**https://drive.google.com/drive/folders/1oNT2ydRH3PkxGuucXbDFZGukx11M3rpv?usp=sharing**

**GANESH G**

**25MML0043**

**LAB – 1**

**AIM:**

The goal of this lab is to get hands-on experience with preparing real-world data for machine learning. Using the Titanic dataset, we'll clean the data, handle missing values, create and transform features, convert categories into numbers, scale values, and finally organize everything into a pipeline — making the dataset ready for building predictive models.

**INTRODUCTION:**

Before building any machine learning model, it’s important to clean and prepare the data — a process called data preprocessing. In this lab, we’ll use the Titanic dataset to practice these skills. This dataset is great for learning because it includes real-world challenges like missing values, different data types, and the need to create new features. Step by step, we’ll get the data ready for modeling.

**REAL WORLD EXAMPLE:**

Predicting Hospital Patient Survival:

1. Load patient records from hospital database
2. Fill missing blood pressure or age from previous visits
3. Create "Risk Score" using age + comorbidities
4. Convert 'Admission Type' or 'Gender' into numeric codes
5. Normalize "Hospital Stay" and "Blood Pressure"
6. Drop irrelevant features (e.g., Patient ID)
7. Split into training (80%) and testing (20%) sets
8. Automate the above for any future patient prediction

**ALGORITHM:**

1. Import necessary libraries

Import pandas, NumPy, seaborn, matplotlib, and scikit-learn tools.

1. Load the dataset

Read the Titanic dataset into a Data Frame.

1. Explore the dataset

View the first few rows, Check column data types, Identify missing values.

1. Handle missing values

Fill or drop missing values depending on the column.

1. Perform feature engineering

Create new features (e.g., family size from siblings/spouses and parents/children).

Extract titles from passenger names if needed.

1. Encode categorical variables

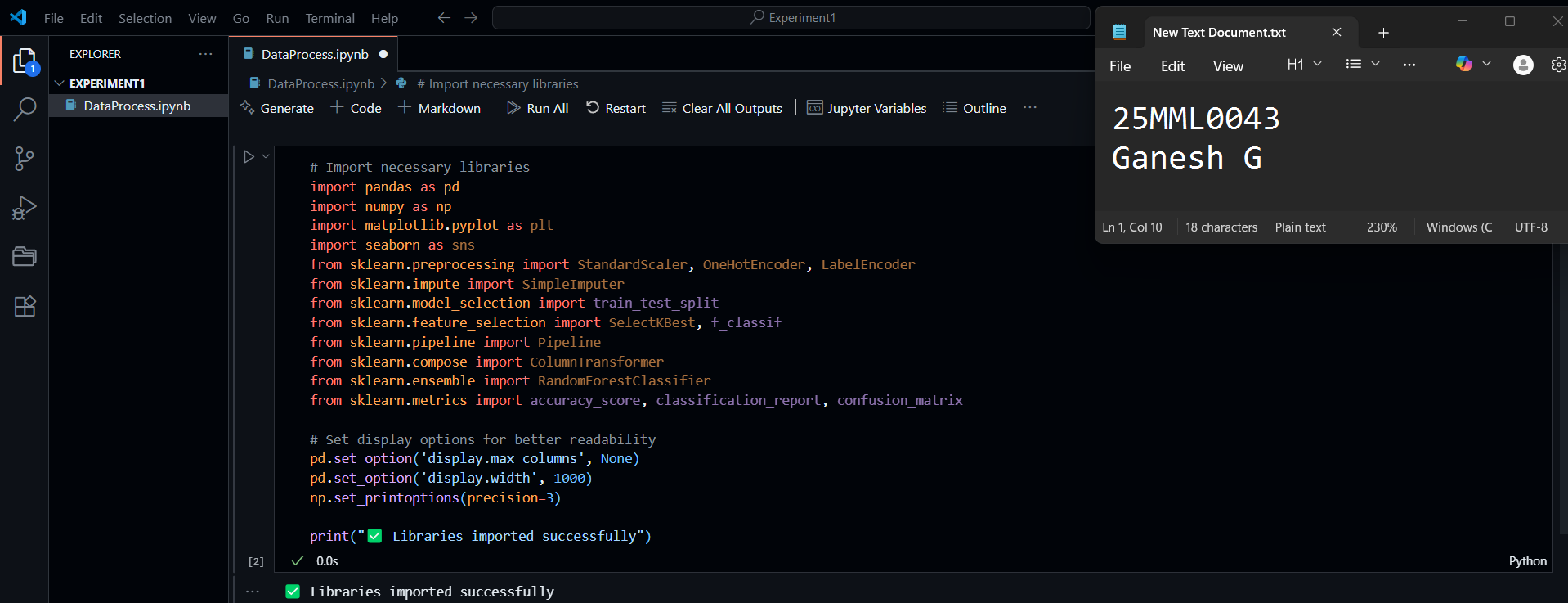
Convert categorical columns like sex, embarked into numeric form using Label Encoding or One-Hot Encoding.

1. Split the dataset

Divide the dataset into training and testing sets (e.g., 80% train, 20% test).

**IMPLEMENTATION AND RESULTS:**

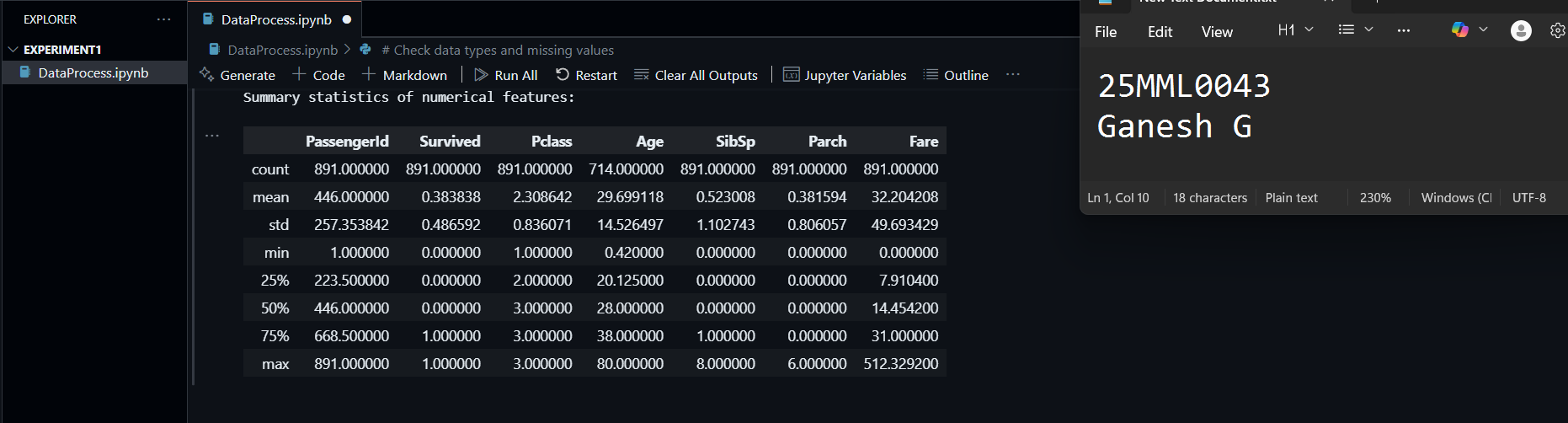
Import necessary libraries:

Data Loading:

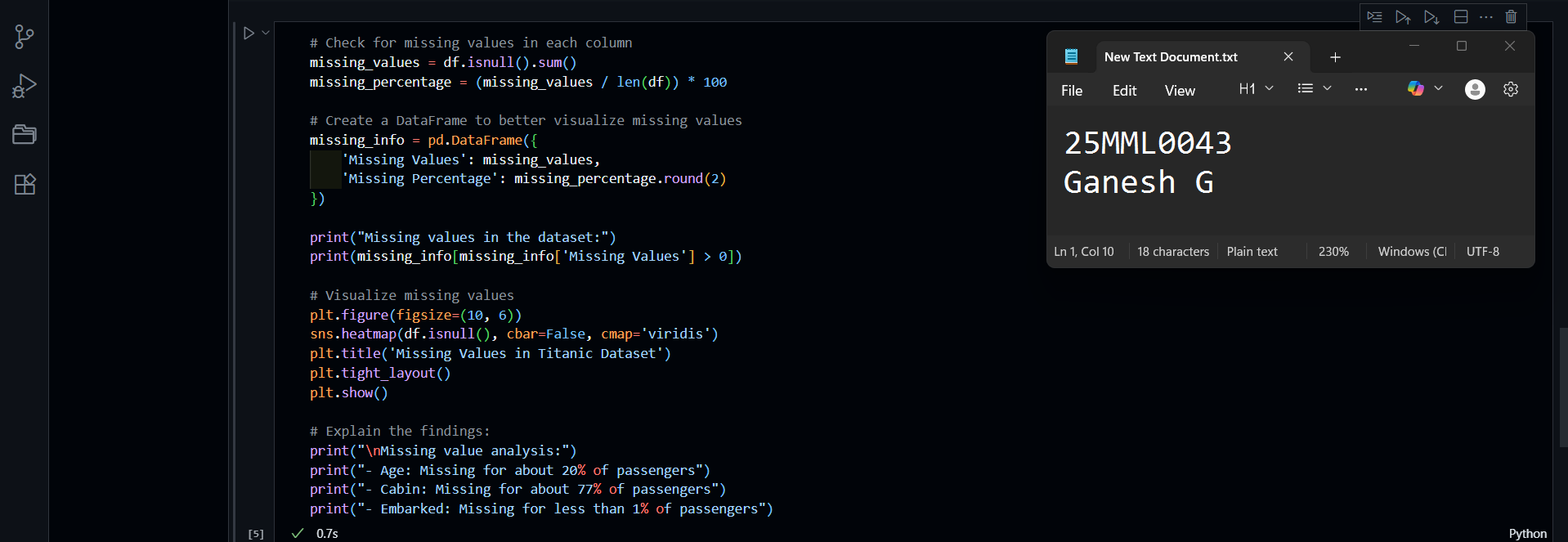


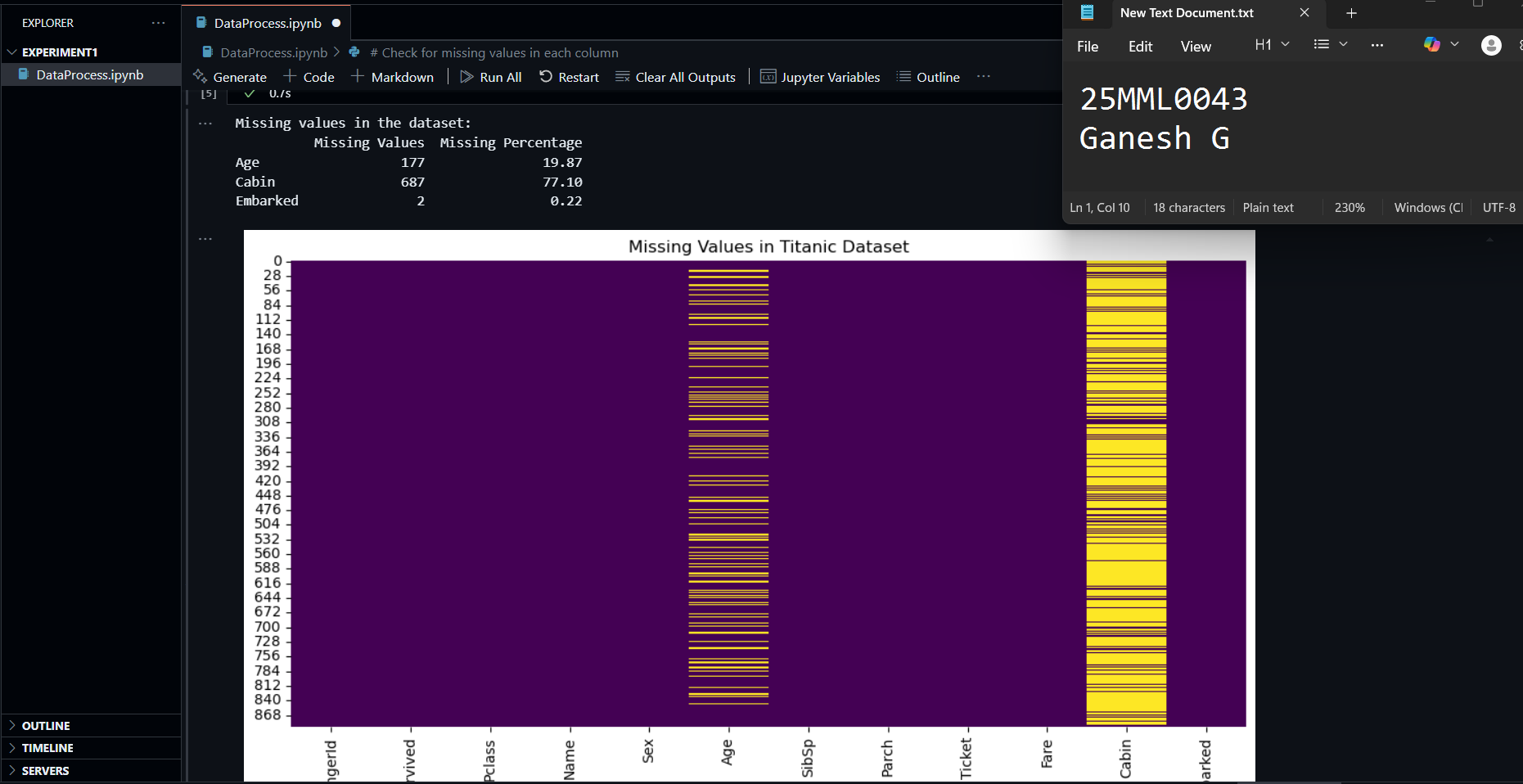
Understanding the Dataset Features:



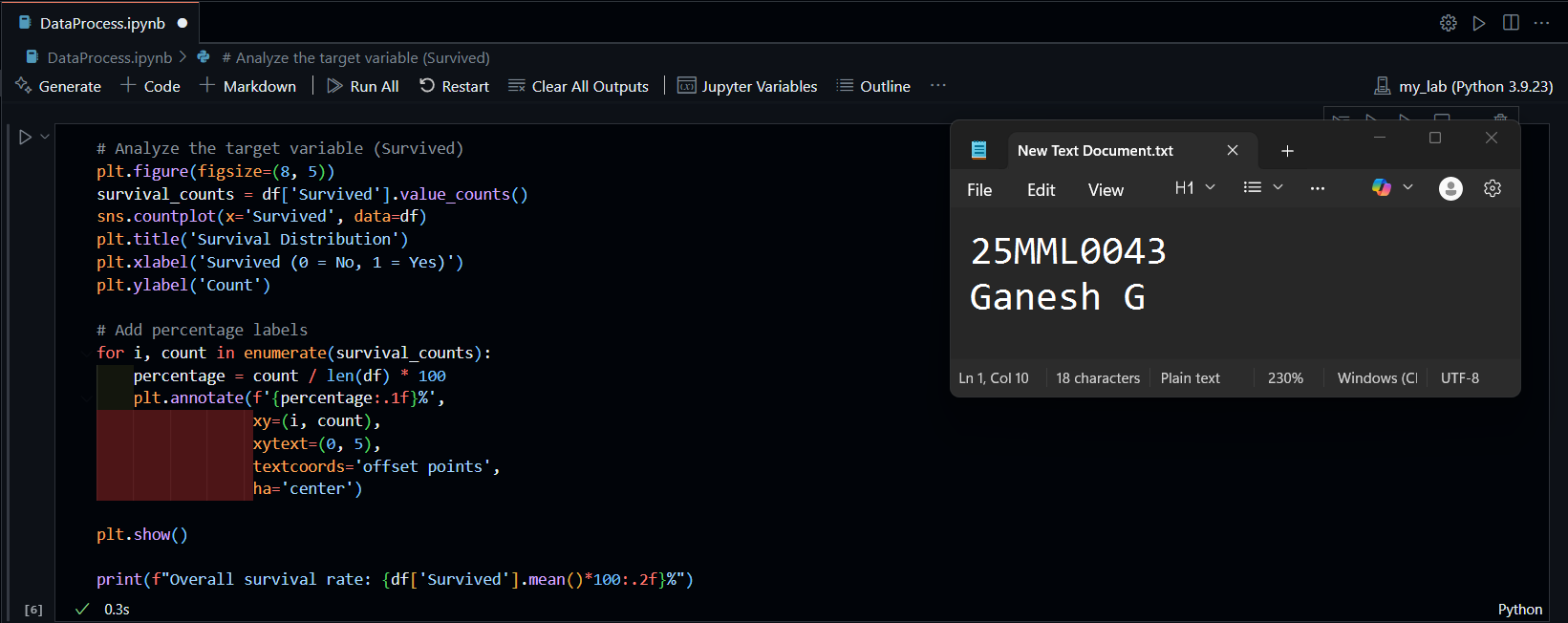


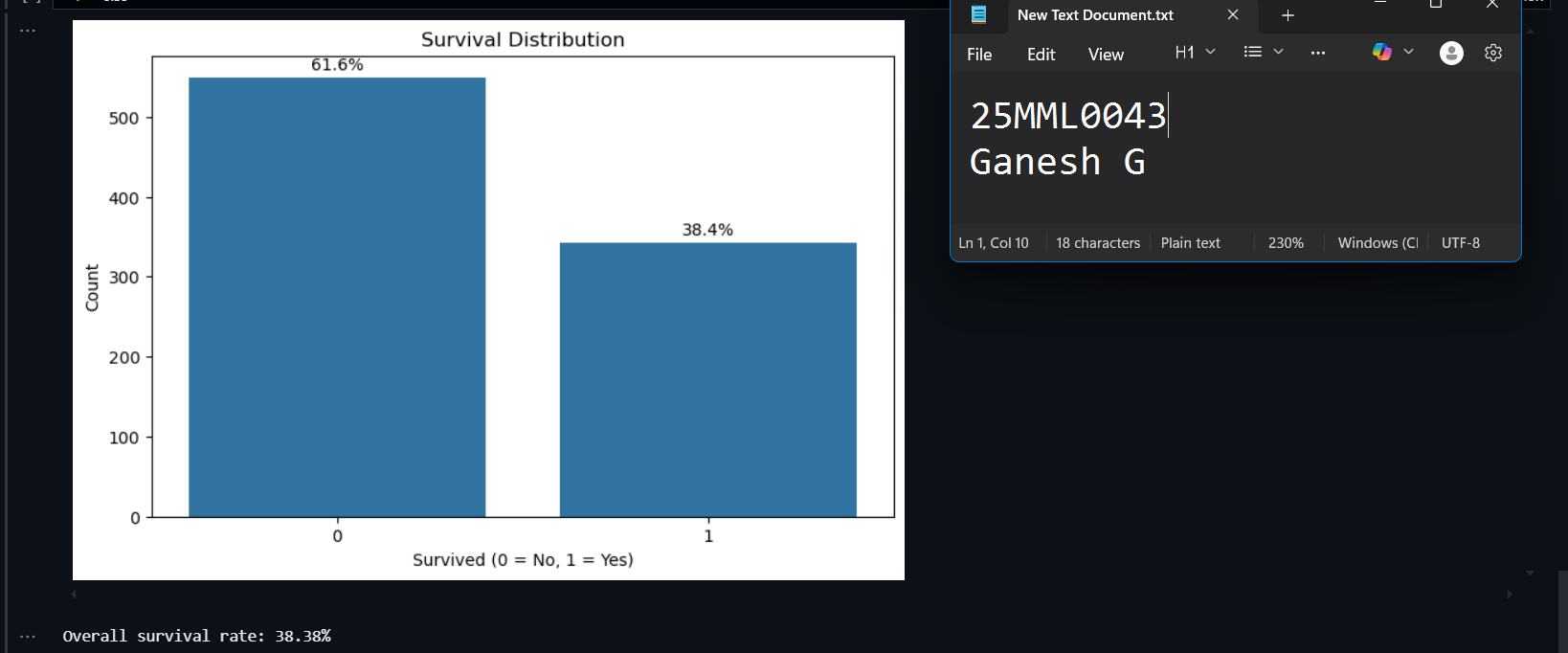
Analyzing Missing Values:





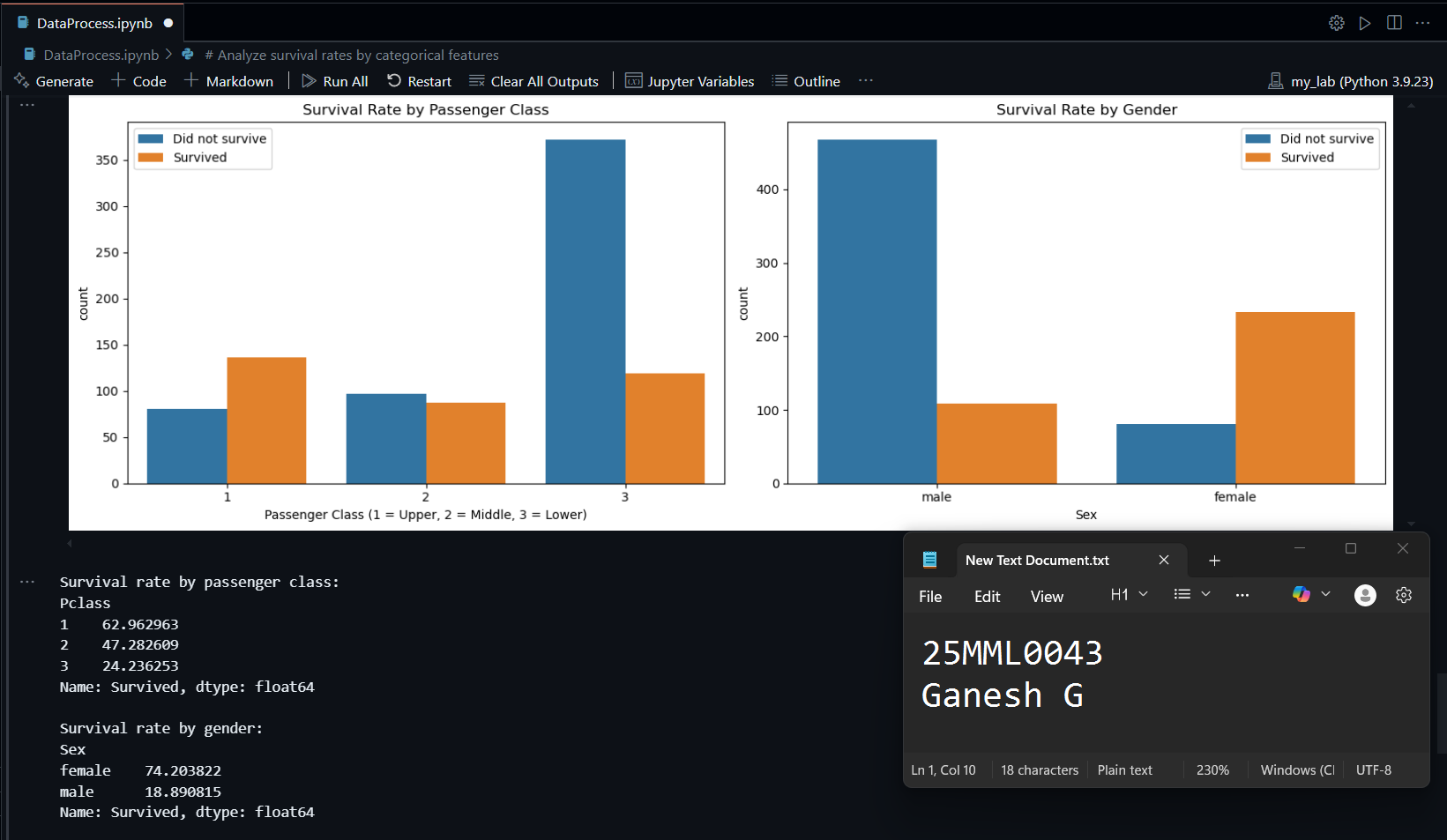
 Exploratory Data Analysis (EDA):

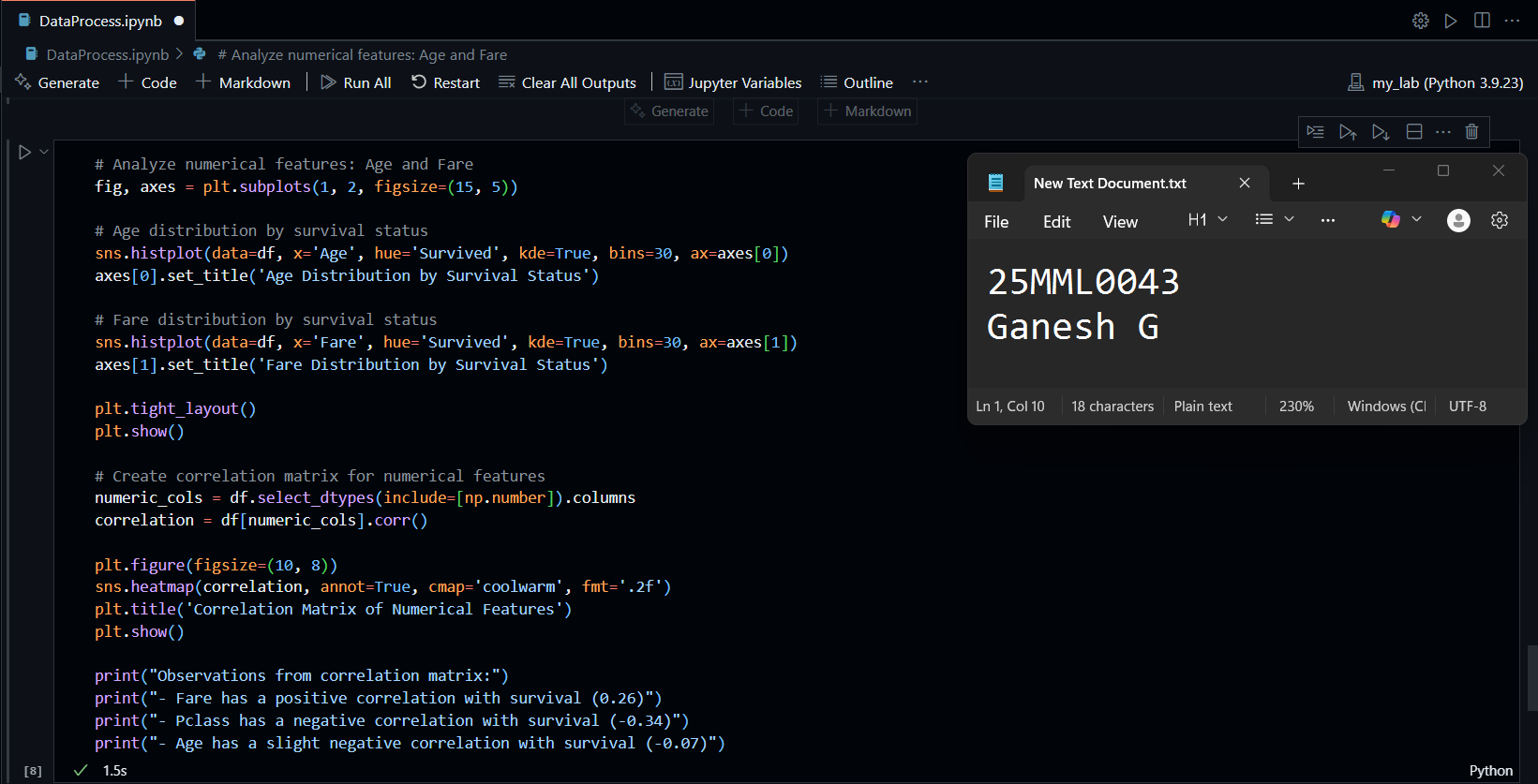


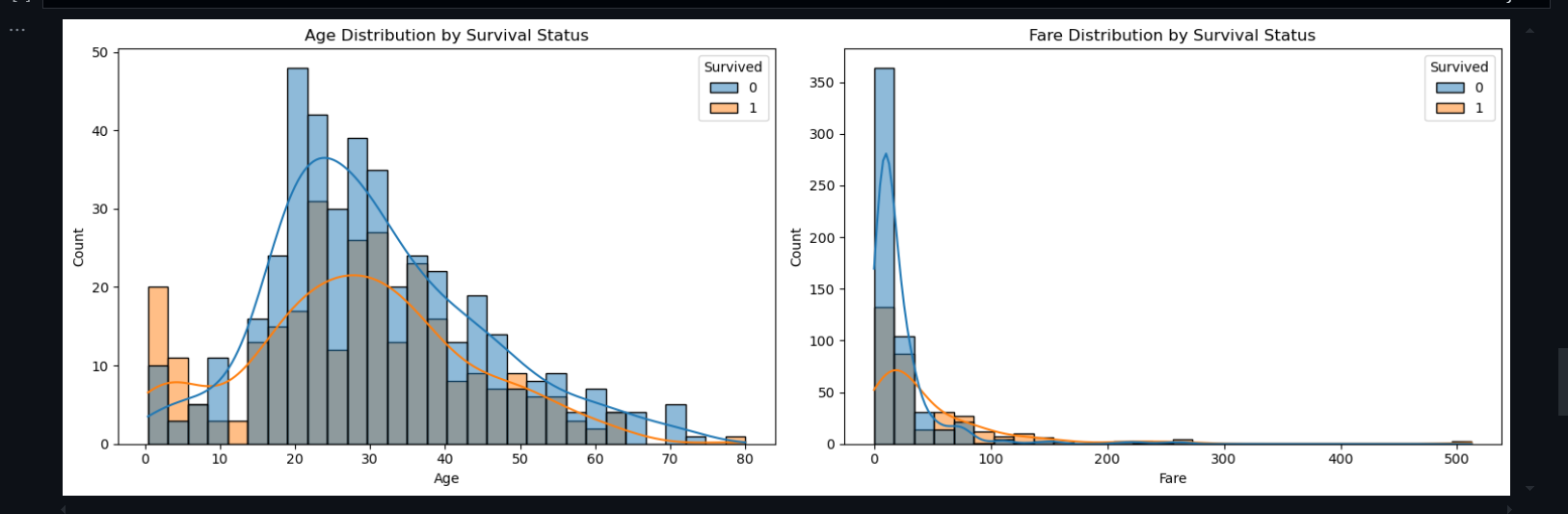


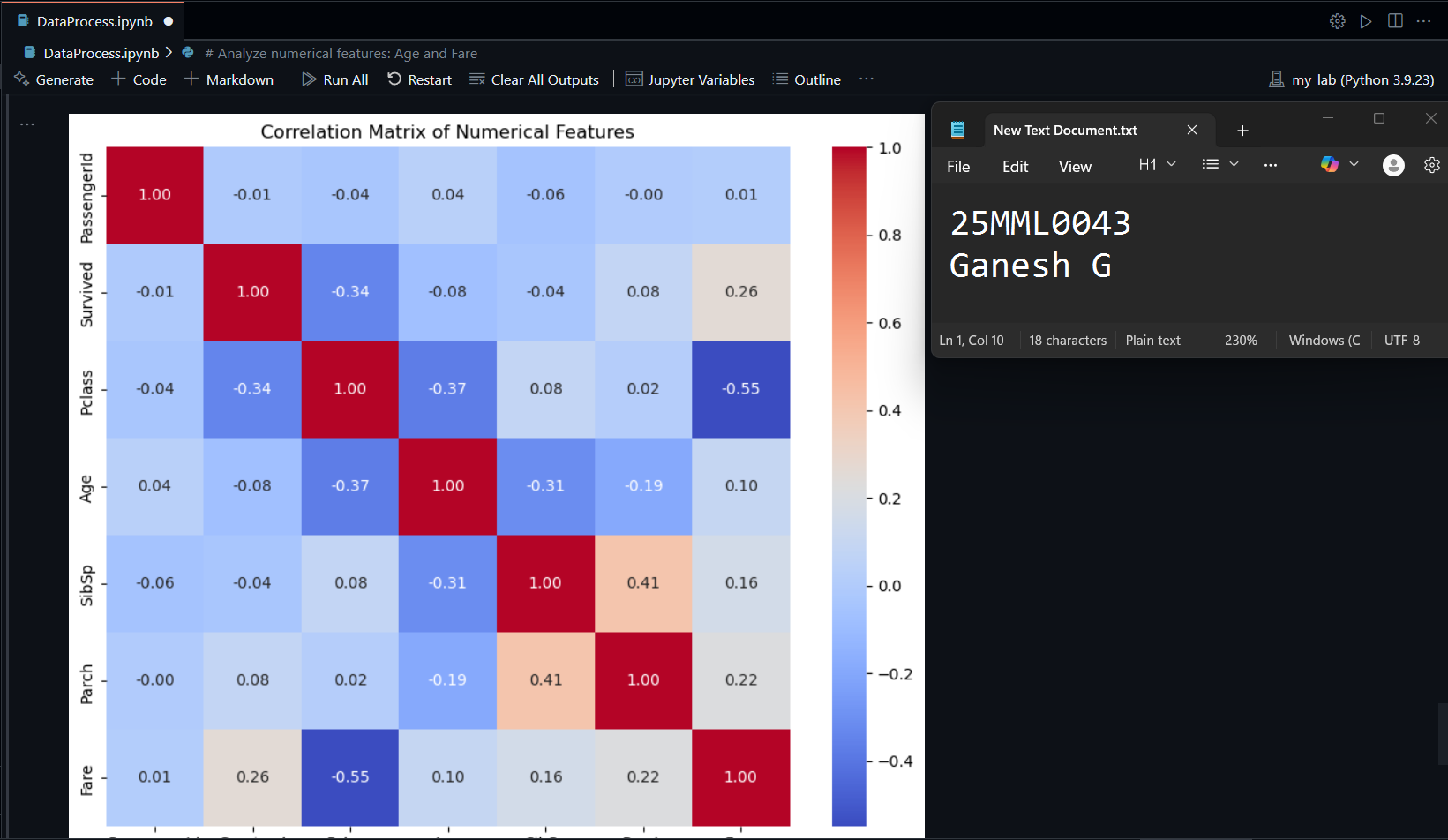
Categorical Features Analysis:

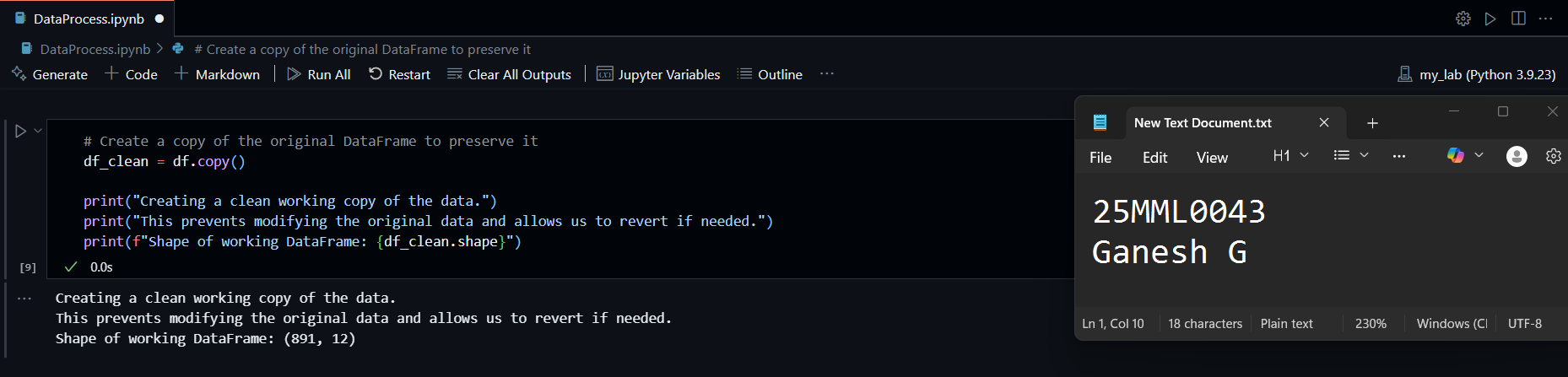


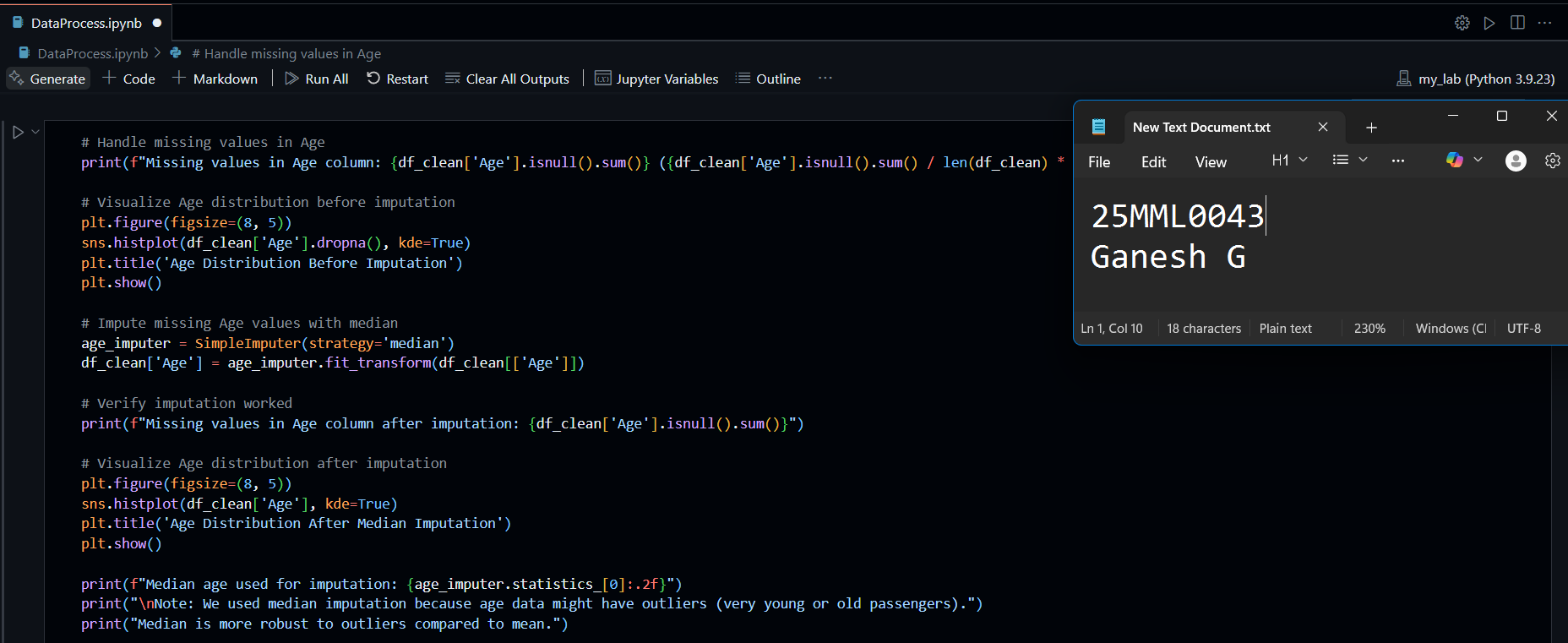


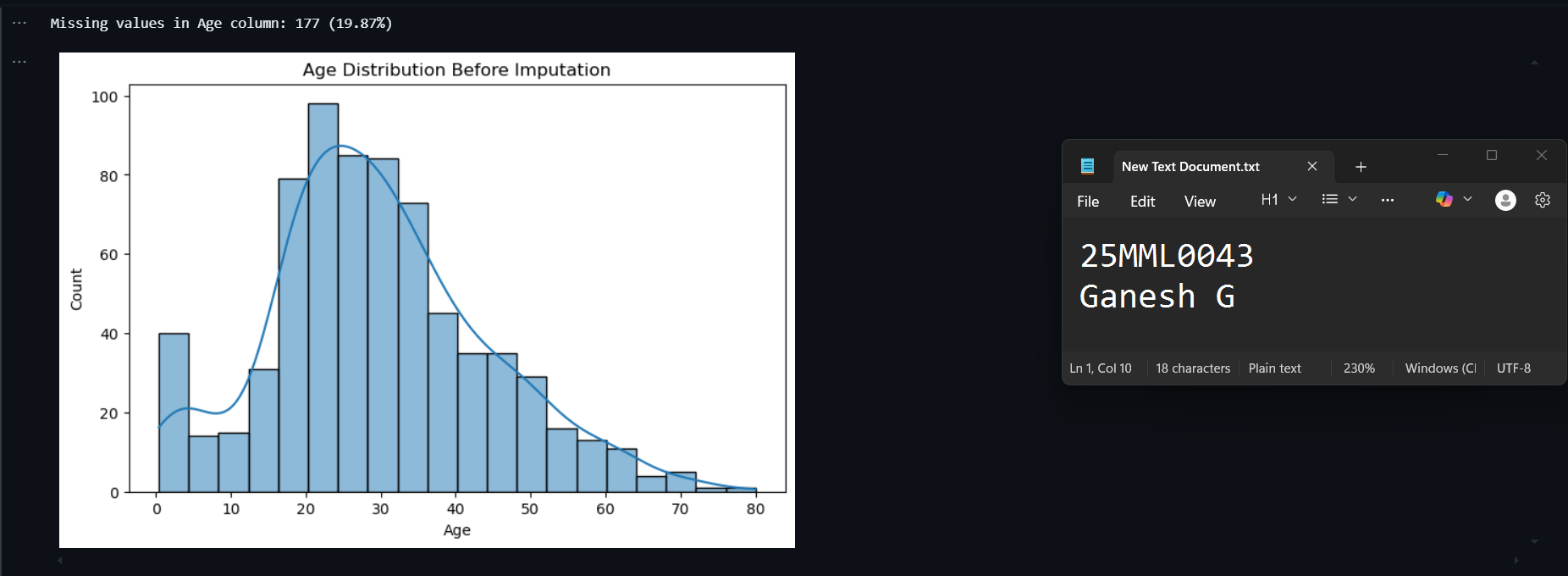


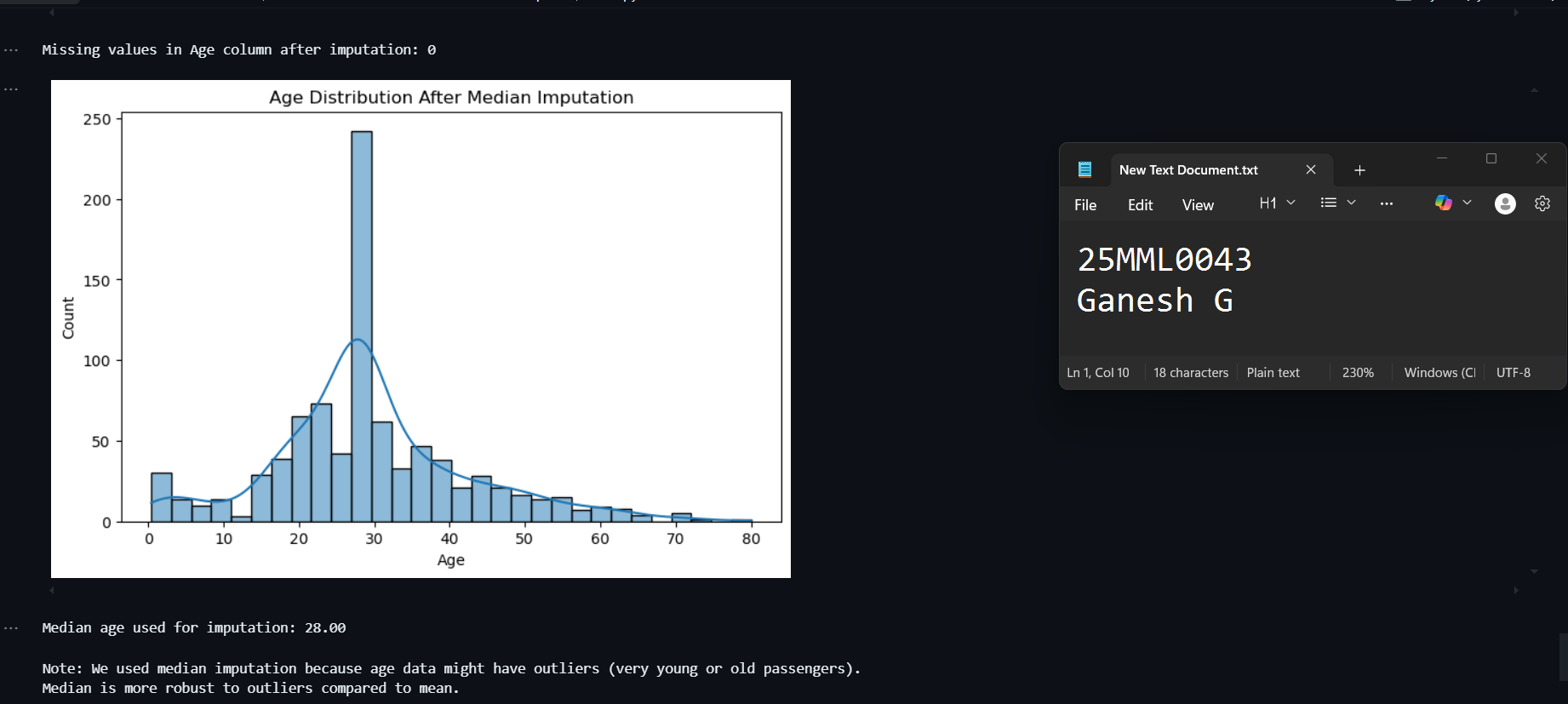


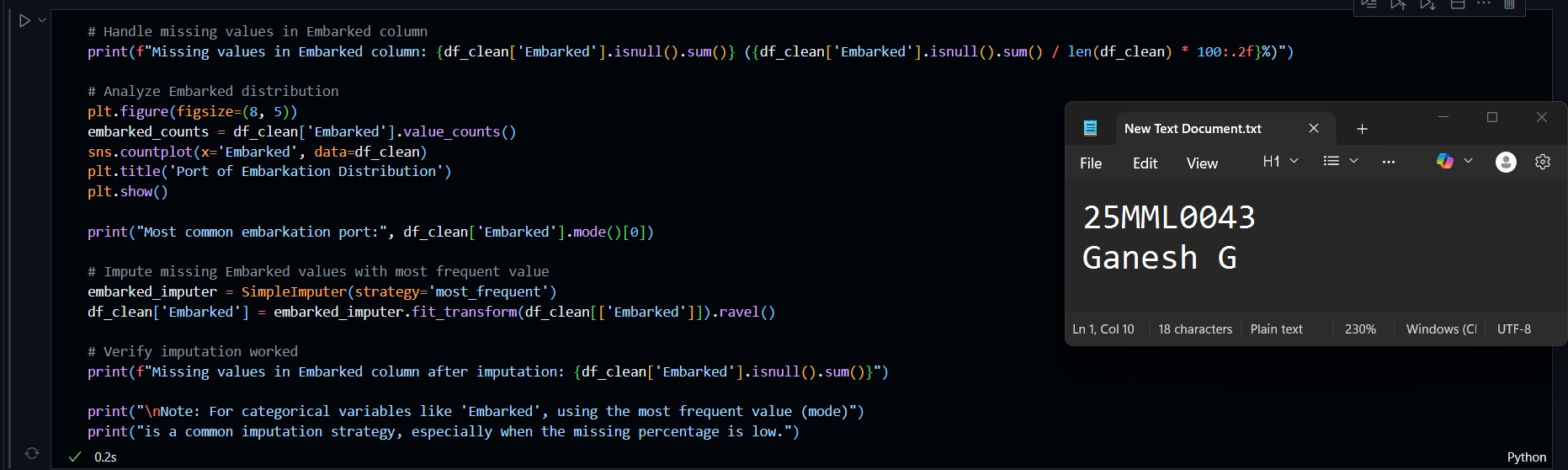


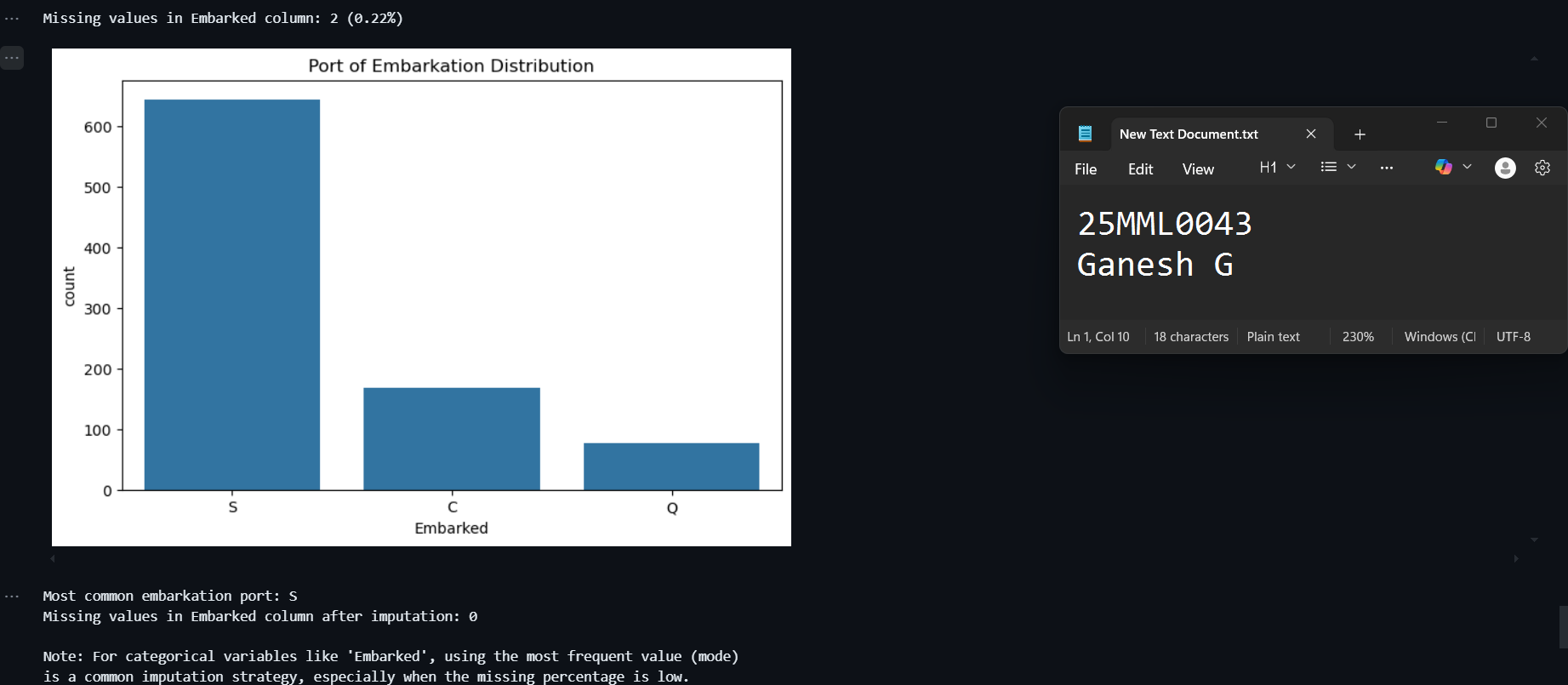


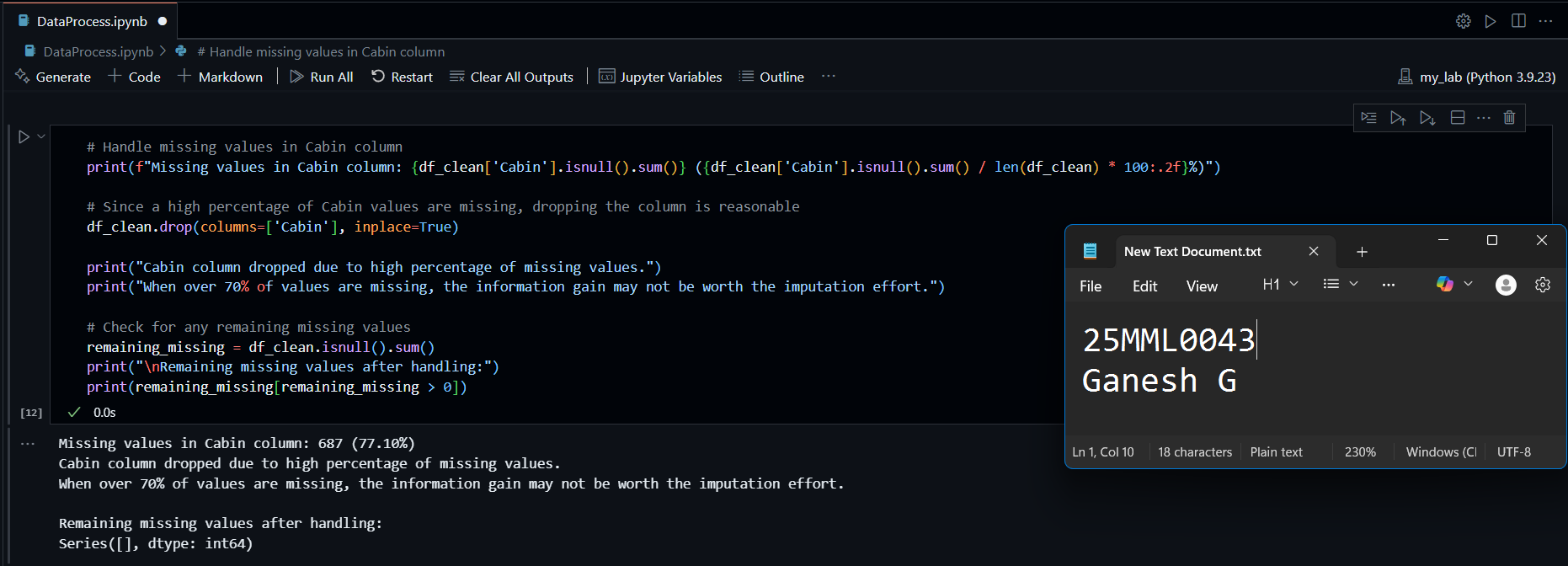


****

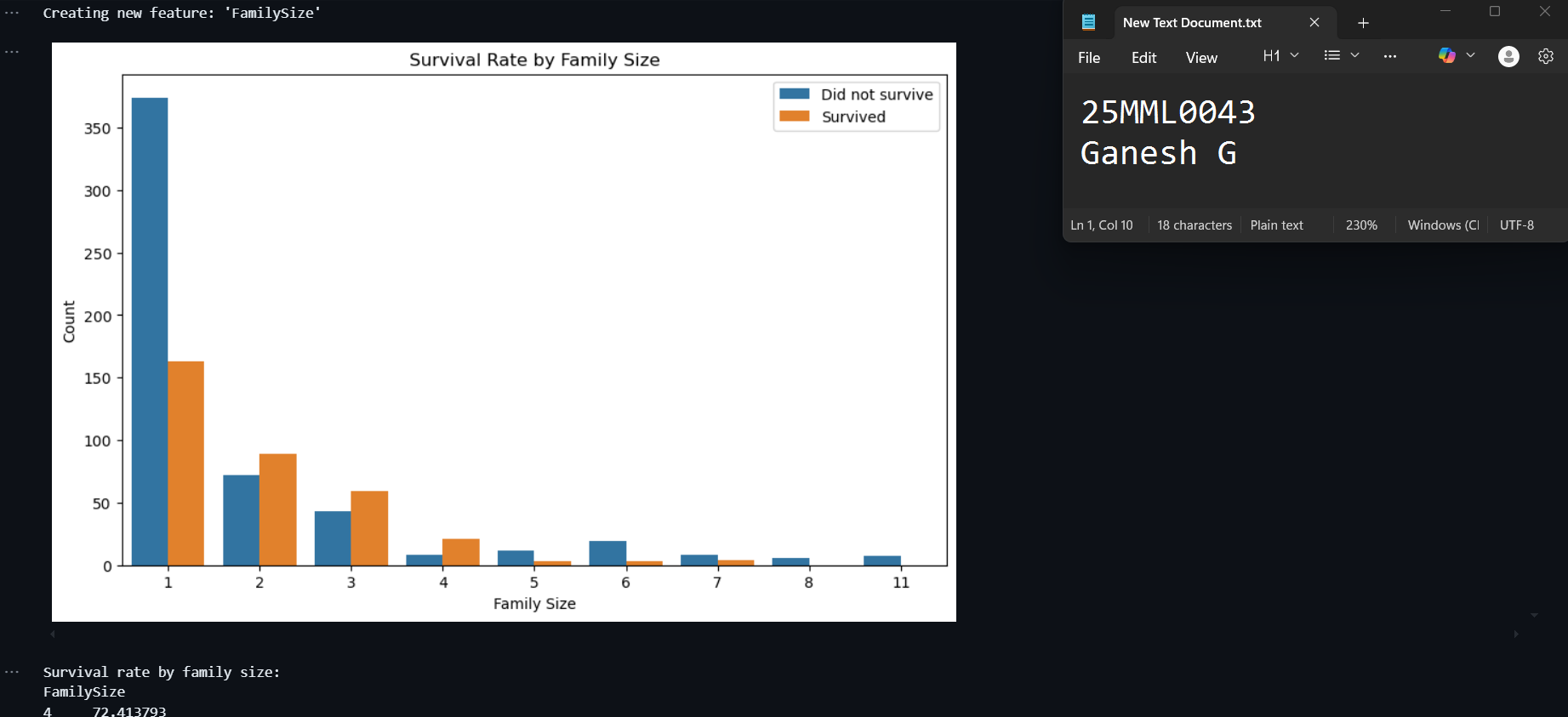
****

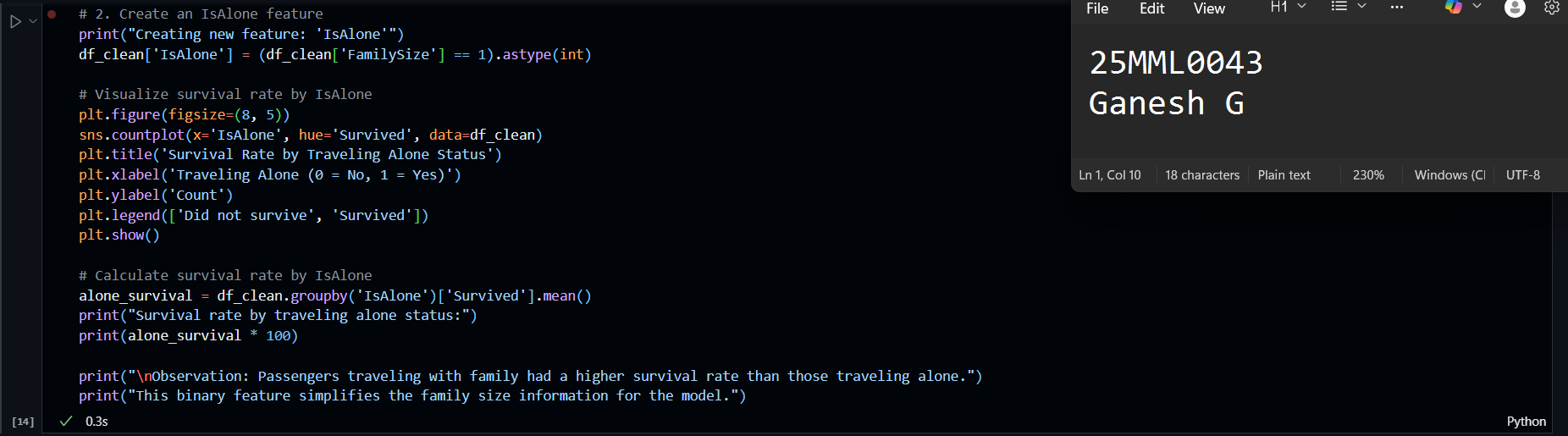
****

****

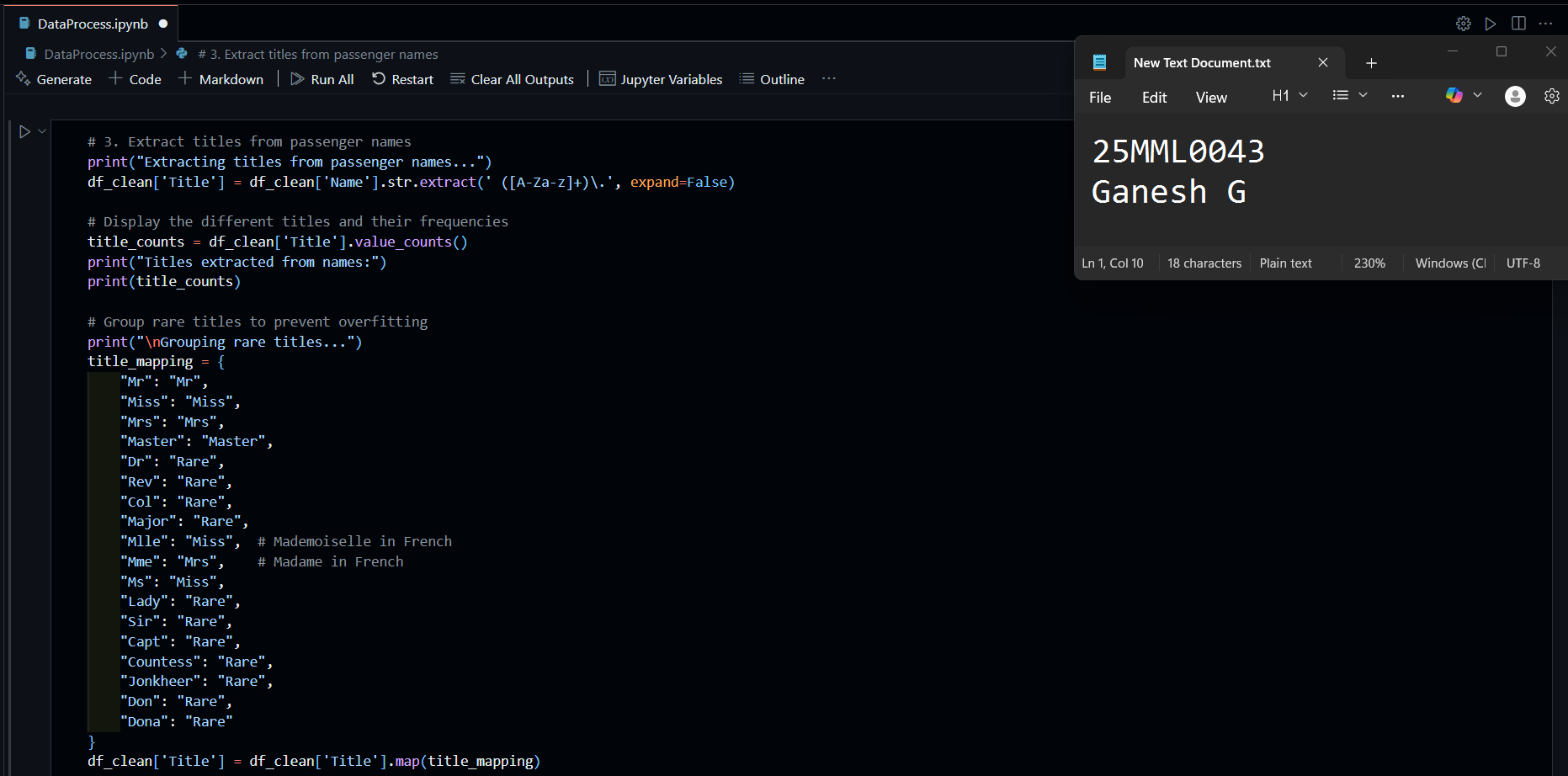
****

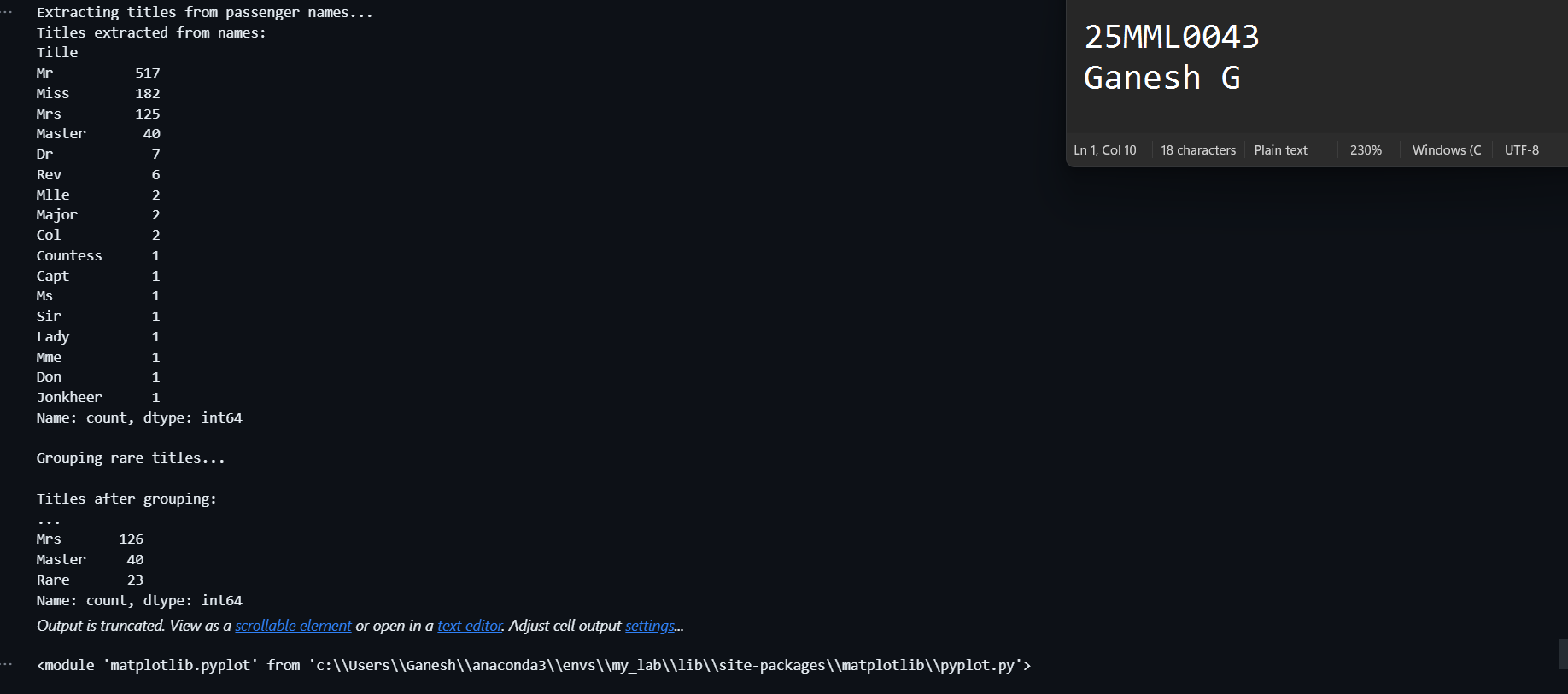
****

****

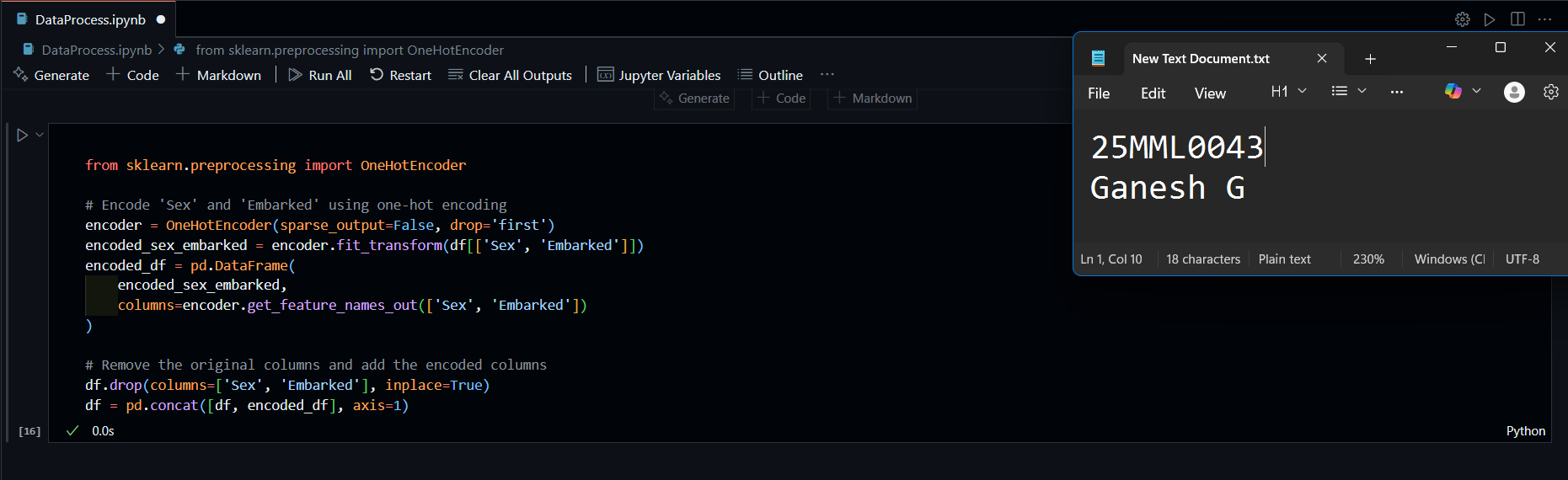
****

****

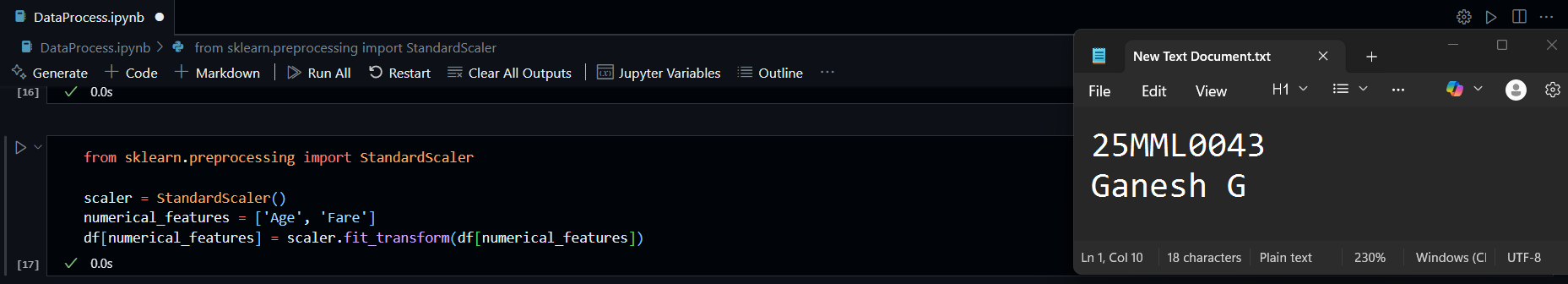
**  
**

****

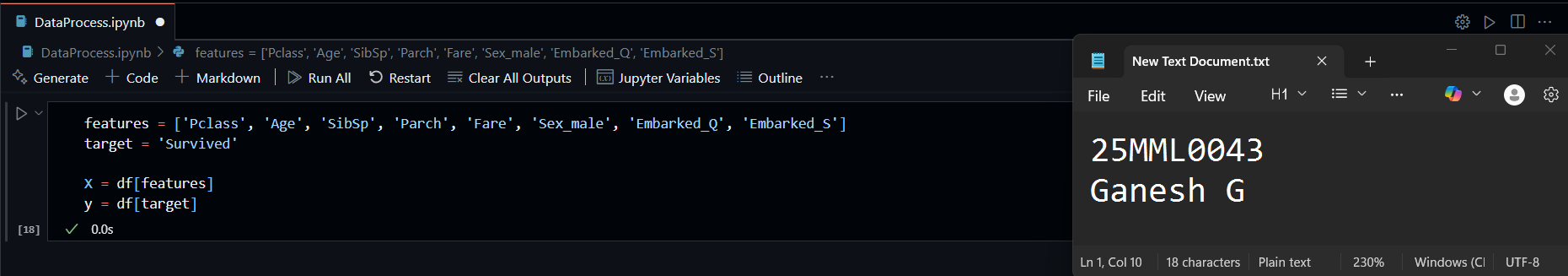
Encode Categorical Variables:



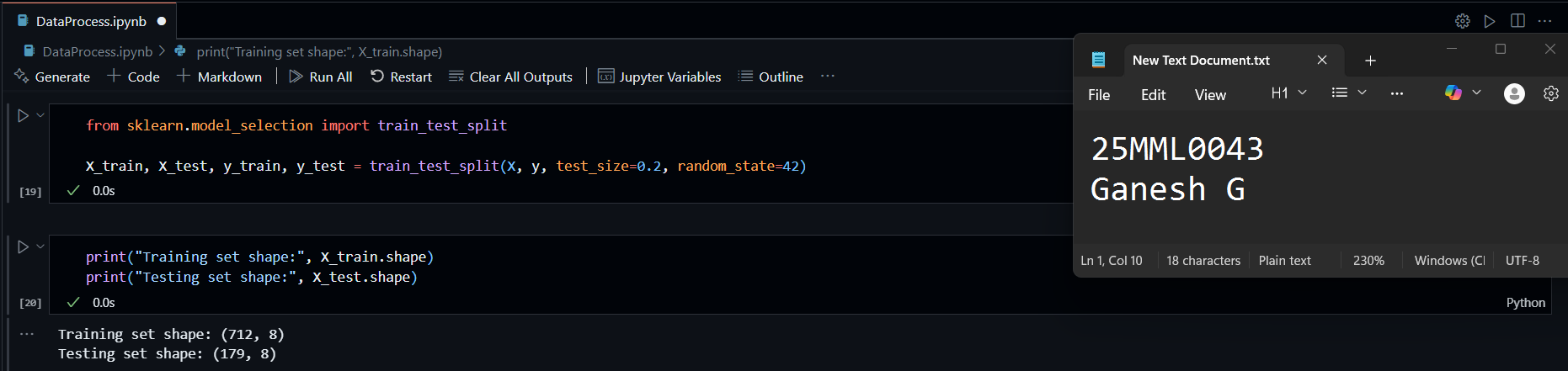
 Scale Numerical Features:



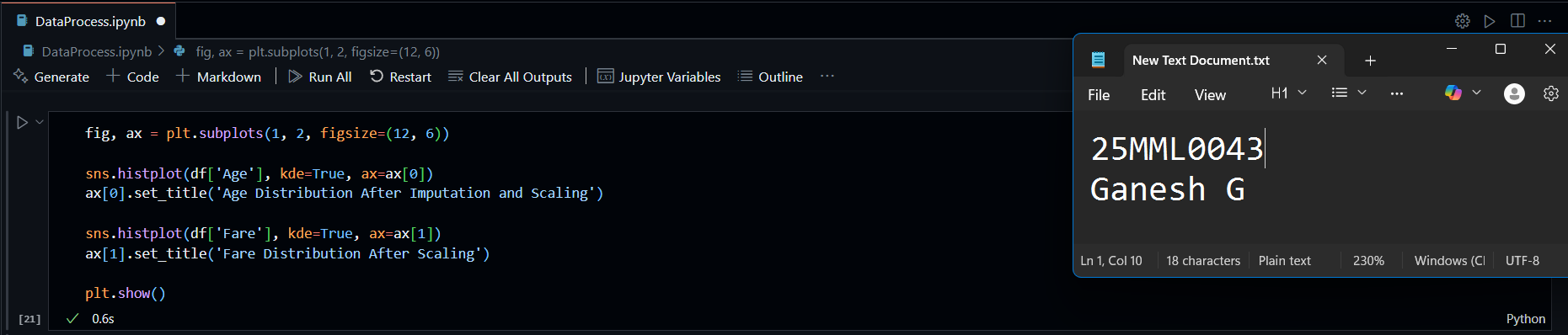
Select Features and Target Variable:

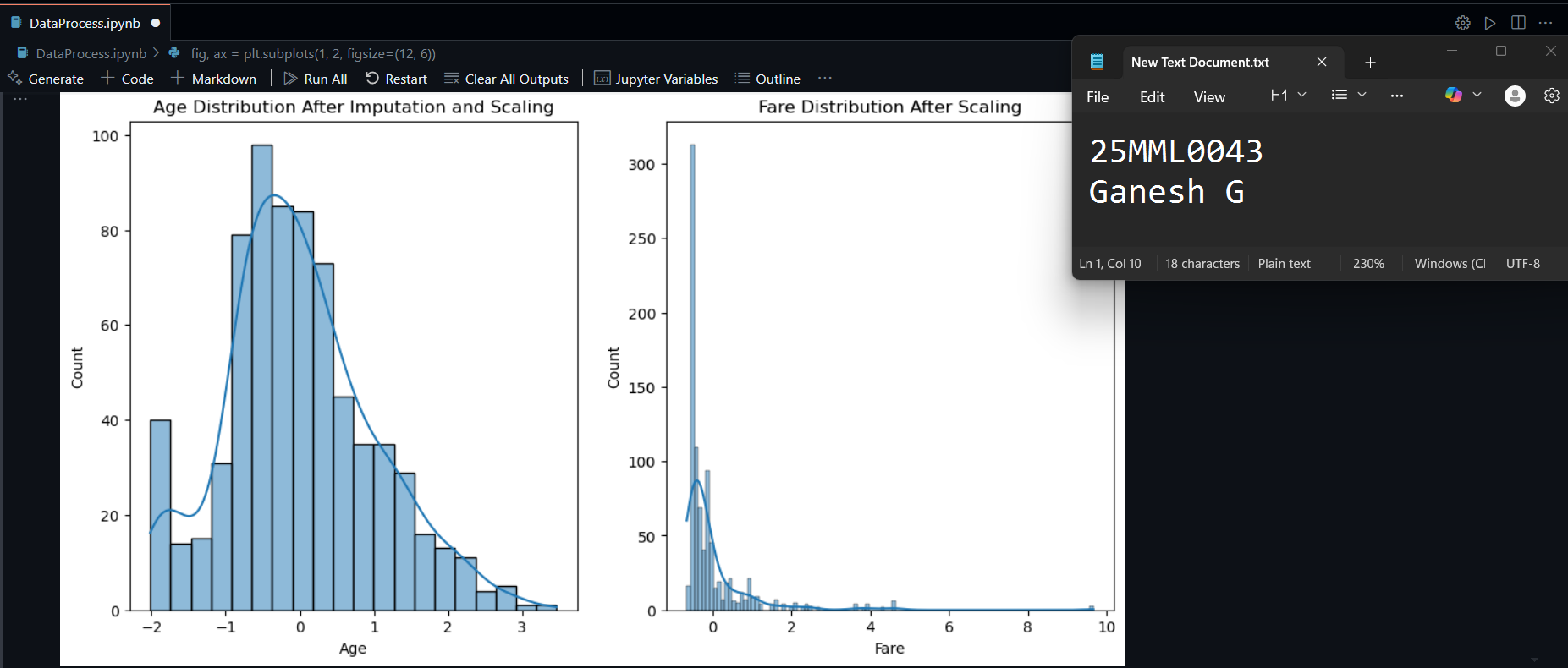


Split the Dataset



Visualize the Distribution of Numerical Features:





**LAB – 2**

**AIM:**

To implement logistic regression for binary classification using the MNIST dataset, and train a model to classify whether a given handwritten digit image is a '0' or a '1'.

**INTRODUCTION:**

Logistic Regression is a widely used algorithm for binary classification tasks. It models the probability that a given input belongs to a particular class. In this lab, we’ll use logistic regression to classify handwritten digits from the MNIST dataset — specifically to distinguish between the digits **‘0’** and **‘1’**. The MNIST dataset contains thousands of 28x28 pixel grayscale images of handwritten digits from 0 to 9. By focusing on just two classes, we can understand how logistic regression works in a straightforward binary setting. This lab will walk through loading the dataset, preprocessing the data, training the model, and evaluating its performance using accuracy and confusion matrices.

**REAL WORLD EXAMPLE:**

Security Access System (Known vs Unknown Faces):

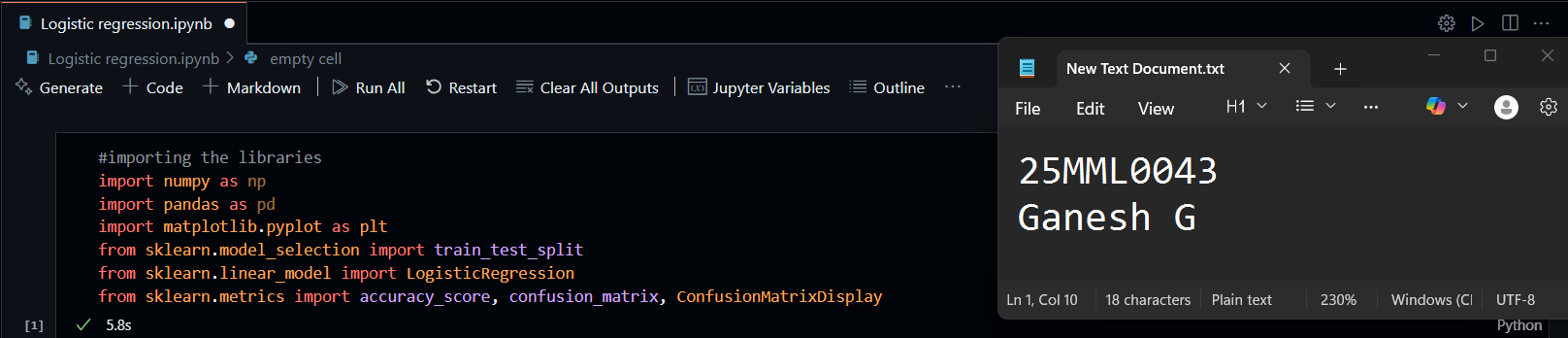
1. Image Input: A webcam or CCTV captures a **grayscale or color face image** of the person at the entrance.
2. Preprocessing: Resize all face images to the same size (e.g., 64×64).
3. Labelling Data: Known as (1) and Unknown as (0)
4. Model Training with Logistic Regression
5. Prediction & Real-Time Use: At the door, the system captures and preprocesses a face image, flattens it into a vector, and feeds it into a logistic regression model. If the prediction is 1, access is granted; if 0, a security alert is triggered.

**ALGORITHM:**

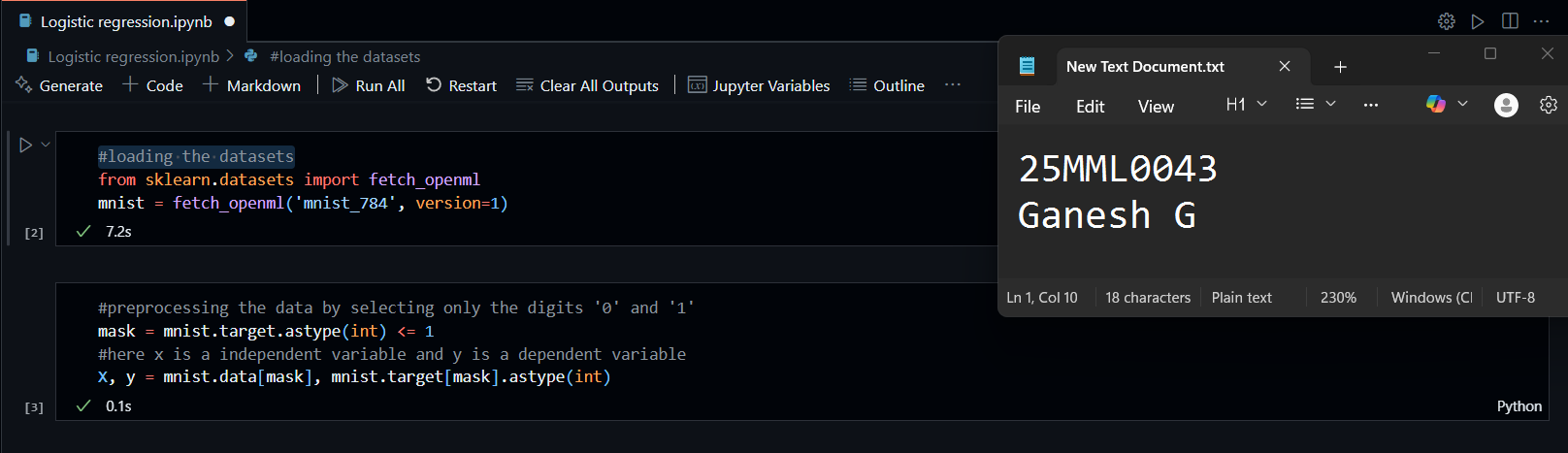
1. Begin by importing the necessary libraries such as NumPy, pandas, matplotlib, and modules from scikit-learn.
2. Load the MNIST dataset using the fetch\_openml() function from sklearn.datasets.
3. Filter the dataset to include only the images labeled as '0' and '1', since this is a binary classification task.
4. Normalize the pixel values of the images by dividing by 255 to bring all values into the range [0, 1].
5. Split the dataset into training and testing sets using train\_test\_split() to prepare for model training and evaluation.
6. Initialize the logistic regression model using LogisticRegression() from sklearn.linear\_model.
7. Train the model by fitting it to the training data using the .fit() method.
8. Predict the labels for the test data using the .predict() method.
9. Evaluate the model’s performance using metrics such as accuracy score and the confusion matrix.
10. Visualize the confusion matrix and some sample predictions to better understand how the model is performing.

**IMPLEMENTATION AND RESULTS:**

Import Libraries



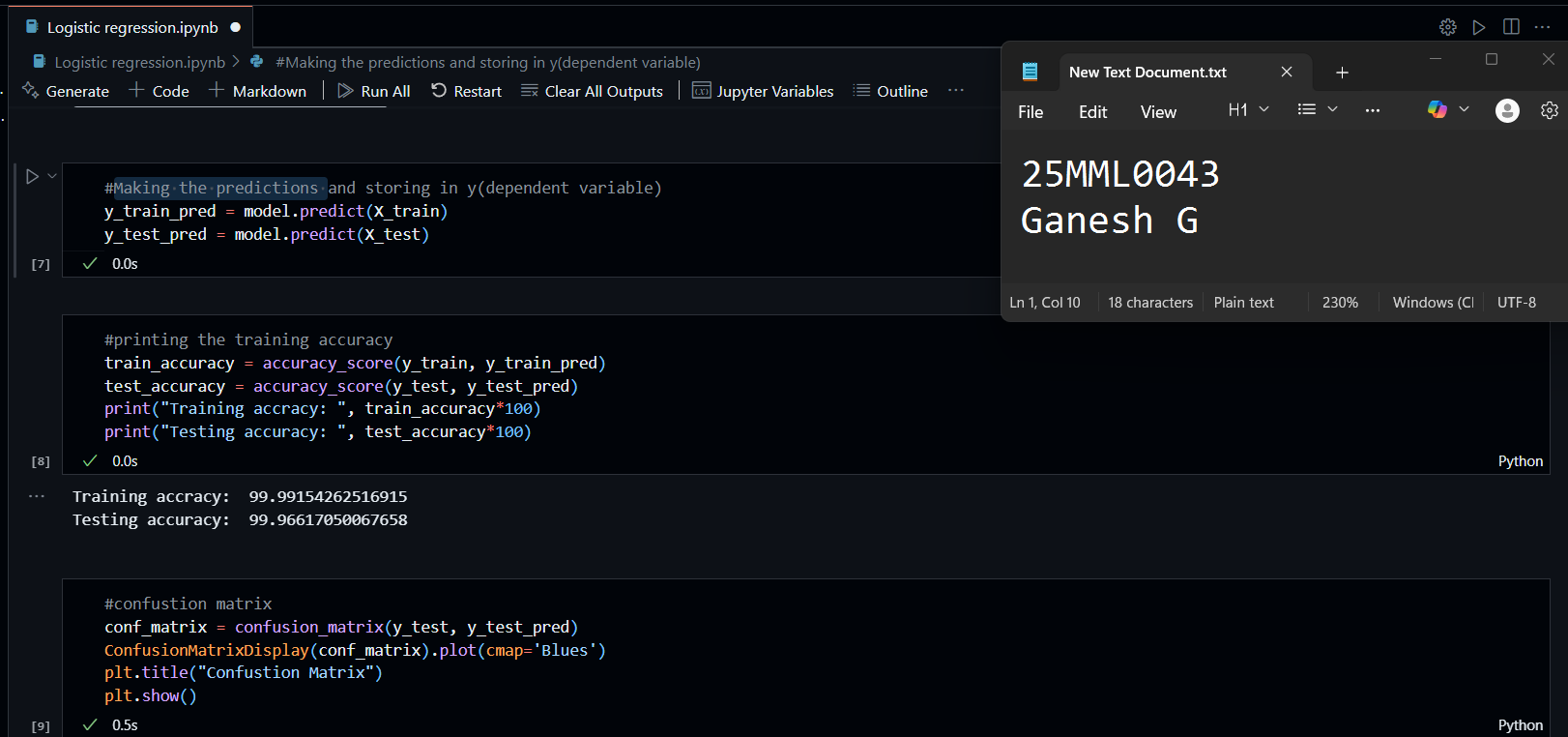
Load and preprocess data

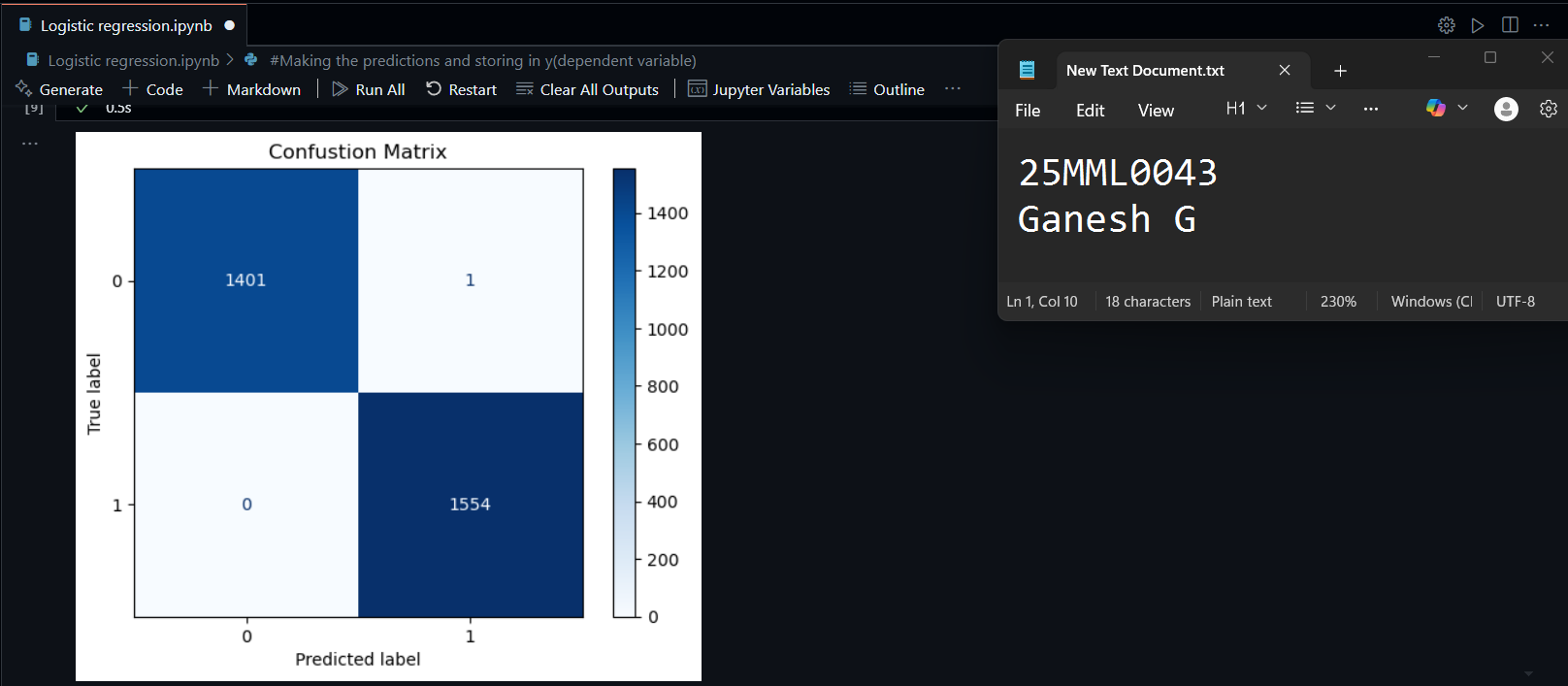


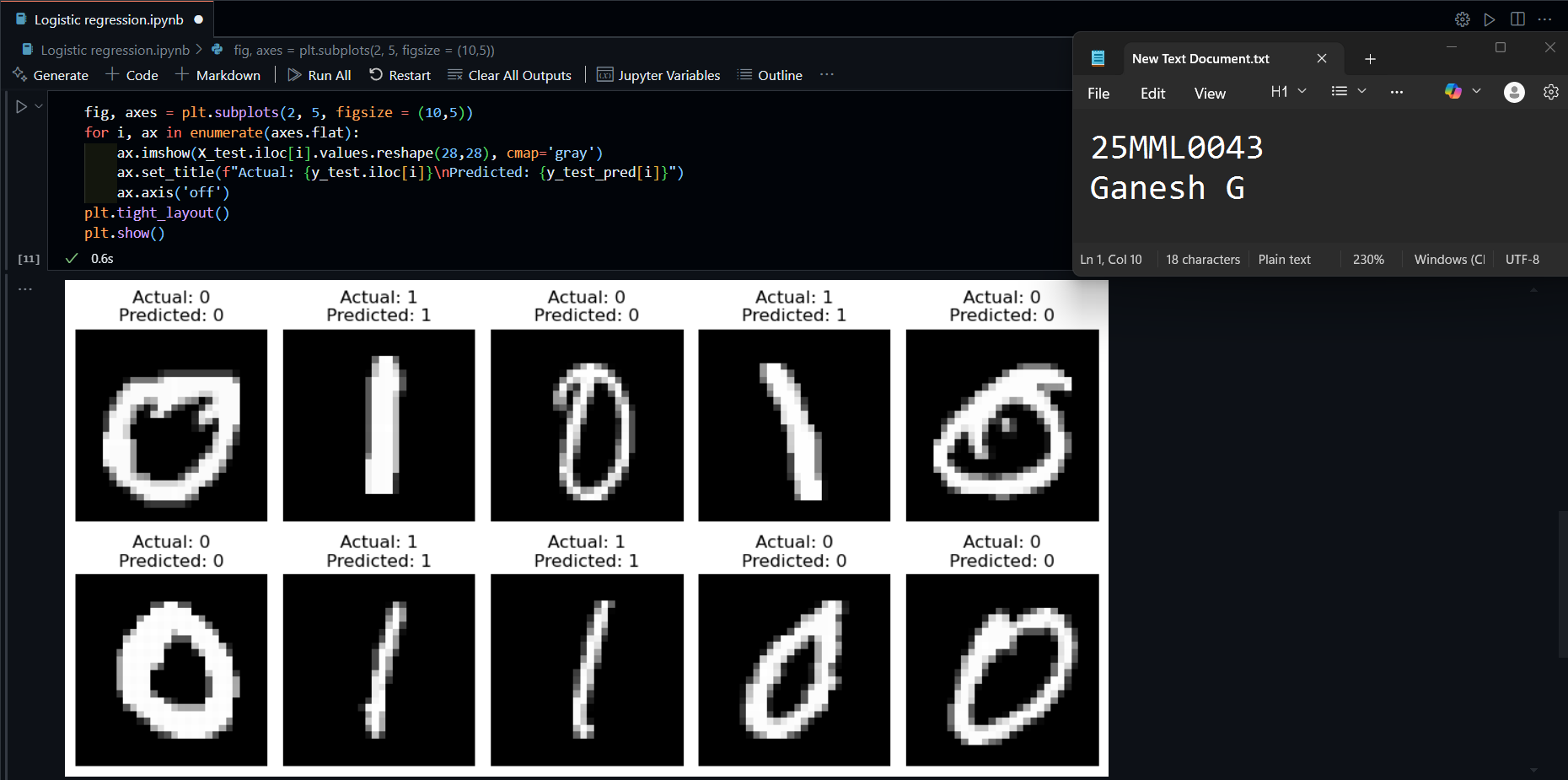
Data Normalization and Training



Making the predictions







**LAB – 3**

**AIM:**

Predicting gold prices using 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.

**INTRODUCTION:**

Gold price prediction is an important task in financial analytics, where historical data is used to forecast future prices. Linear regression is a simple yet powerful algorithm that models the relationship between input variables (such as oil price, exchange rates, stock indices, etc.) and a continuous target variable (gold price). In this experiment, we will use historical gold price data, apply preprocessing, and train a linear regression model to make predictions.

**REAL WORLD EXAMPLE:**

Predicting Used Car Prices Using Linear Regression:

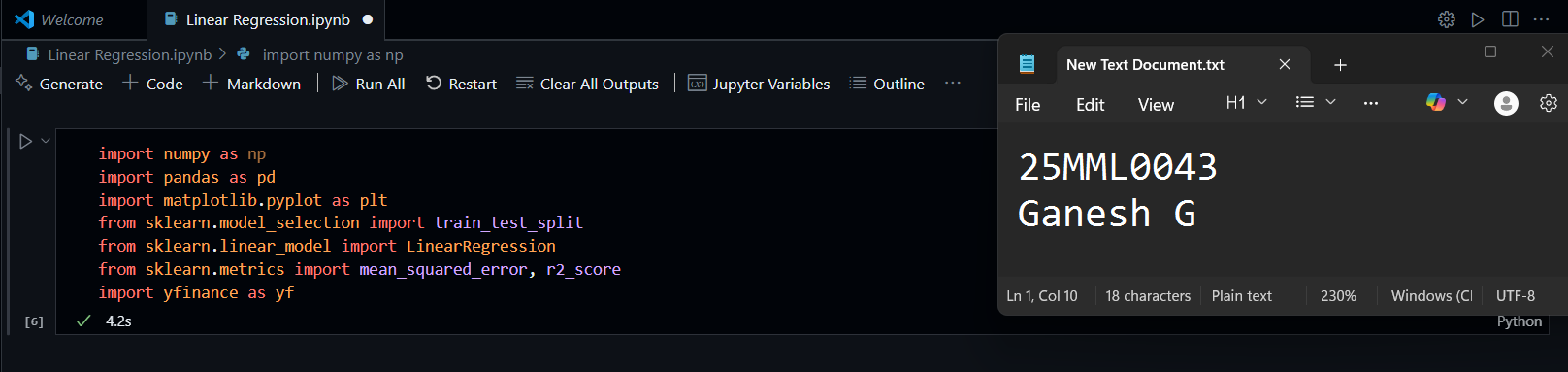
1. Data Collection:The company collects a dataset of thousands of used car listings from their platform
2. Data Preprocessing
3. Train Linear Regression Model
4. Prediction Example: Suppose a seller enters the following car info
5. Model Evaluation: Mean Absolute Error (MAE), Mean Squared Error (MSE)

**ALGORITHM:**

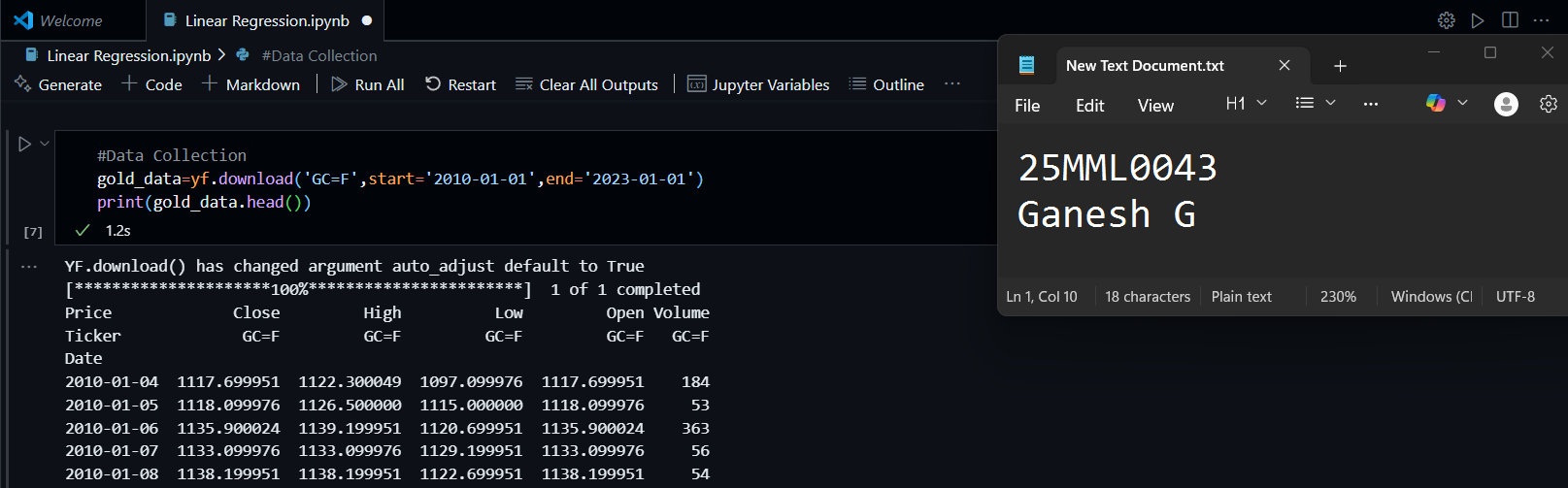
1. Begin by importing the necessary libraries including pandas, NumPy, matplotlib, seaborn, and Scikit-learn modules for regression and evaluation.
2. Load the historical gold price dataset into a DataFrame.
3. Convert the date column to datetime format and set it as the index for time-series structure (if applicable).
4. Explore and visualize the data to understand trends and detect any missing values or anomalies.
5. Create lag features, such as using the previous day's gold price to predict the current day's price.
6. Drop rows with missing values caused by lagging.
7. Define the input features (X) and target variable (y) for the regression model.
8. Split the dataset into training and testing sets using train\_test\_split() to evaluate model performance.
9. Initialize and train the LinearRegression model using the training data.
10. Use the trained model to predict gold prices on the test dataset.
11. Evaluate the model using metrics such as Mean Squared Error (MSE) and R-squared (R²) score.
12. Plot and compare the actual and predicted prices to visually assess model performance.

**IMPLEMENTATION AND RESULTS:**

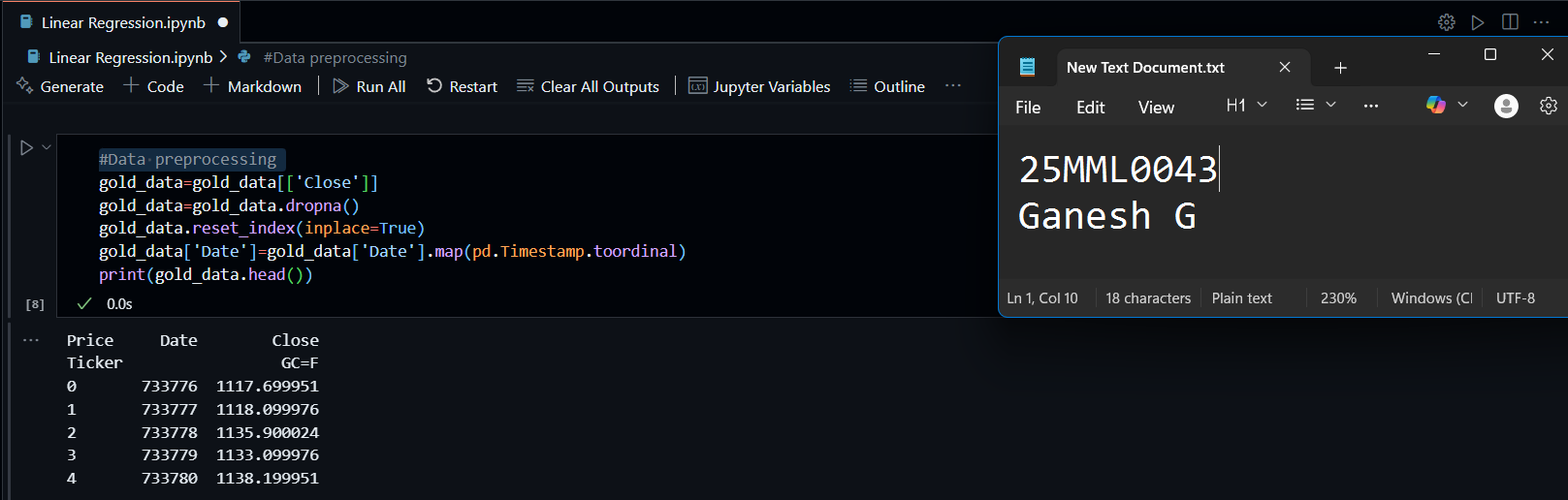
Import Libraries



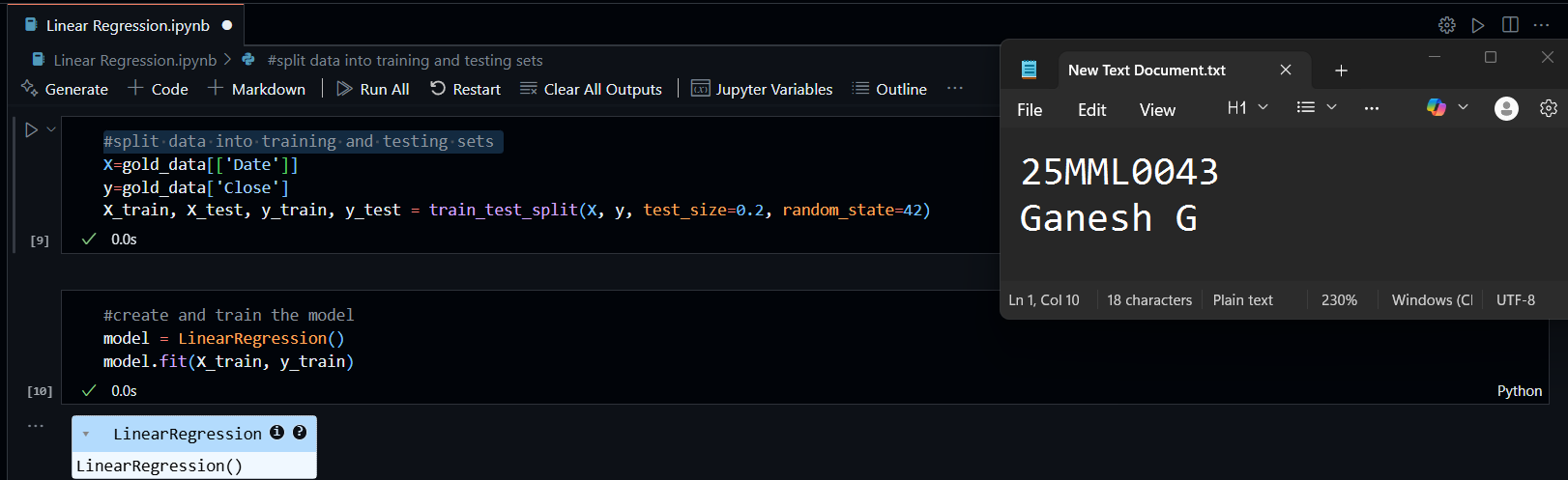
Data Collection



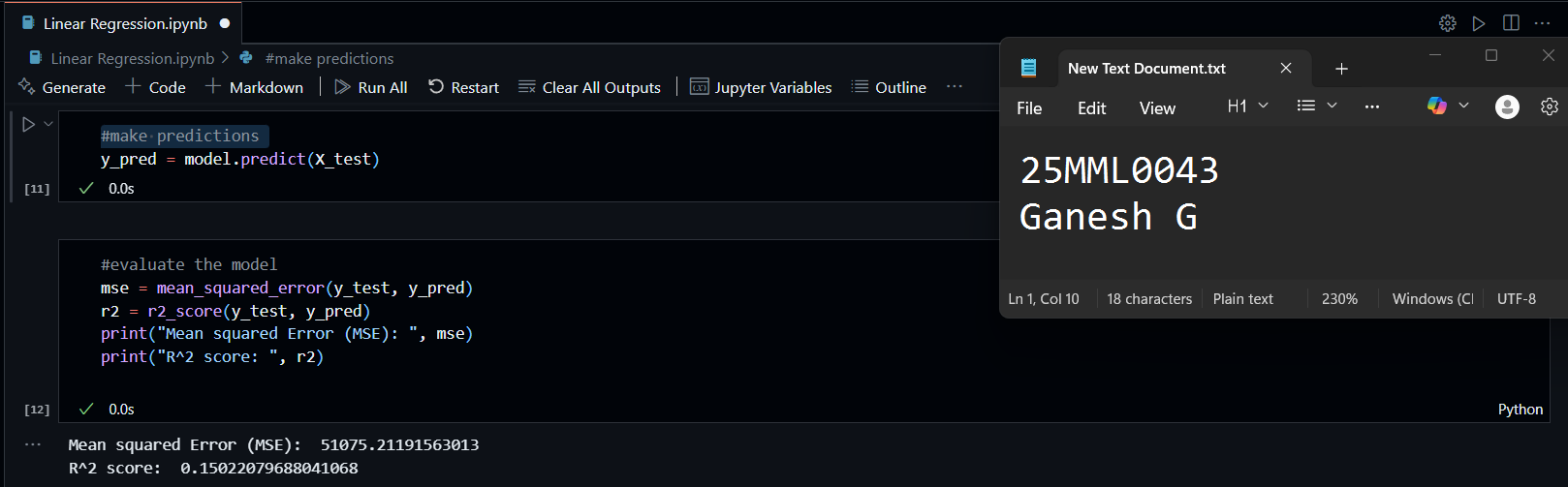
Data preprocessing



split data into training and testing sets



make predictions



visualize the results

