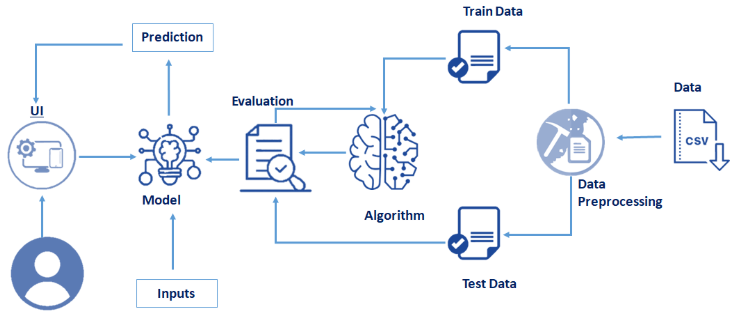


**CREDIT CARD APPROVAL PREDICTION USING MACHINE LEARNING**

The Credit Card Approval Prediction system is a machine learning-based solution designed to evaluate the likelihood of a credit card application being approved or denied. By analyzing key applicant features such as income, credit score, employment status, and financial history, the system aims to support financial institutions in streamlining the approval process, minimizing risk, and enhancing customer satisfaction

**Technical Architecture:**

****

**Project Flow:**

* User interacts with the UI to enter the input.
* Entered input is analyzed by the model which is integrated.
* Once the model analyses the input, the prediction is showcased on the UI

Define Problem / Problem Understanding

* Specify the business problem
* Business requirements
* Literature Survey
* Social or Business Impact

Data Collection & Preparation

* Collect the dataset
* Data Preparation

Exploratory Data Analysis

* Descriptive statistical
* Visual Analysis

Model Building

* Training the model in multiple algorithms
* Testing the model

Performance Testing & Hyperparameter Tuning

* Testing model with evaluation metrics
* Comparing model accuracy before & after applying hyperparameter tuning

Model Deployment

* Save the best model
* Integrate with Web Framework

Project Demonstration & Documentation

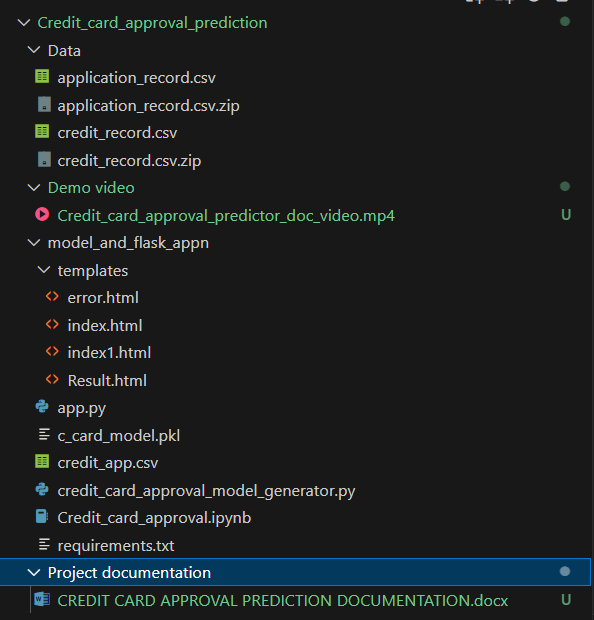
* Record explanation video for project end-to-end solution
* Project documentation – Step-by-step project development procedure

**PRIOR KNOWLEDGE:**

One must have prior knowledge of the topics to complete this project

* [Supervised Learning](https://www.geeksforgeeks.org/supervised-machine-learning/)
* [Decision Tree](https://www.geeksforgeeks.org/decision-tree-introduction-example/)
* [Hyper-parameter tuning techniques](https://www.geeksforgeeks.org/machine-learning/how-to-tune-a-decision-tree-in-hyperparameter-tuning/)
* [Evaluation metrics](https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/)
* [Flask Basics](https://www.youtube.com/watch?v=lj4I_CvBnt0&embeds_referring_euri=https%3A%2F%2Fskillwallet.smartinternz.com%2F&source_ve_path=OTY3MTQ)

**PROJECT STRUCTURE:**

****

**PROBLEM DEFINITION:**

Credit card applications are typically assessed based on several financial and personal criteria, including income, credit score, employment status, and existing debts. Traditionally, this evaluation process is manual and prone to subjectivity, delays, and inconsistencies. In a fast-paced financial environment, there is a growing need for automated, accurate, and data-driven solutions to assess creditworthiness.

This project aims to develop a machine learning-based credit card approval prediction system that can automatically predict whether an applicant is likely to be approved or denied. By training the model on historical application data, the system can learn patterns that indicate approval or rejection, enabling financial institutions to make faster, fairer, and more reliable decisions.

**Model Impact**: The ultimate goal is to streamline the approval process, reduce default risk, and enhance customer satisfaction by providing timely and objective results.

**DATA COLLECTION AND PREPARATION:**

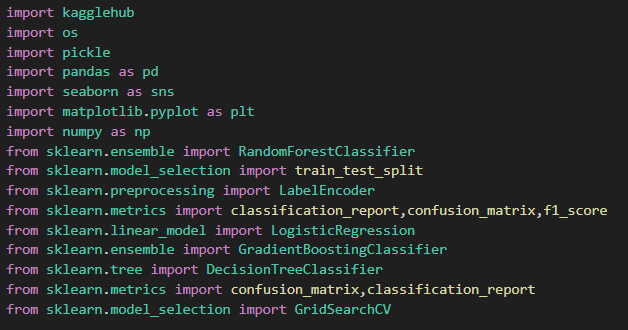
**Collect the dataset:**

Dataset: [Credit Card Approval Prediction](https://www.kaggle.com/code/namphuengauawatcharo/credit-card-approval-prediction/data)

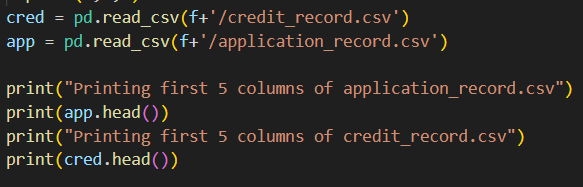
There are many popular open sources for collecting the data. Ex: kaggle.com, UCI repository, etc. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset or click the text given above.

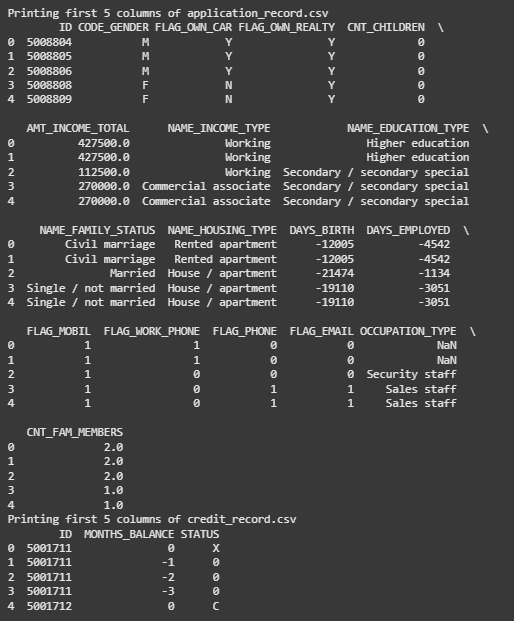
Link: <https://www.kaggle.com/namphuengauawatcharo/credit-card-approval-prediction/data>

**Importing the libraries**

****

**Read the Dataset:**

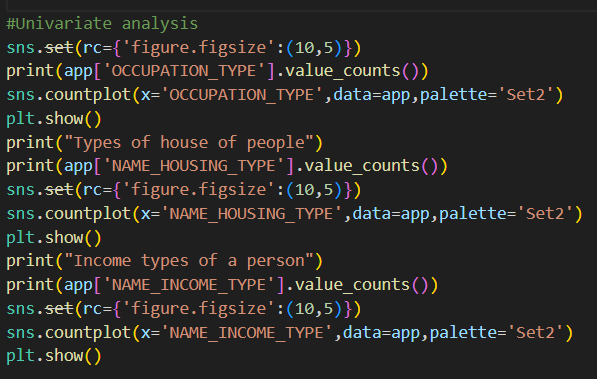
****

****

**EXPLORATORY DATA ANALYSIS:**

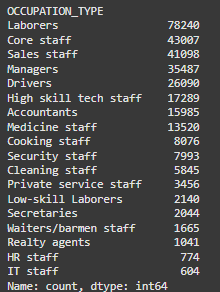
**UNIVARIATE ANALYSIS:**

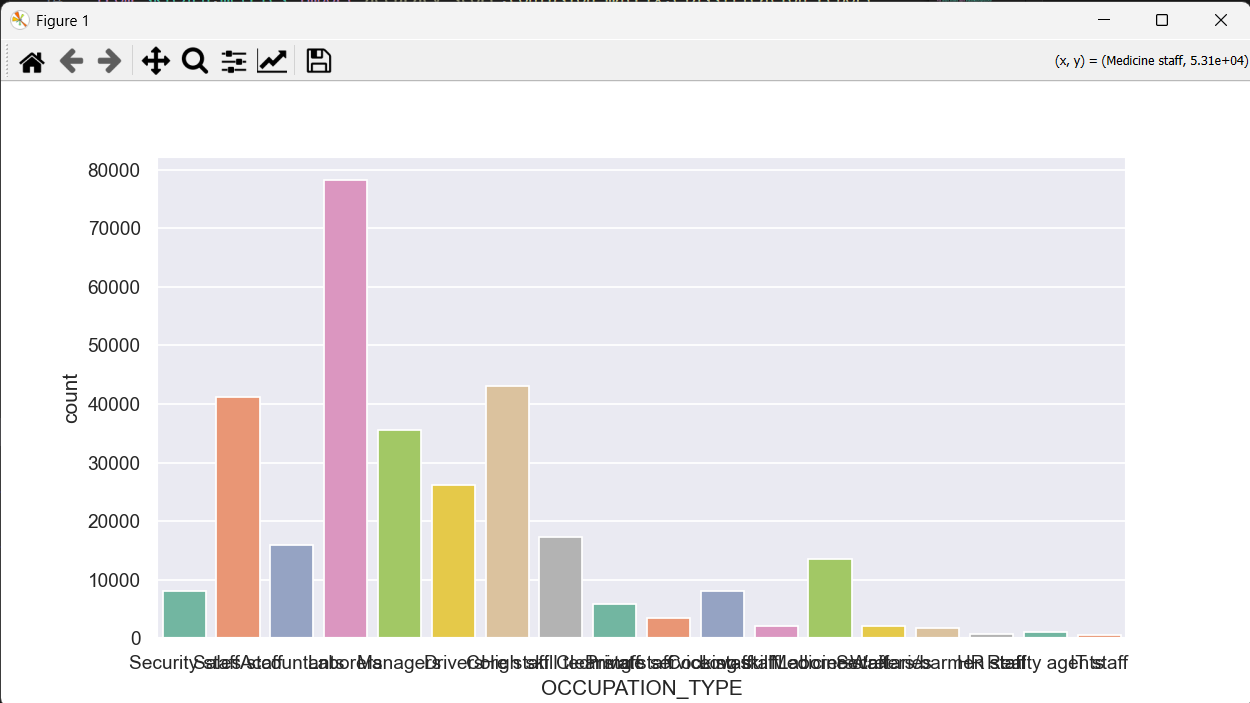
Univariate analysis was conducted to understand the distribution of individual categorical features using both counts and visualizations.

****

Occupation Type:

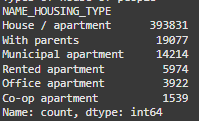
Frequencies of different occupations were displayed using value\_counts() and visualized with a count plot

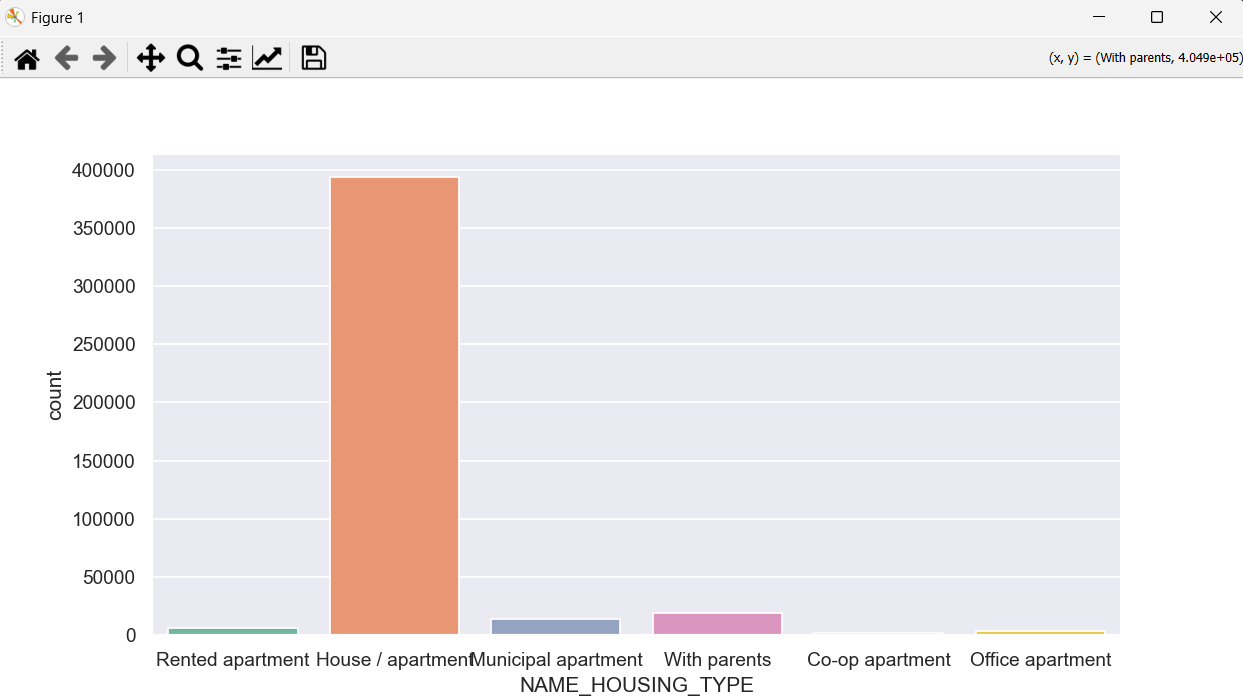
****

****

Housing Type:

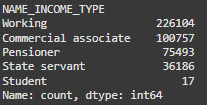
Different types of housing were analyzed to understand the residential situation of applicants.

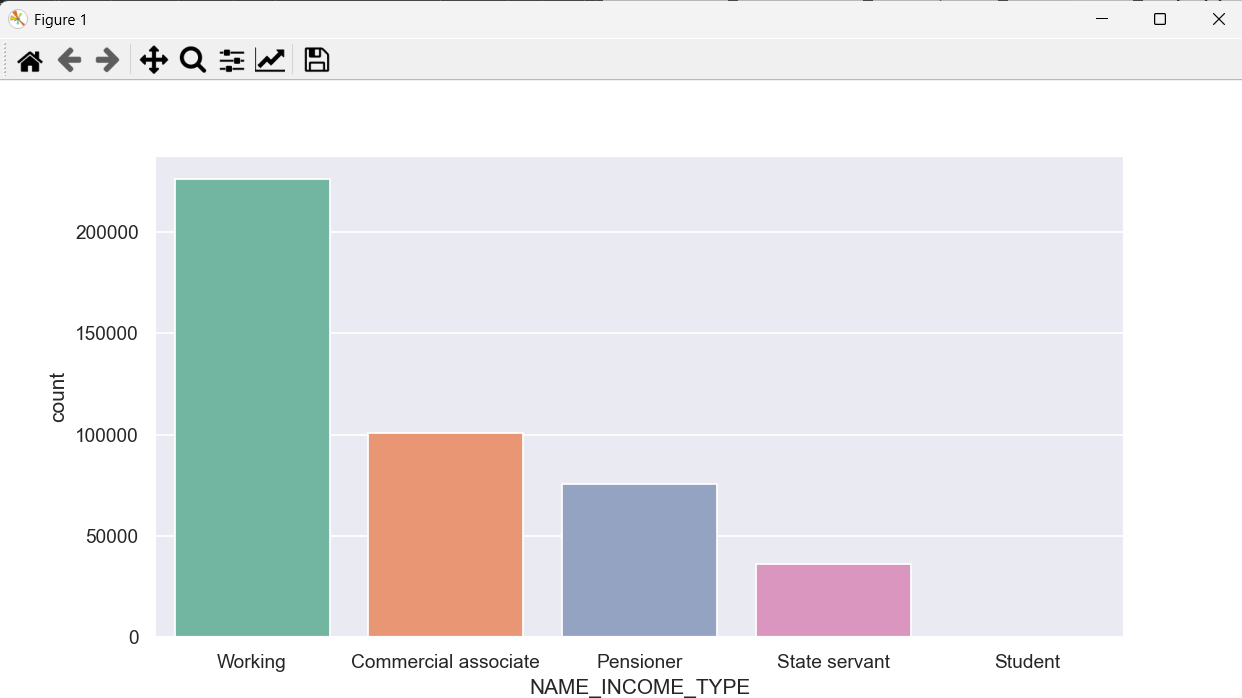
****

****

Income Type of a Person:

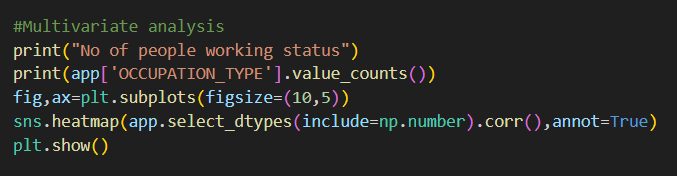
Analyzed the type of income sources such as "Working", "Pensioner", "Student", etc…

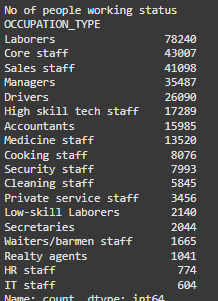
****

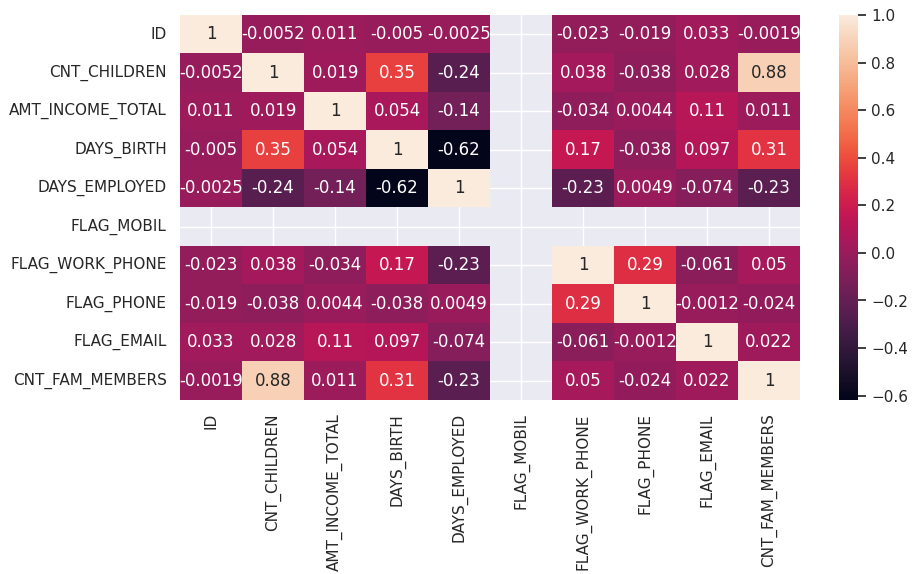
****

**MULTIVARIATE ANALYSIS:**

A correlation heatmap was created using numerical features to examine the relationships between different financial and demographic variables.

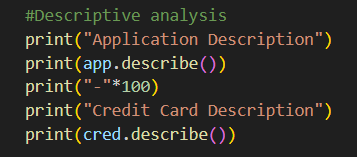
****

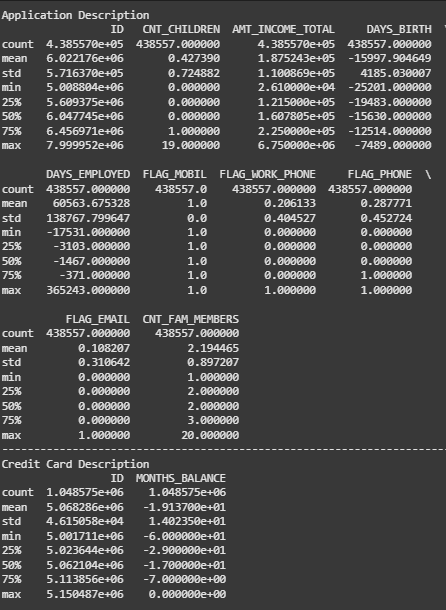
****

****

**DESCRIPTIVE ANALYSIS:**

Basic statistical summaries were generated for both datasets (app and cred) using the .describe() method.

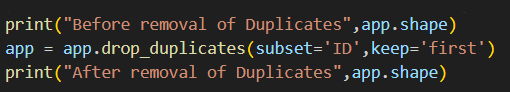
****

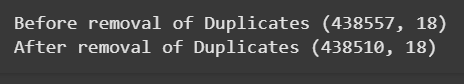
****

**DATA PREPARATION:**

**Drop Unwanted features:**

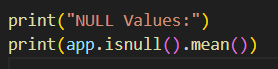
Checked and removed duplicate entries based on ID

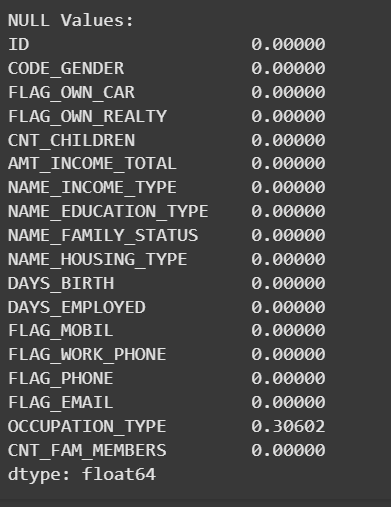
****

****

**Handling Missing Values:**

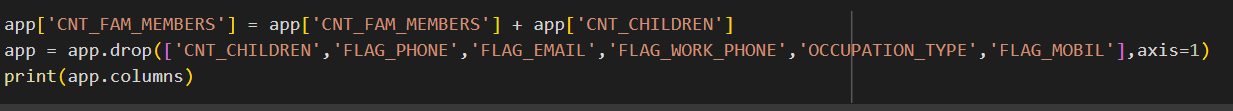
Displayed null value proportions using .isnull().mean()

****

****

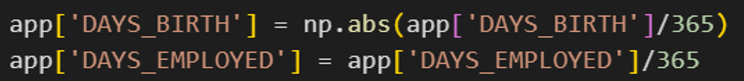
**DATA CLEANSING:**

* Combined children and family members count into one unified column CNT\_FAM\_MEMBERS.
* Dropped unnecessary or redundant columns like CNT\_CHILDREN, FLAG\_PHONE, etc.

****

Converted days-based columns to years:

* DAYS\_BIRTH: Converted to absolute age in years.
* DAYS\_EMPLOYED: Converted to years of employment.

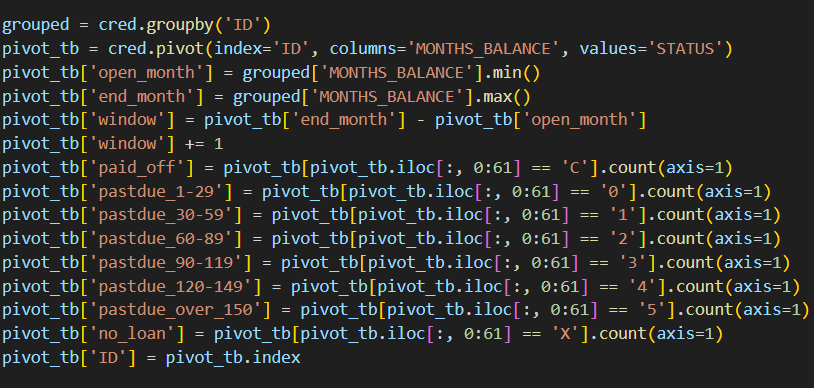
****

Mapped categorical variables to broader, simplified categories:

* Housing Type: Grouped all housing types under a common label "House/apartment" or "With parents".
* Income Type: Grouped into "Working", "Pensioner", and "Student".
* Education Type: Mapped into "secondary", "Higher education", and "Academic degree".
* Family Status: Mapped to "Single" or "Married".

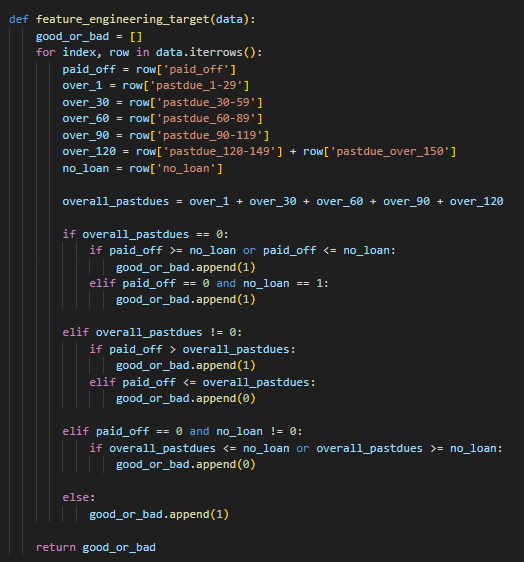
****

Converted STATUS values to meaningful metrics using a pivot table:

****

**FEATURE ENGINEERING:**

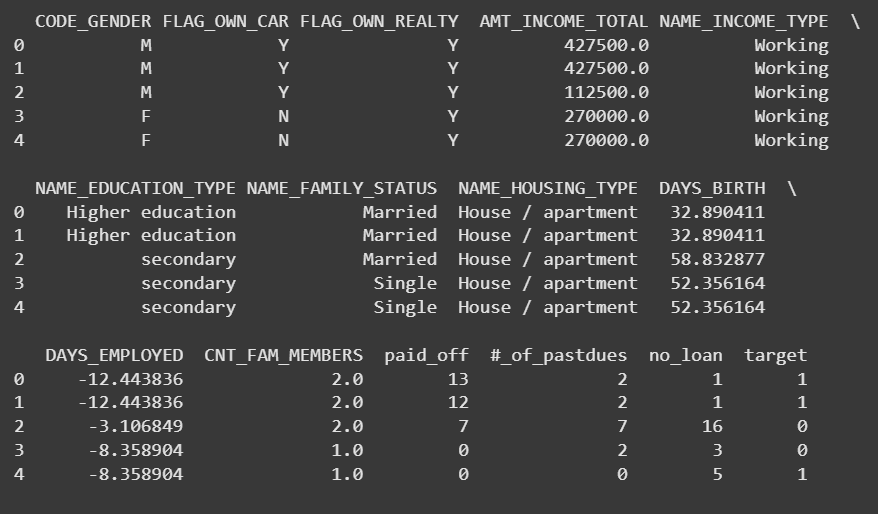
Engineered the final target column using a function feature\_engineering\_target():  
Logic-based on comparing paid-off months vs past-due months  
Labeled each ID as 1 (Good credit) or 0 (Bad credit)

****

**DATA MERGING:**

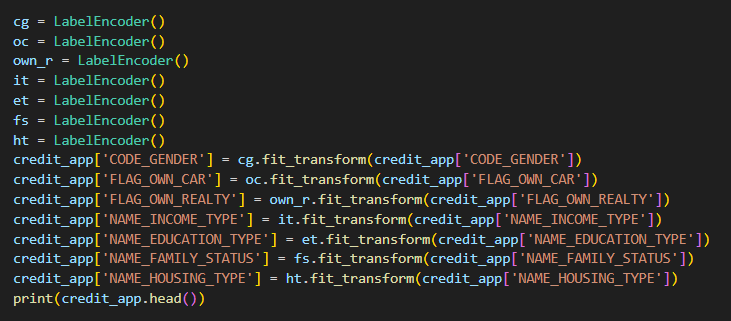
The target DataFrame was merged with the cleaned app dataset on ID and the ID column was dropped

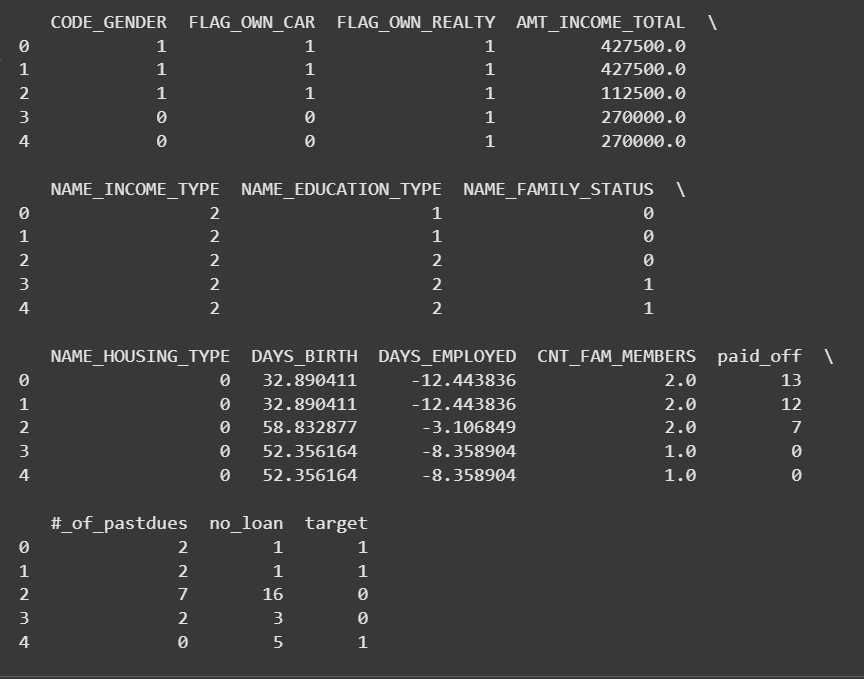
****

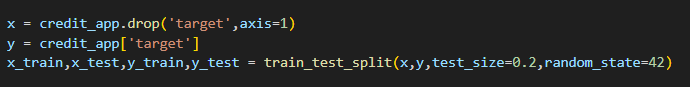
****

**HANDLING CATEGORICAL VALUES:**

Before training machine learning models, all categorical variables must be converted into a numerical format. To achieve this, Label Encoding was applied to all relevant categorical columns in the dataset using LabelEncoder from sklearn.preprocessing

****

****

**SPLITTING DATA INTO TRAIN AND TEST:**

**MODEL BUILDING:**

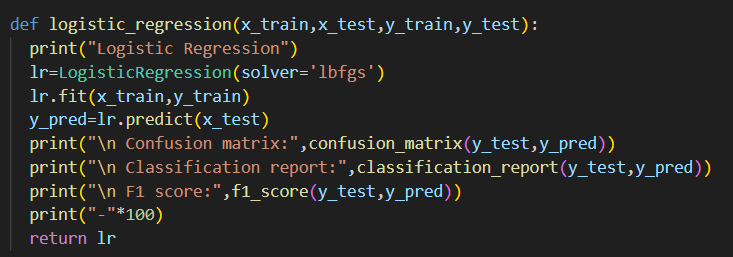
**Training the model in multiple algorithms**

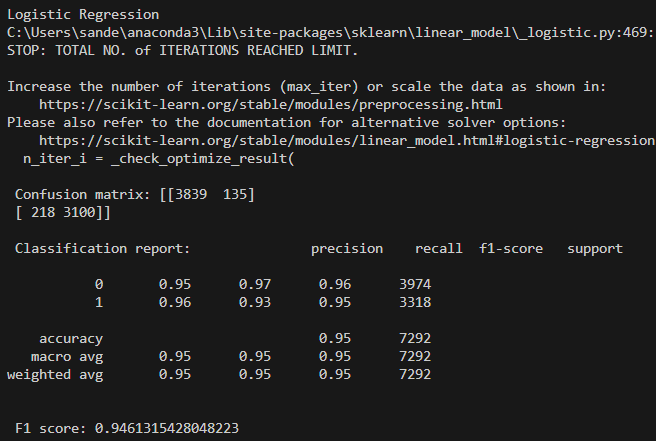
To determine the most effective classification algorithm for predicting credit card approval, five different machine learning models were trained and evaluated: Logistic Regression, Random Forest, XGBoost, Decision Tree. Each model was evaluated using standard metrics including confusion matrix, classification report, and F1 score. The best-performing model was saved for deployment.

**Logistic Regression**

A function named logistic\_regression was defined. It performs the following steps:

* Initializes the LogisticRegression() model using the 'lbfgs' solver.
* Trains the model on the training set using .fit().
* Predicts outcomes on the test set using .predict().
* Evaluates the results with confusion matrix, classification report, f1 score

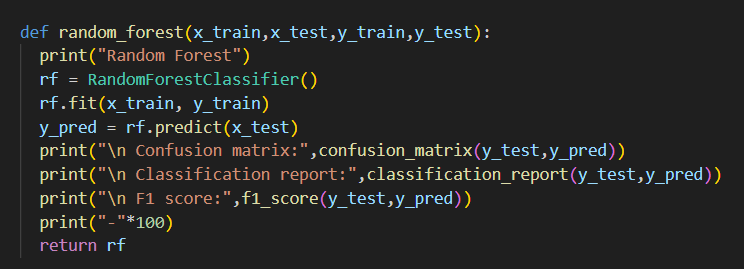


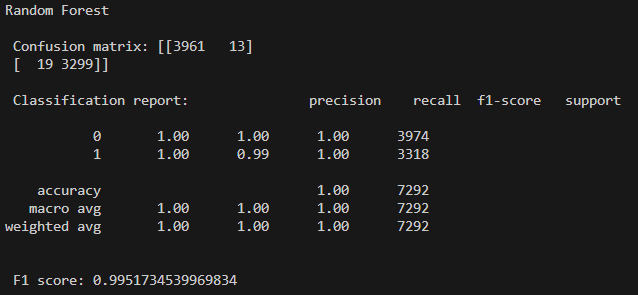


**Random Forest Classifier**

The random\_forest function:

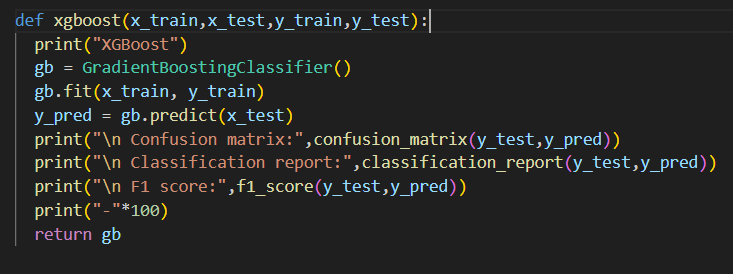
* Initializes a RandomForestClassifier() with default parameters.
* Trains the model on the training set.
* Predicts outcomes on the test set.
* Evaluates the results with confusion matrix, classification report, f1 score

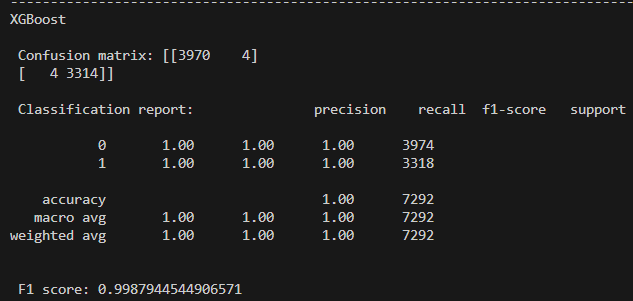




**XGBoost (Gradient Boosting)**

* The xgboost function uses GradientBoostingClassifier()
* Trains the model on the training data.
* Predicts test outcomes and evaluates the results using the same evaluation metrics.

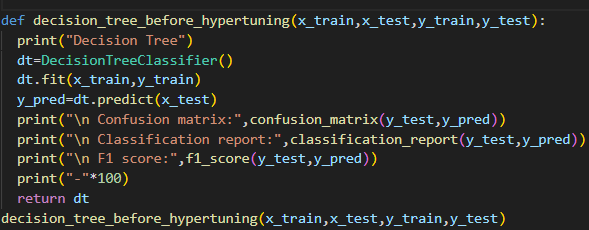


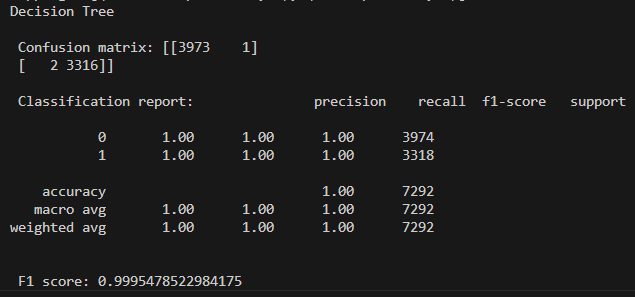


**Decision Tree Classifier Before Hypertuning:**

In the desicion\_tree\_before\_hypertuning function:

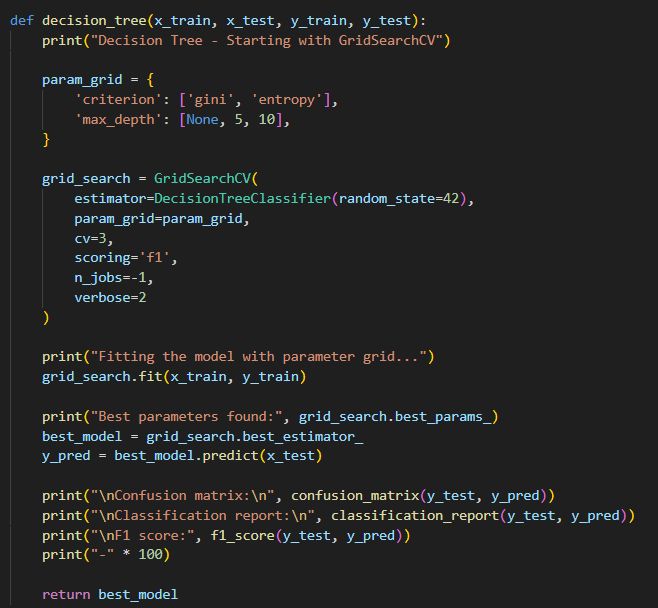
* A DecisionTreeClassifier() is initialized and trained.
* Predictions are made and evaluated using confusion matrix, classification report, and F1 score.

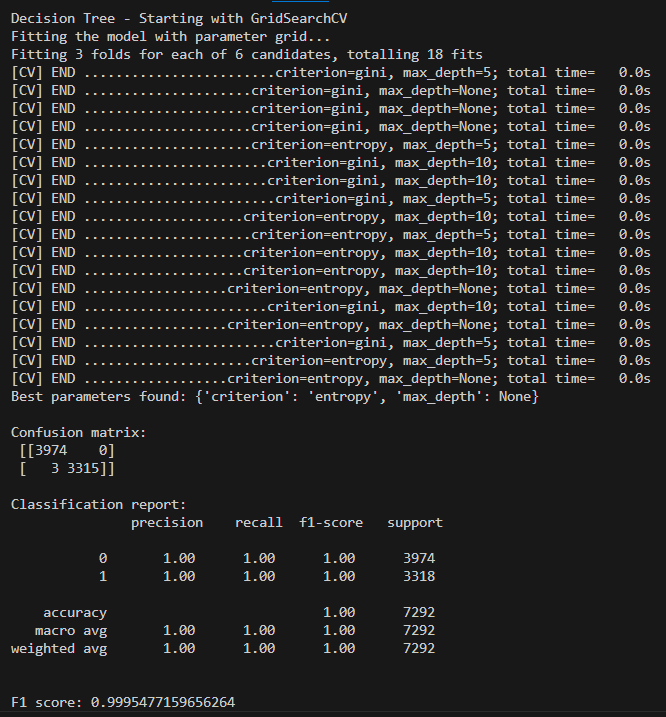
****

****

**Decision Tree Classifier After Hypertuning:**

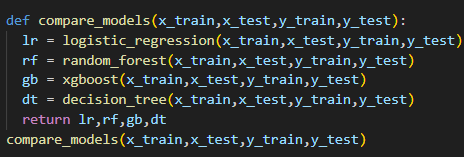
To improve the performance of the Decision Tree classifier, hyperparameter tuning was performed using GridSearchCV, a technique that systematically searches through specified parameter combinations to find the best configuration.

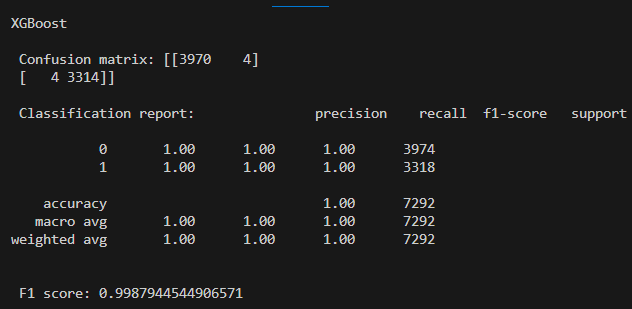
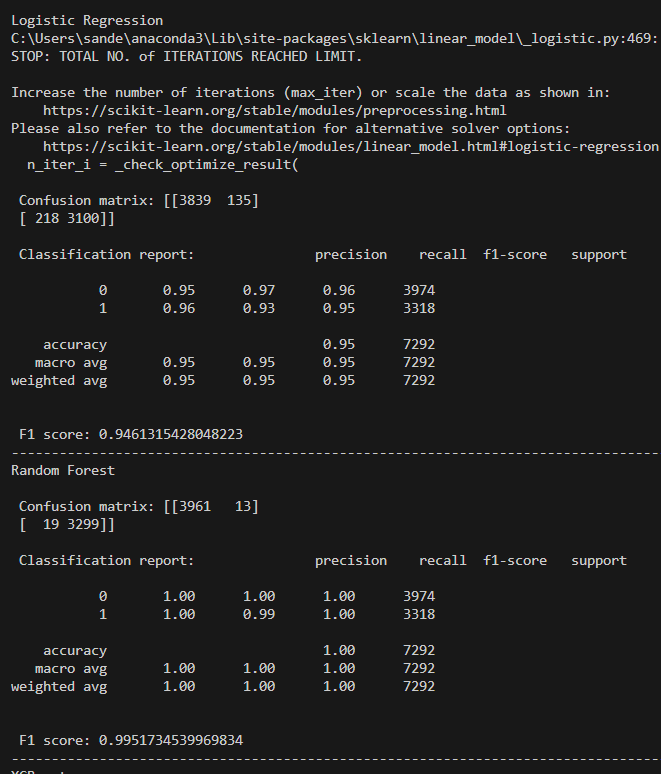


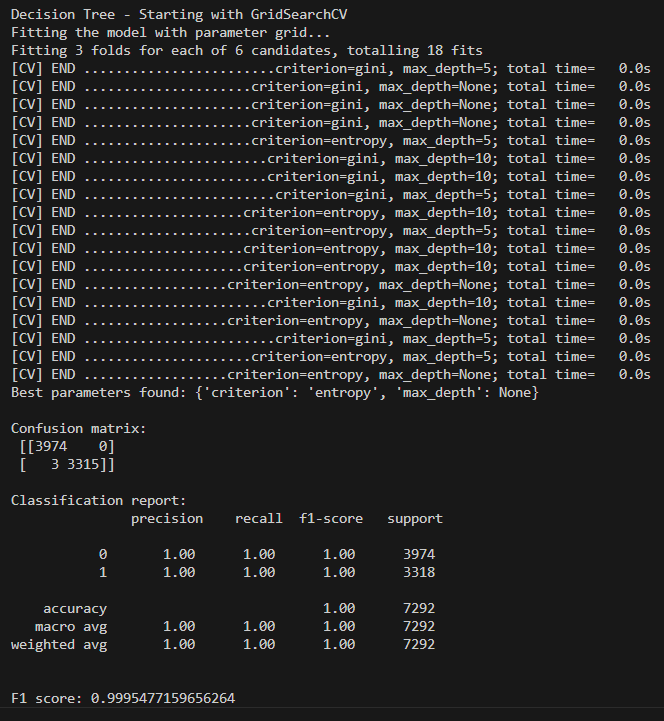


**COMPARE MODELS:**

* Calls each of the model functions (logistic\_regression, random\_forest, xgboost, desicion\_tree)
* Prints evaluation metrics (confusion matrix, classification report, F1 score) for comparison
* Returns all trained models for further use

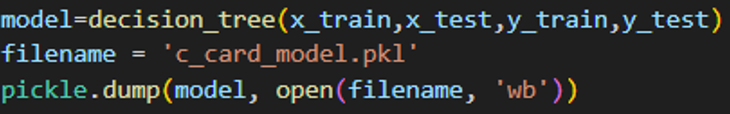
****

****

****

**Best model:** HypertunedDecision Tree with F1 score of 0.9995477159656264

**SAVE THE MODEL:**

****

**APPLICATION BUILDING:**

This section has the following tasks

* Building HTML Pages
* Building server-side script
* Run the web application

**BUILDING HTML PAGES:**

**index.html:**

This is the main landing page for a Credit Card Approval Prediction web application. It uses Bootstrap for styling and features a navigation bar, a section introducing the project, and a form with a "Prediction" button.

**index1.html**

This file contains a detailed and user-friendly Loan Default Prediction form. Built with Bootstrap, it collects comprehensive user information such as gender, car ownership, income, employment duration, and more. The form includes tooltips for guidance and is designed for submitting data to a loan default prediction model.

**error.html**

This is a simple error display template, used with a Python backend (Flask). It shows a specific message if the user left a required field blank or displays any other error message passed from the server. The template uses Jinja2-style templating for dynamic error handling.

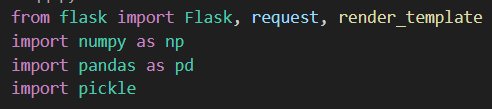
**Result.html**

This file displays the prediction result for the credit card approval process. It uses Bootstrap for styling and shows a centered message indicating whether the user is approved or not, with the result dynamically inserted via templating (e.g., {{prediction}}). It serves as the final feedback page after form submission and model evaluation.

**Build python code(app.py)**

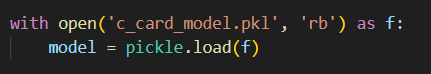
This Flask web application provides a user-friendly interface to predict whether a user is eligible for a credit card, based on input features. It loads a pre-trained machine learning model “c\_card\_model.pkl” and uses it to classify input data submitted via a web form.

**Import the libraries**

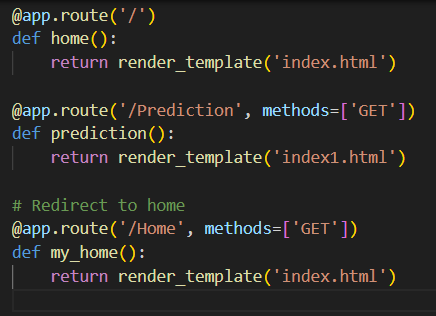
****

Flask: Web framework used to create the web server and handle routes.  
NumPy & Pandas: Used to process the input features and format them into a DataFrame.  
Pickle: Used to load the trained machine learning model saved in .pkl format.

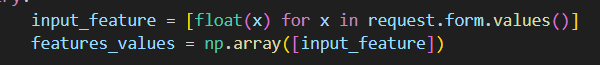
**Load the file:**

****

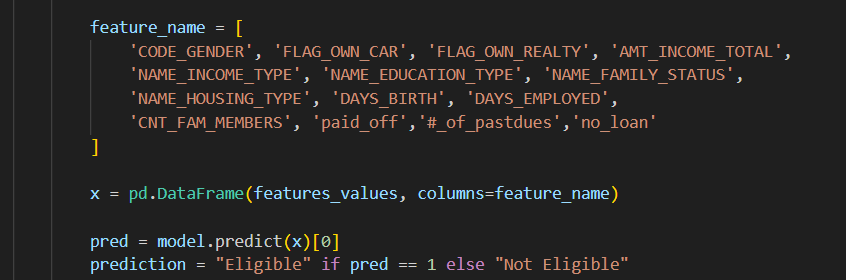
**Render HTML pages:**

****

**Getting the inputs from the form and storing it in a numpy array:**

****

**Converting the data into a dataframe:**

****

Converts the numeric output into a label:  
1 → "Eligible"  
0 → "Not Eligible"

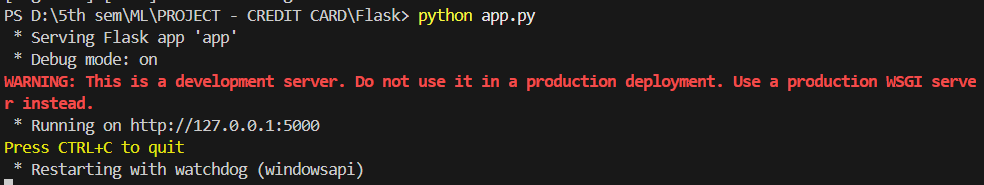
Renders the result.html page with the result.

If any error occurs (e.g., wrong input), it renders error.html with the error message

**Main Function:**

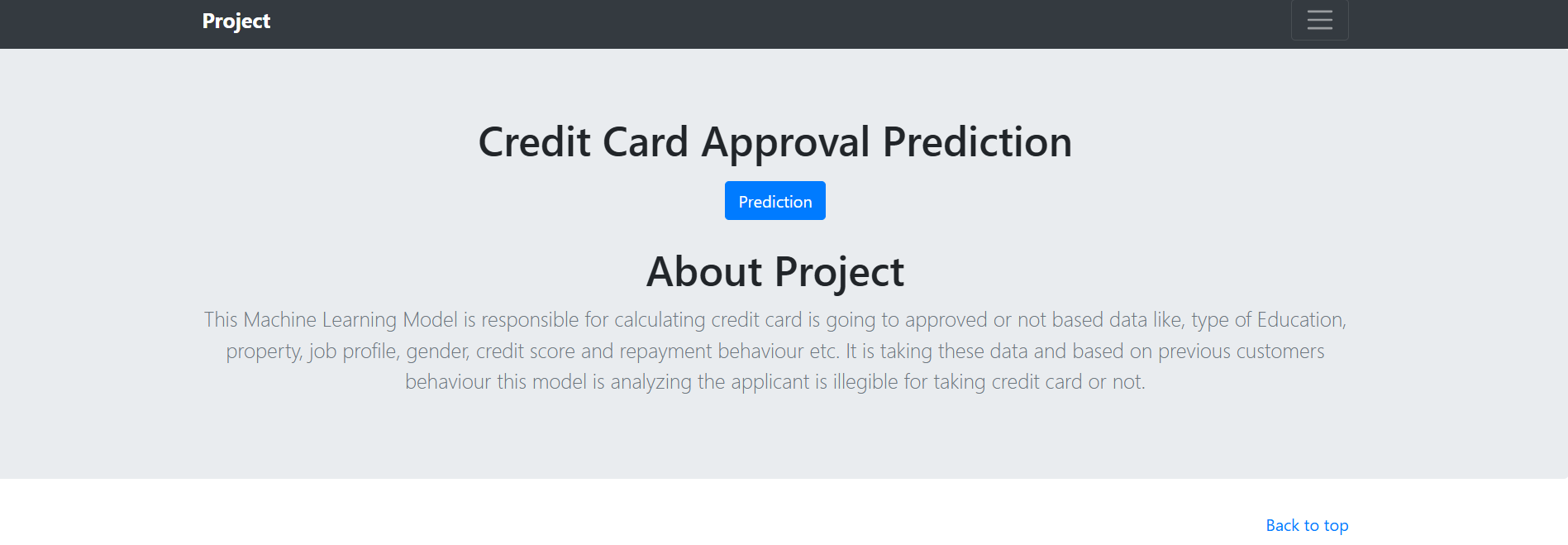
****

**Run the web application:**

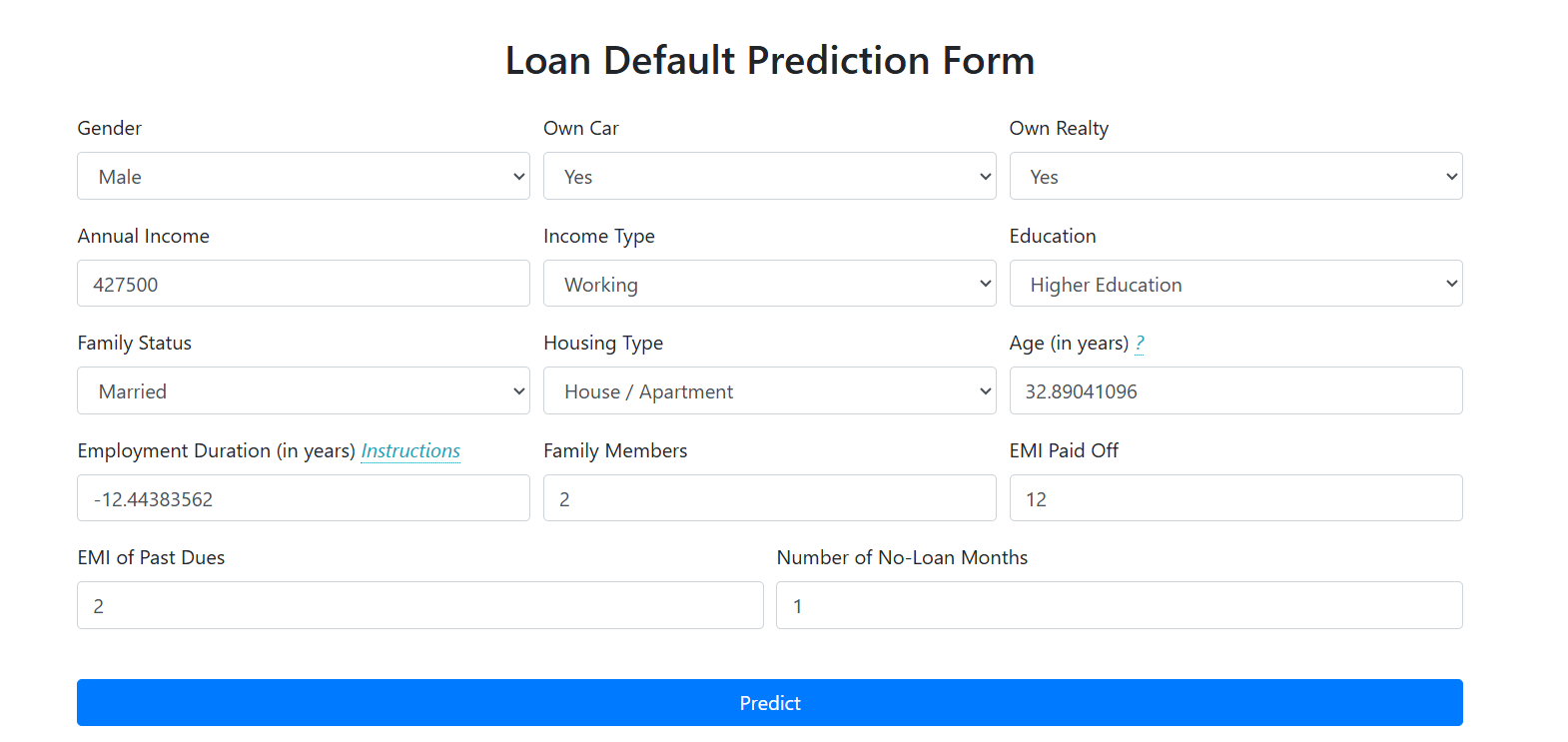
****

**USER INTERFACE:**

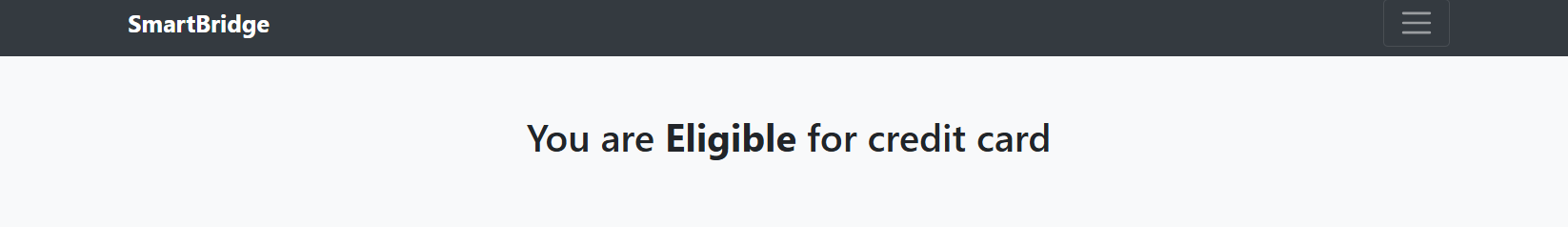
Navigate to localhost to view your web page.   
Click on the "Predict" button to enter user data and check whether the user is eligible or not for obtaining credit card

****

**Input #1:**

****

**Predicted Data:**

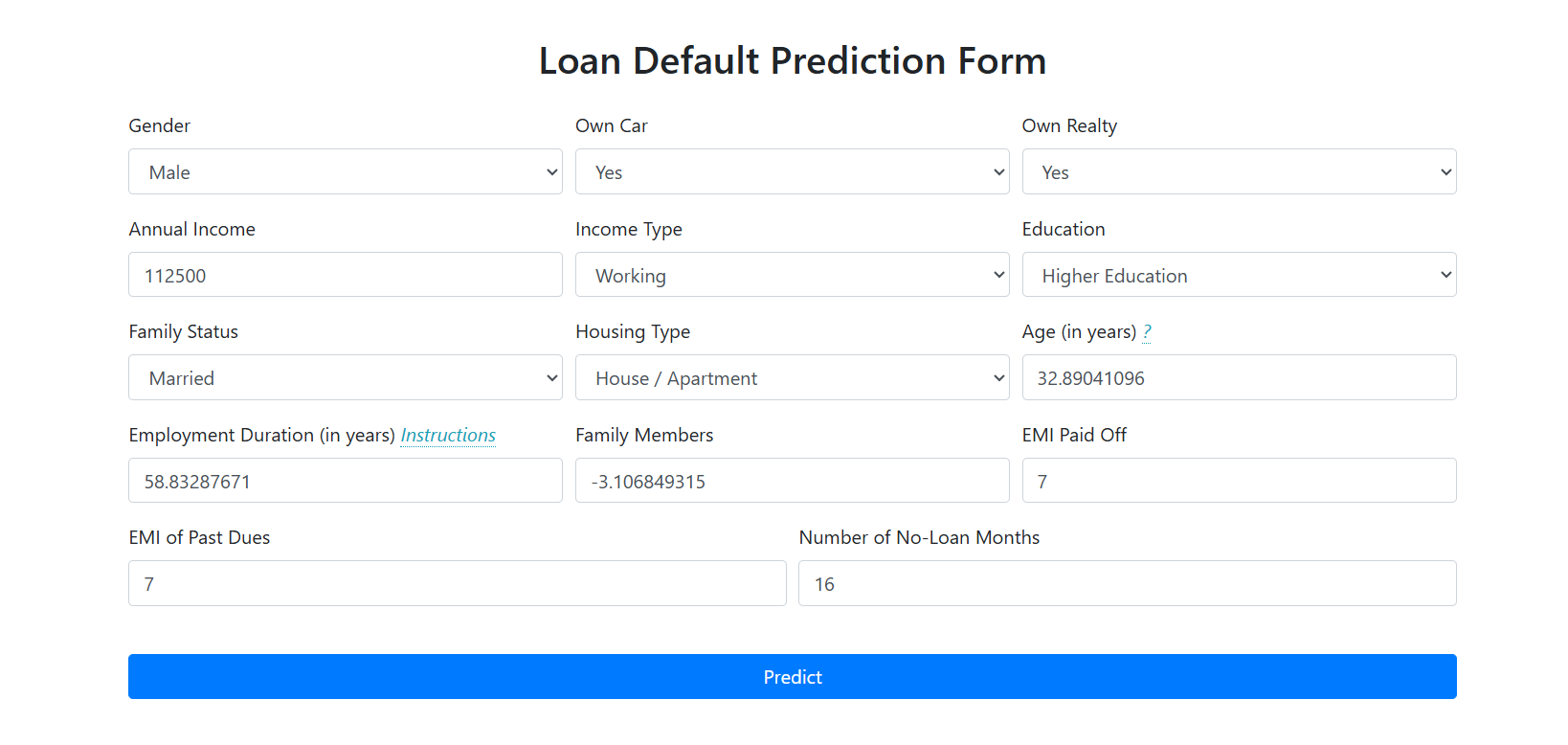
****

**Actual Data:**

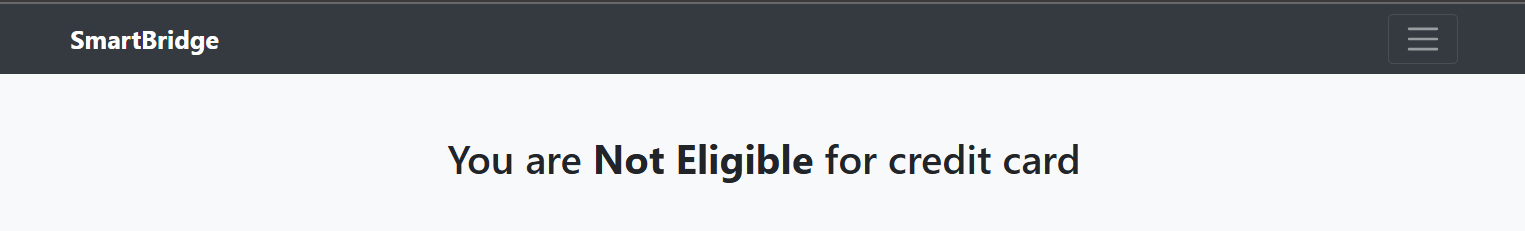
Target = 1 -> Eligible



**Input #2:**

****

**Predicted data:**

****

**Actual Data:**

Target = 0 -> Not eligible

****