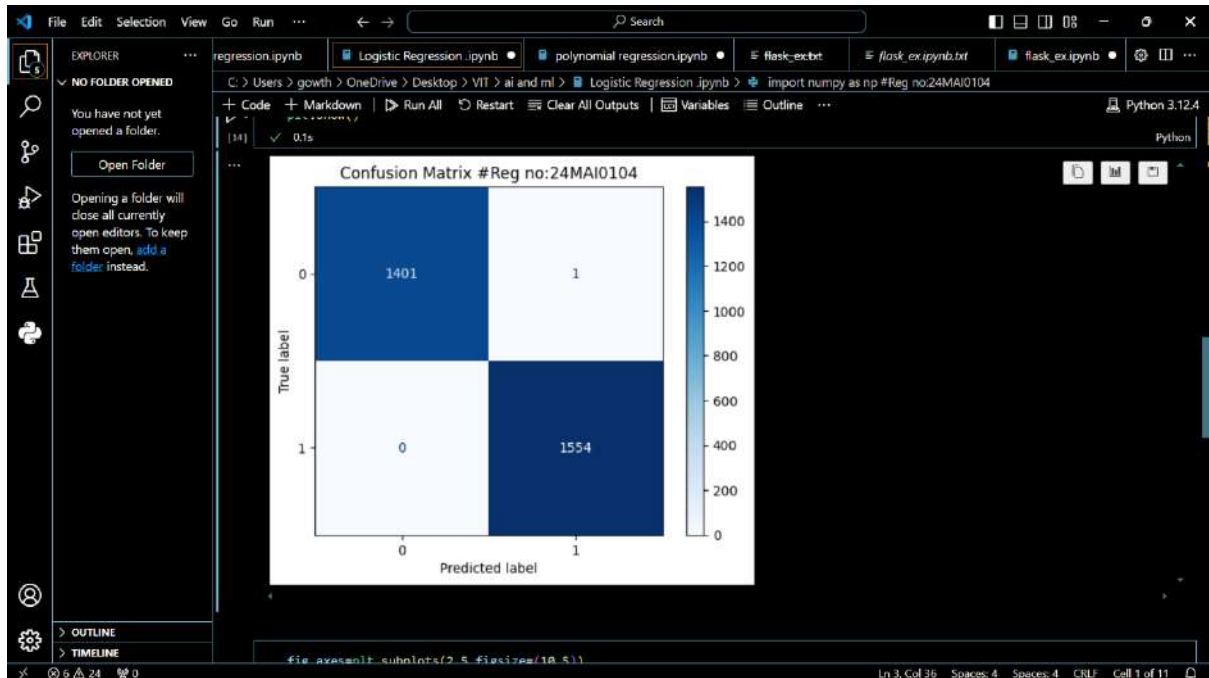
Output: 0.0s



Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

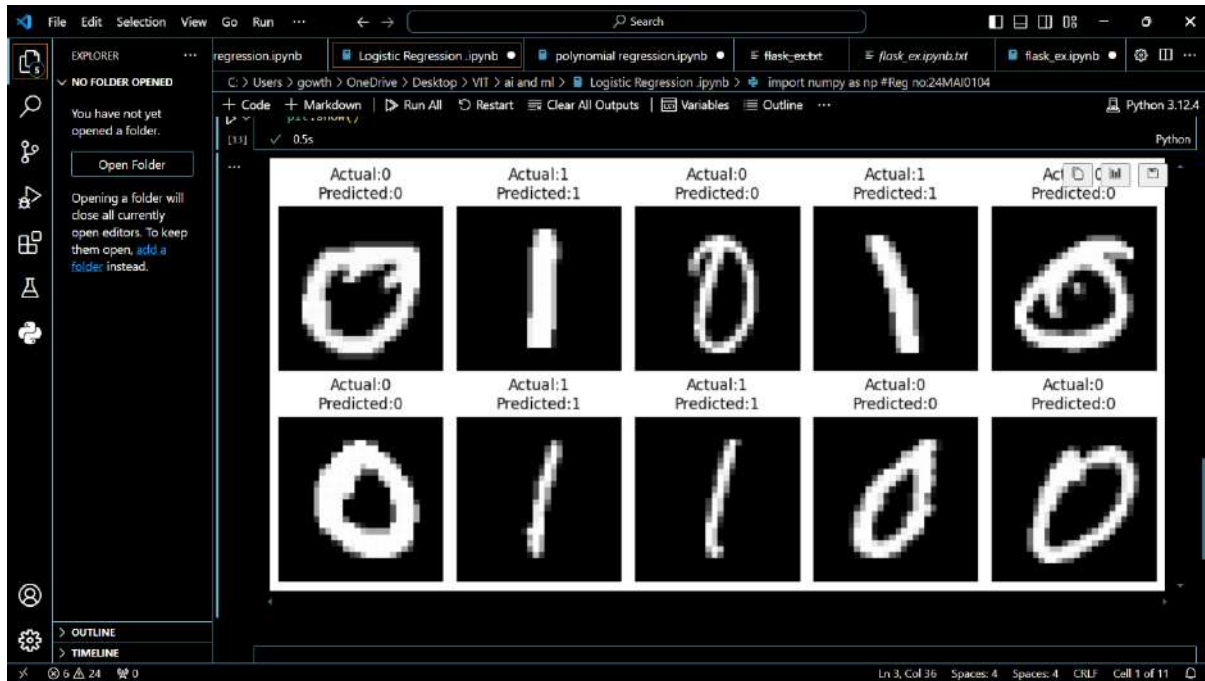
Name: GOWTHAM.J



Artificial Intelligence and Machine Learning Laboratory

Reg no:24MAI0104

Name: GOWTHAM.J



CONCLUSION :

In conclusion, using Logistic Regression for classifying images of handwritten digits '0' and '1' from the MNIST dataset proves to be a straightforward yet effective approach for binary classification tasks. This method leverages the simplicity and interpretability of Logistic Regression while demonstrating its application in the context of image classification.

Key Points:

1. **Model Performance:** Logistic Regression provides a reasonable baseline performance for binary classification tasks, achieving decent accuracy in distinguishing between '0' and '1' digits based on pixel intensity values.
2. **Interpretability:** The coefficients learned by Logistic Regression offer insights into which pixels contribute positively or negatively to the likelihood of a digit being classified as '1' versus '0'. This interpretability is valuable for understanding the model's decision-making process.
3. **Implementation Ease:** Logistic Regression is relatively easy to implement and computationally efficient, making it suitable for initial experimentation and as a baseline comparison against more complex models.