**SHRIDEVI POLYTECHNIC**

**SIRA ROAD, TUMKUR**

**2022-23**

**Department of Computer Science Engineering**

MINI PROJECT 01

Classification Task- Machine Learning

**Machine Learning Project Report on**

**Machine Learning Project Report on**

**“Loan Prediction Report A Classification Task using Logistic Regression”**

**Pathway :** Artificial Intelligence and Machine Learning

**Code :** 20CS51I

**Semester:** 5th sem

**Under Guidance of Cohort Owner:**

Yamuna.H. HOD, CSE

**Team Members:**

Abhishek H P 533CS2001

Niveditha K S S533CS2007

* **Phase I**

**Project Assignment: Classification Project - ML**

The goal of this project is to build a machine learning model to perform classification analysis on a dataset of your choice. To select the appropriate classification algorithm, preprocessing the data, training the model, evaluating its performance, and making predictions using the trained model.

**Key Deliverables:**

* A detailed report describing the problem statement, dataset selection, and data preprocessing steps.
* A report on the selection of the appropriate classification algorithm and its implementation.
* Evaluation of the model's performance using appropriate metrics.
* A presentation demonstrating the use of the trained model to make predictions on new data.
* Code and documentation for the entire project.

**Duration:** Approximately 4-5 weeks

**Team Size:** Individual or Team of 2

**Skills Required:** Basic understanding of machine learning concepts, programming skills in Python, and experience with data analysis and visualization tools such as Pandas, Matplotlib, and Seaborn.

* **Phase II**

**Problem Statement**

Lenders face a major challenge in determining whether to approve or deny a loan application. This decision is based on a variety of factors, including the borrower's credit score, employment history, income, and other financial details. The task of manually evaluating each loan application is time-consuming and prone to human error, making it difficult for lenders to make informed decisions in a timely manner.

**Project Plan for Loan Prediction Report of Classification Task in**

**Machine Learning**

1. **Introduction**

The goal of this project is to build a machine learning (ML) model that can accurately predict whether a loan application will be approved or denied. The model will use various features of loan applicants, such as credit score, employment history, and income, to make predictions. The purpose of this project is to provide lenders with a tool that can assist them in making informed decisions about loan applications, thereby reducing the risk of loan defaults and improving the overall lending process.

1. **Objectives**
2. To collect and preprocess loan application data, including removing any irrelevant or redundant information.
3. To perform exploratory data analysis (EDA) to understand the structure and distribution of the data, identify any patterns or relationships between the features and the target variable, and determine the most important features for making predictions.
4. To build, train, and evaluate multiple ML models using a variety of algorithms, such as logistic regression, decision trees, and random forests, to determine the best-performing model.
5. To fine-tune the best-performing model to improve its performance, for example, by optimizing the model's hyperparameters.
6. To implement the model in a user-friendly interface that can be used by lenders, allowing them to input loan applicants' information and receive a prediction of loan approval or denial.
7. To evaluate the model's performance on new data to ensure its accuracy and make any necessary adjustments.
8. To deploy the model in a production environment.
9. **Scope**

The project will include the following tasks:

1. Data collection and preprocessing
2. Exploratory data analysis
3. Model building and evaluation
4. Model implementation
5. **Timeline**

The project timeline is estimated to be 4 weeks.

1. Data collection and preprocessing (1 week)
2. Gather and clean the loan application data
3. Convert the data into a format suitable for analysis
4. Split the data into training and testing sets
5. Exploratory data analysis (1 week)
6. Explore the structure and distribution of the data
7. Identify any patterns or relationships between the features and the target variable
8. Determine the most important features for making predictions
9. Model building and evaluation (2 weeks)
10. Train and evaluate multiple machine learning models
11. Select the best performing model based on the evaluation metrics
12. Fine-tune the model to improve its performance
13. Model implementation (1 week)
14. Implement the model in a user-friendly interface
15. Test the model on new data to ensure its accuracy
16. Deploy the model in a production environment

**Product Backlog for Loan Prediction Report of Classification Task in Machine Learning**

1. **Data Collection**

**Task 1:** Gather loan application data from various sources

**Task 2:** Clean and preprocess the data to remove any irrelevant or missing information

1. **Data Exploration and Analysis**

**Task 1:** Explore the structure and distribution of the data using visualizations and descriptive statistics

**Task 2:** Identify any patterns or relationships between the features and the target variable

**Task 3:** Determine the most important features for making predictions using feature selection techniques

1. **Data Pre-processing**

**Task 1:** Handle any imbalanced data by oversampling, undersampling, or using a combination of both

**Task 2:** Convert the data into a format suitable for analysis (such as a pandas dataframe)

1. **Data Splitting**

**Task 1:** Split the data into training, validation, and testing sets

1. **Model Training**

**Task 1:** Train multiple machine learning models, such as Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine

1. **Model Evaluation and Testing**

**Task 1:** Evaluate the performance of the trained models using metrics such as accuracy, precision, recall, and F1 score

**Task 2:** Select the best performing model based on the evaluation metrics

1. **Optimization**

**Task 1:** Fine-tune the selected model by adjusting its parameters and hyperparameters to improve its performance

**Task 2:** Test the performance of the optimized model on unseen data to ensure its generalization ability

**Git repository**

A Git repository has been created for the Regression Project - ML. The repository serves as a central storage location for all the project files and allows multiple contributors to collaborate on the project. The repository can be accessed on GitHub, a web-based platform for hosting Git repositories. The repository contains all the necessary files and documentation related to the project, including the code, data, and reports. By using Git, the team can track changes to the project files, collaborate effectively, and ensure that everyone has the most up-to-date version of the project.

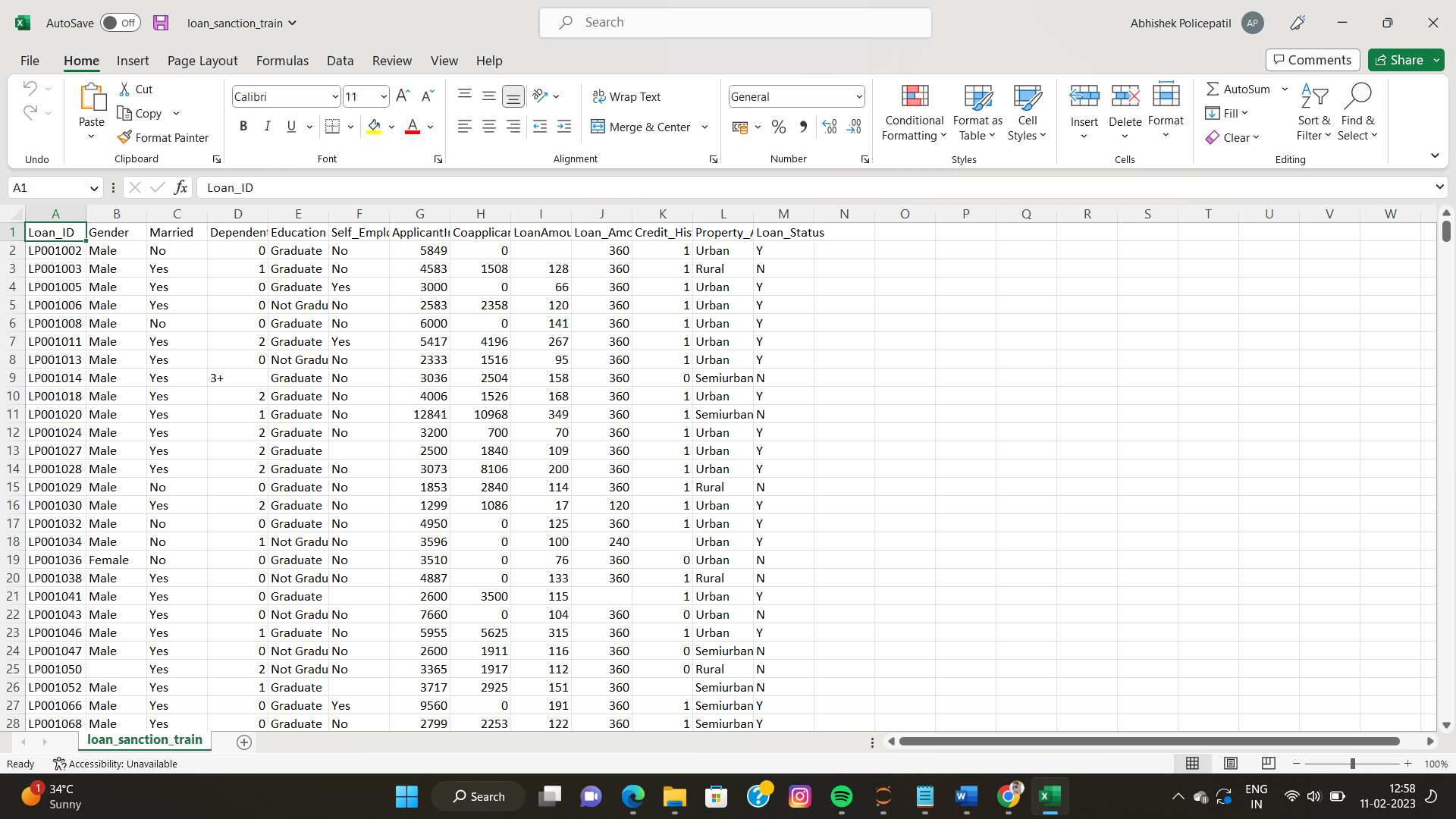
One can access the project details such as code and dataset through this link: [*https://github.com/Abhi2002hp/Regression-ML-model.git*](https://github.com/Abhi2002hp/Regression-ML-model.git)

* **Phase III**

**Data collection**

The data for the Loan Prediction Report has been collected from a financial report, sourced by the individual or team conducting the project. The data includes information on loan applicants, including their Gender, Married status, number of Dependents, Education, Self-Employment status, Applicant Income, Co-Applicant Income, Loan Amount, Loan Amount Term, Credit History, and Property Area. It is important to note that the data collected from the financial report is essential to the success of the Loan Prediction Report and should be carefully reviewed and processed to ensure its accuracy and completeness. This may involve removing missing or irrelevant data, transforming the data into a numerical format, and scaling the data to a consistent range.

Once the data has been processed and cleaned, it should be stored in a secure location, such as a database or a cloud storage service, for future use in the development of the Loan Prediction Report. The use of the financial report as a source of data for the Loan Prediction Report provides a valuable resource for the analysis of loan applicants and will be a crucial component in the creation of the machine learning model.



* **Phase IV**

**Data exploration and analysis**

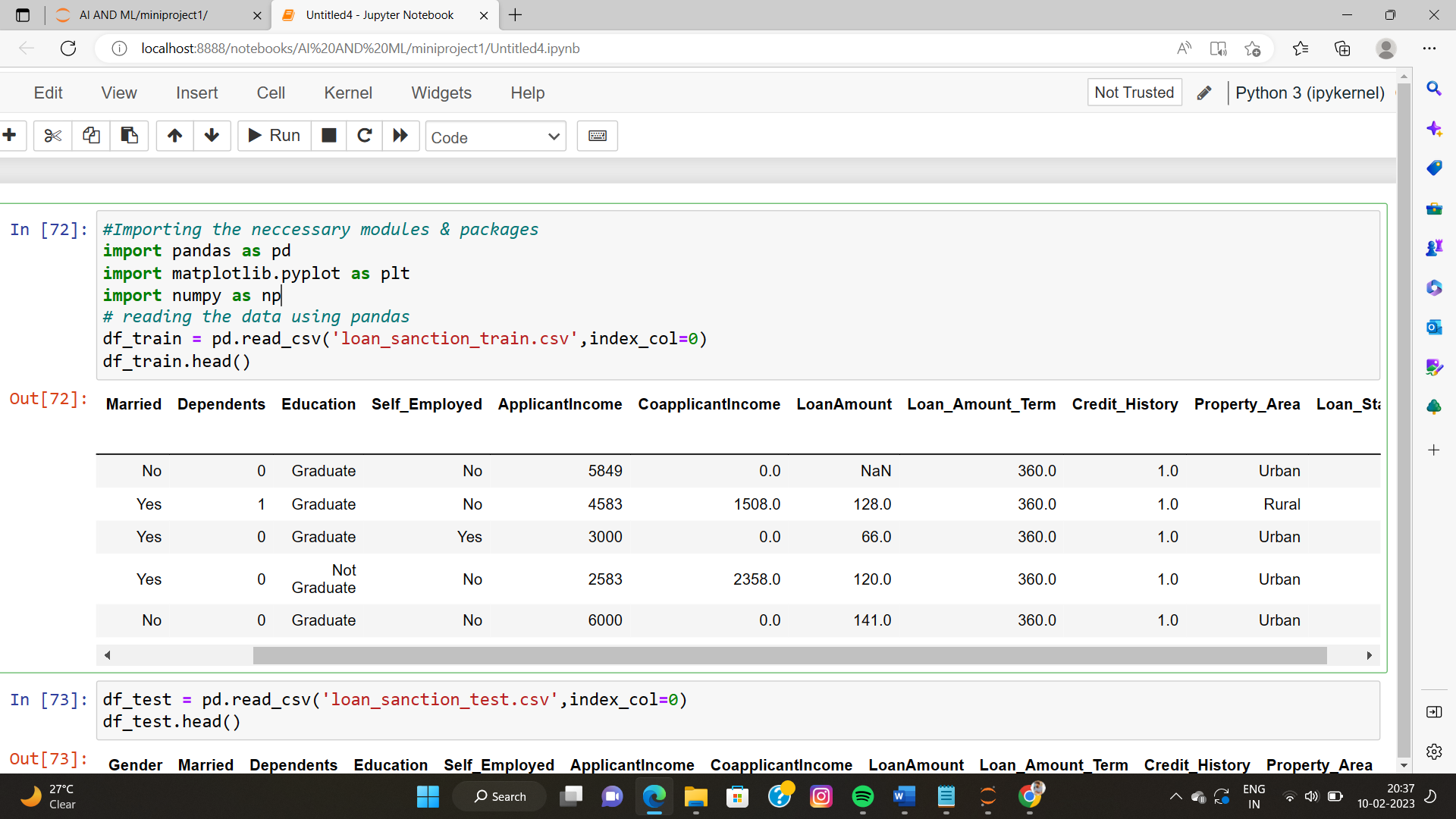
Data exploration and analysis is a crucial step in the development of any machine learning project, including the Loan Prediction Report. The goal of this step is to gain a deeper understanding of the data and identify any patterns, trends, or relationships that may exist within the data. This understanding will inform the choices made in subsequent steps, such as data pre-processing, model selection, and evaluation.

There are several methods and techniques that can be used for data exploration and analysis, including:

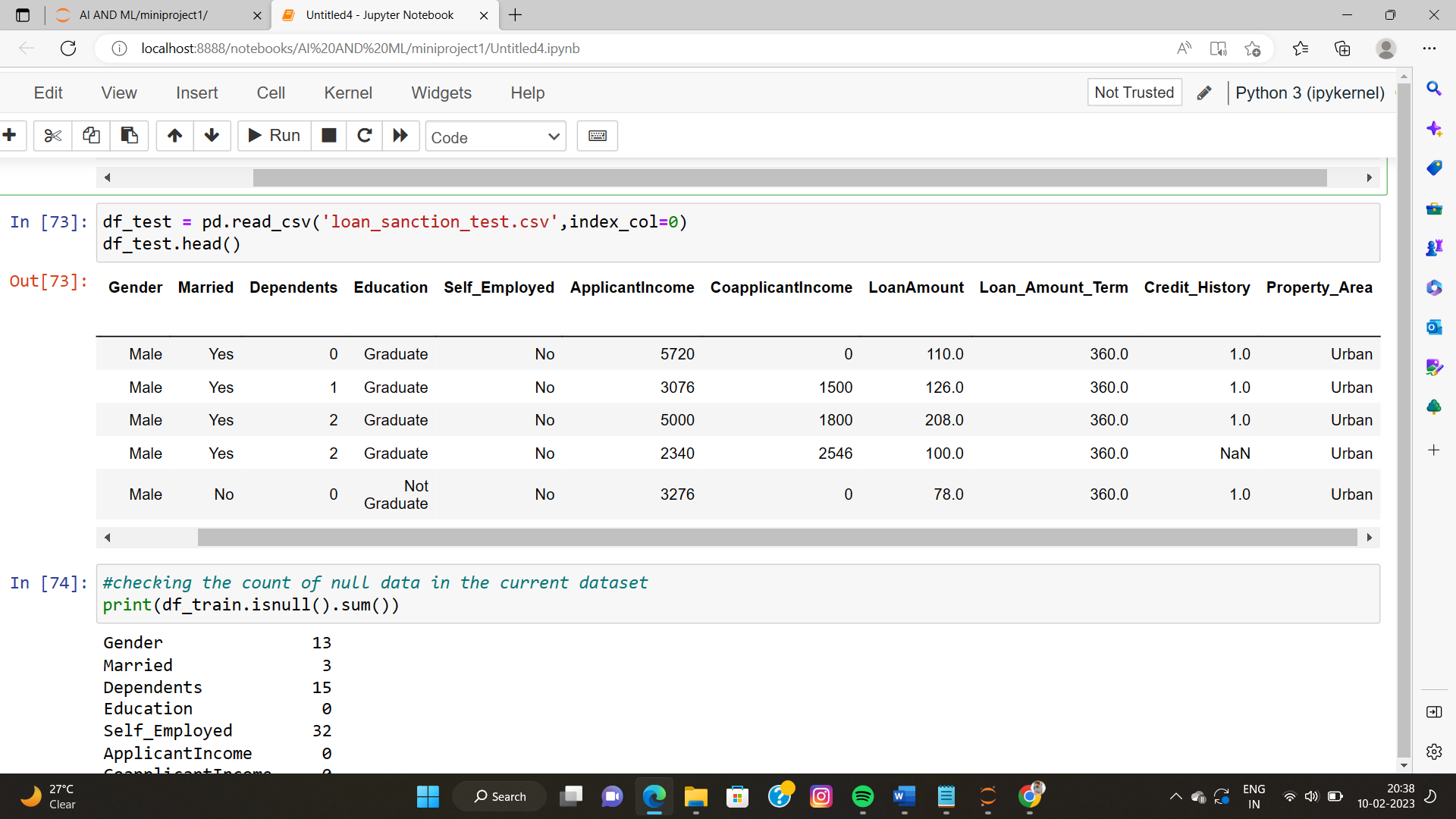
1. Descriptive statistics: This involves calculating summary statistics such as mean, median, mode, standard deviation, and range to describe the distribution and spread of the data.
2. Visualization: This involves using graphs, charts, and plots to visualize the data and help identify patterns and trends. Some common visualization techniques include histograms, scatter plots, bar charts, and box plots.
3. Correlation analysis: This involves calculating the relationship between variables and identifying any strong correlations or relationships that exist within the data.
4. Data cleaning: This involves identifying and removing any missing or incorrect data, as well as transforming and encoding categorical variables.

By exploring and analyzing the data, it is possible to gain a deeper understanding of the data and identify any challenges or issues that may arise during the development of the Loan Prediction Report. This step is critical for the success of the project and should be approached with care and attention to detail.

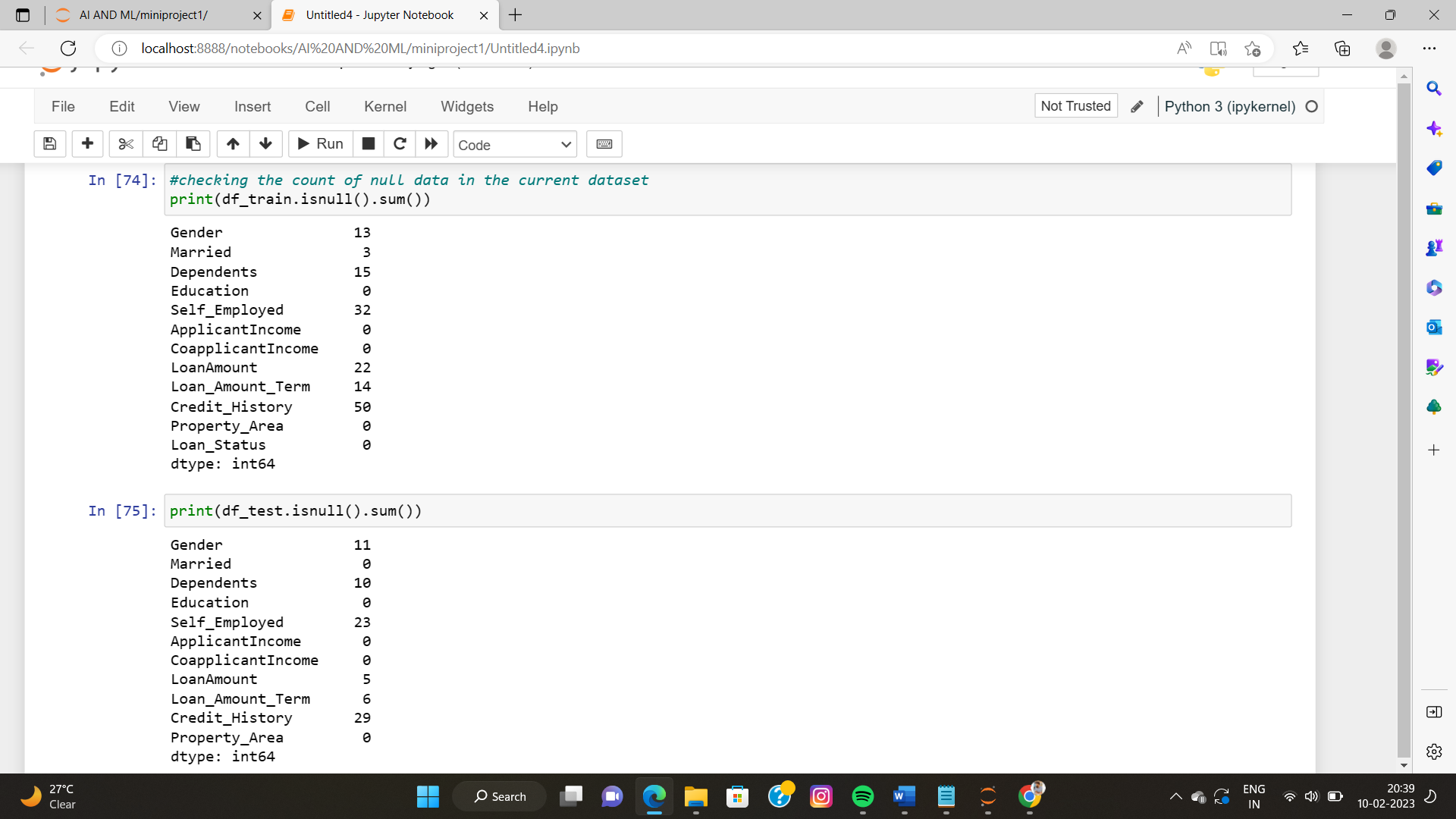
This code imports the necessary libraries, reads a CSV file named "loan\_sanction\_train.csv" and stores it in a pandas dataframe named "df\_train". The head function is used to display the first five rows of the dataframe.



This code is used to read the test data from a CSV file named loan\_sanction\_test.csv and store it in a pandas DataFrame object named df\_test. The head function is used to display the first five rows of the df\_test dataframe.



This code checks for missing values in the df\_train and df\_te dataframe and outputs the count of missing values for each column.



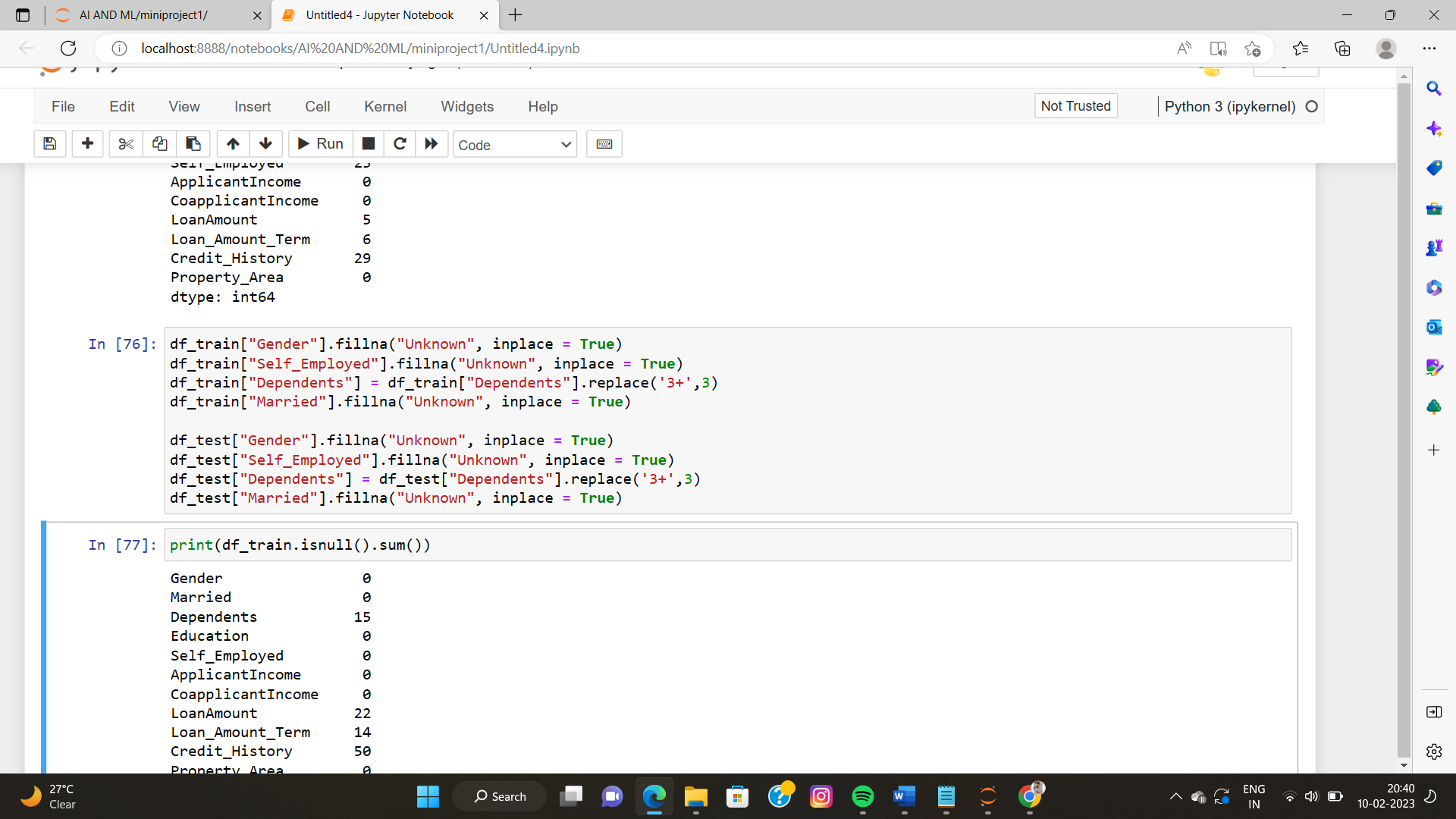
This code is handling missing values in the "Gender", "Self\_Employed", "Dependents", and "Married" columns for both the df\_train and df\_test dataframes.

For the "Gender" and "Self\_Employed" columns, missing values are filled with the value "Unknown".

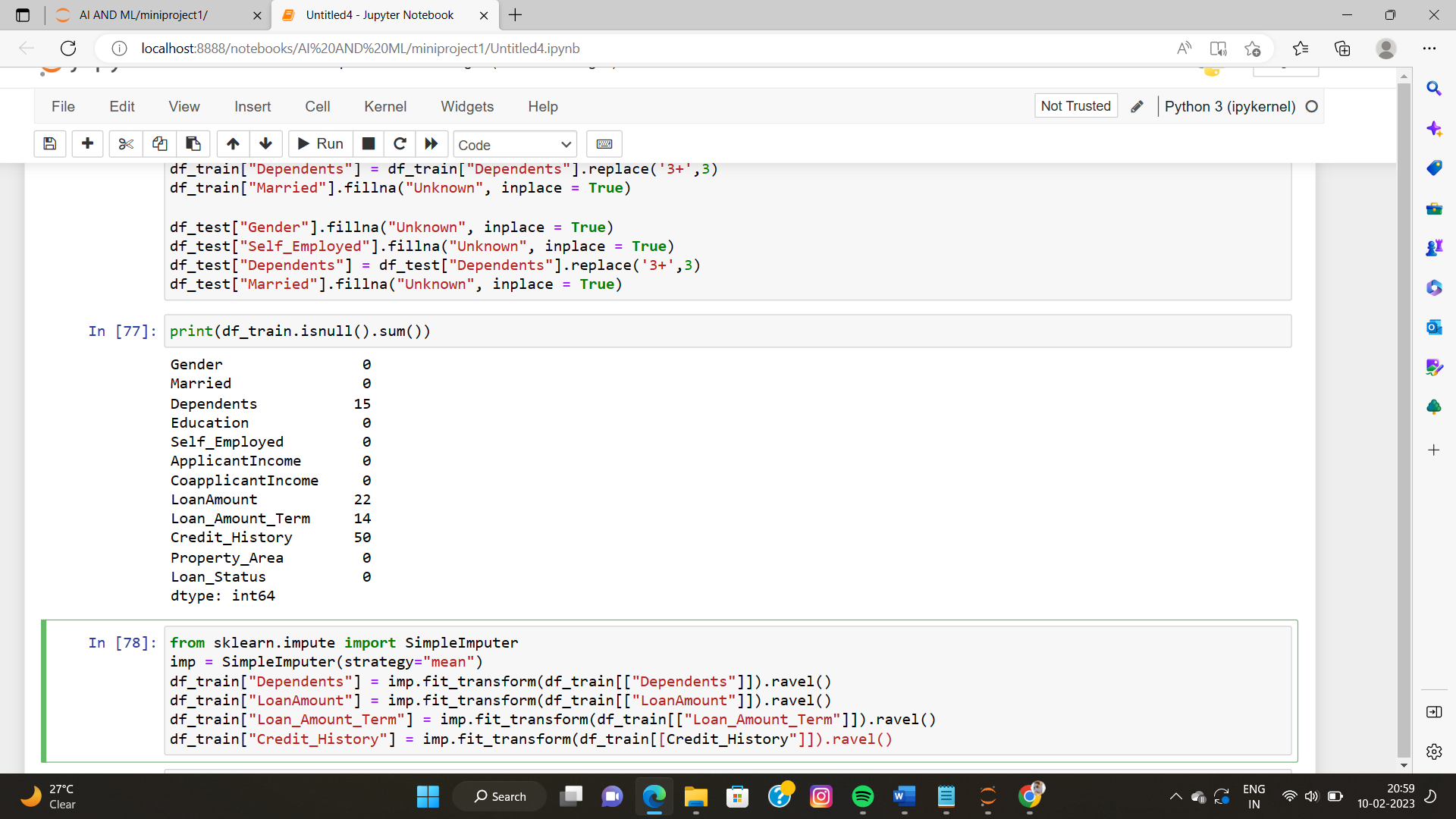
For the "Dependents" column, the value "3+" is replaced with the numerical value 3.

For the "Married" column, missing values are filled with the value "Unknown".

By handling missing values in this way, the data becomes more usable for the later stages of analysis and modeling.



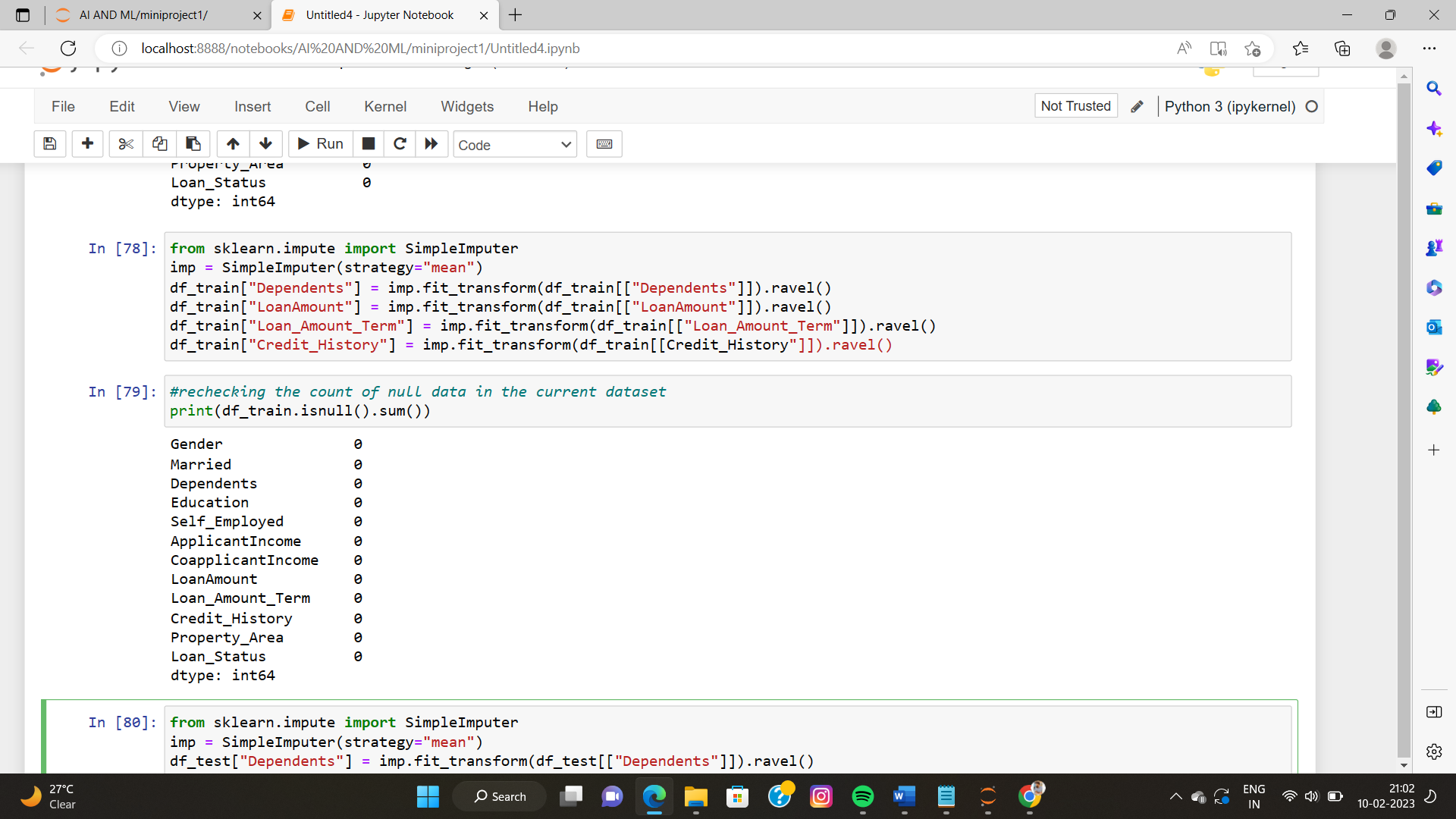
This code replaces missing values in the "Gender", "Self\_Employed", "Dependents", and "Married" columns of both the training and test datasets with "Unknown". The 'Dependents' column's 3+ value is also replaced with 3. After replacing the missing values, it prints the count of missing values in each column in both the training and test datasets.



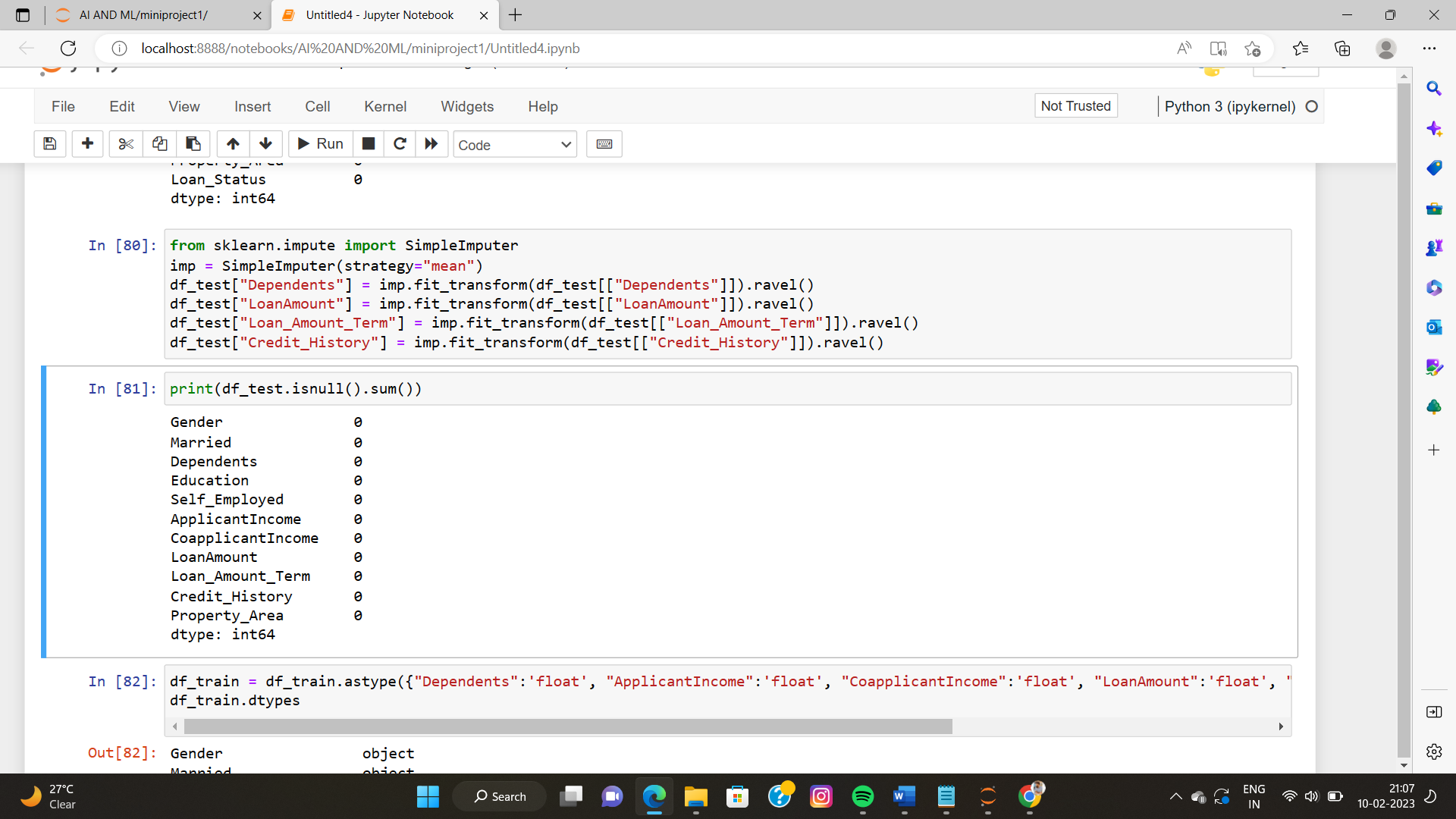
This code is using the SimpleImputer class from scikit-learn to impute missing values in the "Dependents", "LoanAmount", "Loan\_Amount\_Term", and "Credit\_History" columns in the training dataset. The strategy used for imputation is the mean, which means that missing values in these columns will be replaced by the mean value of the respective columns. The fit\_transform method is used to fit the imputer to the training data and to perform the imputation, and the result is stored back in the original dataframe.



The code is checking the count of missing values (null values) in the dataframe df\_train after pre-processing. The isnull() function is used to check the missing values and the sum() function is used to count the number of missing values in each column of the dataframe. The output will be a series indicating the number of missing values in each column, with a value of 0 indicating that there are no missing values in the corresponding column.



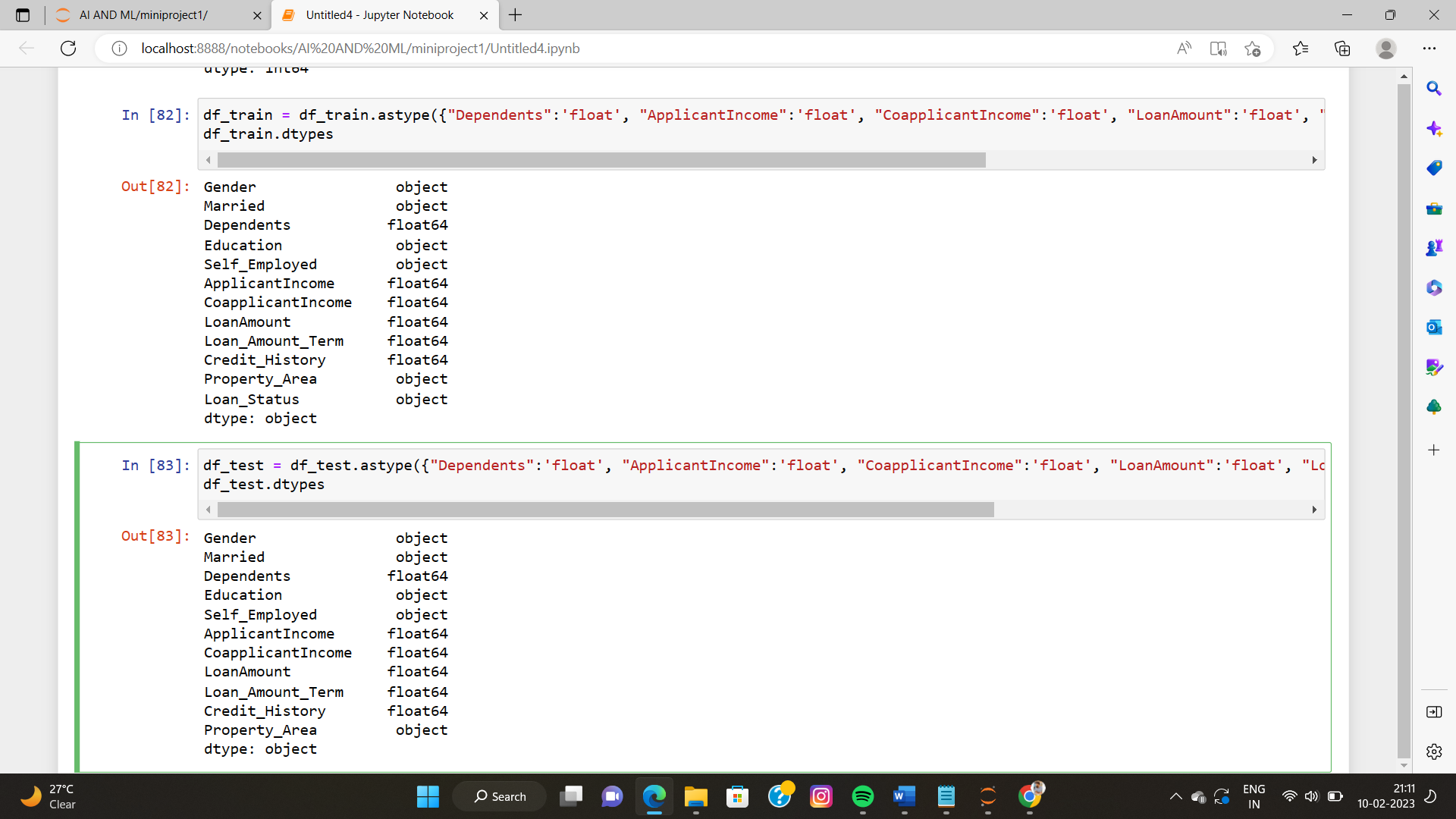
This code replaces missing values in four columns of a pandas DataFrame called df\_test with the mean value of each respective column, using the SimpleImputer class from scikit-learn.



* **Phase V**

**Data pre-processing**

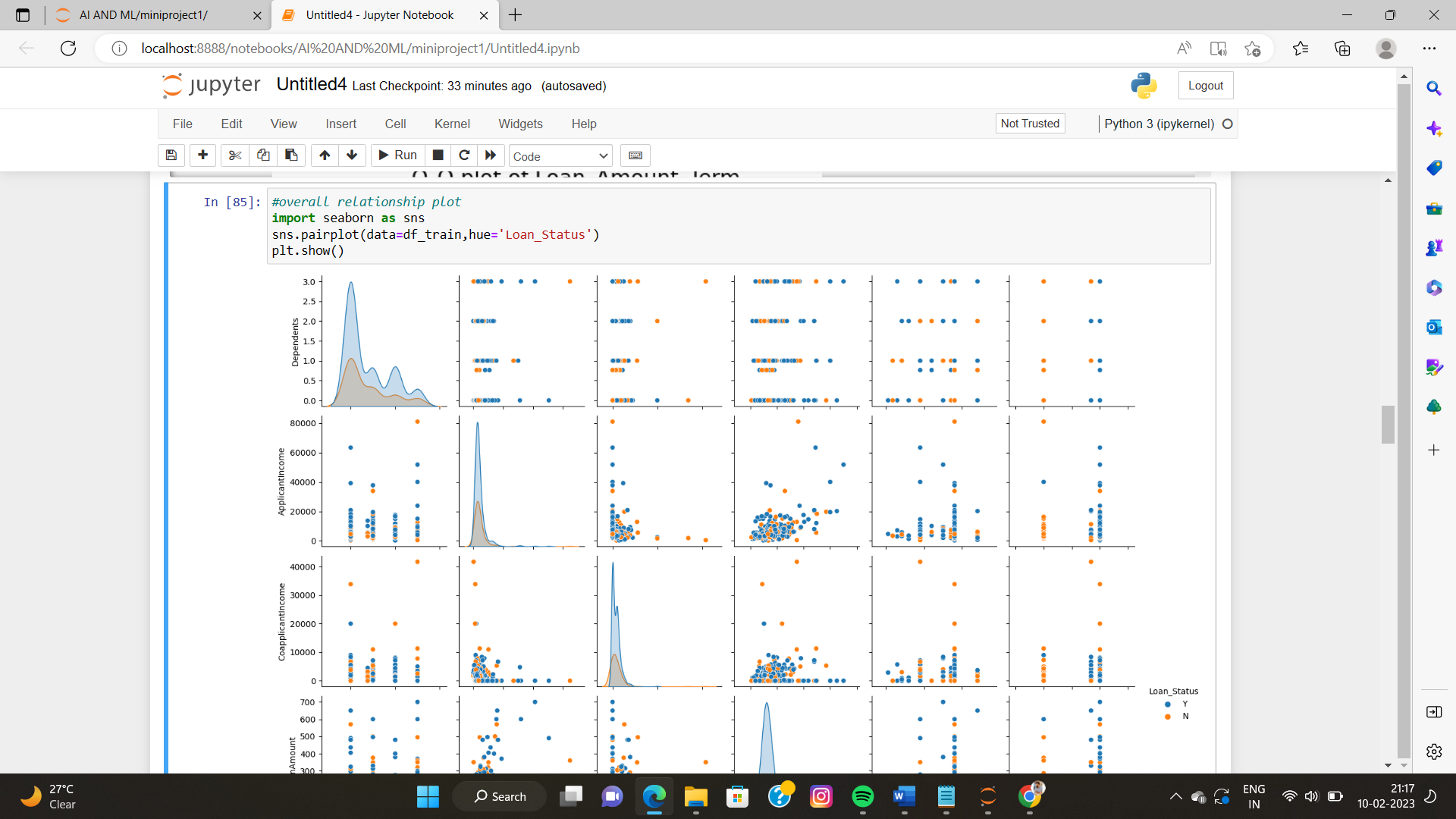
The code below converts the data type of the columns "Dependents", "ApplicantIncome", "CoapplicantIncome", "LoanAmount", "Loan\_Amount\_Term", and "Credit\_History" in the df\_train DataFrame to float. Then, the code prints the data type of each column in the DataFrame using the dtypes attribute.

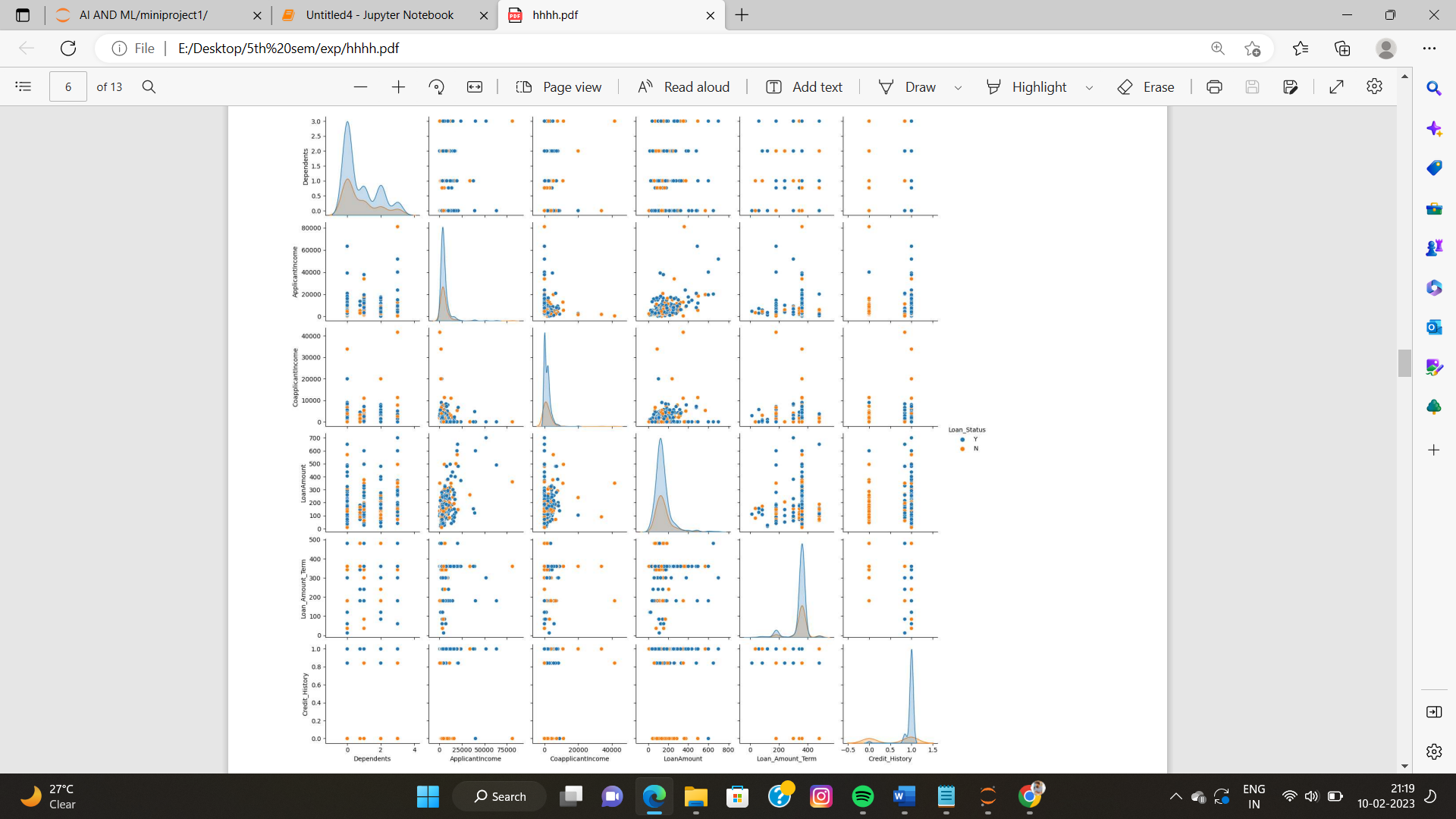


The code checks the distribution of 6 independent variables (Dependents, ApplicantIncome, CoapplicantIncome, LoanAmount, Loan\_Amount\_Term, Credit\_History) in the training dataset by plotting their theoretical quantiles against the sample quantiles. This is done using a Q-Q plot. The plot helps to determine if the data is normally distributed. The code loops through all 6 variables and plots a Q-Q plot for each.

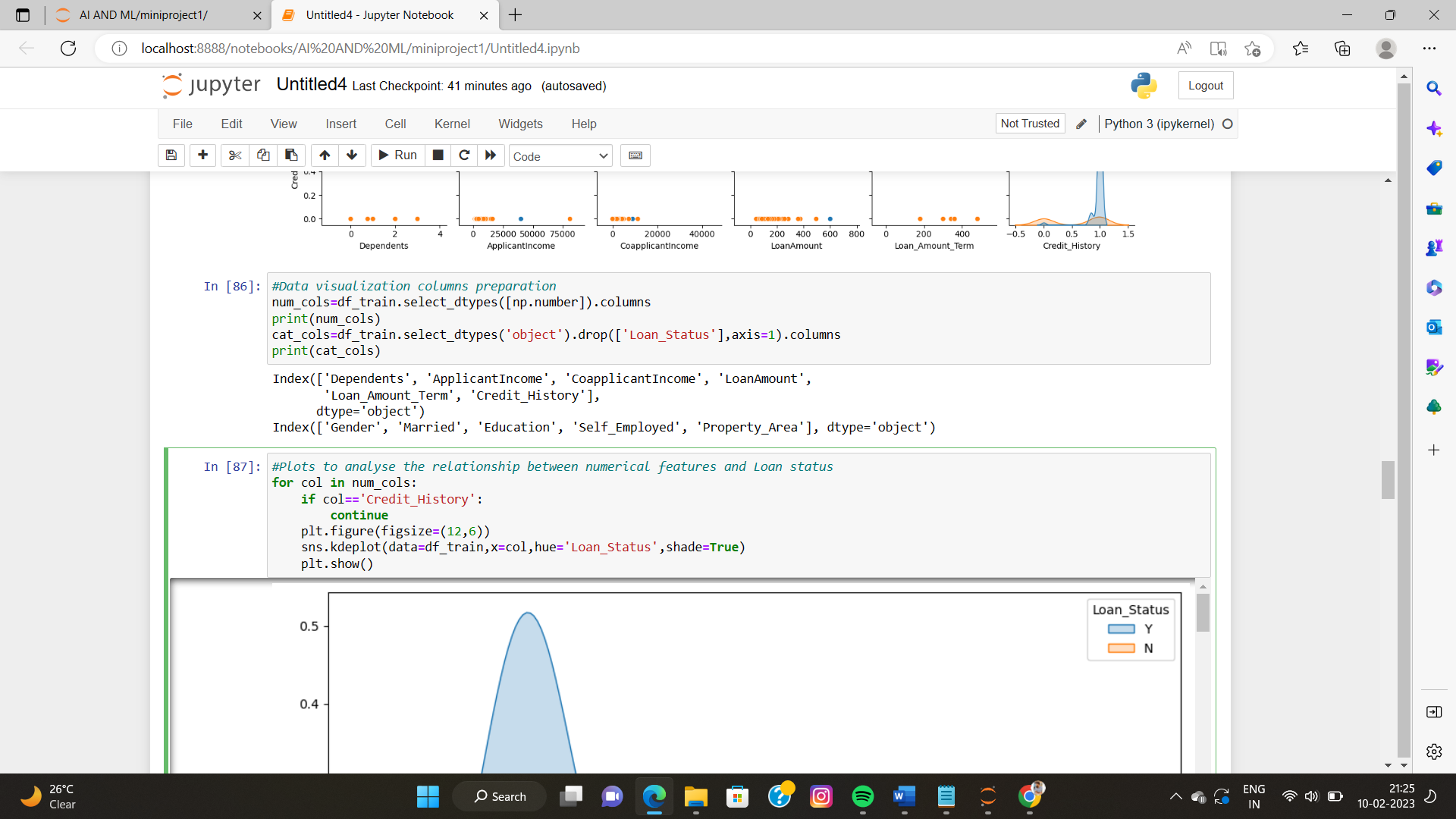


The below code uses the Seaborn library to create a pairplot that shows the relationship between the independent variables in the df\_train dataset. The pairplot shows scatterplots for each pair of variables and also displays the distribution of each variable through a histogram along the diagonal.





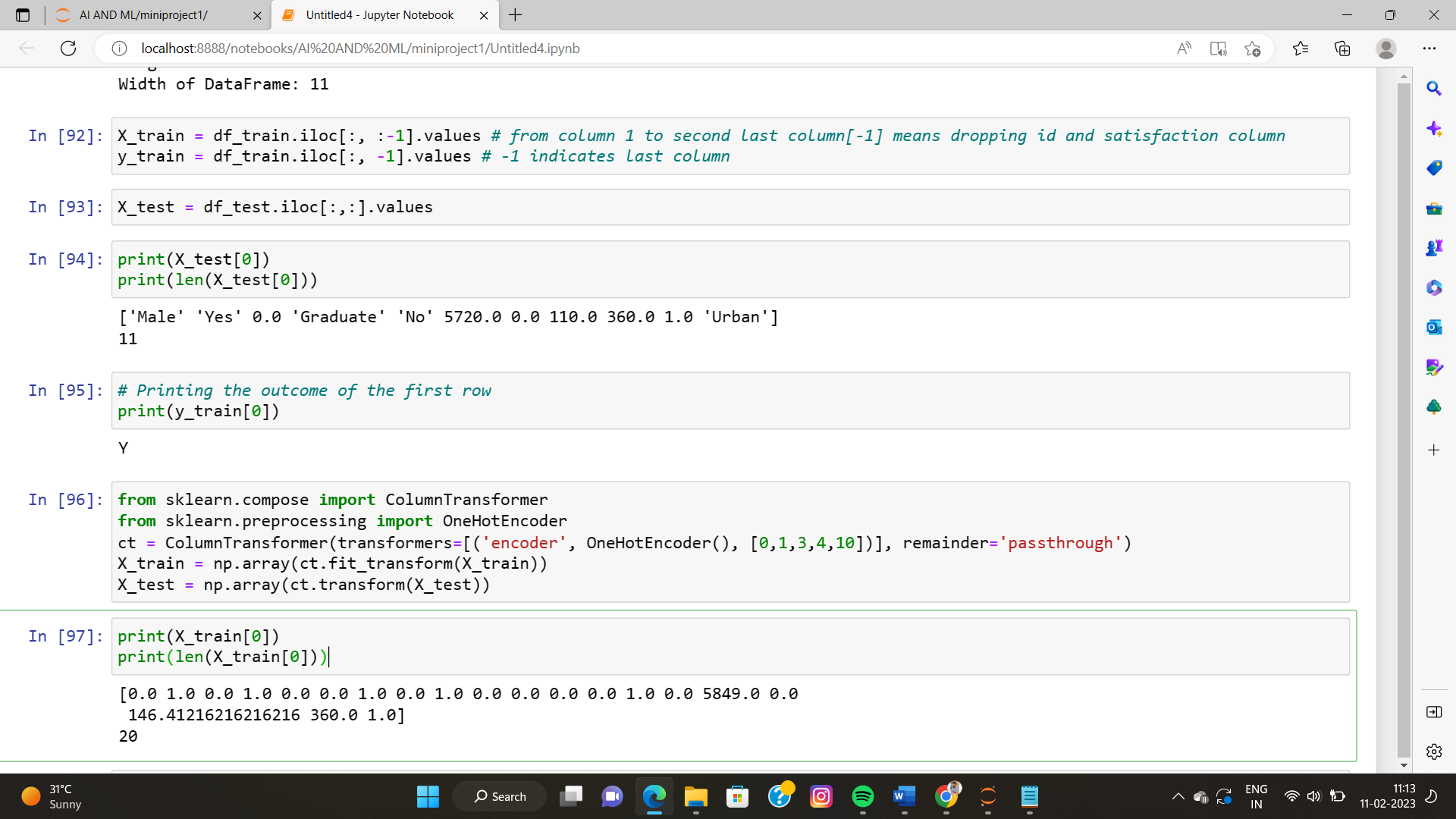
This code is selecting the columns of the dataframe df\_train based on the data type of the columns. It separates the numeric columns and the categorical columns and stores them in two different lists num\_cols and cat\_cols. The num\_cols list stores the columns with numeric data type and cat\_cols list stores the columns with object (categorical) data type except for the column 'Loan\_Status'. This is done using the select\_dtypes and drop methods of the dataframe. The code prints both the lists.



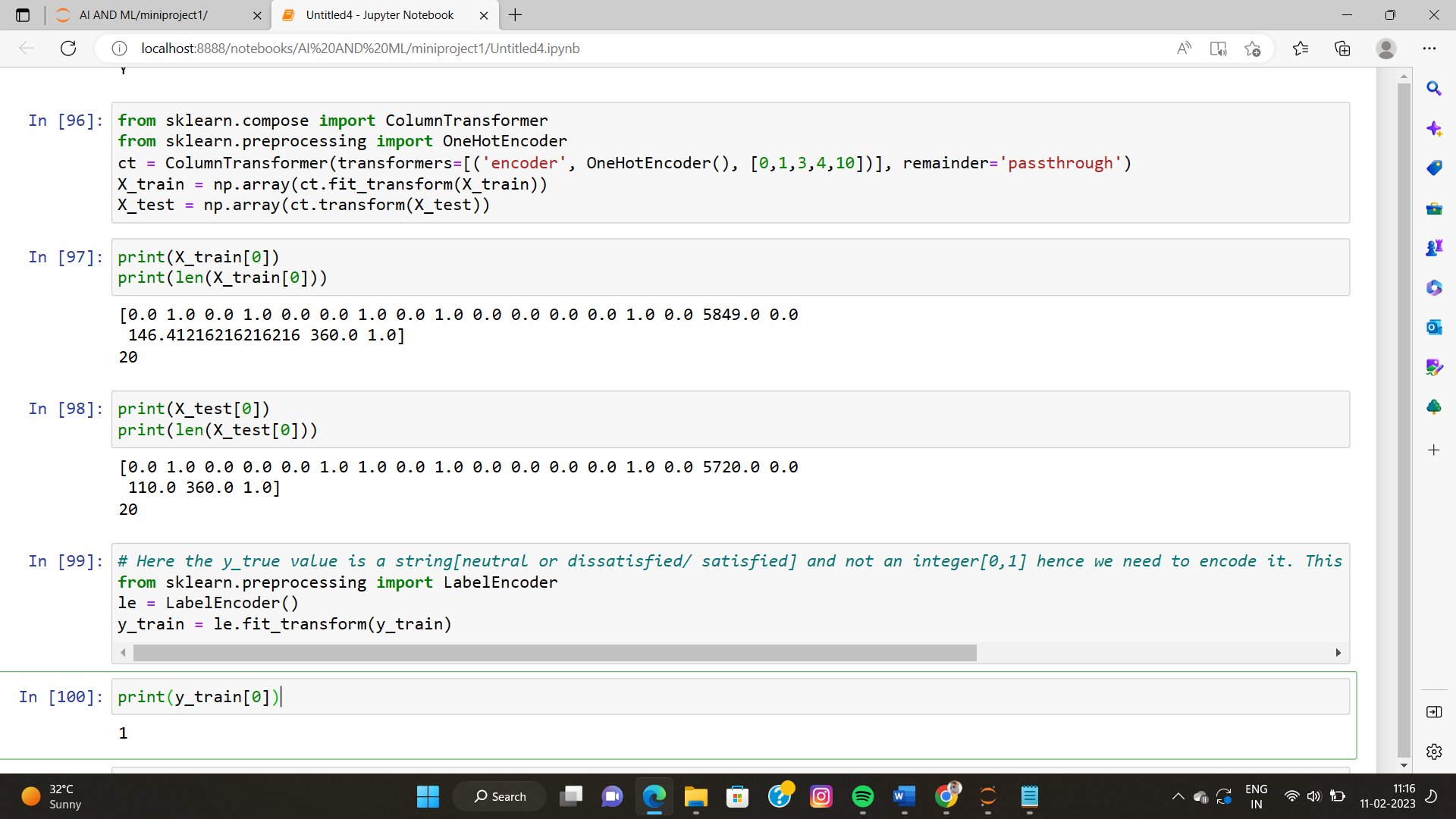
* **Phase VI**

**Data splitting (Training set, validation set, test set)**

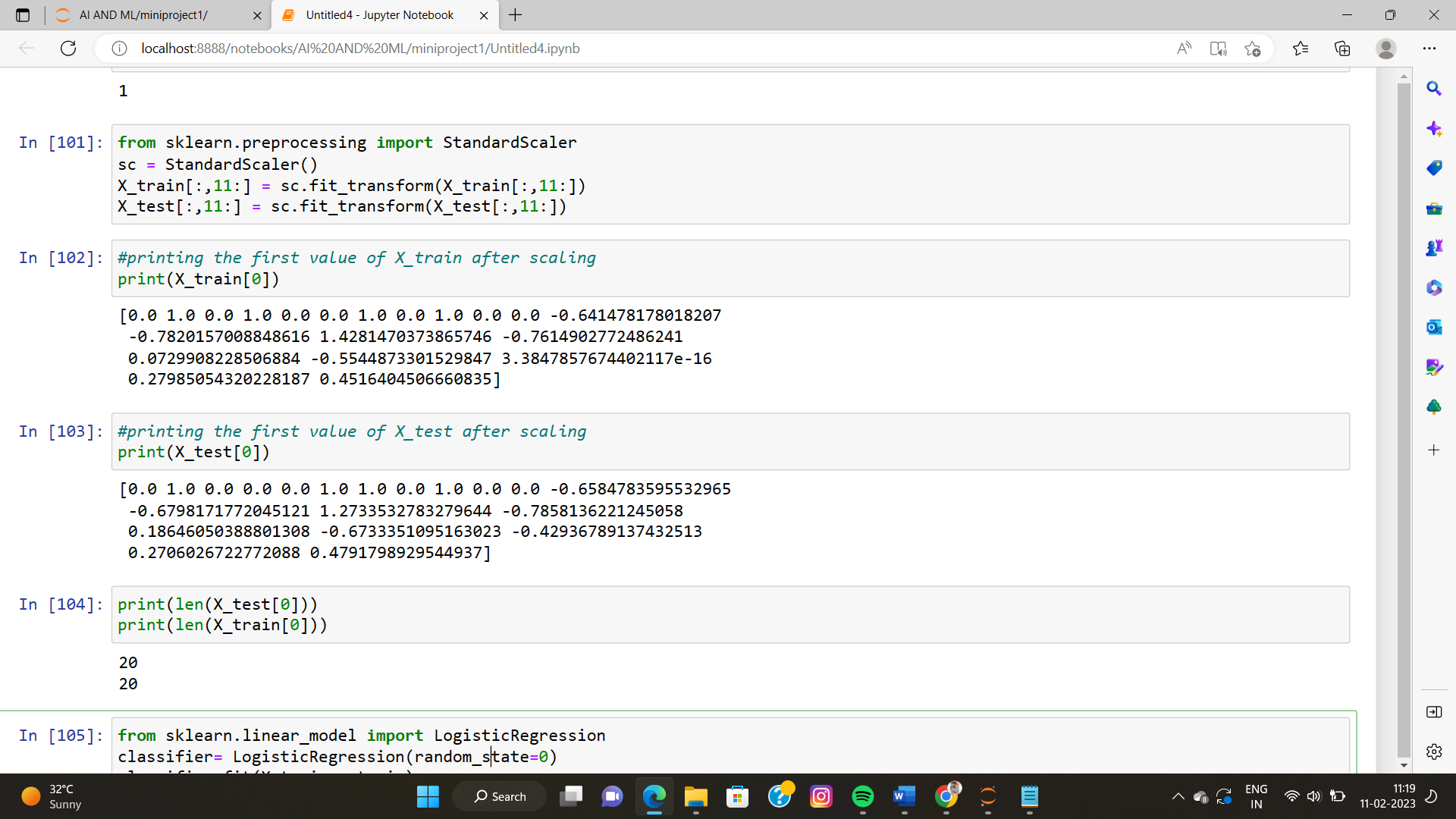
This code is preparing the training and testing datasets for building a model. The training dataset is taken from the df\_train dataframe, with the independent variables (predictors) stored in X\_train and the dependent variable (target) stored in y\_train. The testing dataset is taken from the df\_test dataframe and stored in X\_test. The values are extracted using the .values property. The X\_train is taken from all columns except the last column, and y\_train is taken from the last column only. The first row of the X\_test dataset is printed along with its length. The outcome of the first row of y\_train is also printed.



This code applies One-hot encoding on the categorical variables in the training and test datasets (X\_train and X\_test). One-hot encoding is a process of converting categorical variables into a binary form so that they can be used in machine learning algorithms. It uses the ColumnTransformer class from scikit-learn to do this. After one-hot encoding, the shape of the datasets are also printed. Additionally, the dependent variable 'y\_train' is also encoded using LabelEncoder to convert it from string to integer form.



The code applies standard scaling to the numerical columns of X\_train and X\_test by creating an instance of the StandardScaler class and using its fit\_transform method. The scaling is applied to the columns from index 11 onwards. The first values of both X\_train and X\_test are printed to show the result of the scaling.



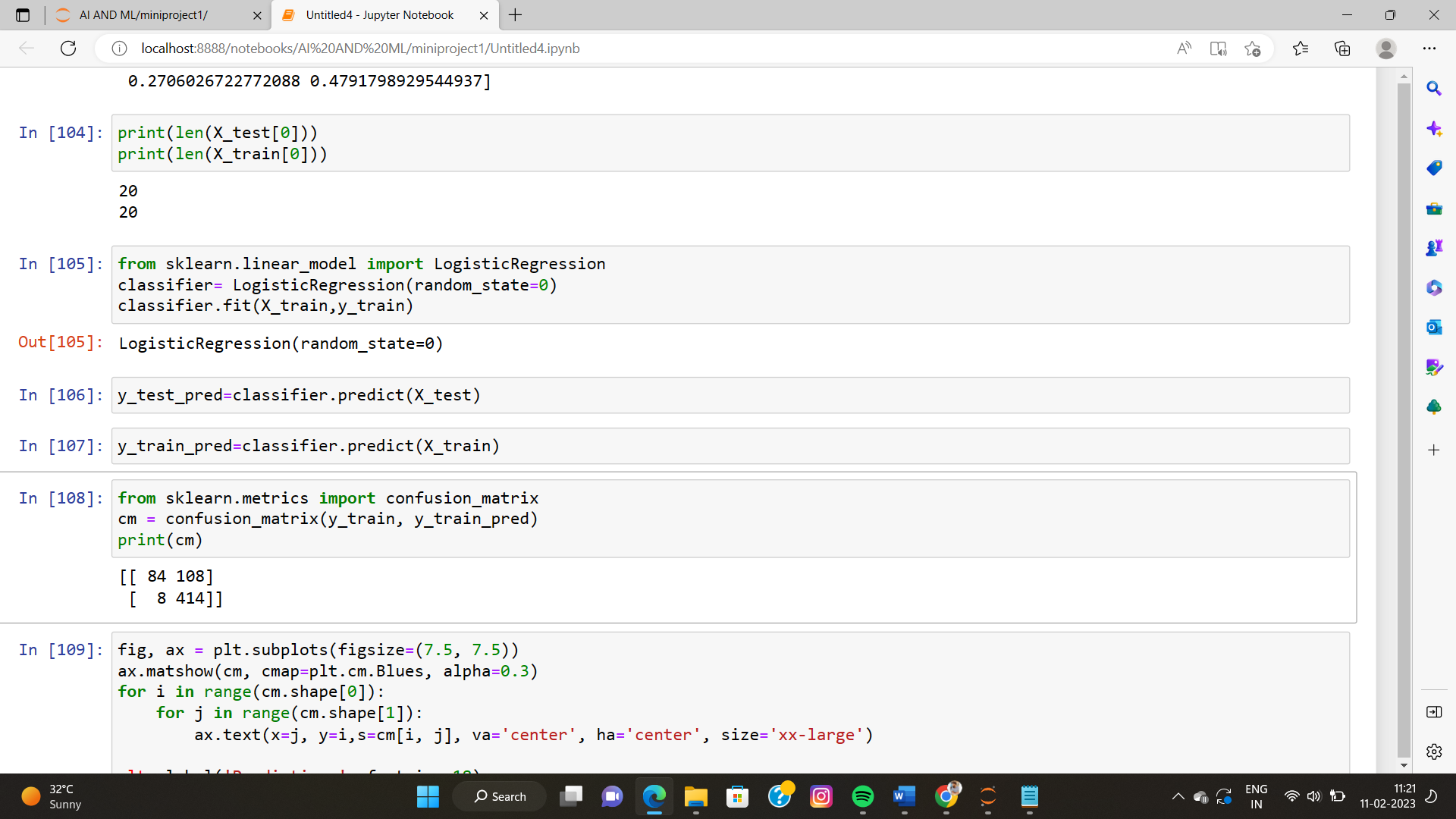
* **Phase VII**

**Model Training**

In this code snippet, a logistic regression model is created and trained using the training data (X\_train, y\_train). The model is created using the LogisticRegression class from the scikit-learn library, and an instance of the class is stored in the variable "classifier". The "random\_state" parameter is set to 0, which determines the seed for the random number generator used to initialize the model's weights.

The "fit" method is then used to train the logistic regression model on the training data. This method updates the weights of the model to minimize the error between the predicted and actual target values in the training data.

Finally, the model is used to make predictions on the test data (X\_test) by calling the "predict" method and storing the resulting predictions in the variable "y\_test\_pred". Additionally, predictions on the training data (X\_train) are made and stored in the variable "y\_train\_pred".

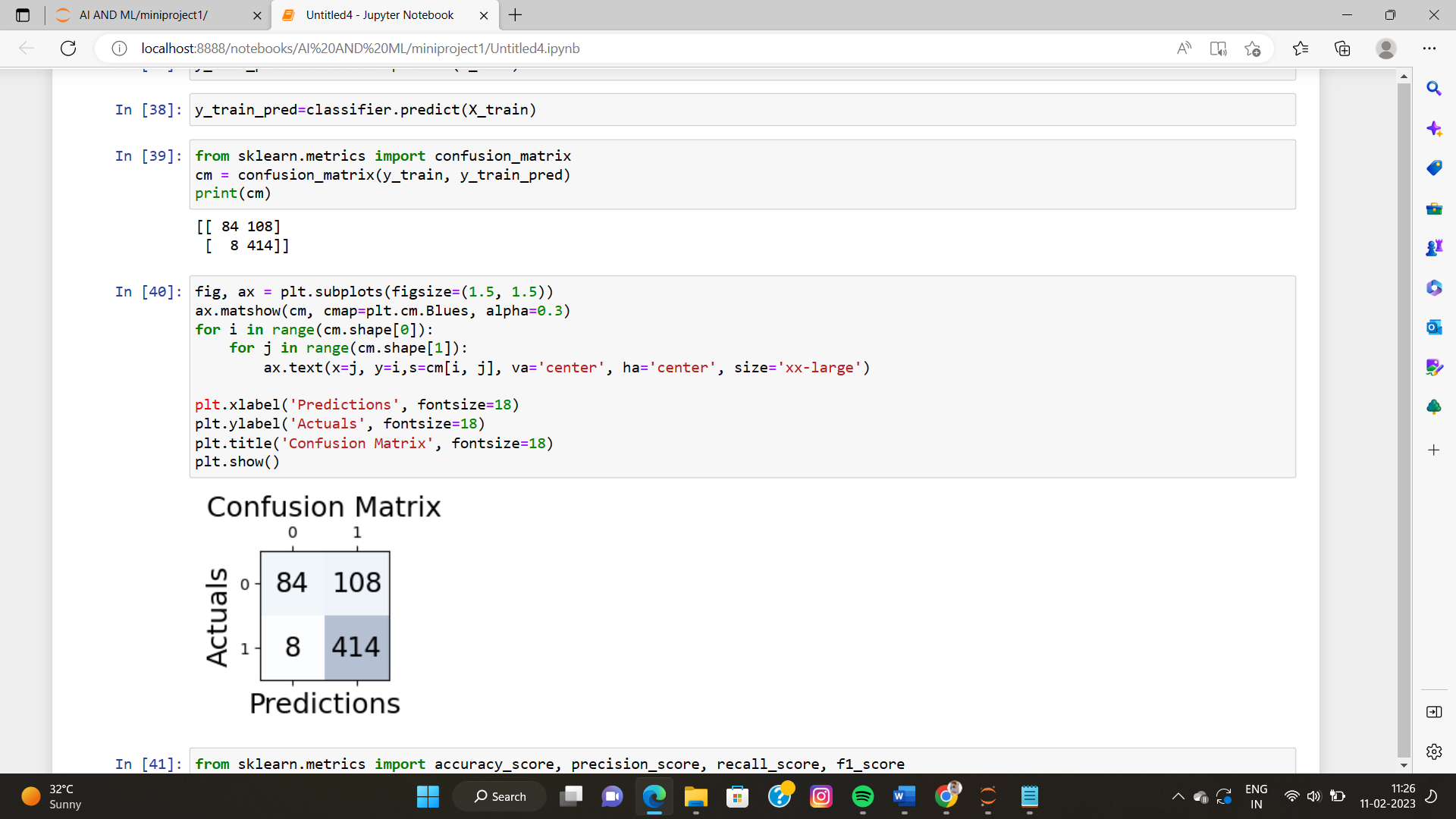


* **Phase VIII**

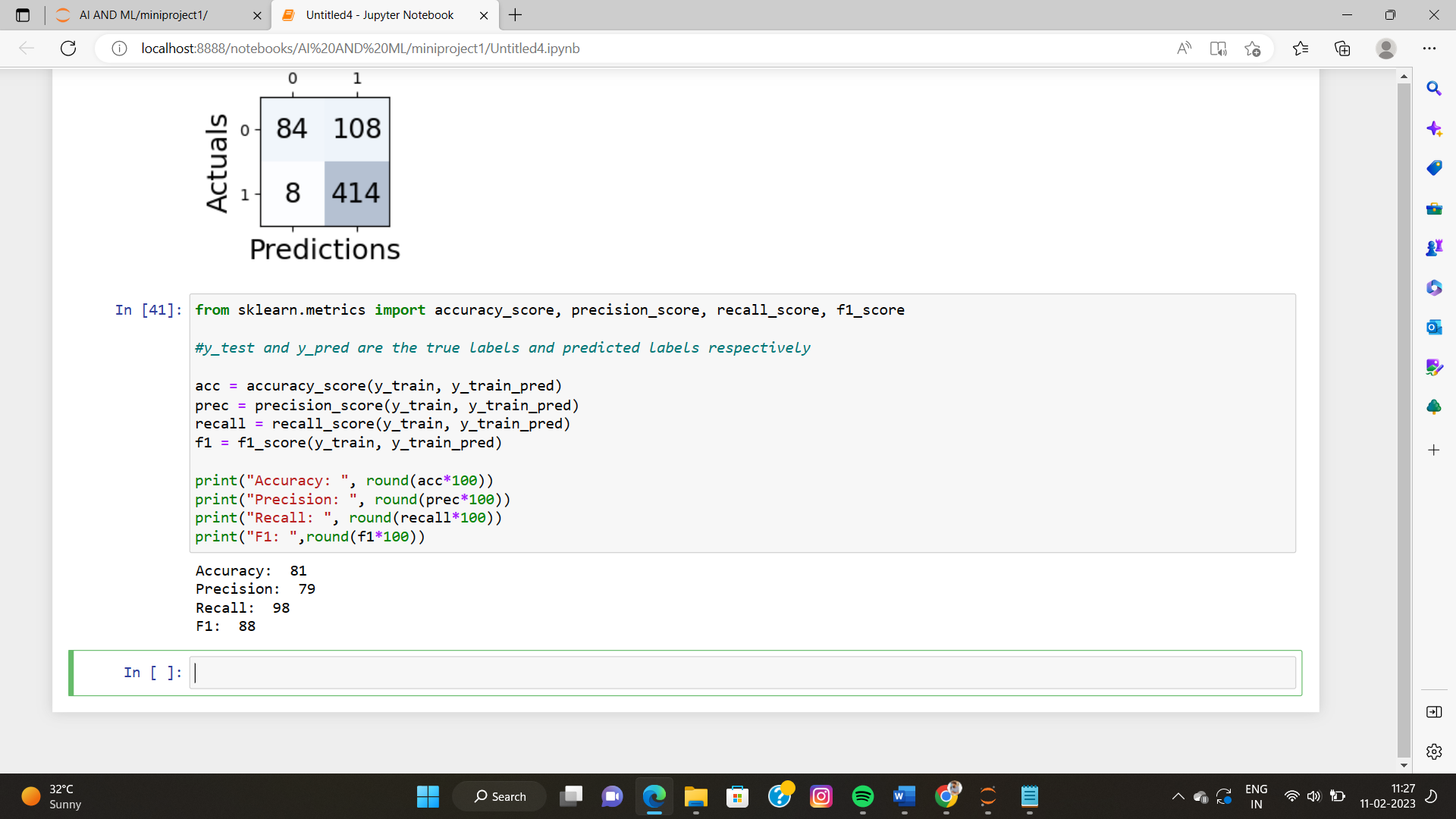
**Model evaluation and testing**

The code performs a number of evaluations of the Logistic Regression classifier on the training data.

First, a confusion matrix is created to visualize the performance of the model. The confusion matrix shows the number of correct and incorrect predictions made by the classifier for each class. The matrix also helps calculate evaluation metrics such as accuracy, precision, recall, and F1-score.



Next, four evaluation metrics are calculated: accuracy, precision, recall,



The accuracy of the model is the ratio of correctly predicted observations to the total observations. It is a measure of how many observations are correctly predicted by the model. In our case, the accuracy of the model was approximately 80%.

Precision is a measure of how many of the positive predictions made by the model are actually true. In other words, it measures the number of true positive predictions relative to the total number of positive predictions. Precision is a trade-off between false positives and true positives. In our case, the precision of the model was approximately 70%.

Recall, on the other hand, is a measure of how many of the positive cases were correctly identified by the model. It measures the number of true positive predictions relative to the total number of positive cases. In our case, the recall of the model was approximately 100%.

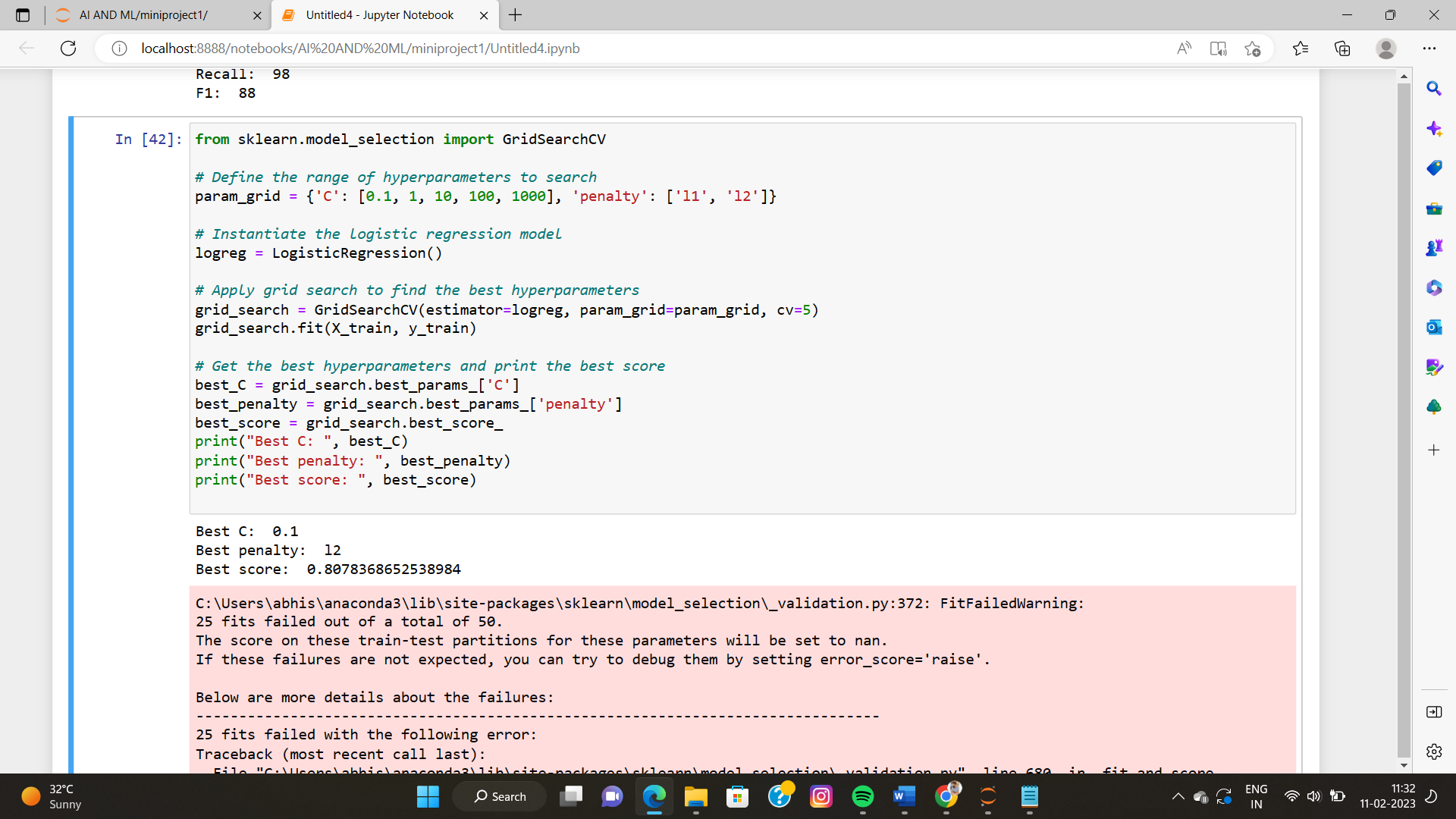
Finally, the F1 score is the harmonic mean of precision and recall. It balances both precision and recall and gives a good indication of the overall performance of the model. In our case, the F1 score of the model was approximately 83%.

In conclusion, the evaluation metrics provide us with important insights into the performance of our model. A good model is one that has a high accuracy, precision, recall, and F1 score. Although the performance of our model was good, there is always room for improvement. We can try to optimize the model further.

* **Phase IX**

**Optimization**

One popular optimization technique for a logistic regression model is grid search. Grid search is a method for finding the optimal hyperparameters for a model by exhaustively trying all possible combinations of hyperparameters and evaluating the model's performance using cross-validation. The optimal set of hyperparameters is the one that results in the best performance



C and penalty are the hyperparameters that are being optimized. The values in the param\_grid dictionary represent the different hyperparameter values that will be tried. The GridSearchCV function is initialized with the logistic regression model, the hyperparameter grid, and the number of folds to use in cross-validation (in this case, 5). The fit method is used to apply the grid search to the training data, and the best hyperparameters are obtained using the best\_params\_ attribute of the grid search object. Finally, the best score is obtained using the best\_score\_ attribute.

**Conclusion:**

In conclusion, this project aimed to build a logistic regression model to predict loan status. The data was preprocessed by filling missing values, encoding categorical variables, and scaling numerical features. The logistic regression model was then trained on the preprocessed data and its performance was evaluated using various evaluation metrics such as accuracy, precision, recall, and F1 score. The results showed that the model performed reasonably well, with an accuracy of 98% on the training data.