**Credit Score Classification**

*A project report submitted to ICT Academy of Kerala*

*in partial fulfillment of the requirements*

*for the certification of*

**CERTIFIED SPECIALIST**

**IN**

**DATA SCIENCE & ANALYTICS**

submitted by

**Haneena V P**



**ICT ACADEMY OF KERALA**

**THIRUVANANTHAPURAM, KERALA, INDIA**

**Dec 2023**

# List of Figures

|  |  |  |
| --- | --- | --- |
| Figure 1 | Implementation Plan | 7 |
| Figure 2 | Exploratory Data Analysis | 11 |
| Figure 3 | Target Value Count Plot | 12 |
| Figure 4 | Target Value Count Pie Chart | 13 |
| Figure 5 | Data Splitting | 14 |
| Figure 6 | Balancing the data |  |
| Figure 7 | Model training | 15 |
| Figure 8 | Cross Validation | 17 |
| Figure 9 | Data Serialization | 18 |
| Figure 10 | Snapshots of Flask Web application Coding | 21 |
| Figure 11 | Loading necessary libraries and trained model | 21 |
| Figure12 | Snapshots of Flask application after deployment-Initial Page | 26 |
| Figure13 | Snapshots of Flask application after deployment-Final Page | 27 |

# List of Abbreviations

|  |  |  |
| --- | --- | --- |
| 1. | EDA | Exploratory Data Analysis |
| 2. | DF | Data frame |
| 3. | SVM | Support Vector Machine |
| 4. | RF | Random Forest Classifier |
| 5. | NB | Naive Bayes |
| 6. | SMOTE | Synthetic minority oversampling |

**Table of Contents**

|  |  |  |
| --- | --- | --- |
|  | Abstract | 5 |
| 1. | Problem Definition | 6 |
| 2. | Introduction | 7 |
| 3. | Literature Survey | 9 |
| 4. | Implementation Plan | 10 |
| 5. | Dataset Preparation | 13 |
| 6. | Model Training And Web App Deployment | 20 |
| 7. | Result | 27 |
| 8. | Conclusion | 28 |
|  | *Reference* | 29 |

# Abstract

Credit scoring is a way of analyzing statistical data used in financial organizations and banks to acquire a person’s creditworthiness. The bestowers generally manipulate it to decide to widen or retract credit. The score plays a significant role in determining the creditworthiness of a person and if he/she can be sanctioned a loan or not. Machine learning techniques help us to predict the credit score more accurately using classification algorithms. Few base and ensemble classification algorithms were used in this research to perform a comparative analysis. The ensemble method incorporates several base classification algorithms like Decision trees, Logistic Regression, Nearest neighbor, Support Vector Machine, etc. to achieve better results. The objective of this paper is to predict the credit score based on different classifier models and evaluate the performance of each model based on the metrics. A comparative analysis is done to identify the best classifier to predict the credit score. . Experimental results prove that the Random Forest classifier model produces better accuracy in ensemble classifiers.

.

# 1. Problem Definition

## 1.1 Overview

Banks and credit card companies calculate your credit score to determine your creditworthiness. It helps banks and credit card companies immediately to issue loans to customers with good creditworthiness. Today banks and credit card companies use Machine Learning algorithms to classify all the customers in their database based on their credit history. To address this situation the project aims to develop a fact-checking platform that utilizes a classification model to distinguish between good, Standard and poor creditworthiness. The platform will provide users with a reliable tool to predict the credit score of an individual based on their financial history and personal information.

## 1.2 Problem Statement

## Given a person’s credit-related information, build a machine learning model that can classify the credit score.The goal of this project is to build a model that can accurately predict the credit score of an individual based on their financial history and personal information. This project is a crucial component of my portfolio as it showcases my ability to work with real-world data, perform data cleaning and pre-processing, and apply machine learning algorithms to solve a practical problem. the process of building a credit score is through a classification model.

**2. Introduction.**

A credit score is a number based on investigating a person’s credit files, to represent the trustworthiness of an individual. The score computes the creditworthiness of borrowers based on their existing information. It is often used by banking, mobile manufacturers, insurance companies, landlords, government departments, and financial institutions such as online lenders. There are three credit scores that banks and credit card companies to label their customers good,standard,poor.A person with a good credit score will get loans from any bank or financial institution. Understanding credit scores and their impact on loan eligibility is essential for individuals seeking credit and financial institutions evaluating loan applications. Maintaining a good credit score through responsible financial management can significantly improve one's chances of securing favorable loan terms and achieving financial goals.

When applying for a loan, lenders typically request permission to check your credit report and use your credit score to assess your creditworthiness. Lenders use credit scores as one of the key factors to assess an individual's eligibility for loans, credit cards, mortgages, and other forms of credit.In this project we do credit score classification and predit the credit worthiness of the individual and whether they are eligible to get a loan.

# 3. Literature Survey

The credit risk scoring model provides a standardized and objective way for lenders to [assess the creditworthiness of individuals and businesses](https://www.highradius.com/resources/Blog/how-to-check-the-creditworthiness-of-a-new-customer/) .By using a credit scoring model, lenders can evaluate the risk of lending money. or extending credit to a borrower, allowing them to make informed decisions about loan terms and interest rates.. The literature survey for this project encompasses the following key areas:

## 3.1 Credit score Classification Datasets

The custom dataset created using various sources such as Mendeley, GitHub, and Kaggle provides a diverse collection of credit score, offering an opportunity to understand the creditworthiness. These datasets have been used by researchers and practitioners to develop and evaluate credit score classification models, providing valuable insights to banks and financial institution.

## 3.2 Exploratory Data Analysis (EDA) with Seaborn and Matplotlib

The utilization of Seaborn and Matplotlib for data visualization and exploratory data analysis has been well-documented in the literature. Studies have demonstrated the effectiveness of these libraries in visualizing textual data characteristics, word distributions, and class imbalances, providing essential insights for model development

## 3.3 Data Preprocessing with Python Libraries

Leveraging Python libraries such as Pandas and NumPy for data preprocessing has been a common practice in the literature. Researchers have utilized these libraries to handle missing values, perform text normalization, and address data quality issues, laying the groundwork for effective model training and evaluation.

## 3.4 Classification Models in Python

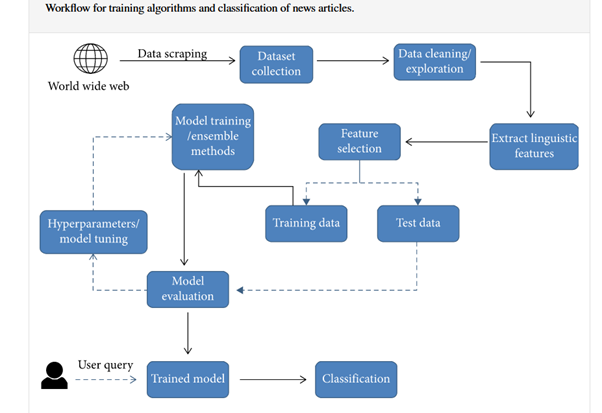
Research papers and practical guides have extensively discussed the implementation of classification models, including logistic regression, random forest classifier, and others, using Python-based machine learning libraries. These resources have highlighted the strengths and limitations of various algorithms for fake news classification tasks.

## 3.5 Python Flask Web Application Development

The development of web applications using Python Flask for deploying machine learning models has gained significant attention in the literature. Scholars and practitioners have outlined methodologies for integrating classification models into web hosting platforms, enabling the creation of user-friendly applications for fact-checking and misinformation detection.

By synthesizing the findings from these key areas, the project aims to leverage the insights from the literature and the practical application of the custom dataset to develop an effective fact-checking platform with a robust credit score classification model. The literature survey serves as a foundation for integrating best practices and insights from prior research into the project's methodology and approach.

# 4. Implementation Plan

****

As shown in the above diagram which indicate the workflow of our project and various intermediate stages. Initially we have to scrape data from various publicly available datasets which can meet our requirements.

Data cleaning is a crucial step in the implementation plan, as it directly impacts the quality and reliability of the subsequent analysis and model training. The purpose of Data cleaning is to get our data ready to analyze and visualize. During this phase, the collected data undergoes comprehensive preprocessing to address various issues such as missing values, outliers, and inconsistencies. Techniques such as imputation, outlier detection, and normalization are employed to ensure that the dataset is in a suitable form for further processing. Additionally, standardizing the data format and structure is essential to facilitate seamless integration with the model and to enable accurate feature extraction and selection. By conducting thorough data cleaning, we aim to enhance the overall robustness and integrity of the dataset, laying a solid foundation for subsequent stages of the implementation plan.

The process of extracting linguistic features involves leveraging available techniques to derive meaningful insights and patterns from textual data. This encompasses tasks such as tokenization, lemmatization, and the extraction of syntactic and semantic structures to capture the inherent linguistic characteristics within the dataset. Furthermore, feature selection plays a pivotal role in identifying the most relevant attributes that contribute to the predictive power of the model. By employing methods such as statistical tests, dimensionality reduction, and domain knowledge, we aim to isolate the most discriminative features while mitigating the impact of noise and redundant information. The synergy between linguistic feature extraction and feature selection is crucial in shaping the input space of the model, ensuring that it encapsulates the most pertinent linguistic properties while discarding superfluous attributes. This holistic approach not only facilitates the construction of a more refined and efficient model but also enhances its interpretability and generalization capabilities, ultimately contributing to the overall efficacy of the web application.

The division of the dataset into training and testing subsets is a fundamental aspect of the implementation plan. This process is pivotal in assessing the model's performance and generalization to unseen data. By allocating a portion of the data for training, the model can learn the underlying patterns and relationships within the dataset. The remaining portion, designated for testing, serves as an independent evaluation set to gauge the model's predictive accuracy and robustness. Striking a balance between the training and testing data is essential to prevent overfitting or underfitting, ensuring that the model achieves a harmonious equilibrium between learning from the training data and exhibiting proficiency on unseen instances. Through this meticulous partitioning, the model's ability to effectively capture and generalize patterns within the data can be systematically evaluated, laying the groundwork for the subsequent phases of model training and evaluation within the implementation plan.

The phase of model training constitutes a critical stage in the implementation plan, wherein the selected machine learning algorithm is exposed to the training data to learn the underlying patterns and relationships. Subsequently, the model's performance is rigorously evaluated using the designated test data, allowing for an assessment of its predictive accuracy and generalization capabilities. Concurrently, hyper-parameter tuning is undertaken to fine-tune the model's configuration, leveraging techniques such as grid search or randomized search to identify the optimal hyper-parameter values that enhance the model's performance. This iterative process of hyper-parameter optimization seeks to maximize the model's predictive prowess while mitigating the risk of overfitting. Upon achieving an optimized configuration, the final model is derived, encapsulating the refined parameters and hyper-parameter settings. This culmination represents the culmination of an intricate process, culminating in the deployment of a robust and finely-tuned model, poised to underpin the functionality of the web application with its adept predictive capabilities.

The development and deployment of the web application using Python Flask represent pivotal phases in the implementation plan. Leveraging the robust capabilities of Flask, a micro web framework, the development process involves the creation of a seamless and interactive interface to showcase the model's predictive functionality. Through Python's extensive libraries and Flask's intuitive structure, the web application is tailored to effortlessly interact with users, offering a user-friendly experience. Upon completion, the deployment phase ensures the accessibility of the web application to a wider audience, enabling users to leverage the predictive model's insights and functionalities. The deployment process may encompass considerations such as server configuration, scalability, and security measures to safeguard the application and its underlying model. By meticulously orchestrating the development and deployment of the web application, the implementation plan culminates in the realization of a dynamic and accessible platform, poised to deliver the model's predictive capabilities to its intended audience.

# 5. Dataset Preparation

The custom dataset selected for analysis, offering an opportunity to understand the credit worthiness of individual based on the information.

## 5.1 Data Preprocessing

For machine learning , it’s necessary to convert raw data into a clean data set, which means we must convert the data set to numeric data. We do this by encoding all the categorical labels to column vectors with binary values. Missing values, or NaNs (not a number) in the data set is handled by either dropping the missing rows or filling them up with a mean or interpolated values. The below mentioned preprocessing steps which we completed for our dataset for preprocessing

### 5.1.1 Drop Columns That Aren’t Useful

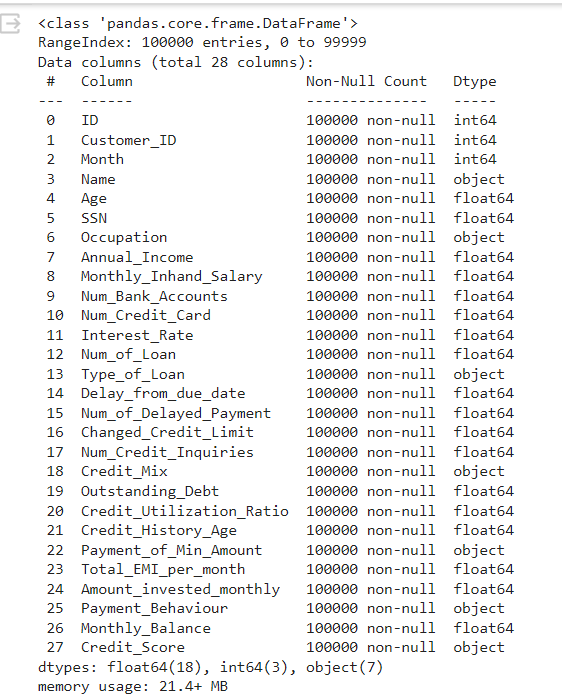
Drop some of the columns which won't contribute much to our machine learning model. Here we dropped the column ID,Name,SSN,Month,Type\_of\_Loan,Credit\_History\_Age.

### 5.1.2 Handling missing data

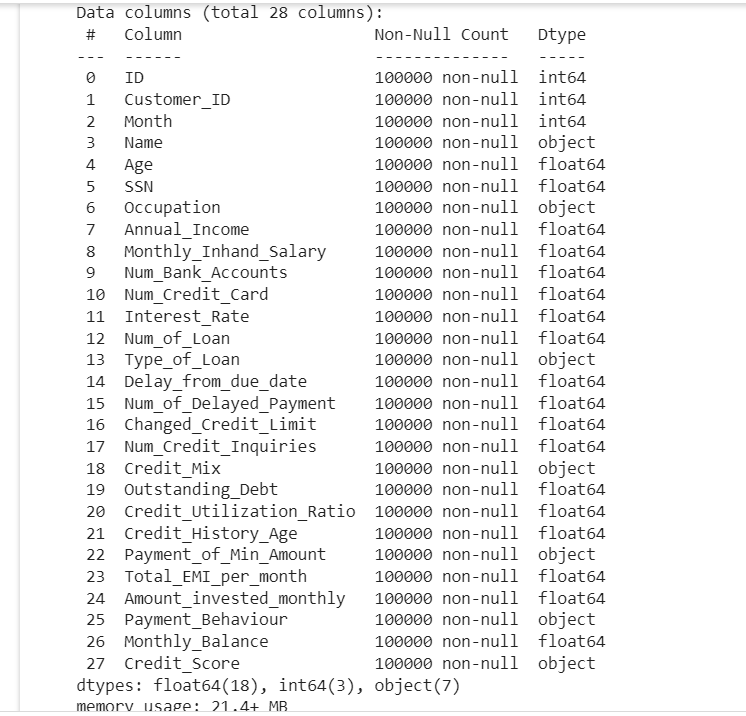
Dealing with missing data is a common and inherent issue in data collection, especially when working with large dataset**s.** If missing values have been found, there are particularly two ways to resolve this issue:

* Either Remove the entire row that contains a missing value. However, removing the entire row can generate a possibility of losing some important data. This approach is useful if the dataset is very large
* Or Estimate the value by taking the mean, median or mode.

Here in our dataset, we did not have any missing values.



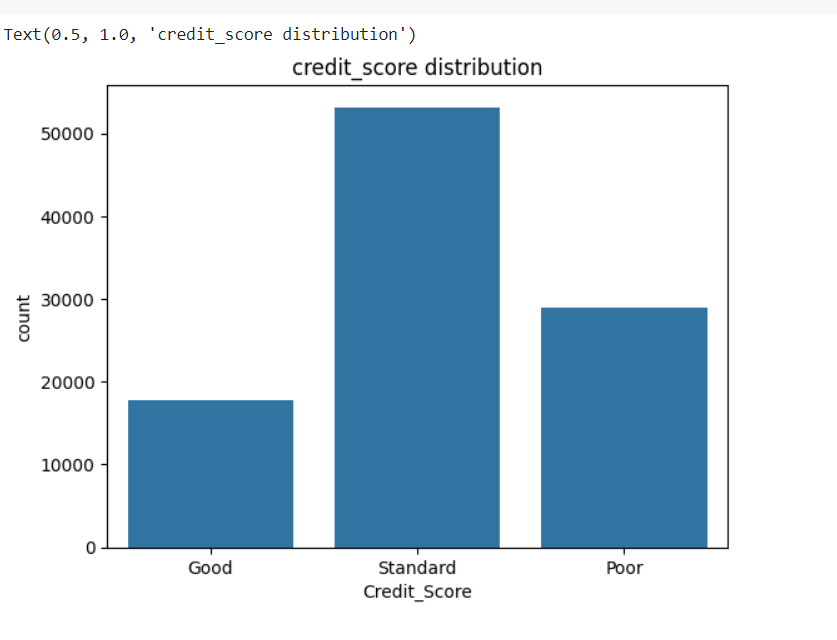
## 5.2 Exploratory Data Analysis

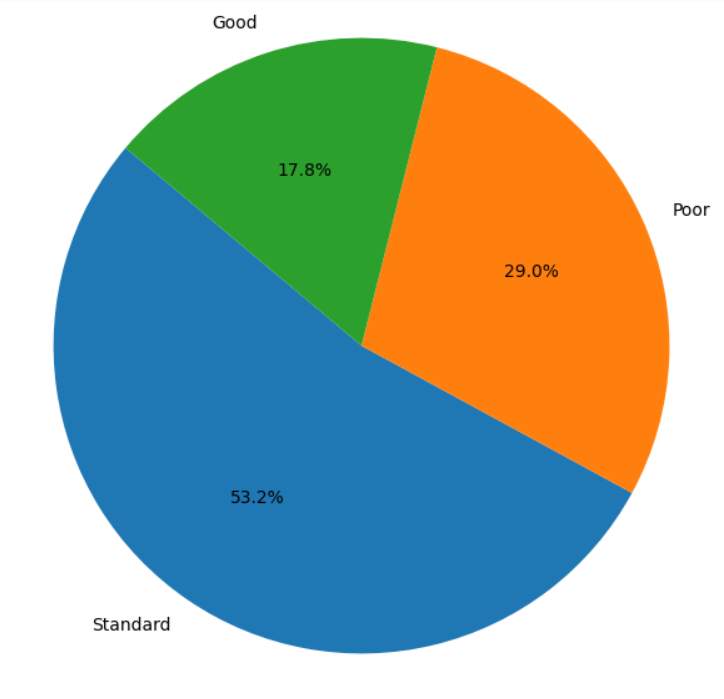
****

The dataset contains 28 columns including both feature sets and a target column, and the dataset contains more than100000 rows. The columns are :

'ID','Customer\_ID','Month','Name','Age','SSN','Occupation,'Annual\_Income','Monthly\_Inhand\_Salary'’, Num\_Bank\_Accounnt’,'Num\_Credit\_Card','Interest\_Rate', 'Num\_of\_Loan', 'Type\_of\_Loan', ‘Credit\_Mix’,'Delay\_from\_due\_date','Num\_of\_Delayed\_Payment','Changed\_Credit\_Limit',Num\_Credit\_Inquiries', 'Outstanding\_Debt','Credit\_Utilization\_Ratio','Credit\_History\_Age','Payment\_of\_Min\_Amount', 'Total\_EMI\_per\_month','Amount\_invested\_monthly','Payment\_Behaviour', 'Monthly\_Balance','Credit\_Score'.

**DISTRIBUTION OF CREDIT SCORE**

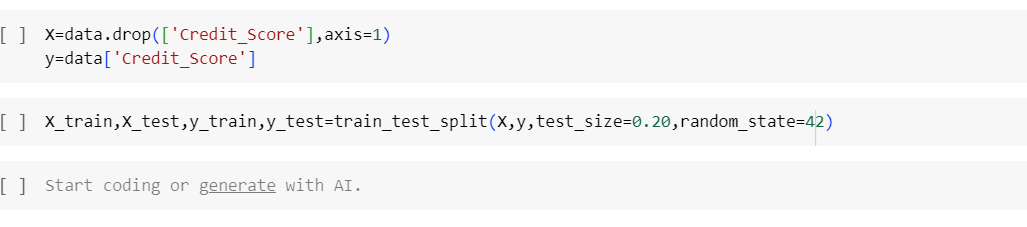




The target column of our dataset contains only three unique values, which are *good* and *standard and poor.* The plot shows the unique values in the target column plotted against their individual counts using seaborn’s countplot. And data values in the target columns are  *good,standard and poor* and their counts are 17828,53174 and 28998 respectively, and the percentage values are 17.8 ,29.and 53.2 respectively.

## 5.3 Data splitting

Here we split the dataset into train and test dataset.



## 5.4 Data Balancing

Balancing a dataset typically refers to adjusting the distribution of classes or categories within the dataset to ensure that each class is represented fairly equally. This is often important in machine learning tasks, especially classification problems, where an imbalanced dataset can lead to biased models that perform poorly on minority classes.

Here are some common techniques for balancing a dataset:

**Random Undersampling**: Randomly remove instances from the majority class(es) until the dataset is balanced. This can be effective but may lead to loss of information.

**Random Oversampling**: Randomly duplicate instances from the minority class(es) until the dataset is balanced. This can lead to overfitting, but it's a simple approach to try.

**Synthetic Minority Over-sampling Technique (SMOTE):** SMOTE works by generating synthetic examples in the minority class based on existing examples. It creates synthetic examples along the line segments joining k minority class nearest neighbors. This helps in creating a more diverse and balanced dataset.

**Adaptive Synthetic Sampling (ADASYN):** ADASYN is an extension of SMOTE that generates more synthetic examples in regions of the feature space where the density of minority examples is low.

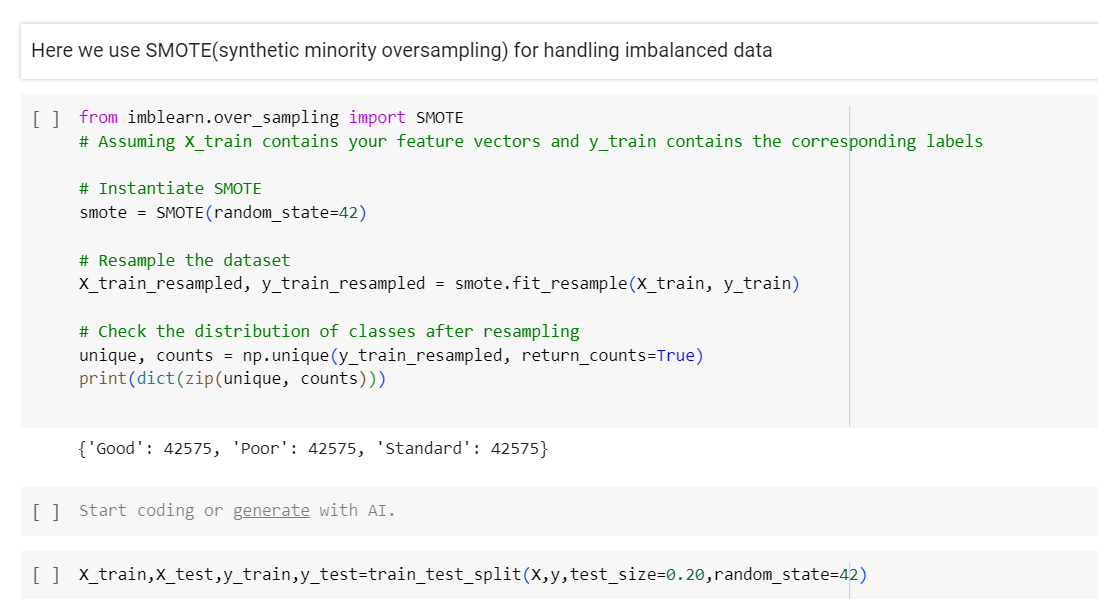
**Class Weighting**: In some algorithms, such as in many classifiers in scikit-learn, you can assign different weights to different classes. This penalizes misclassifications of the minority class more heavily, effectively giving it more importance during training.

**Data Augmentation:** For certain types of data like images or text, you can artificially increase the size of the minority class by applying transformations such as rotation, flipping, cropping, or adding noise.

**Ensemble Methods**: Utilize ensemble methods like bagging or boosting with resampling techniques, which can help in creating balanced subsets during each iteration of the ensemble.

**Collect More Data**: If possible, gather more data for the minority class(es) to balance out the dataset naturally.

The choice of method depends on the specific dataset, the problem at hand, and the algorithms being used. It's often a good idea to experiment with different techniques to see which one works best for your particular situation. Additionally, it's crucial to evaluate the performance of your model on a separate test set to ensure that the balancing technique you choose does not introduce biases or adversely affect generalization.

Here we use SMOTE for balancing the data. 

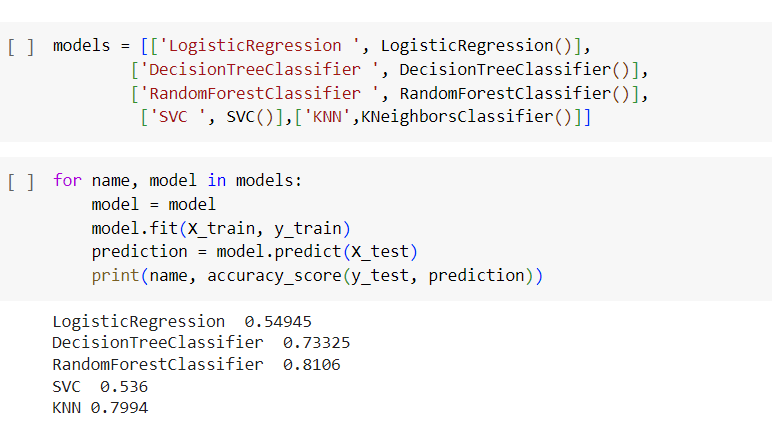
# 6. Model Training And Deployment

After the completion of dataset cleaning and preprocessing, then next immediate to follow model training and getting a good accuracy score. And a successfully evaluated dataset can be used in the web application development using data serialization, which follows deploying the web application in a suitable platform.

## 6.1 Model Training

A machine learning training model is a process in which a machine learning (ML) algorithm is fed with sufficient training data to learn from. Here in our dataset we performed various machine learning algorithms which include SVM(Support Vector Machine), KNN(K Nearest Neighbor) classifier, Decision tree and Random Forest Classifiers.

Below shows training the accuracy code comparison for SVM, KNN, DT and RF classifiers.



We have got an accuracy score for SVM as 53.6% , for Rf as 81.06% , for KNN it's 79.9% and for DT its 73.3%. Based on the score , we decided to go with the hyperparameter tuning.

## 6.2 Hyperparameter Tuning

Hyperparameter tuning is a critical aspect of machine learning model development. Hyperparameters are configuration settings that govern the learning process of a machine learning algorithm. Unlike model parameters, which are learned from the data during training (such as weights in a neural network), hyperparameters are set prior to training and control aspects like model complexity, learning rate, regularization strength, etc.

Hyperparameter tuning involves the process of finding the optimal set of hyperparameters that result in the best performance of the machine learning model on a validation dataset. The goal is to maximize the model's performance metrics, such as accuracy, precision, recall, or F1 score, depending on the specific problem being addressed.

There are several techniques for hyperparameter tuning, including:

**Manual tuning**: This involves manually selecting hyperparameters based on domain knowledge and intuition. However, this approach can be time-consuming and may not always yield the best results.

**Grid search**: Grid search involves defining a grid of hyperparameter values and evaluating the model's performance for each combination of values. This method exhaustively searches through all possible combinations, making it computationally expensive but thorough.

**Random search**: Random search involves randomly sampling hyperparameter values from predefined ranges and evaluating the model's performance for each sampled combination. While it may not be as exhaustive as grid search, random search is more computationally efficient and often yields good results.

**Bayesian optimization**: Bayesian optimization is an iterative approach that uses probabilistic models to search for the optimal set of hyperparameters. It balances exploration and exploitation to efficiently navigate the hyperparameter space and find promising regions.

**Automated machine learning (AutoML):** AutoML platforms automate the process of hyperparameter tuning along with other steps in the machine learning pipeline, such as feature engineering and model selection. These platforms leverage various optimization techniques to find the best hyperparameters automatically.

The choice of hyperparameter tuning technique depends on factors such as the size of the dataset, computational resources available, and the complexity of the model. It's important to strike a balance between computational cost and the quality of the tuned model when selecting a hyperparameter tuning method.Here in this project we use Grid search and Random search for hypertuning.

## 6.3 Cross Validation

Cross-validation is a resampling technique used to assess the performance and generalization ability of a machine learning model. It is particularly useful when working with limited data or when aiming to evaluate how well a model will perform on unseen data.

In cross-validation, the dataset is split into multiple subsets or folds. The model is trained on a subset of the data (training set) and then evaluated on the remaining subset (validation set). This process is repeated multiple times, with each fold serving as the validation set exactly once. The performance metrics (e.g., accuracy, precision, recall) are averaged over all the folds to obtain a more reliable estimate of the model's performance.

The most common type of cross-validation is k-fold cross-validation, where the dataset is divided into k equal-sized folds. The model is trained and evaluated k times, each time using a different fold as the validation set and the remaining k-1 folds as the training set.

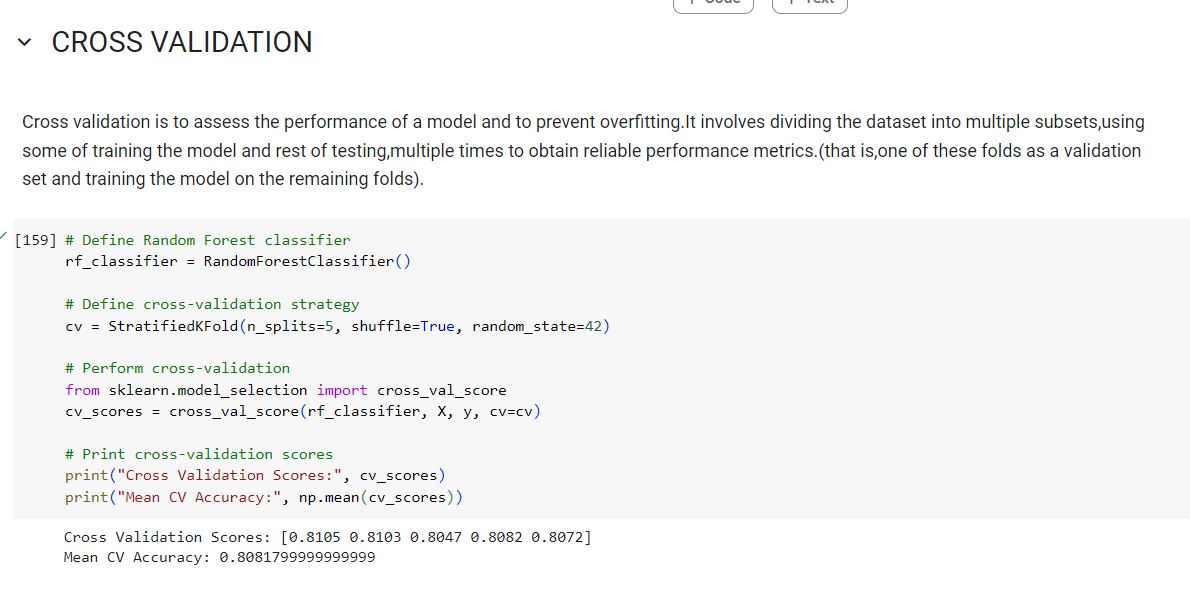
Other variants of cross-validation include:

**Leave-One-Out Cross-Validation (LOOCV):** Each data point is used as the validation set once, while the rest of the data is used for training. LOOCV is computationally expensive for large datasets but provides an unbiased estimate of model performance.

**Stratified Cross-**Validation: Ensures that each fold has a similar distribution of target classes as the original dataset. This is particularly useful for imbalanced datasets where one class may dominate the data.

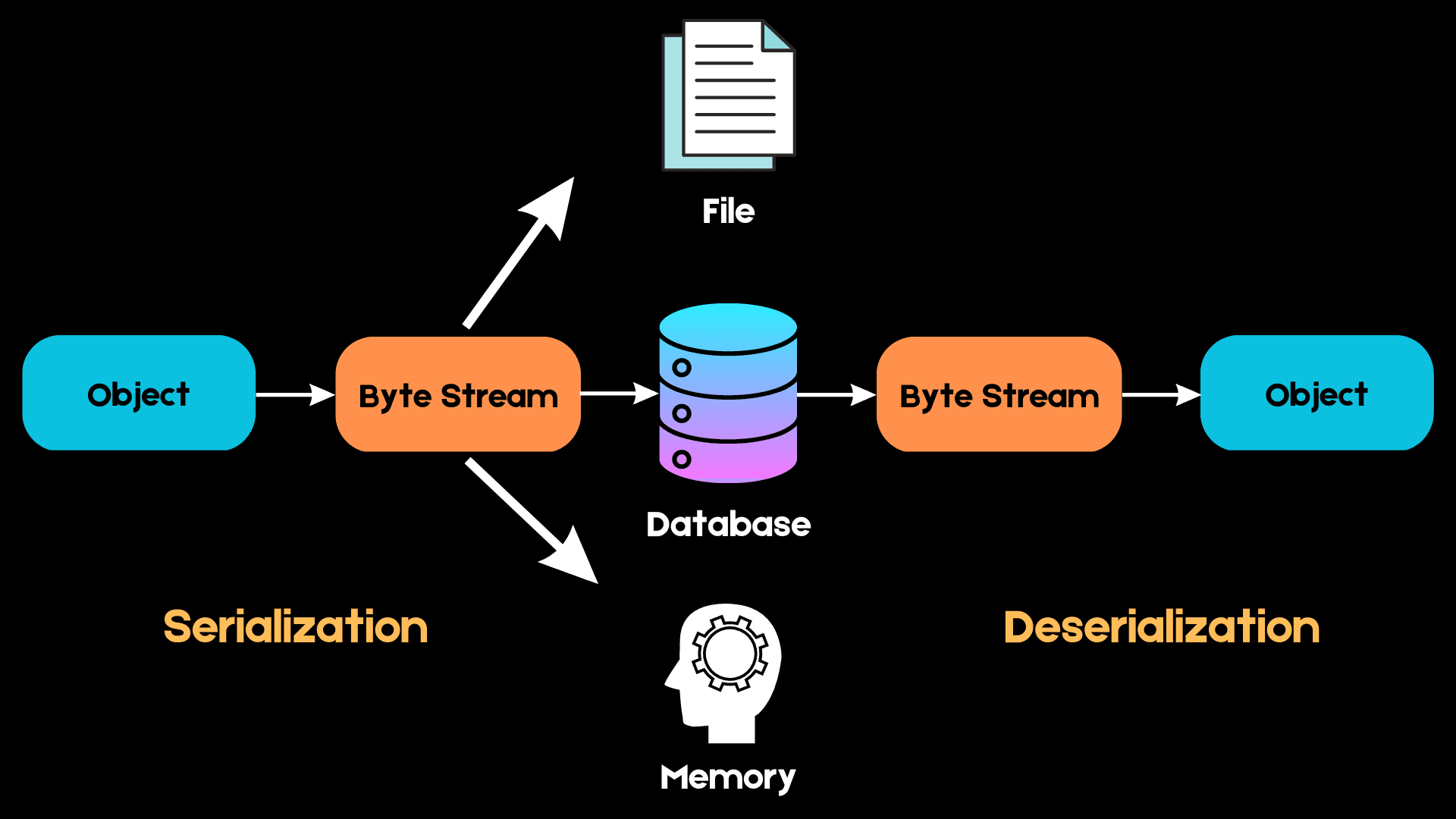
**Repeated Cross**-Validation: Involves repeating the cross-validation process multiple times with different random splits of the data. This helps to obtain more reliable performance estimates, especially with smaller datasets.

Cross-validation helps to assess how well a model generalizes to new, unseen data and provides insights into its robustness and stability. It is commonly used for hyperparameter tuning, model selection, and comparing different machine learning algorithms. By averaging performance over multiple folds, cross-validation provides a more accurate estimate of a model's performance compared to a single train-test split.Here we use Stratified KFold cross validation.



## 6.4 Data Serialization using pickle

Serialization is essential in order to use a trained model in a web application, which directly converts a python coded model into a byte stream file that is more convenient using different programmes or platforms. Here is simple diagram showing how serialization works:



The next time we want to access the same data structure, this sequence of bytes must be converted back into the high-level object in a process known as **deserialization**.

We can use formats such as JSON, XML, HDF5, and Pickle for serialization. In this project we are using a pickle library for the serialization process. The reason to choose pickle library is that is native to the python environment and also one of the most capable serialization algorithms available out there.

Here we use the *dump* method from the pickle library by referring to the trained model and save as a pickle file with an ‘*.pkl’* extension.



### 6.4.1 Advantages of using Pickle to serialize objects

* Unlike serialization formats like JSON, which cannot handle tuples and datetime objects, Pickle can serialize almost every commonly used built-in Python data type. It also retains the exact state of the object which JSON cannot do.
* Pickle is also a good choice when storing recursive structures since it only writes an object once.
* Pickle allows for flexibility when deserializing objects. We can easily save different variables into a Pickle file and load them back in a different Python session, recovering your data exactly the way it was without having to edit your code.

### 6.4.2 Disadvantages of using Pickle

* Pickle is unsafe because it can execute malicious Python callables to construct objects. When deserializing an object, Pickle cannot tell the difference between a malicious callable and a non-malicious one. Due to this, users can end up executing arbitrary code during deserialization.
* Pickle is a Python-specific module, and we may struggle to deserialize pickled objects when using a different language.
* According to multiple benchmarks, Pickle appears to be slower and produces larger serialized values than formats such as JSON and ApacheThrift.

## 6.5 Web Page Designing

The web page design of our Flask web app deployment is tailored to provide a seamless and intuitive user experience. The design elements and features are aimed at facilitating easy interaction and information presentation. Below is an overview of the key components of our web page design.

The content section of our web page allows users to input their informations into their corresponding colums. Upon submission, the system processes the input and provides relevant information,whether the person is eligible for loan or not and also provide their creditscore. Our web page includes easy-to-use navigation elements to enhance the user experience. Users can submit their informations, clear the input area, and again provide the necessary informations using the provided buttons .

We have incorporated visual elements such as the company logo and icons to improve the aesthetics and overall appeal of the web page. The use of icons for buttons and file input enhances the interactivity and visual appeal.

Upon submission of informations, credit worthiness is displayed in a visually appealing manner. The design of our web page aims to create a user-friendly environment where users can seamlessly interact with the application and obtain meaningful insights from their input.

This structured description provides an overview of the web page design and its key elements, allowing readers to understand the approach taken to create an engaging and functional user interface for the Flask web app deployment.

In our Flask web app deployment, the web page design is complemented by custom CSS styling to enhance the visual appeal and user interaction.

## 6.6 Flask Web Application

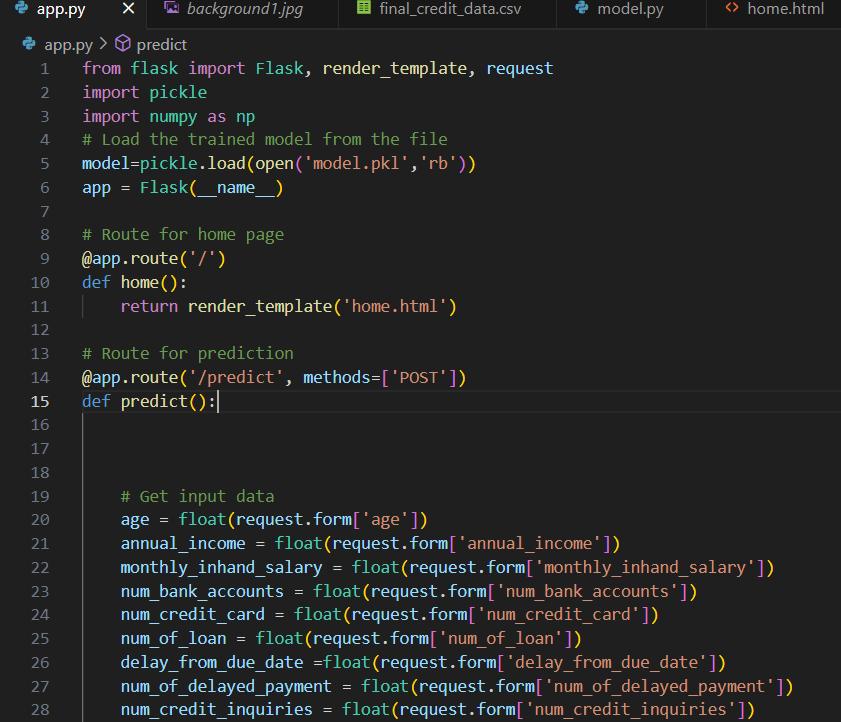
In the project, a Flask application has been developed to know the credit worthiness and eligibility of individuals using a pre-trained model that was mentioned earlier. The following steps outline the creation and functionality of the Flask application:

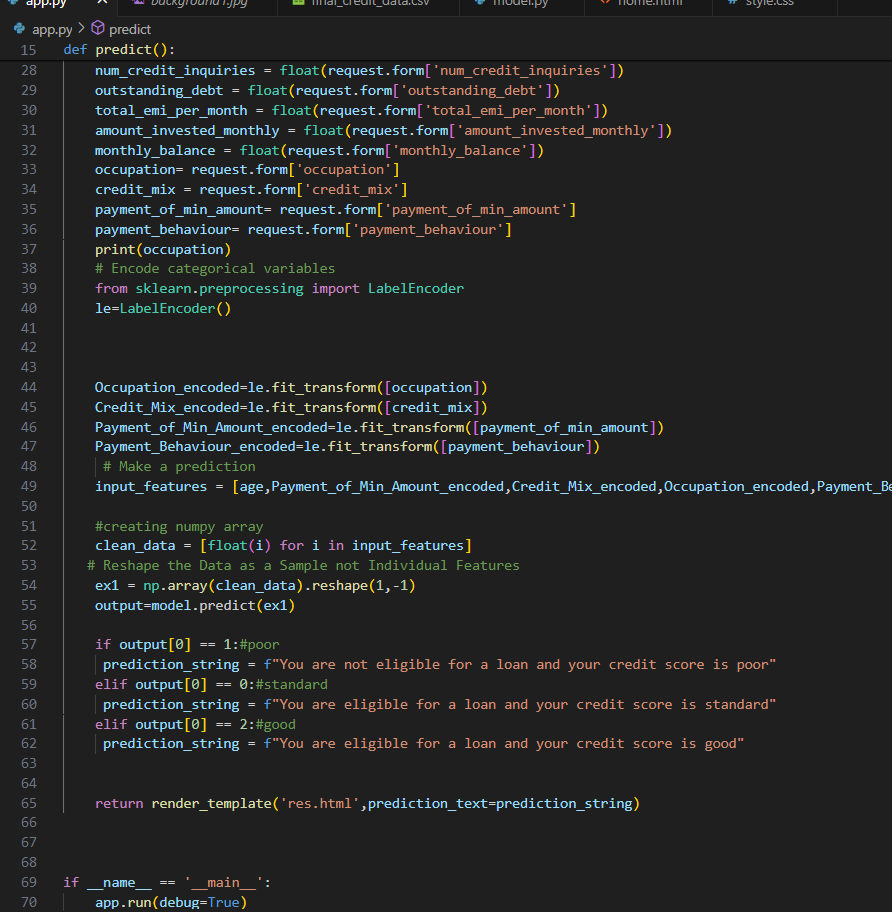
* **Importing Libraries and Modules:** The necessary libraries including Flask, request, render\_template, pickleare imported at the beginning of the script.
* **Initializing the Flask App:** The Flask application is initiated using Flask(name), creating an instance of the Flask class.
* **Loading the Classifier:** The pre-trained classifier are loaded using pickle.load to deserialize the saved model files.
* **Defining the Routes and Views:** The ‘/’ route is defined to handle both GET and POST requests. On POST request, the colums to update informations is processed, and classified using the loaded classifier. The results along with relevant statistics are passed to the ‘*home.html’* template for display.
* **Running the Flask App:** The main block contains app.run(port=8000) to run the Flask app on port 8000.

By incorporating Flask in the project, the team has developed a robust web application that leverages a pre-trained model for credit score classification, providing users with real-time insights and analysis. On the local host it is giving the expected result

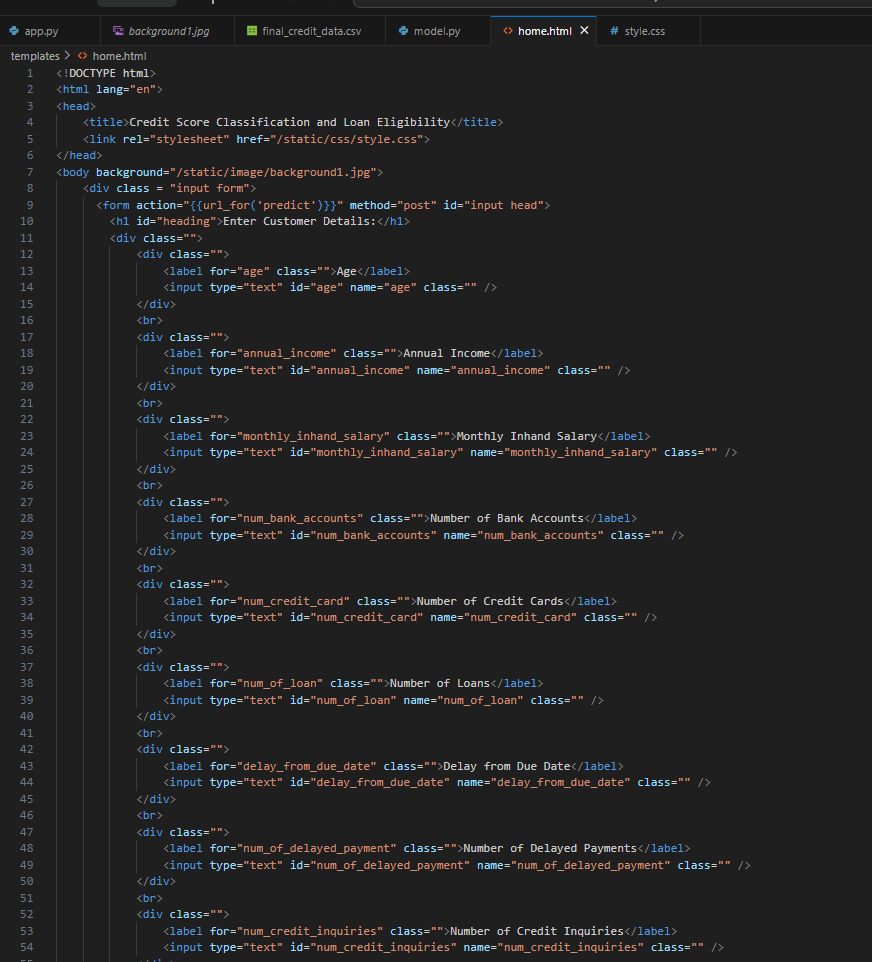
**6.6.1 Snapshots of Flask Web application Coding :**

**Loading necessary libraries and trained model**

****

****

**home.html web page code (main page)**

****

****

## 6.6 Deploying the Flask Web Application

In order to deploy our application we need a website hosting service provider. First priority was the service which should be free since it is a project model. And we choose to host in *pythonanywhere.com*, which is lightweight and easy to configure apart from providing limited free service.

Within the PythonAnywhere dashboard, a new web app was created, and the Flask web framework was selected as the preferred framework for our application. Additionally, the appropriate Python version was specified to align with our project requirements.

The web app was configured by setting up a virtual environment and specifying the project path, ensuring that all necessary dependencies and project files were appropriately organized within the PythonAnywhere environment.

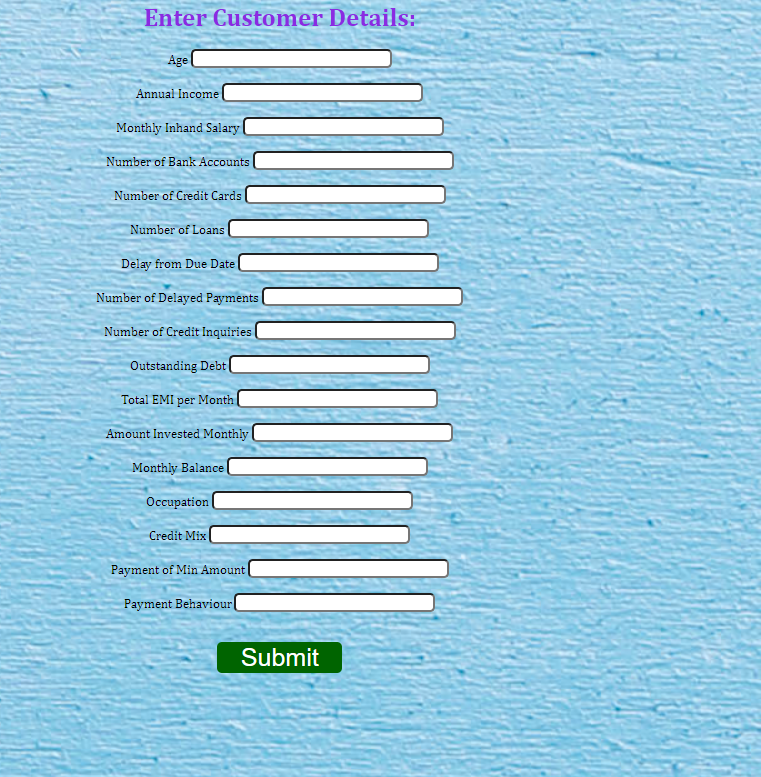
Critical Flask application files, including the Python script, HTML templates, and other essential components, were uploaded to the PythonAnywhere account. This step ensured that the entire application, along with its resources, was seamlessly integrated into the hosting environment.

Following the configuration and file upload process, the web app was initiated from the PythonAnywhere dashboard, marking the commencement of our Flask application's availability over the internet.

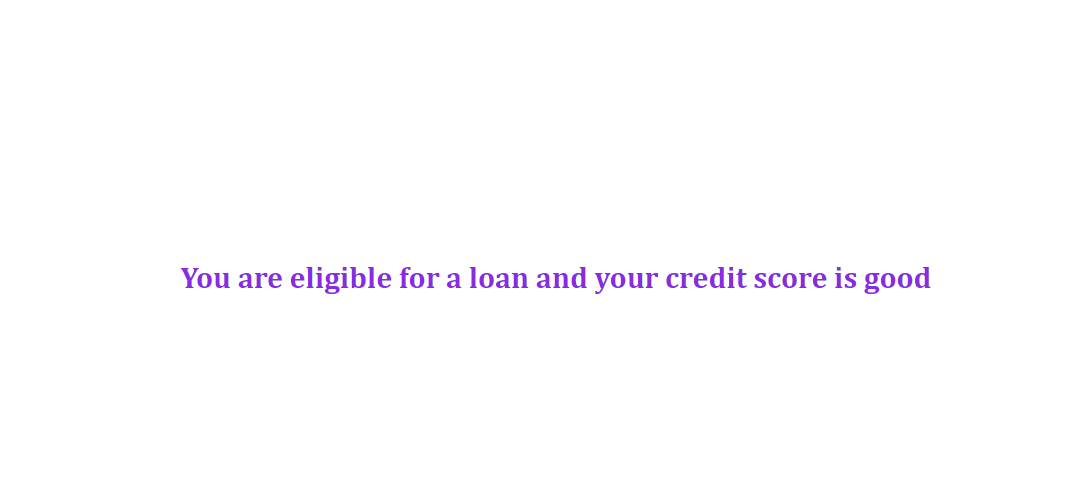
**Web Url Links to our python web application after deployment**

Link: <https://loaneligibility.pythonanywhere.com/>

**6.6.1 Snapshots of Flask application after deployment :**

****

**Initial page**

****

**Final page after submitting necessary details.**

# 7. Result

Upon successful deployment, the hosted Flask application became accessible via a unique domain provided by PythonAnywhere. Users could interact with the application through a standard web browser, enabling seamless access to credit score classification and to know the credit worthiness.

The successful hosting of our Flask application on PythonAnywhere has extended the reach of our credit score classification system, making it readily available to bank or financial institution across the internet. This hosting solution aligns with our project's objective of ensuring widespread accessibility and usability.

# 8. Conclusion

Our project focused on the development and deployment of credit score classification system using machine learning and web technologies. Through the collaborative efforts ,i successfully achieved the following key outcomes:

1. System Development:

We designed and implemented a robust credit score classification system and loan eligibility that leverages machine learning techniques to analyze the persons credit worthiness and their loan eligibility. The system's intuitive user interface allows seamless interaction, making it accessible to a wide audience.

1. Technology Integration:

By integrating machine learning models with a Flask web application, we created a cohesive platform that empowers banks or finanacial institution to submit provided details about the individual for classification, receive real-time results, and contribute to a dynamic dataset for continuous model improvement.

1. Deployment on PythonAnywhere:

The hosting of our Flask application on PythonAnywhere further extended the reach of our credit score classification system, ensuring its availability and accessibility to banks or finanacial institution across the internet. This deployment marked a significant milestone in making our system widely available.

1. User Engagement and Impact:

Throughout the development and deployment phases, we prioritized user engagement and usability, aiming to provide a valuable tool for individuals seeking efficient credit score categorization. The successful deployment and accessibility of our system align with our vision of creating a practical and impactful solution.

1. Future Prospects:

As we conclude this phase of the project, we recognize the potential for future enhancements and iterations. This includes expanding the dataset, refining the machine learning models, and incorporating user feedback to further improve the accuracy and relevance of the classification system.

In conclusion, our project has realized its primary objectives of developing, deploying, and making accessible a credit score classification system that harnesses the power of machine learning and web technologies. The successful collaboration, technical innovation, and commitment to user-centric design have culminated in a valuable solution with the potential for continued evolution and positive impact.

# References

**Dataset: credit score dataset provided.**

**Python Libraries**

1. <https://www.numpy.org>
2. <https://www.pandas.pydata.org/>
3. <https://www.matplotlib.org/stable/>
4. <https://www.matplotlib.org/stable/tutorials/pyplot.html>
5. <https://www.seaborn.pydata.org/>
6. <https://www.flask.palletsprojects.com/en/3.0.x/>