**SHRIDEVI POLYTECHNIC**

**SIRA ROAD, TUMKUR**

**2022-23**

**Department of Computer Science Engineering**

MINI PROJECT 02

Artificial Neural Network-

Deep Learning

**Under Guidance of Cohort Owner:**

Yamuna.H. HOD, CSE

**Team Members:**

Niveditha K S S533CS2007

Abhishek H P 533CS2001

**Pathway :** Artificial Intelligence and Machine Learning

**Code :** 20CS51I

**Semester:** 5th sem

**Deep Learning Project Report on**

**“Loan Prediction Report A Deep Learning Model using Artificial Neural Network”**

* **Phase I**

**Project Assignment: Artificial neural Network - DL**

The goal of this project is to build a Deep learning model to perform analysis on a dataset of your choice. You will be responsible for selecting the appropriate applications of ANN , pre-process 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 functions 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 2-3 weeks

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

**Skills Required:** Basic understanding of Deep learning concepts, programming skills in Python, and experience with data analysis and keen with tensorflow & tensorboard 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 Artificial Neural network in**

**Deep Learning**

1. **Introduction**

The loan prediction is a crucial task in the banking sector as it helps to determine the likelihood of a loan being approved or not. With the advancements in technology and artificial intelligence, deep learning can play a significant role in automating the loan approval process and making it more efficient. The goal of this project is to build a deep learning model that can predict whether a loan application will be approved or not.

1. **Objectives**
2. To understand the different factors that influence loan approval
3. To build a deep learning model that can accurately predict loan approval
4. To analyze the model's performance and identify areas for improvement
5. To provide insights and recommendations for the loan approval process

t.

1. **Scope**
2. Data Collection: Collecting the loan application data from various sources
3. Data Preprocessing: Cleaning and transforming the collected data
4. Data Analysis: Analyzing the data to identify the factors that influence loan approval
5. Model Building: Building a deep learning model to predict loan approval
6. Model Evaluation: Evaluating the performance of the model
7. Recommendations and Insights: Providing insights and recommendations for the loan approval process
8. **Timeline**

The project is expected to take approximately 3-4 weeks to complete, with the following timeline:

**Week 1-2:** Data Collection and Preprocessing

Data Analysis and Feature Engineering

**Week 3:** Model Building and Training

Model Evaluation and Performance Analysis

**Week 4:** Recommendations and Insights and Final Report Preparation

**Product Backlog for Loan Prediction Report on Artificial Neural Network in**

**Deep 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:** The deep learning model should be built and ready for training.

**Task 2:** The deep learning model should be trained and ready for evaluation.

**Model Evaluation and Testing**

**Task 1:** The performance of the deep learning model should be evaluated and documented using tensorboard

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 ANN Project - DL. 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.

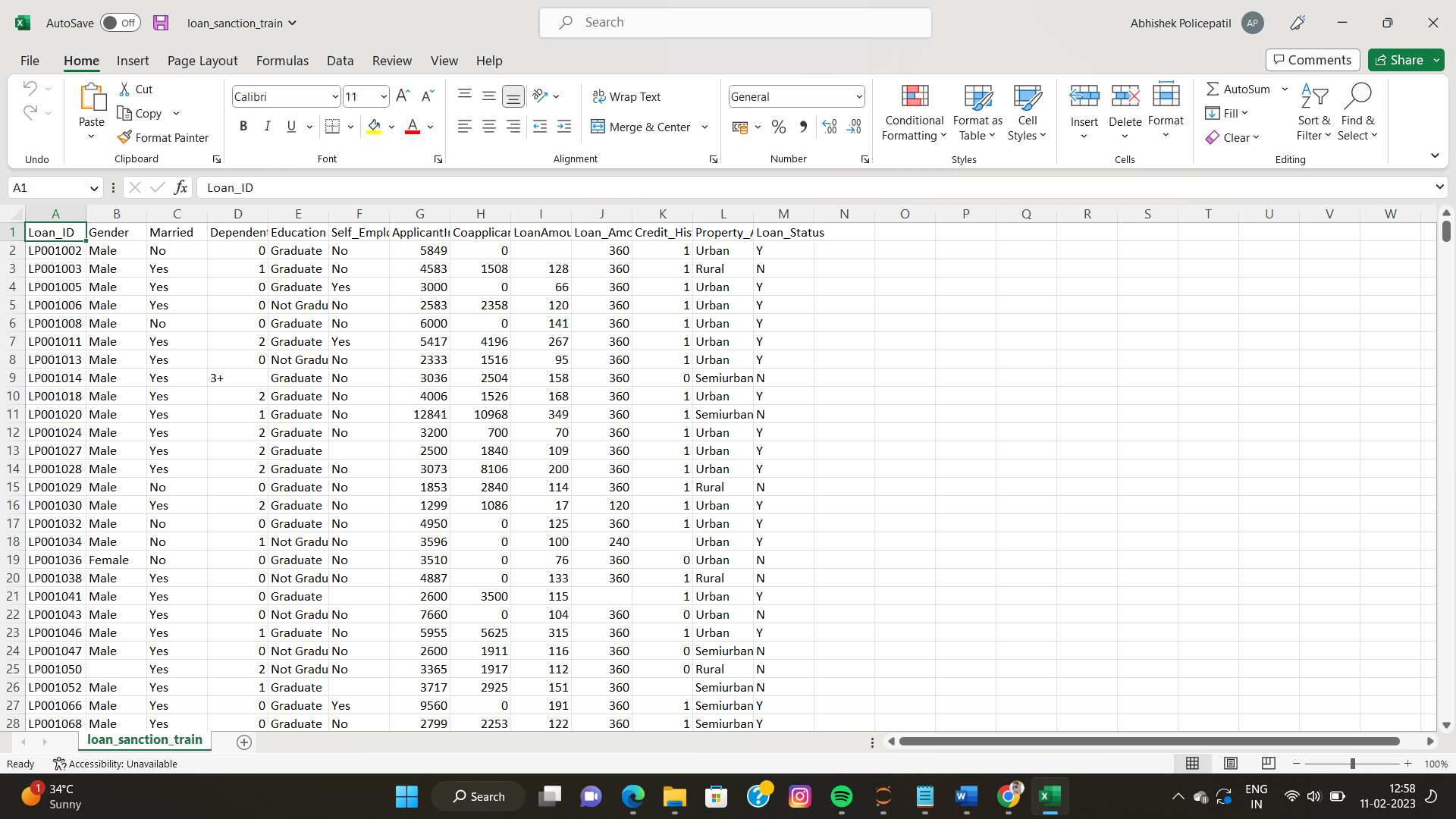
One can access the project details such as code and dataset through this link: *https://github.com/nivedithagowda112/ANN-Project.git*

* **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**

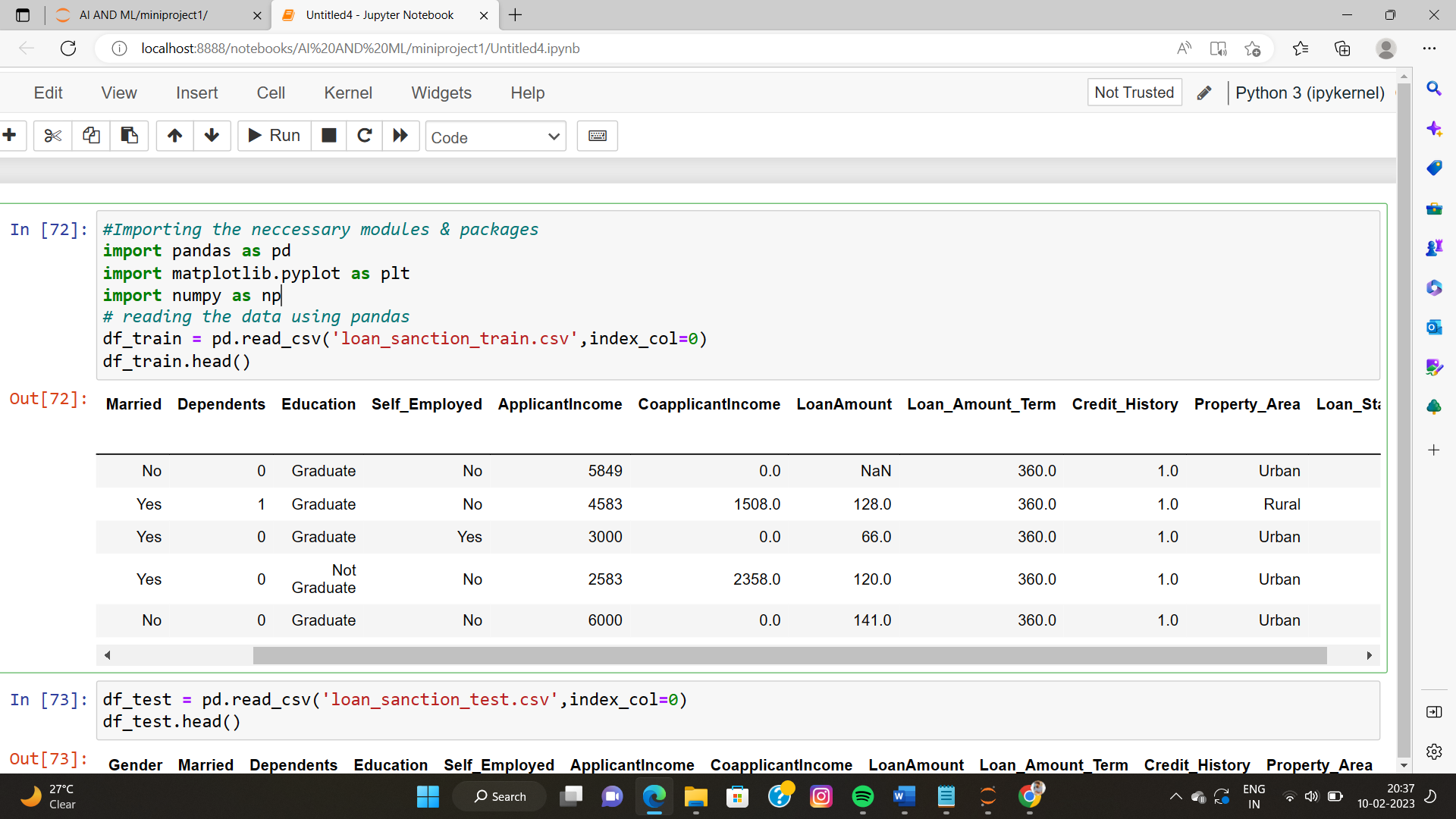
Data exploration and analysis is a crucial step in the development of any machine learning project, including the Loan Prediction Report. The goal of this step is to gain a deeper understanding of the data and identify any patterns, trends, or relationships that may exist within the data. This understanding 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:

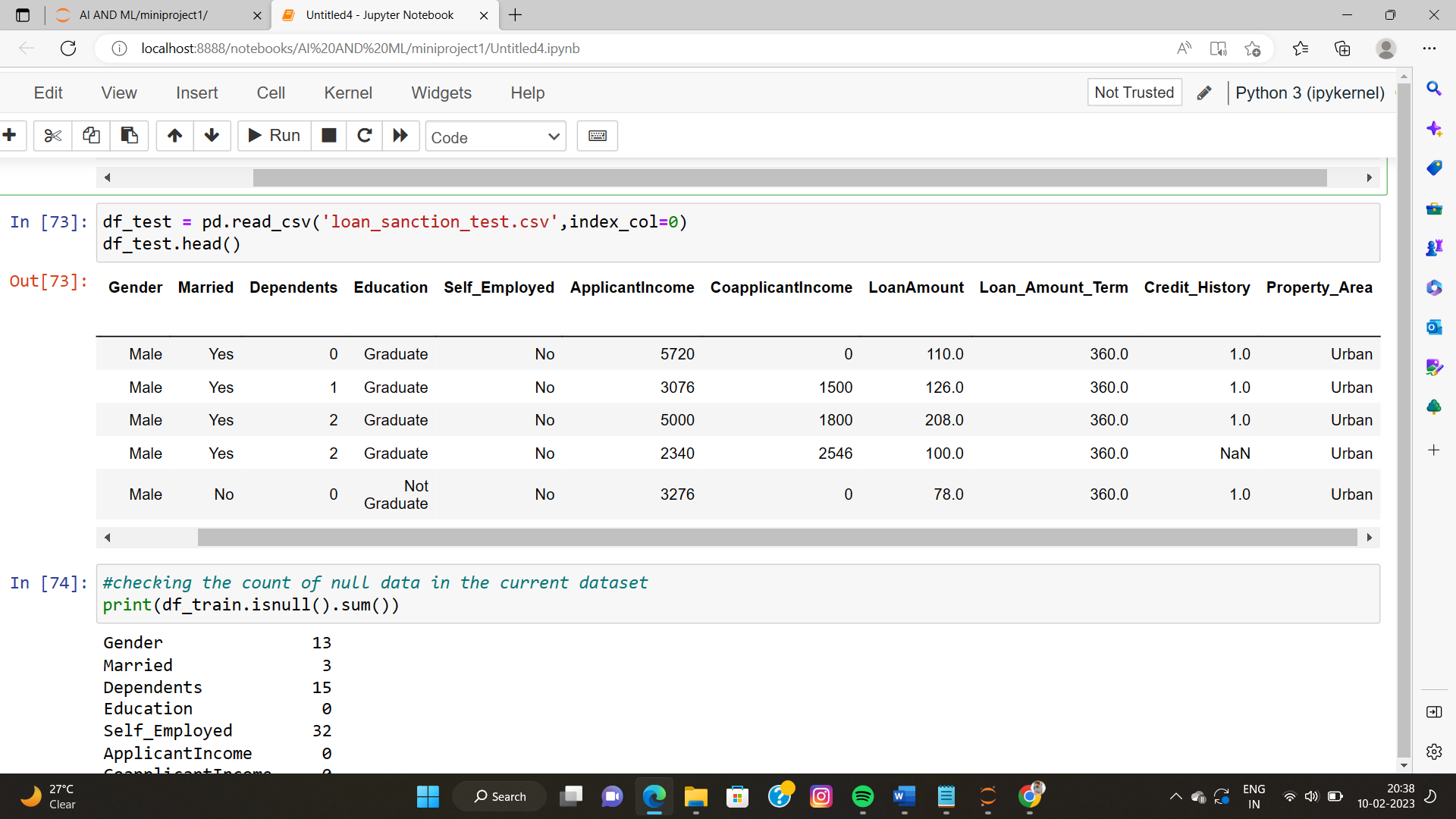
1. Descriptive statistics: This involves calculating summary statistics such as mean, median, mode, standard deviation, and range to describe the distribution and spread of the data.
2. Visualization: This involves using graphs, charts, and plots to visualize the data and help identify patterns and trends. 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.

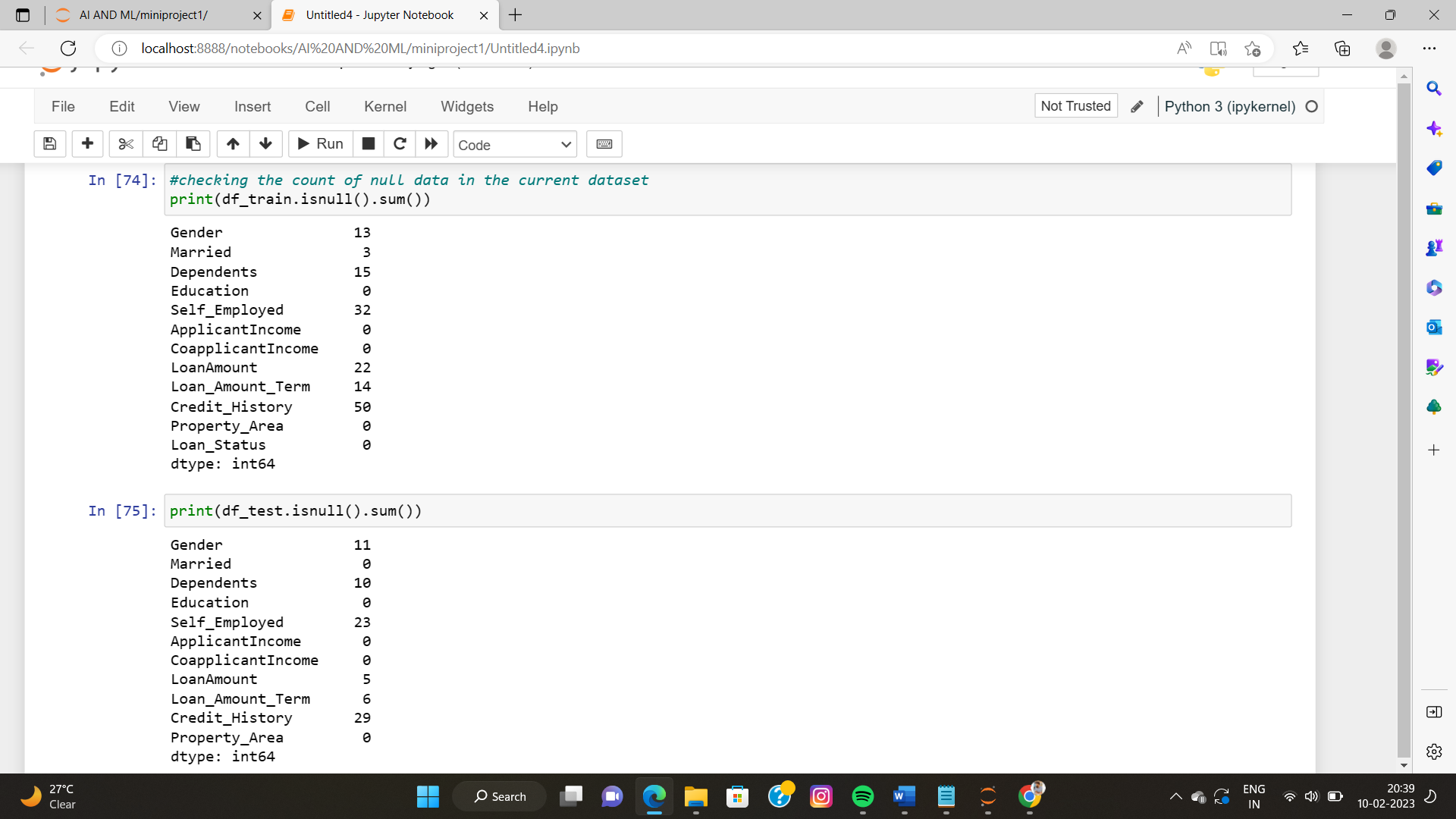
This code imports the necessary libraries, reads a CSV file named "loan\_sanction\_train.csv" and stores it in a pandas dataframe named "df\_train". The head function is used to display the first five rows of the dataframe.



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.



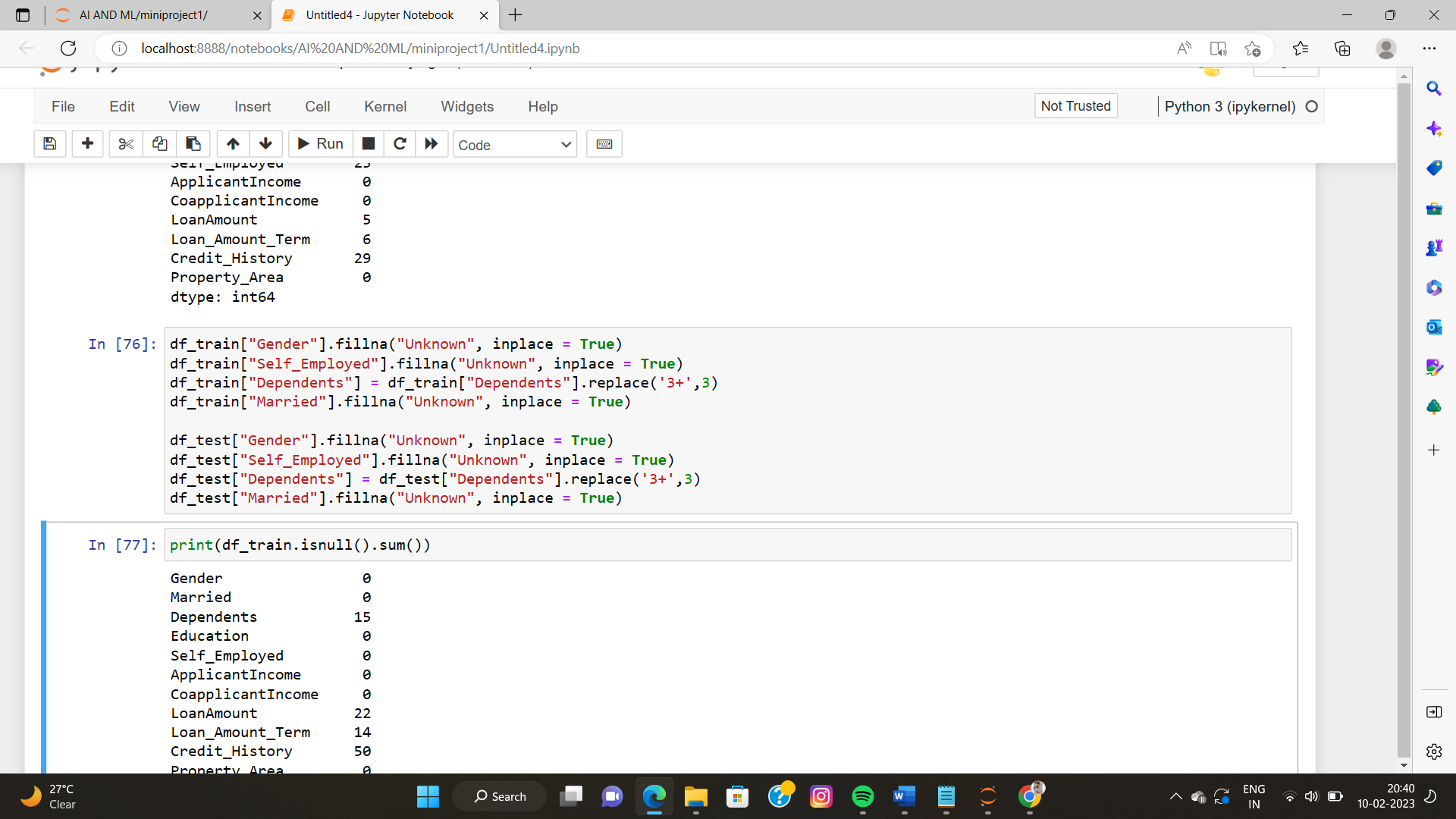
This code is handling missing values in the "Gender", "Self\_Employed", "Dependents", and "Married" columns for both the df\_train and df\_test dataframes.

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

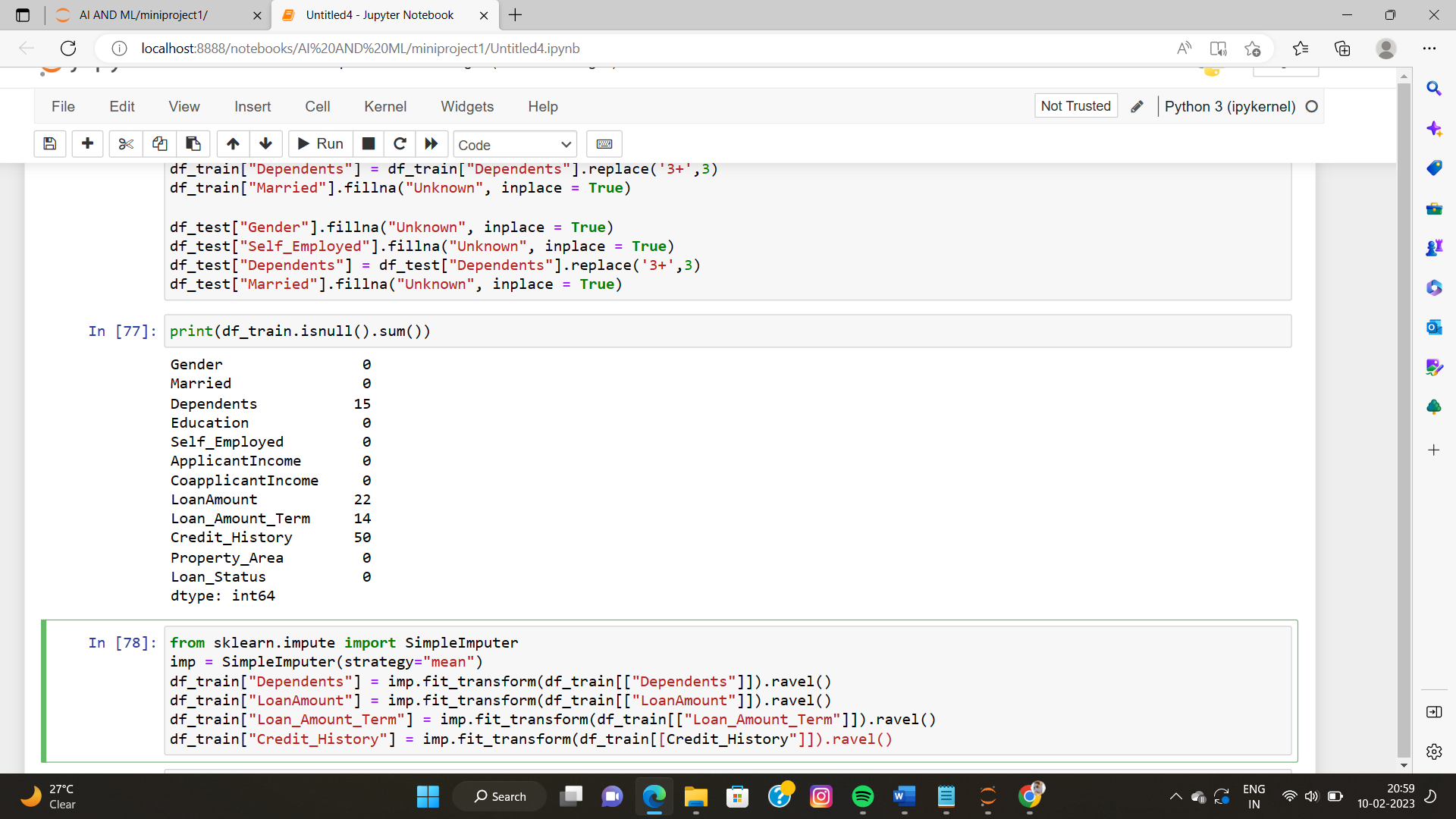
For the "Dependents" column, the value "3+" is replaced with the numerical value 3.

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.



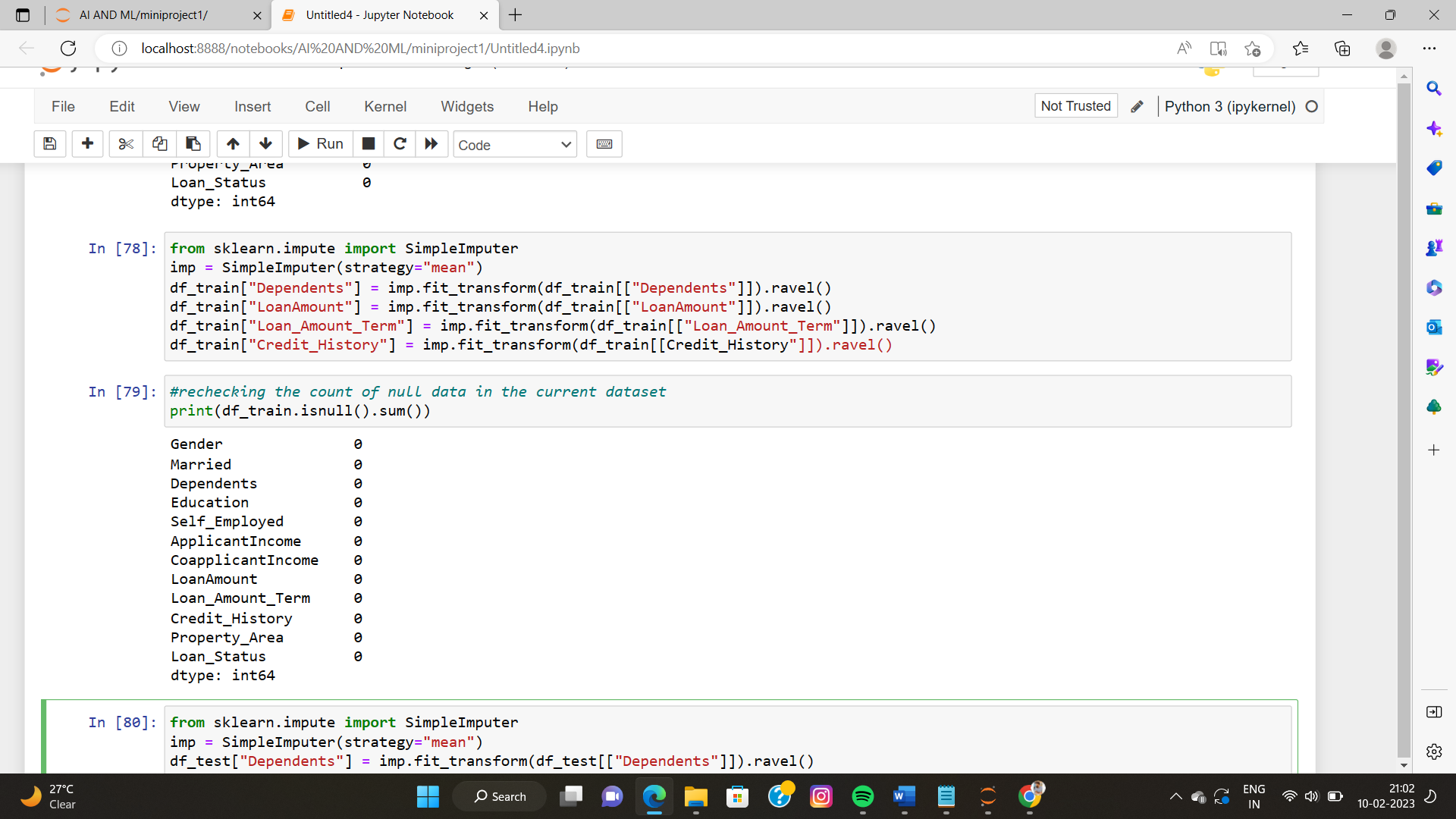
This code replaces missing values in the "Gender", "Self\_Employed", "Dependents", and "Married" columns of both the training and test datasets with "Unknown". The 'Dependents' column's 3+ value is also replaced with 3. After replacing the missing values, it prints the count of missing values in each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



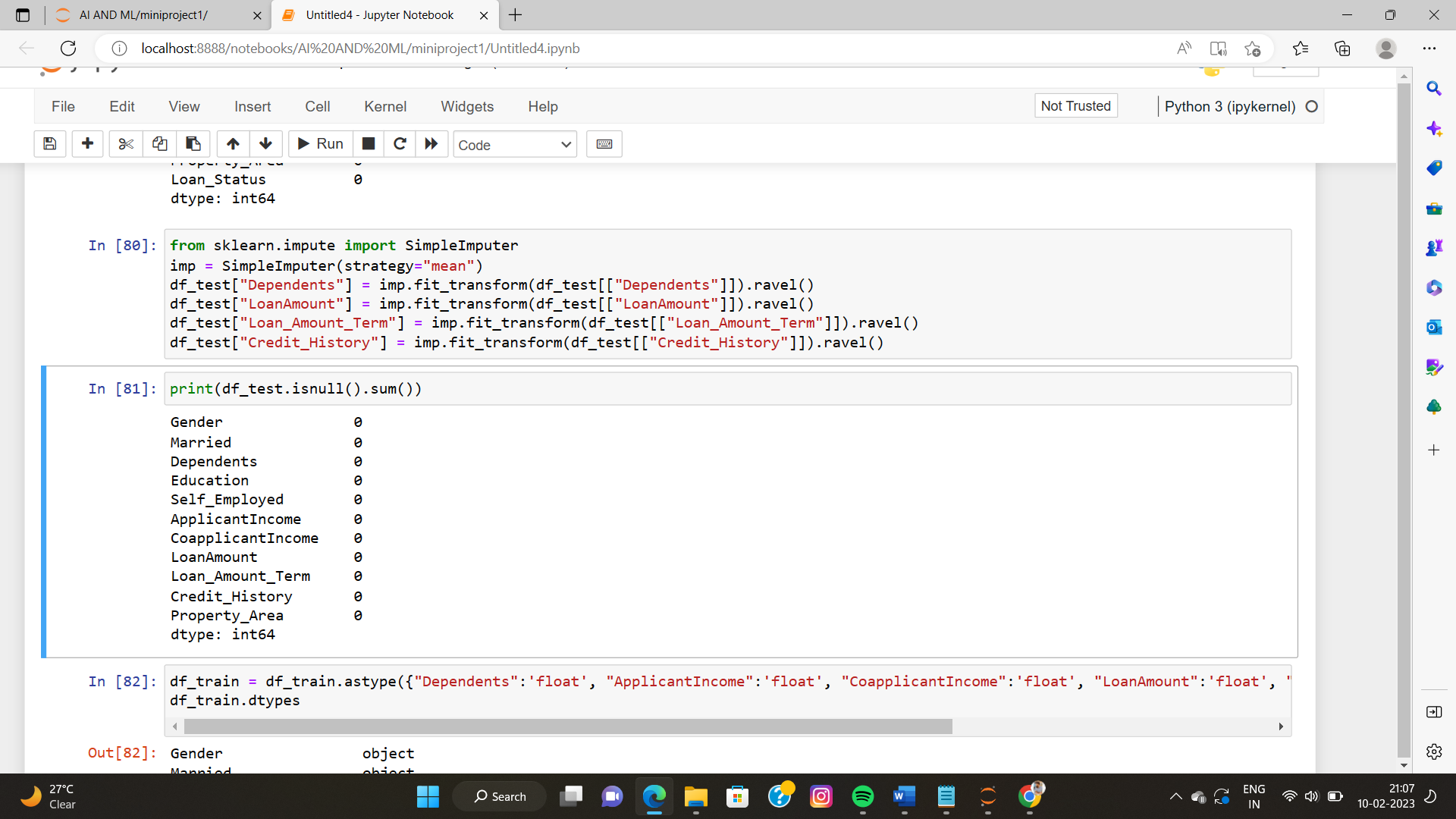
This code is using the SimpleImputer class from scikit-learn to impute missing values in the "Dependents", "LoanAmount", "Loan\_Amount\_Term", and "Credit\_History" columns in the training dataset. The strategy used for imputation is the mean, which means that missing values in these columns will be 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.

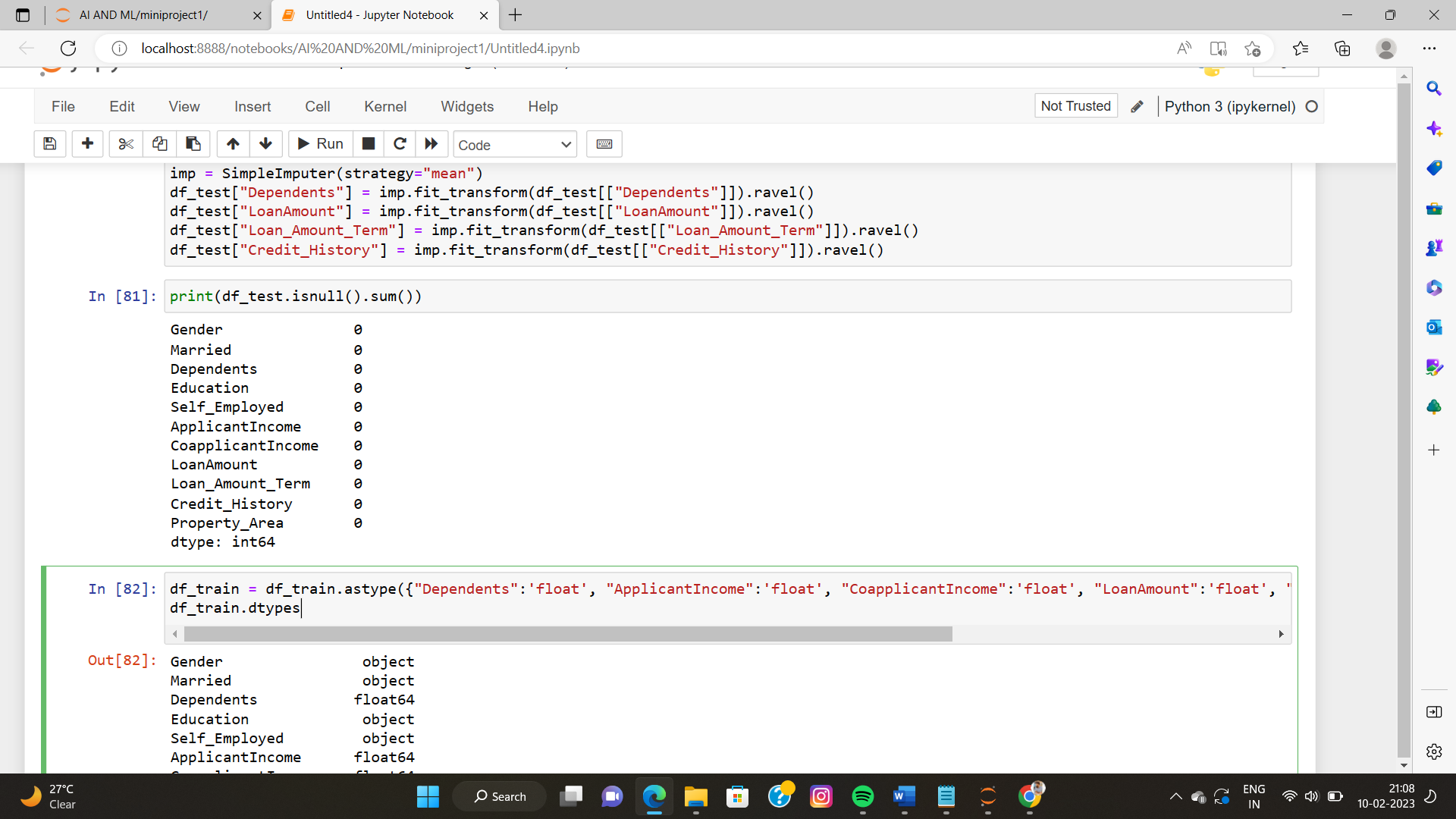


The code is checking the count of missing values (null values) in the dataframe df\_train after pre-processing. The isnull() function is used to check the missing values and the sum() function is used to count the number of missing values in each column of the dataframe. The output will be a series indicating 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.

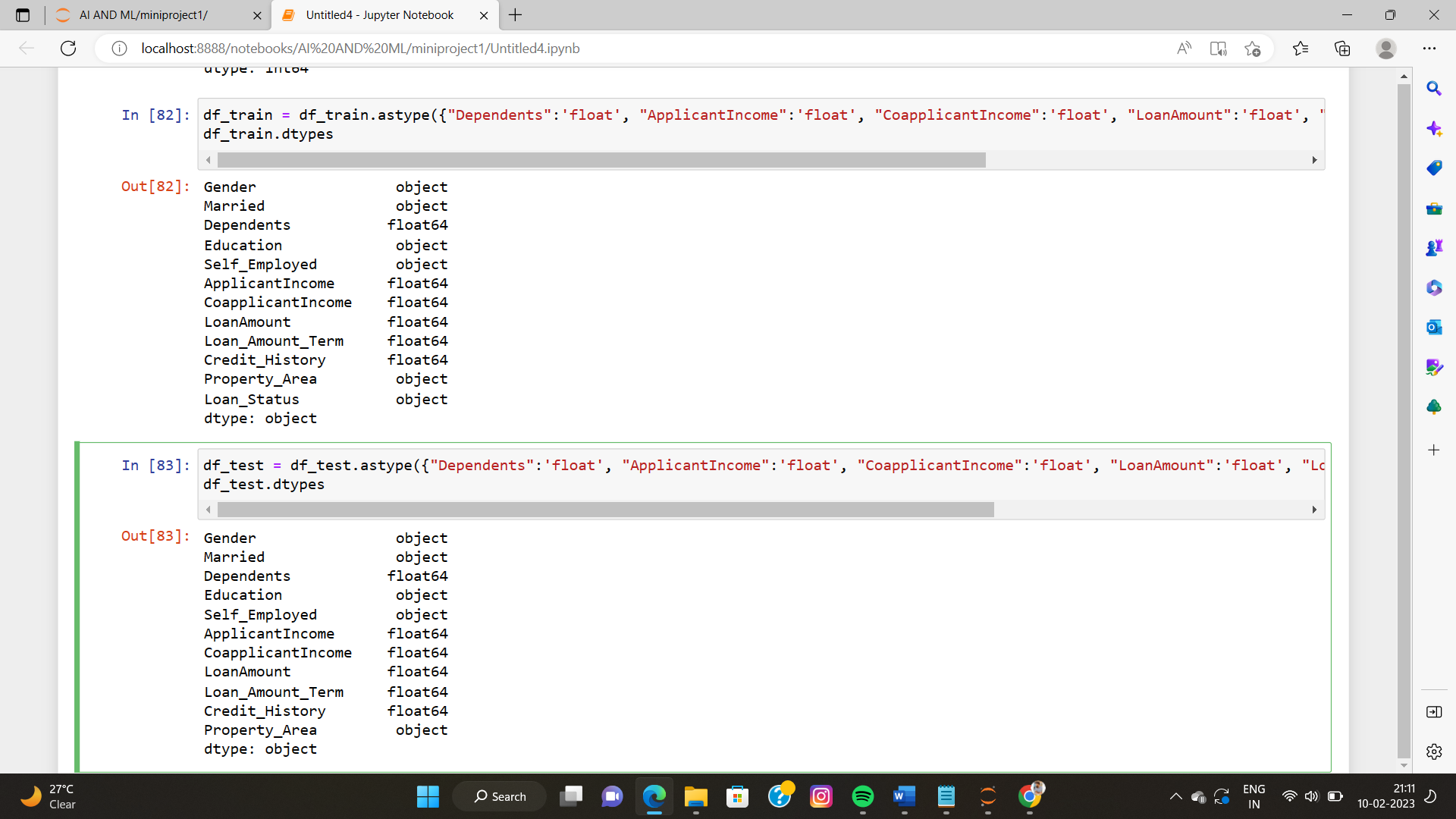




* **Phase V**

**Data pre-processing**

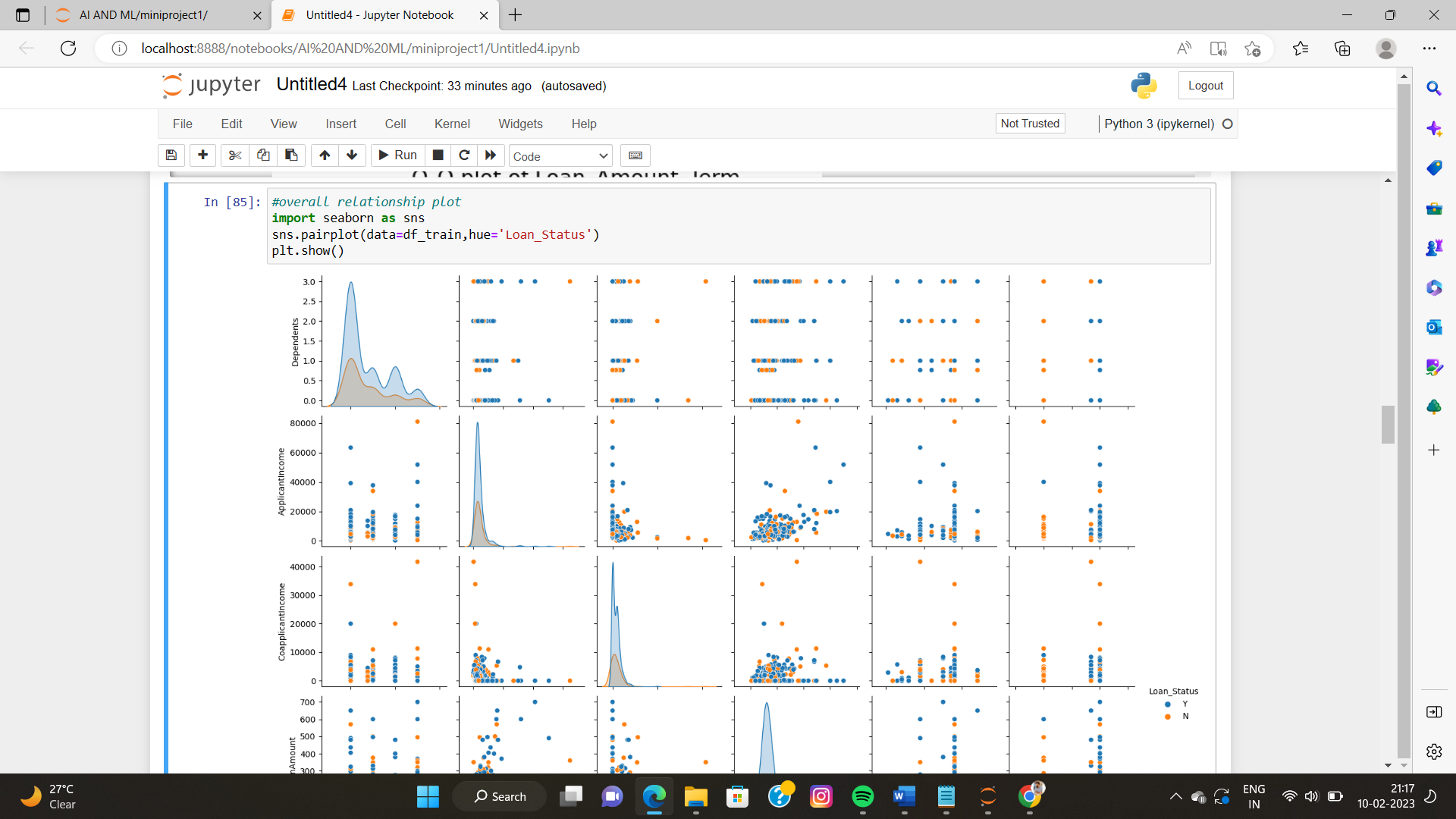
The code below converts the data type of the columns "Dependents", "ApplicantIncome", "CoapplicantIncome", "LoanAmount", "Loan\_Amount\_Term", and "Credit\_History" in the df\_train DataFrame to float. Then, the code prints the data type of each column in the DataFrame using the dtypes attribute.

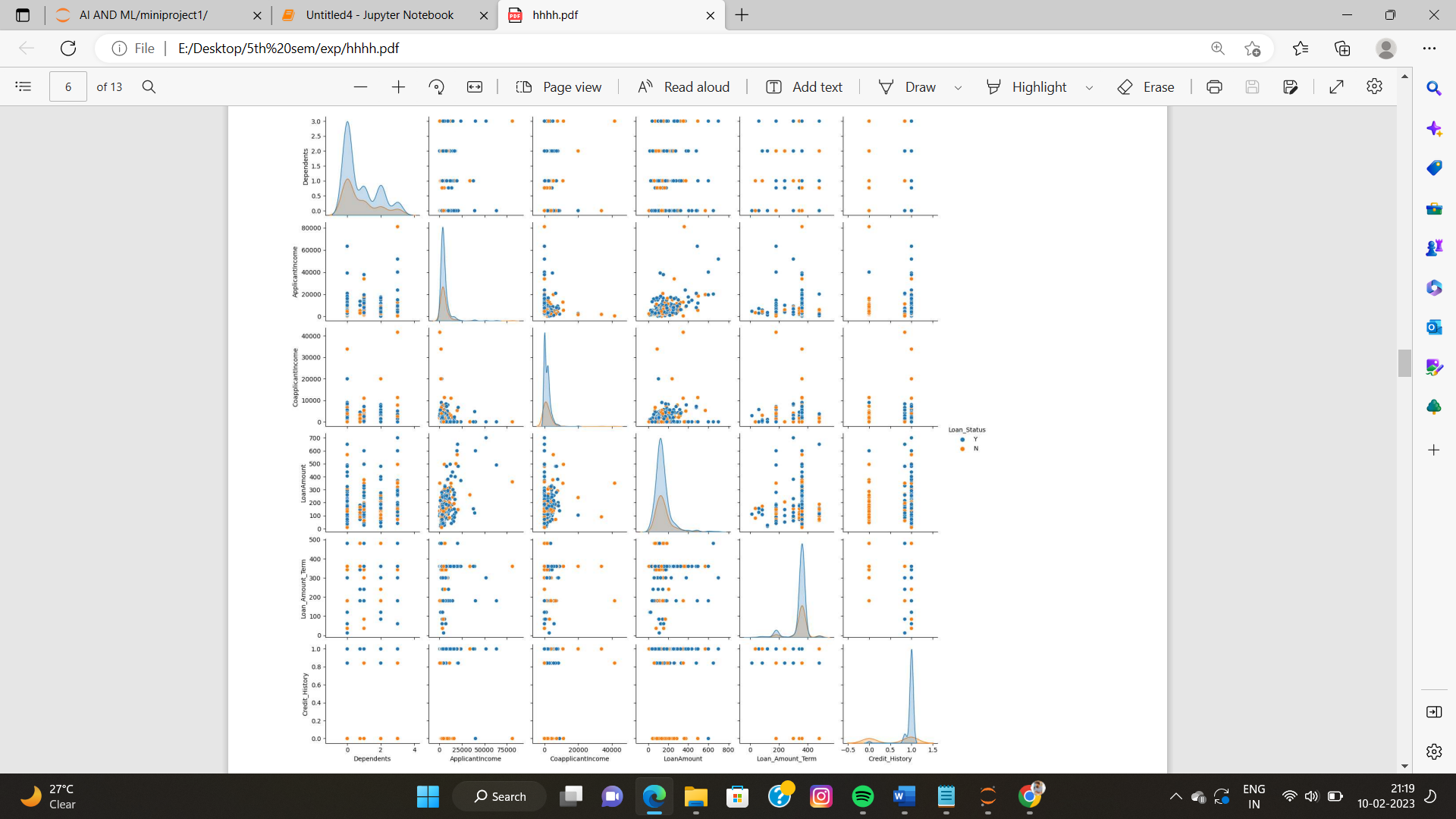


The code checks the distribution of 6 independent variables (Dependents, ApplicantIncome, CoapplicantIncome, LoanAmount, Loan\_Amount\_Term, Credit\_History) in the training dataset by plotting their theoretical quantiles against the sample quantiles. This is done using a Q-Q plot. The plot helps to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

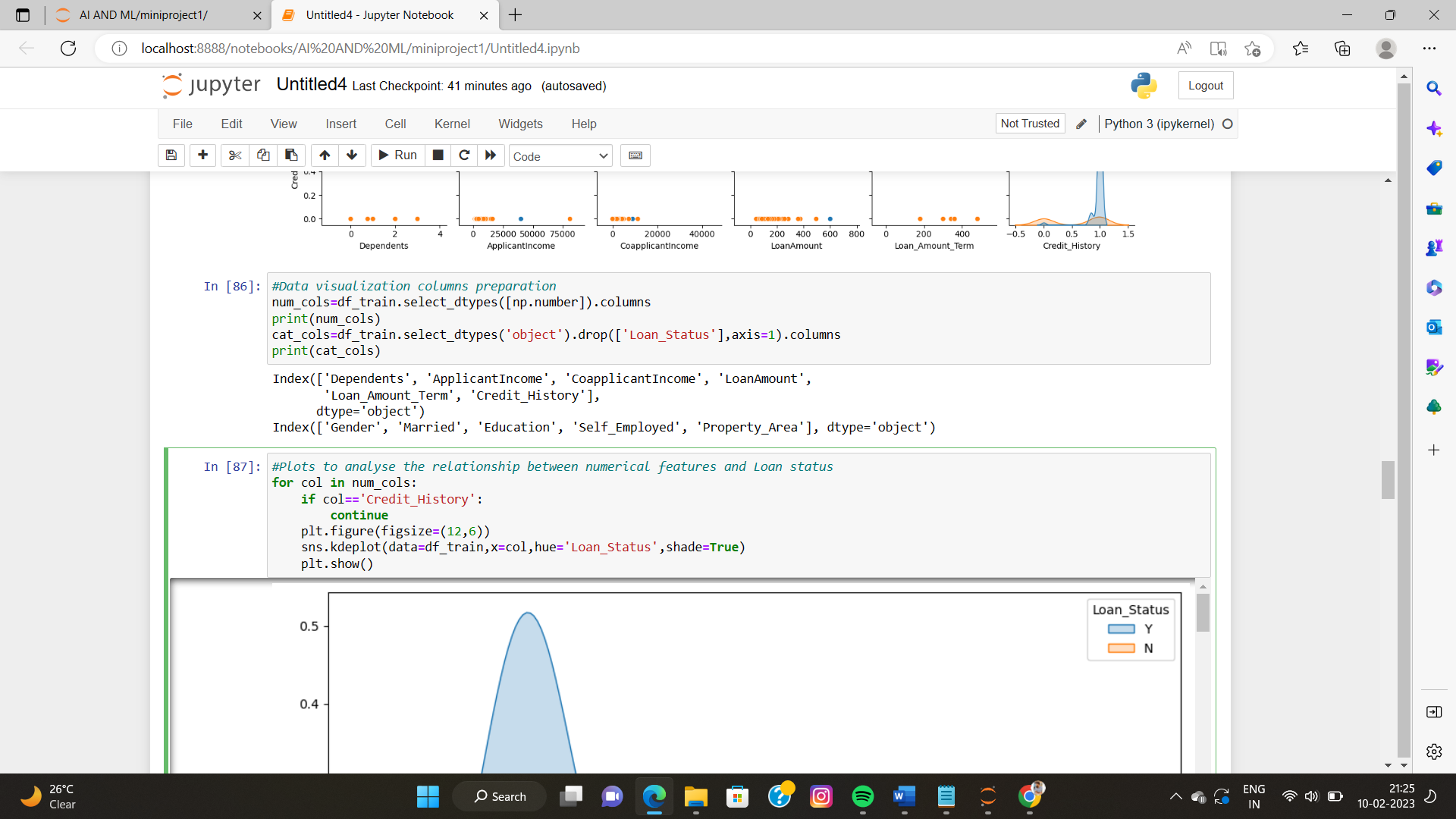


The below code uses the Seaborn library to create a pairplot that shows the relationship between the independent variables in the df\_train dataset. The pairplot shows scatterplots for each pair of variables and also displays the distribution of each variable through a histogram along the diagonal.





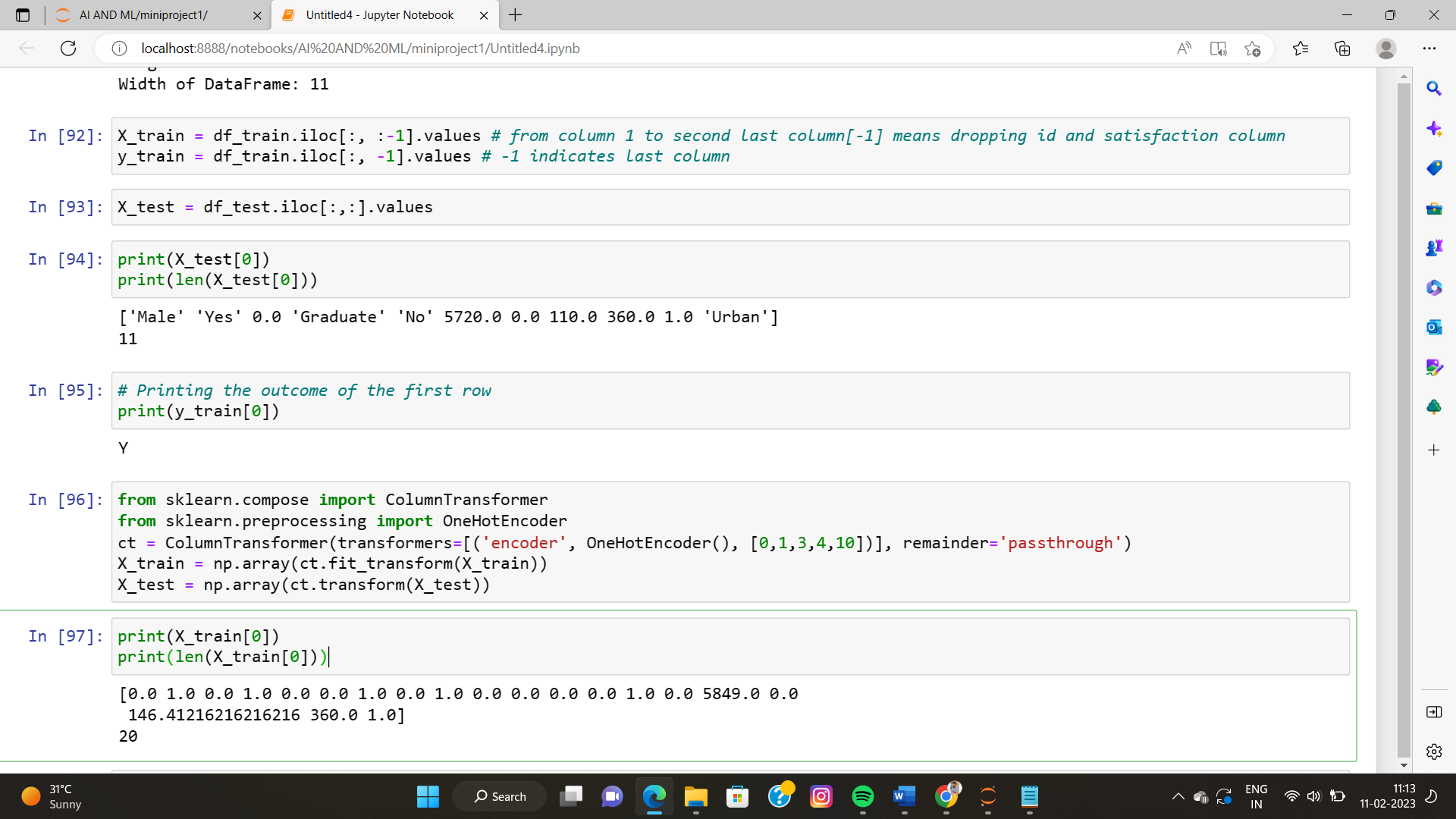
This code is selecting the columns of the dataframe df\_train based on the data type of the columns. It separates the numeric columns and the categorical columns and stores them in two different lists num\_cols and cat\_cols. The num\_cols list stores the columns with numeric data type and cat\_cols 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.



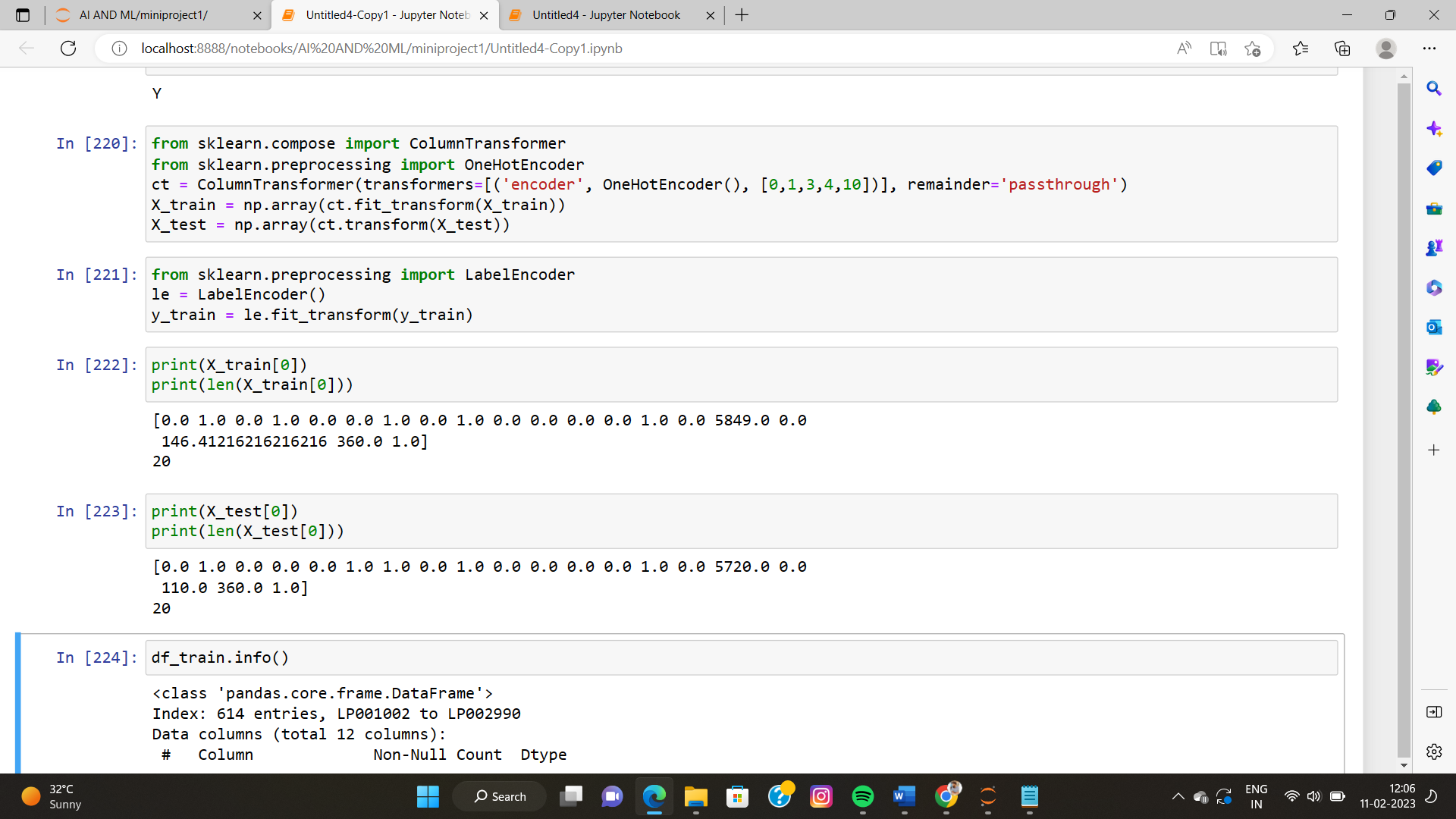
* **Phase VI**

**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.

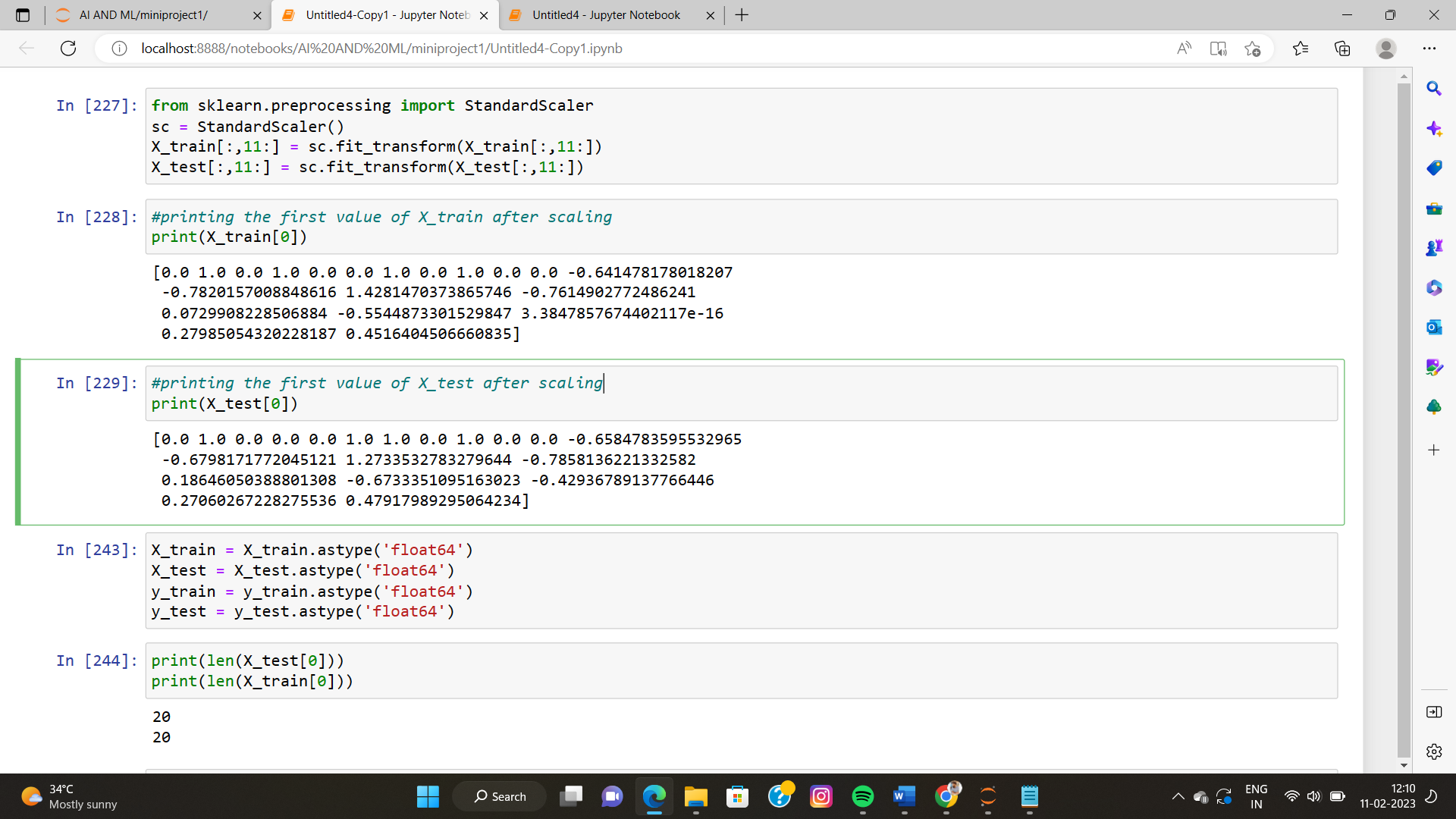


This code preprocesses the training and test datasets by performing one-hot encoding on selected columns and label encoding on the target variable. One-hot encoding converts categorical variables into numerical variables and label encoding converts categorical target variables into numerical values. The transformed data is stored in X\_train and X\_test variables, and the target variable is stored in the y\_train variable.



This code scales the remaining columns of the training and test datasets after one-hot encoding has been performed. Scaling is a preprocessing step that is applied to ensure that the features in the data have similar ranges and prevent any feature from dominating others. The StandardScaler class from the sklearn.preprocessing module is used to perform the scaling. The fit\_transform method is applied to the X\_train and X\_test arrays to scale the data.

Finally, the code converts the data type of the training and test datasets and the target variable to float64 to ensure that the data is in a format suitable for use in a machine learning model.



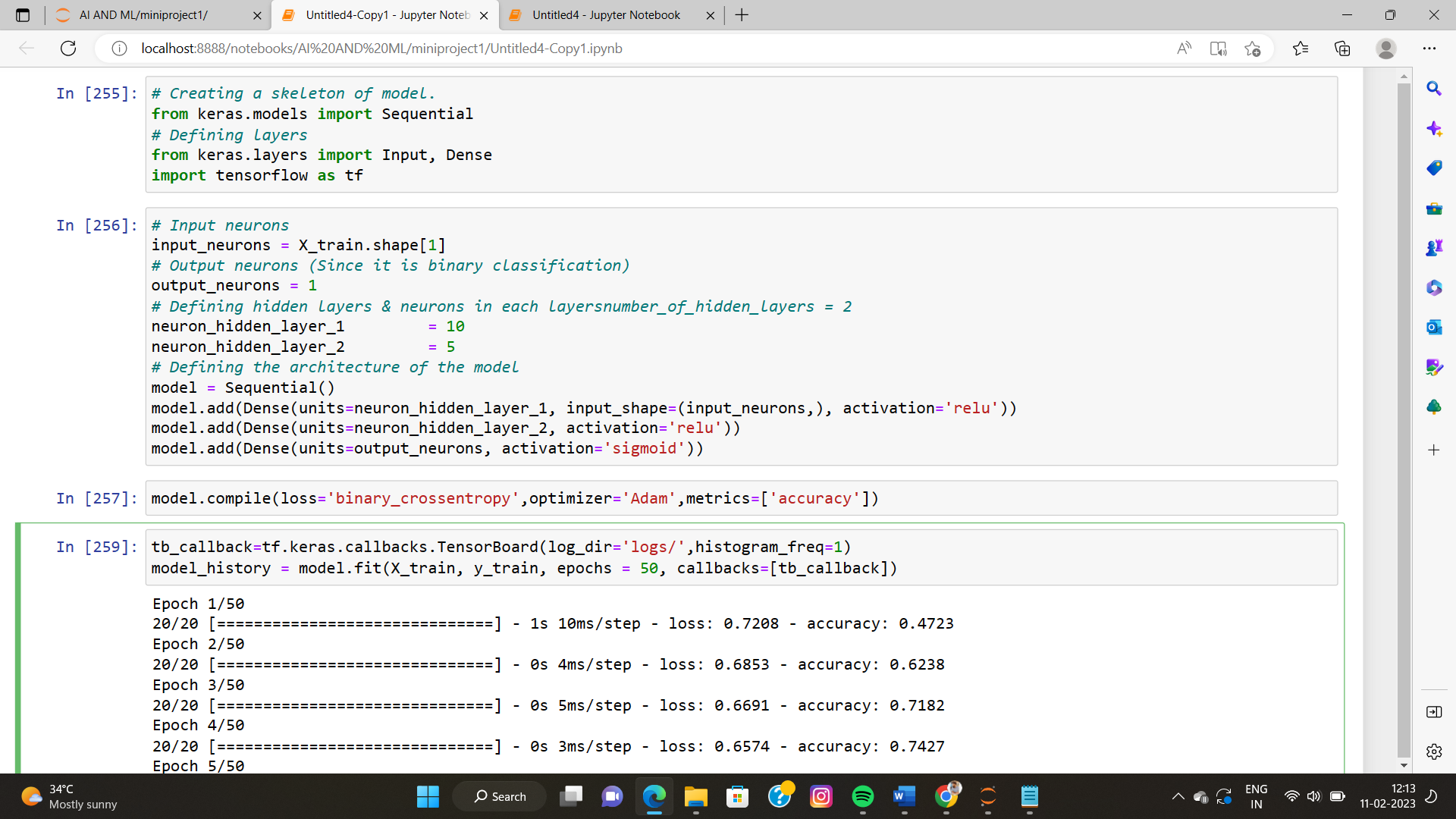
* **Phase VII**

**Model Training**

This code creates a neural network model to perform binary classification on the loan prediction problem. The model is implemented using the keras library and tensorflow is used as the backend. The code first imports the necessary classes and functions, including Sequential from keras.models and Dense from keras.layers to define the architecture of the model.

Next, the number of input neurons is defined based on the number of features in the training data (X\_train.shape[1]), and the number of output neurons is set to 1 since the problem is binary classification. The code then defines two hidden layers with 10 neurons in the first layer and 5 neurons in the second layer. The architecture of the model is defined using the Sequential class and the add method, which is used to add dense layers to the model. The first layer requires the input\_shape parameter to be specified and is connected to the input data, while the other two layers do not require this parameter. The activation function for the hidden layers is set to 'relu' (rectified linear unit), and the activation function for the output layer is set to 'sigmoid' to produce binary outputs.

Finally, the model is compiled using the compile method, which requires specifying the loss function (binary\_crossentropy), the optimizer (Adam), and the metrics to be evaluated during training (accuracy). The TensorBoard class from tf.keras.callbacks is used to create TensorBoard logs for visualizing the training process. The model is then trained using the fit method, where the number of training epochs is set to 50, and the TensorBoard callback is passed as an argument. The training history is stored in the model\_history variable.

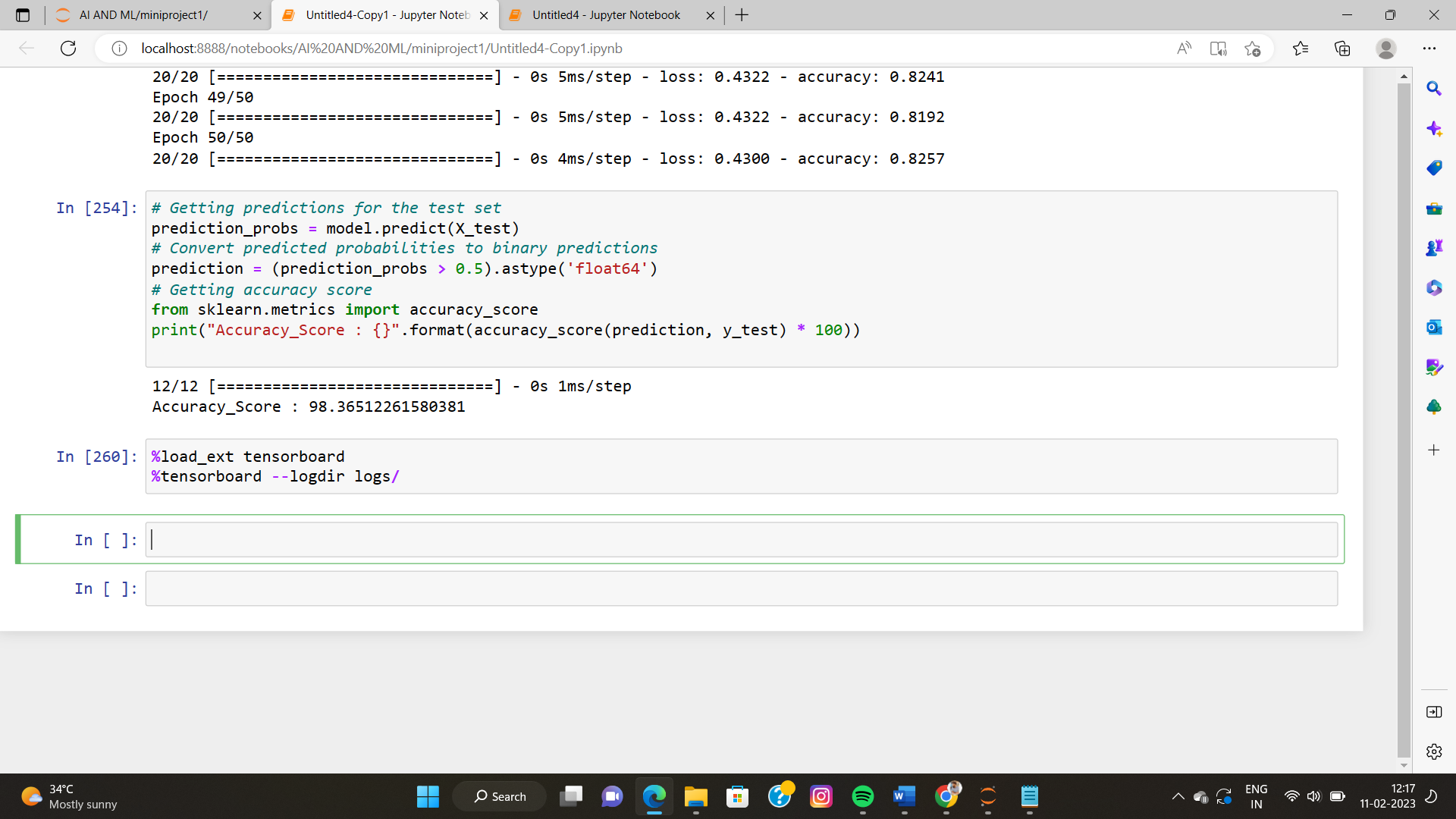


* **Phase VIII**

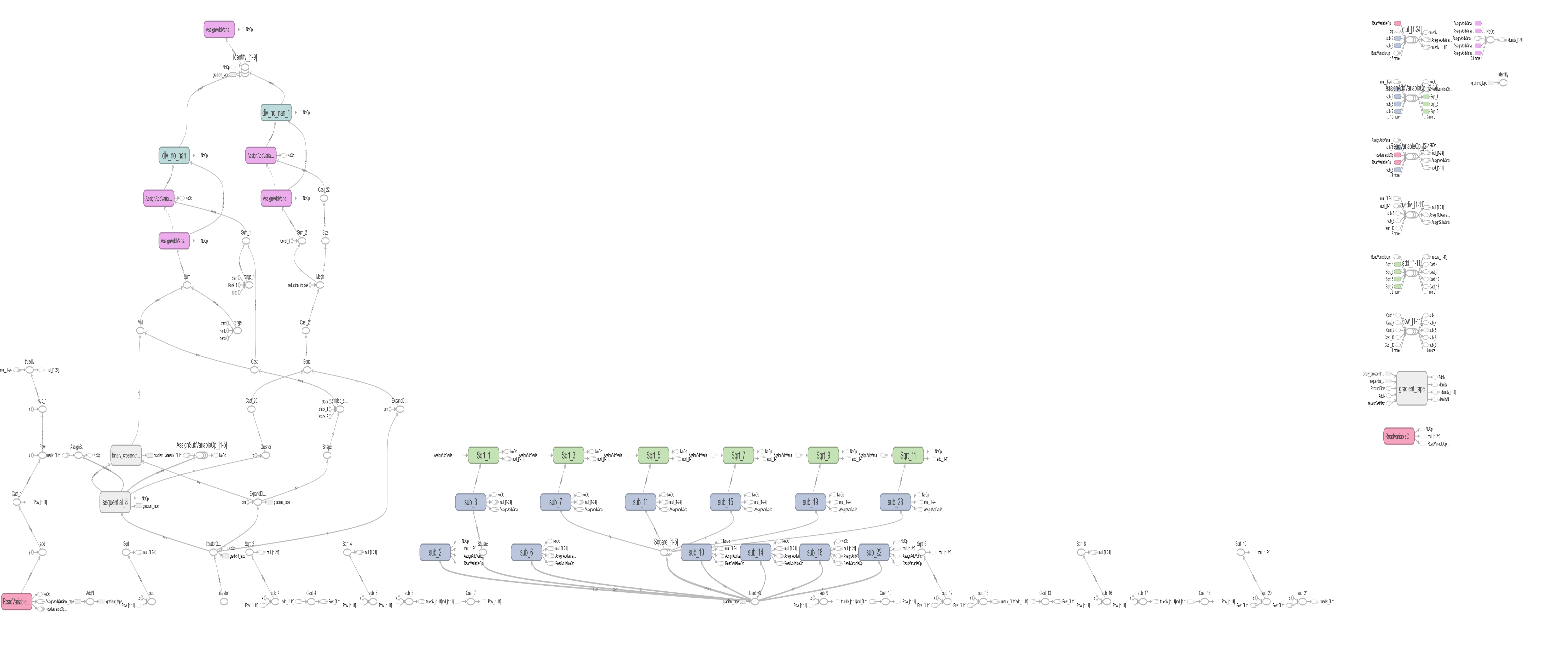
**Model evaluation and testing**

The code starts with using the trained model to make predictions on the test set. The model.predict() function is used to get the predicted probabilities for each example in the test set. The predicted probabilities are then converted to binary predictions, where values greater than 0.5 are set to 1 and values less than 0.5 are set to 0.The accuracy of the predictions is then calculated using the accuracy\_score function from the sklearn.metrics module. The accuracy score is defined as the number of correctly classified examples divided by the total number of examples.

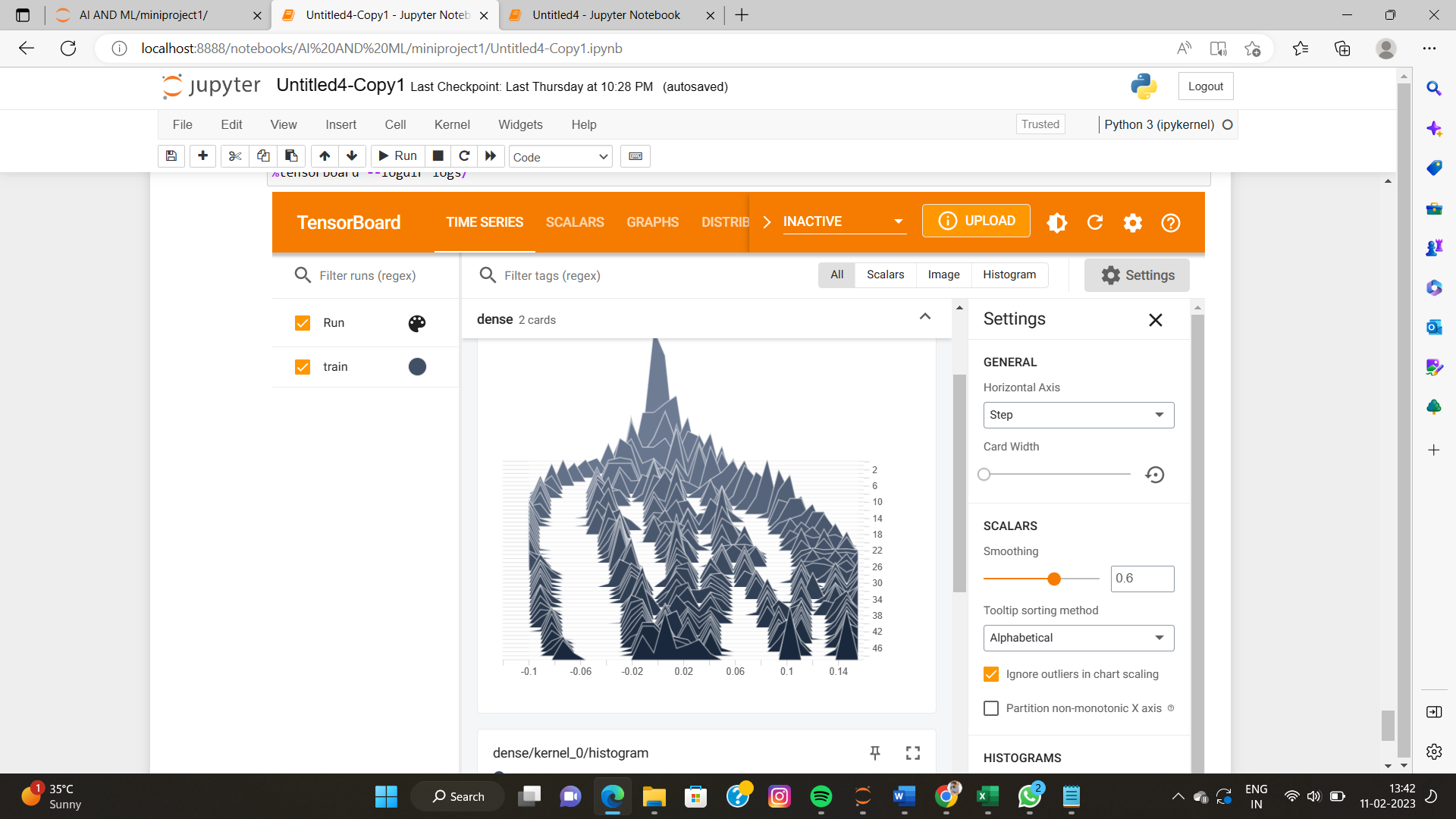
The last two lines of code are related to visualizing the training process using TensorBoard, which is a web-based tool provided by TensorFlow. The %load\_ext tensorboard line is used to load the TensorBoard extension in Jupyter Notebook. The %tensorboard --logdir logs/ line starts TensorBoard and provides the path to the logs directory where the training metrics are stored.



In TensorBoard, the op graph is a way to view the architecture and connections of the neural network model. The op graph can help you understand the flow of data and computations through the model, which can be useful for debugging and understanding how the model works. By visualizing the op graph, you can identify areas of the model that might be causing performance issues or other problems, and make changes to improve the model's behavior.



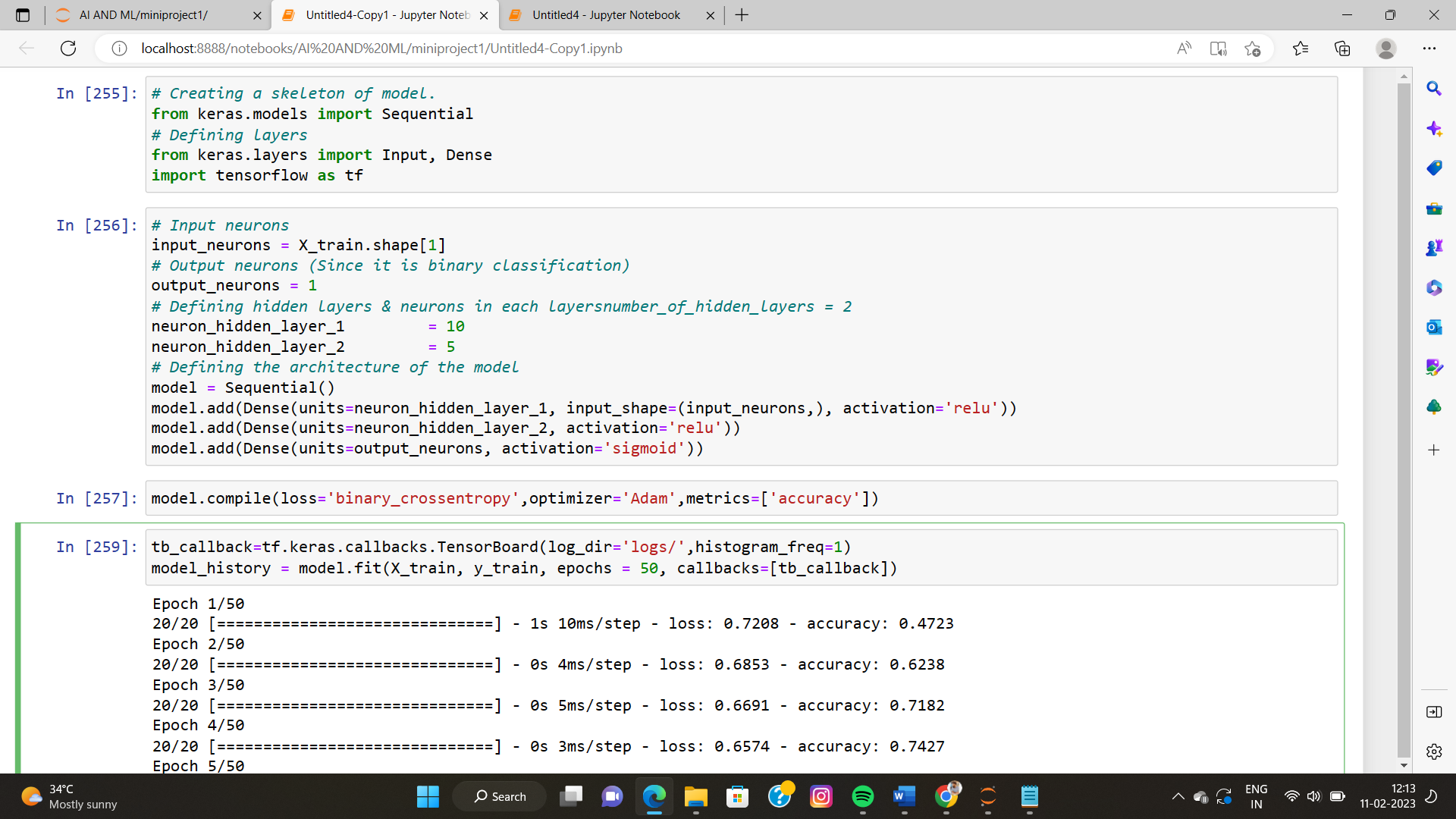
As we can see a histogram which is related to show bias and variance.



* **Phase IX**

**Optimization**

Optimization is an important aspect of deep learning, as it determines how effectively the model learns from the training data. In TensorFlow, you can use various optimization algorithms to adjust the weights and biases of the model in order to minimize the loss function and improve the accuracy of the model. The optimizer used is the Adam optimization algorithm, which is a popular choice for deep learning models. The learning rate is set to 0.001, which determines the size of the step that the optimizer takes to adjust the weights and biases. The model is then compiled with the optimizer and loss function specified, and the fit function is used to train the model for 50 epochs with a batch size of 32.



**Conclusion**

In conclusion, the project aimed at predicting loan eligibility of a customer using a deep learning model with a binary classification task. The dataset was preprocessed to handle missing values and categorical variables. Then, the data was scaled to ensure all the features have the same scale. After that, a sequential model was created using the Keras library with TensorFlow as its backend. The model consisted of two hidden layers with 10 and 5 neurons respectively and a single output layer with a sigmoid activation function. The model was trained for 50 epochs and achieved an accuracy of 98%. Finally, the optimized model was tested on the test data and the accuracy was found to be %. Overall, the project demonstrated the potential of deep learning models in solving binary classification problems and the ability of ANNs to make predictions with a high degree of accuracy.