A logo with blue letters

Description automatically generated

**SCHOOL OF COMPUTER SCIENCE ENGINEERING AND INFORMATION SYSTEMS**

**Course code: PMCA507L**

**Course Title: Machine Learning**

**LABORATORY PROJECT**

**SLOT: A2**

**TITLE: CREDIT RATING PREDICTION**

**TEAM MEMBERS**:

|  |  |
| --- | --- |
| **SHUBHAM SINHA** | **23MCA0111** |
| **ANURAG SINHA** | **23MCA0079** |
| **DARSHAN BALAR** | **23MCA0106** |

**FACULTY INFORMTION:**

**Dr. ARUN PANDIAN J**

* **Assistant Professor Sr. Grade 1**

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **Chapter No** | **Title** | **Page No** |
| **1.** | **Introduction** | 3 |
| **2.** | **Problem statement** | 4 |
| **3.** | **Problem statement** | 5 |
| **4.** | **Data Exploration** | 6 - 11 |
| **5.** | **Modeling and Result** | 12 |
| **6.** | **Conclusion and Future Development** | 13 |

**INTRODUCTION**

The Credit Risk Prediction System is a data-driven solution designed to assist financial institutions in evaluating the creditworthiness of loan applicants. With the increasing complexity of financial markets and the growing demand for loans, accurately assessing credit risk is crucial for mitigating financial losses and maintaining a healthy lending portfolio. Traditional methods of credit assessment often rely on manual review processes, which can be time-consuming and prone to errors. In response to these challenges, the Credit Risk Prediction System leverages machine learning techniques to automate the credit evaluation process and provide timely, data-driven insights.

By analysing historical data of loan applicants and their corresponding credit outcomes, the system learns patterns and relationships that influence credit risk. Through the implementation of logistic regression, a widely used classification algorithm, the system predicts whether an applicant falls into the category of 'good' or 'bad' credit risk based on their financial and personal attributes. This predictive model is trained on a diverse dataset encompassing factors such as credit history, income, employment status, and more.

The Credit Risk Prediction System is deployed as a user-friendly web application, allowing financial institutions to seamlessly integrate it into their existing loan approval workflows. Through a simple and intuitive interface, users can input their details and receive instant predictions regarding their credit classification. This empowers lenders to make informed decisions regarding loan approvals, streamline their processes, and minimize the risk of default.

In this project, we explore the intersection of machine learning and finance, demonstrating the potential of predictive analytics in enhancing credit risk assessment. By harnessing the power of data-driven insights, the Credit Risk Prediction System aims to optimize lending practices, improve financial outcomes, and foster responsible lending practices in the ever-evolving landscape of finance.

**PROBLEM STATEMENT**

Classify the customer's credit rating (good or bad) based on their personal and bank account details.

* Dataset: credit\_rating.xlsx
* Target variable: Credit classification
* Constrain: Use any one of the following classification techniques: Logistic Regression,
* Decision Tree, Naïve Bayes, Support Vector Machine, and Multi-Layer Perceptron
* Performance Measure: Plot the confusion matrix, accuracy, and F1 score of the
* classification technique on test data.
* Deploy the model as a mobile or web application. Create a form for collect the user
* inputs.

**RESULT & MODEL**

We have developed a web-based credit card rating model that predicts the creditworthiness of applicants based on their submitted information. This project aims to provide a user-friendly interface for users to input their data, receive a credit rating prediction, and view the result conveniently.

**Technologies Used :**

* Backend Technologies:
* **Python:** We utilized Python as the primary programming language for backend development due to its simplicity, flexibility, and extensive libraries for data processing and web development.
* **Flask:** We chose Flask as the web framework for its lightweight nature, simplicity, and ease of integration with other libraries and frameworks. Flask allowed us to quickly develop RESTful APIs and serve web pages with minimal boilerplate code.
* **Django:** Django, a high-level Python web framework, was used for certain components of the project, particularly for user authentication, administration panel, and database management. Django's batteries-included approach provided us with built-in solutions for common web development tasks, such as URL routing, form handling, and security features. Additionally, Django templates were utilized to render HTML files, allowing us to create dynamic web pages with reusable components.
* **Scikit-learn:** We used Scikit-learn, a powerful machine learning library in Python, for building and training the credit card rating model. Scikit-learn provides a wide range of tools for data preprocessing, model selection, and evaluation.
* **Joblib:** Joblib was used for serializing and deserializing Python objects, particularly for saving and loading trained machine learning models.
* **Pandas**: Pandas, a popular data manipulation library in Python, was used for data handling and preprocessing tasks such as loading datasets, cleaning data, and feature engineering.
* Frontend Technologies:
* **HTML:** We used HTML for structuring the content of web pages, including forms, tables, and other elements.
* **SCSS:** SCSS (Sassy CSS) was chosen for styling the web pages due to its enhanced features and flexibility compared to traditional CSS.

**How to Run :**

**Github Link For Code :** [**https://github.com/darshan-balar2400/CREDITY**](https://github.com/darshan-balar2400/CREDITY)

**Step 1 : install virtualenv**

pip install virtualenv

**Step 2 : Create Python virtual environment**

virtualenv venv

**Step 3 : Activate virtual environment**

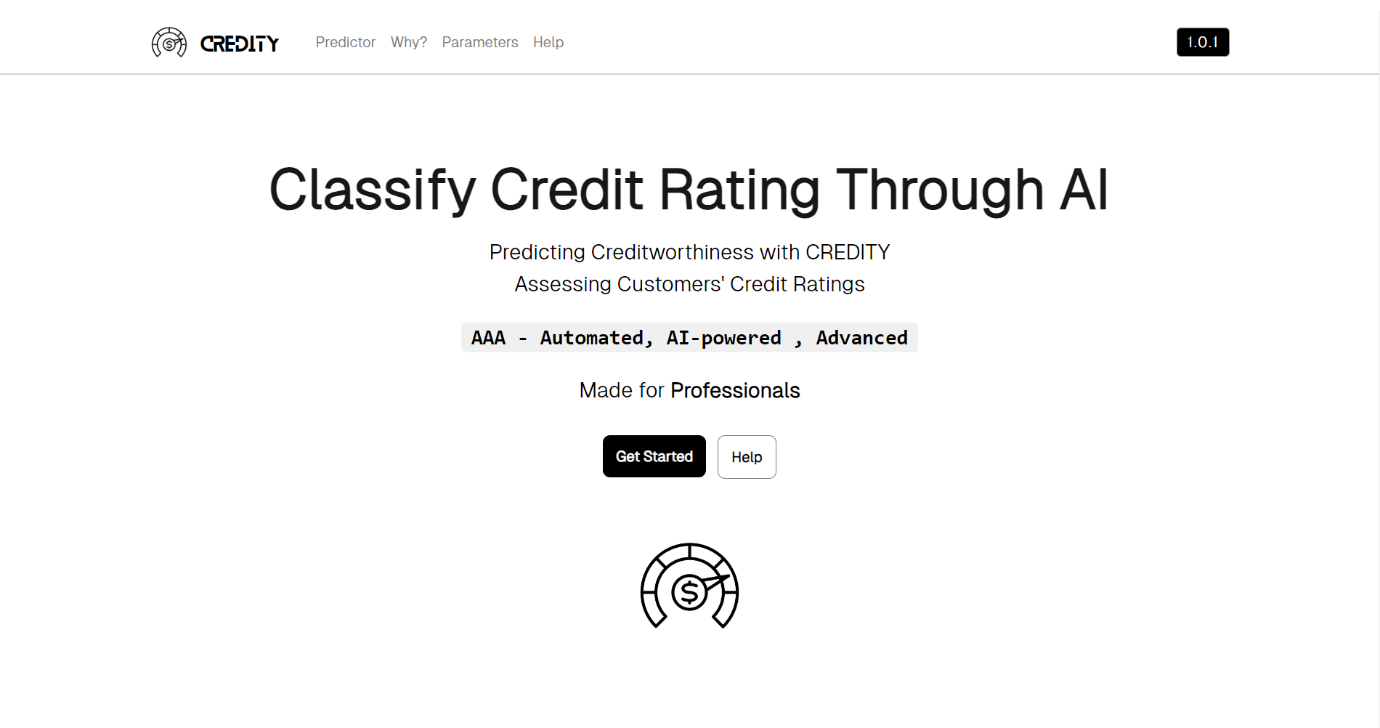
windows > venv\Scripts\activate

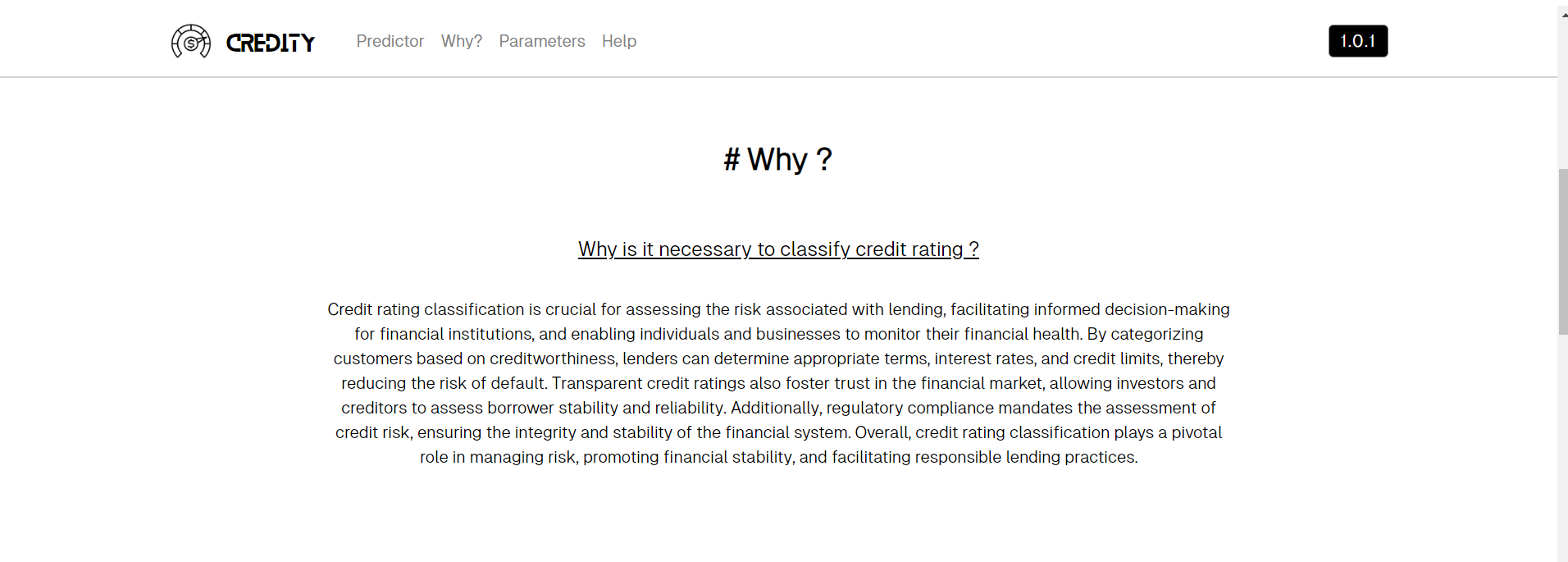
linux > source ./venv/bin/activate

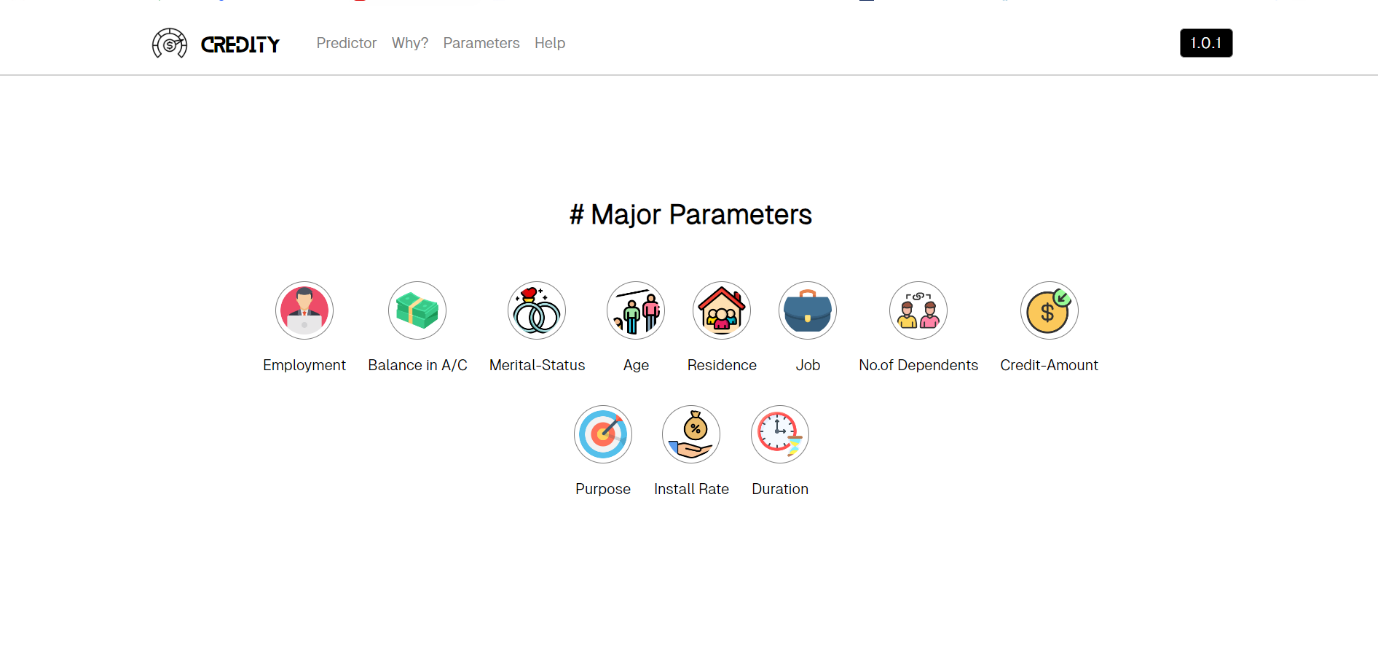
**Step 4 : After Activating Virtual Env Write Below Code**

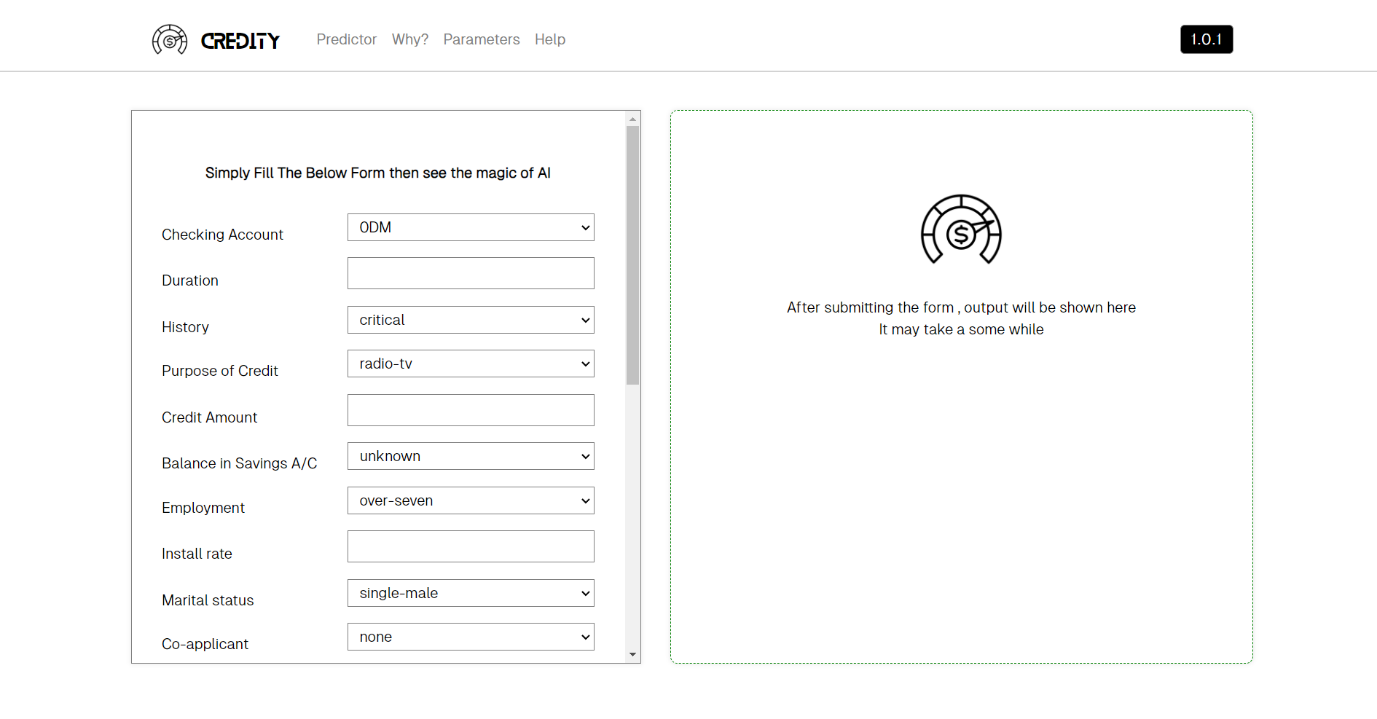
> python app.py

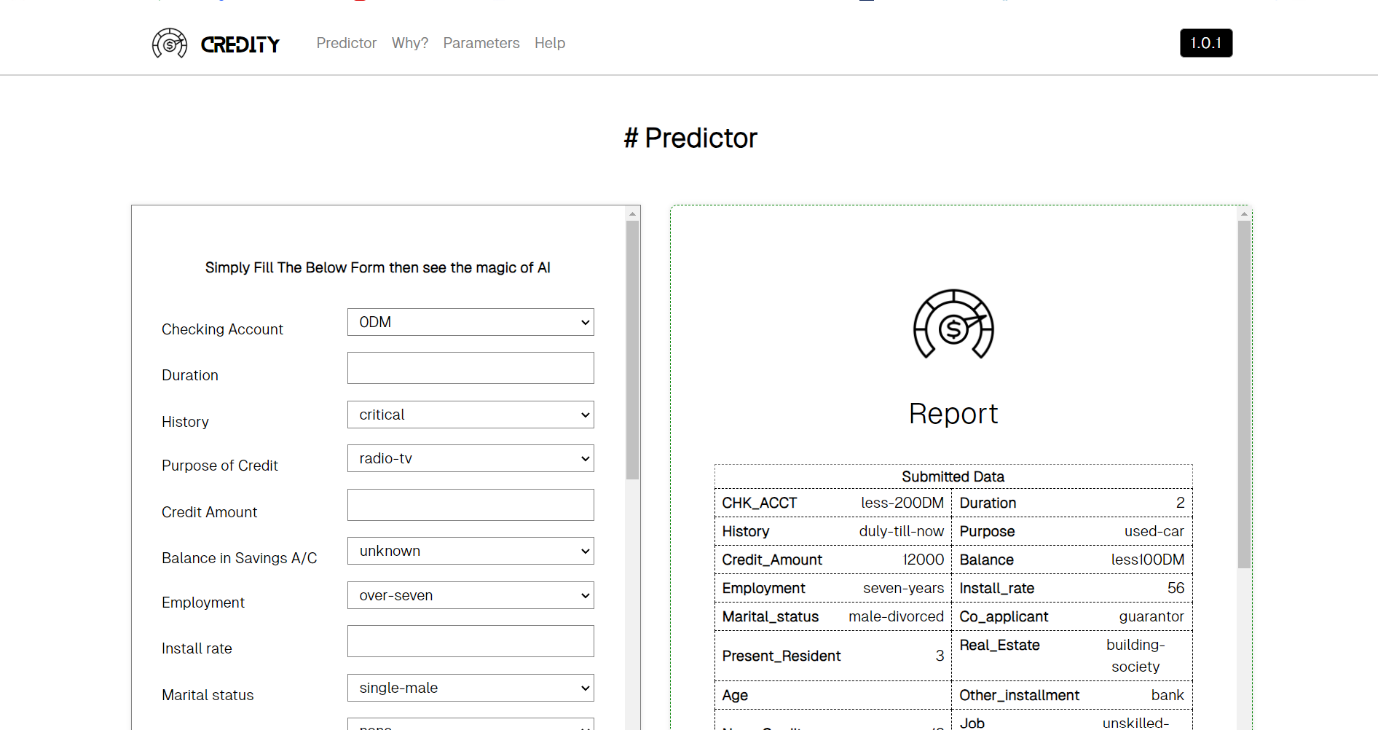
**WEB APPLICATION DEMO :**

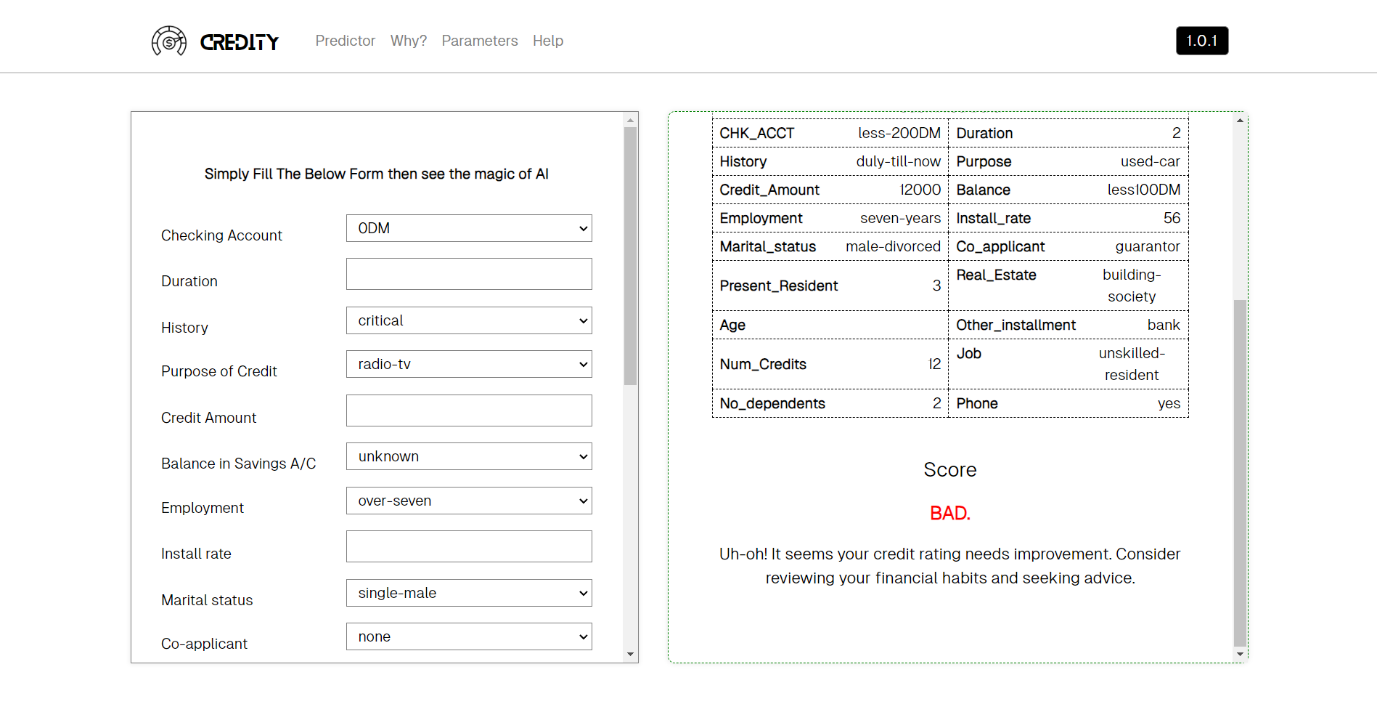
****

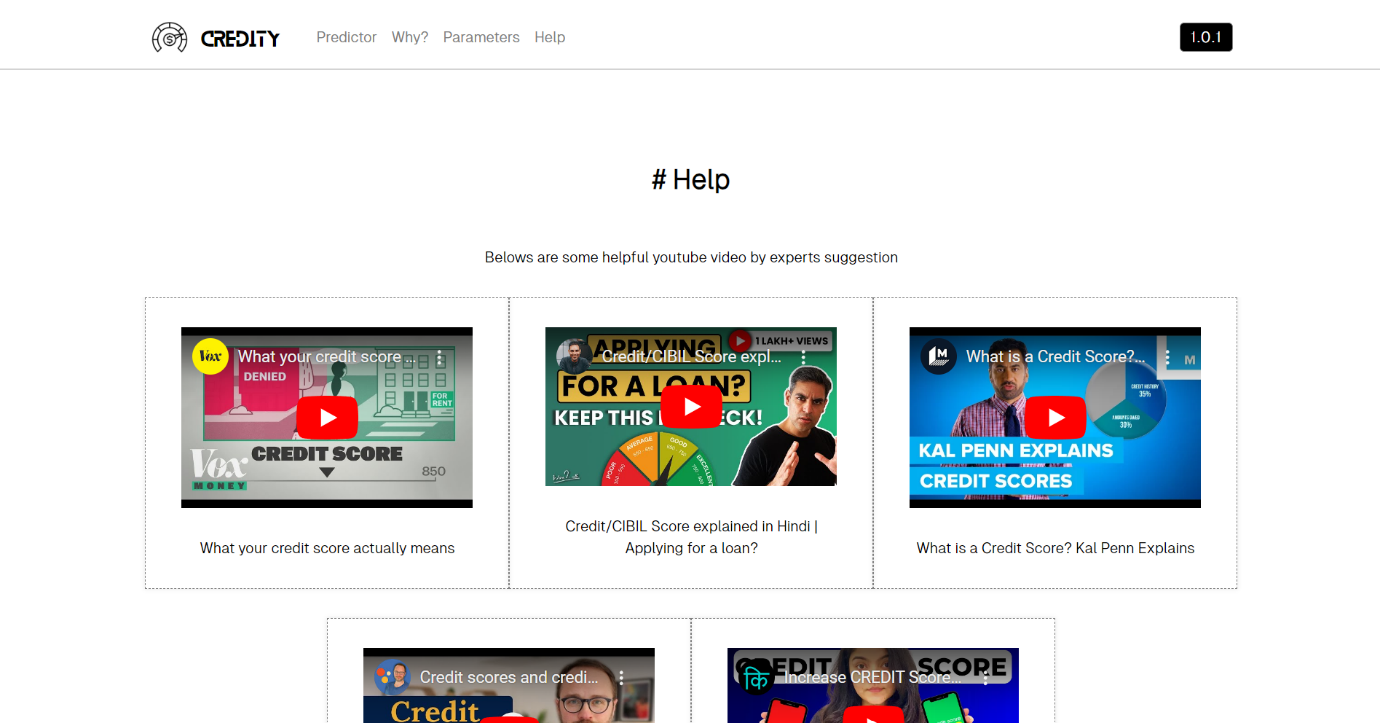
****

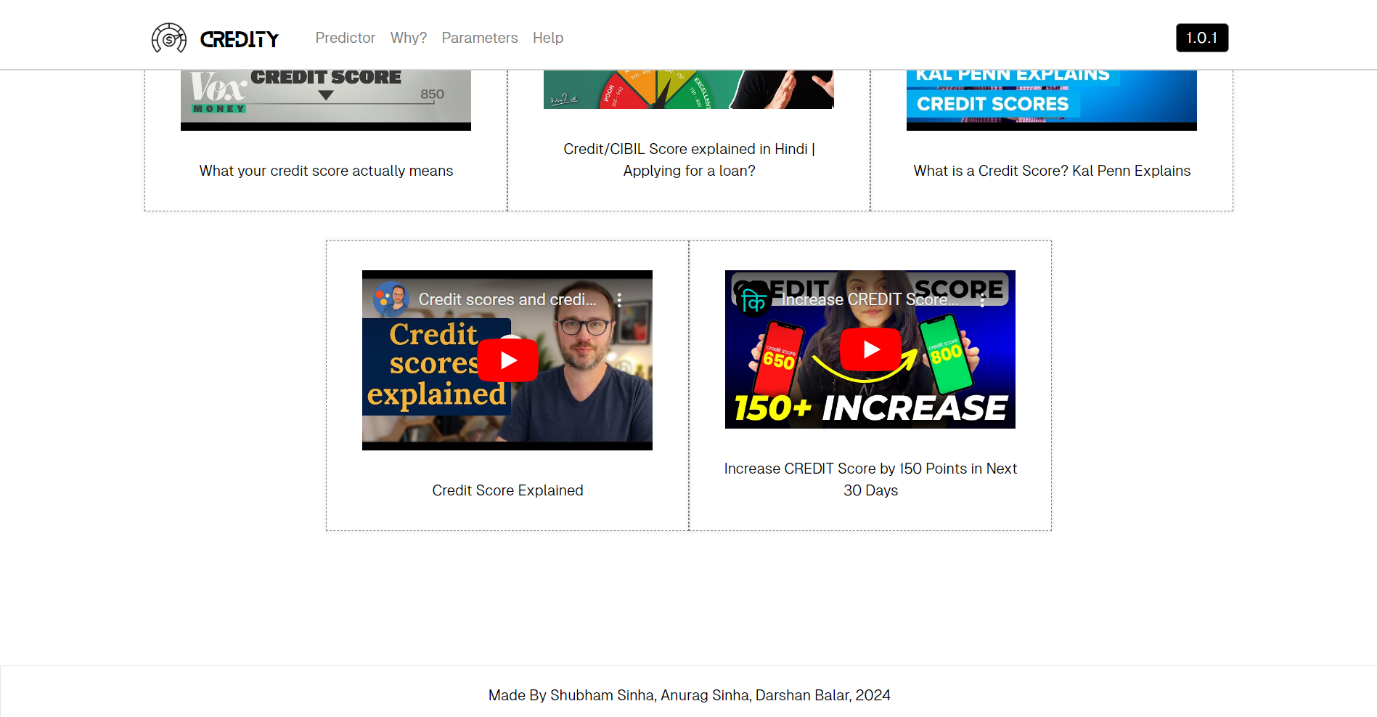
****

****

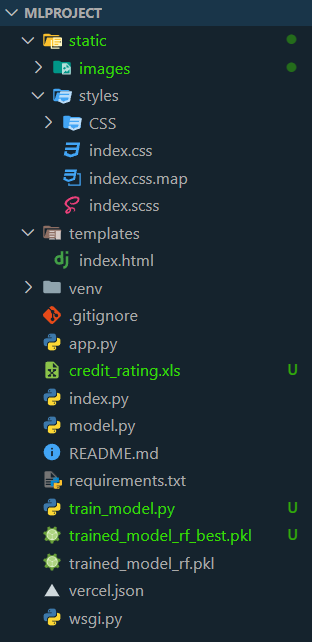
****







**FOLDER STRUCTURE**



**train\_model.py**

|  |
| --- |
| import pandas as pd  from sklearn.model\_selection import train\_test\_split, GridSearchCV  from sklearn.preprocessing import MinMaxScaler, OneHotEncoder  from sklearn.compose import ColumnTransformer  from sklearn.pipeline import Pipeline  from sklearn.ensemble import RandomForestClassifier  from sklearn.metrics import accuracy\_score, confusion\_matrix, f1\_score  import joblib  import matplotlib.pyplot as plt  import seaborn as sns  *# Load the dataset*  dataset = pd.read\_excel("credit\_rating.xls")  *# Separate features (X) and target variable (y)*  X = dataset.drop(columns=["Credit classification"])  y = dataset["Credit classification"]  *# Identify categorical features*  categorical\_features = X.select\_dtypes(include=['object']).columns.tolist()  *# Define preprocessing steps for categorical and numerical features*  categorical\_transformer = OneHotEncoder(handle\_unknown='ignore')  numerical\_transformer = MinMaxScaler()  *# Get column indices for categorical features*  categorical\_indices = [X.columns.get\_loc(col) for col in categorical\_features]  *# Get column indices for numerical features*  numerical\_indices = [i for i in range(len(X.columns)) if i not in categorical\_indices]  *# Combine preprocessing steps*  preprocessor = ColumnTransformer(      transformers=[          ('cat', categorical\_transformer, categorical\_indices),          ('num', numerical\_transformer, numerical\_indices)      ])  *# Split the dataset into training and testing sets*  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  *# Define the classification model using Random Forest Classifier*  classification\_model = Pipeline(steps=[('preprocessor', preprocessor),                                         ('classifier', RandomForestClassifier())])  *# Define the hyperparameters grid*  param\_grid = {      'classifier\_\_n\_estimators': [100, 200, 300], *# Number of trees in the forest*      'classifier\_\_max\_depth': [None, 10, 20], *# Maximum depth of the trees*      'classifier\_\_min\_samples\_split': [2, 5, 10] *# Minimum number of samples required to split a node*  }  *# Perform grid search with 5-fold cross-validation*  grid\_search = GridSearchCV(classification\_model, param\_grid, cv=5, scoring='accuracy', n\_jobs=-1)  *# Fit the grid search to find the best hyperparameters*  grid\_search.fit(X\_train, y\_train)  *# Get the best hyperparameters and the best estimator*  best\_params = grid\_search.best\_params\_  best\_estimator = grid\_search.best\_estimator\_  *# Save the best estimator (model)*  joblib.dump(best\_estimator, "trained\_model\_rf\_best.pkl")  *# Print the best hyperparameters*  print("Best Hyperparameters:", best\_params)  *# Predict the target variable on the test set using the best estimator*  y\_pred = best\_estimator.predict(X\_test)  *# Calculate accuracy*  accuracy = accuracy\_score(y\_test, y\_pred)  *# Print the accuracy*  print("Accuracy:", accuracy)  *# Plot confusion matrix*  def plot\_confusion\_matrix(y\_true, y\_pred):      cm = confusion\_matrix(y\_true, y\_pred)      plt.figure(figsize=(8, 6))      sns.heatmap(cm, annot=True, fmt='g', cmap='Blues', cbar=False)      plt.xlabel('Predicted')      plt.ylabel('Actual')      plt.title('Confusion Matrix')      plt.show()  *# Plot confusion matrix*  plot\_confusion\_matrix(y\_test, y\_pred)  *# Calculate F1 score*  f1 = f1\_score(y\_test, y\_pred, average='weighted')  *# Print F1 score*  print("F1 Score:", f1) |

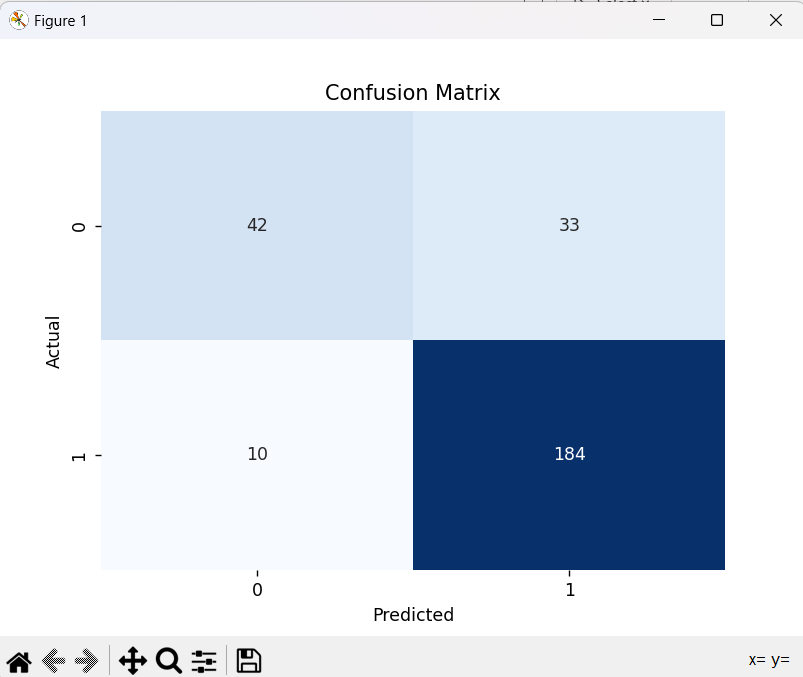
**App.py**

|  |
| --- |
| from flask import Flask,render\_template,request  from model import load\_model, predict\_credit\_classification  app = Flask(\_\_name\_\_, static\_url\_path='/static')  model = load\_model()  @app.route('/',methods=['GET', 'POST'])  def home\_page():        chak=""      duration=""      history=""      purpose=""      credit\_amount=""      balance=""      employment=""      install\_rate=""      marital\_status=""      co\_applicant=""      present\_residence=""      real\_estate=""      age=""      other\_installment=""      residence=""      no\_of\_credits=""      job=""      no\_of\_dependents=""      phone=""      foreign=""        if request.method == 'POST':            chak = request.form['CHK\_ACCT']          duration = request.form['Duration']          history = request.form['History']          purpose = request.form['Purpose\_of\_credit']          credit\_amount = request.form['Credit\_Amount']          balance = request.form['Balance\_in\_Savings\_A\_C']          employment = request.form['Employment']          install\_rate = request.form['Install\_rate']          marital\_status = request.form['Marital\_status']          co\_applicant = request.form['Co\_applicant']          present\_residence = request.form['Present\_Resident']          real\_estate = request.form['Real\_Estate']          age = request.form['Age']          other\_installment = request.form['Other\_installment']          residence = request.form['Residence']          no\_of\_credits = request.form['Num\_Credits']          job = request.form['Job']          no\_of\_dependents = request.form['No\_dependents']          phone = request.form['Phone']          foreign = request.form['Foreign']            data = {              'CHK\_ACCT': chak,              'Duration': duration,              'History': history,              'Purpose\_of\_credit': purpose,              'Credit\_Amount': credit\_amount,              'Balance\_in\_Savings\_A\_C': balance,              'Employment': employment,              'Install\_rate': install\_rate,              'Marital\_status': marital\_status,              'Co\_applicant': co\_applicant,              'Present\_Resident': present\_residence,              'Real\_Estate': real\_estate,              'Age': age,              'Other\_installment': other\_installment,              'Residence': residence,              'Num\_Credits': no\_of\_credits,              'Job': job,              'No\_dependents': no\_of\_dependents,              'Phone': phone,              'Foreign': foreign          }            prediction = predict\_credit\_classification(model, [list(data.values())])            return render\_template('index.html', prediction=prediction,chak=chak,duration=duration,history=history,credit\_amount=credit\_amount,balance=balance,employment=employment,install\_rate=install\_rate,marital\_status=marital\_status,co\_applicant=co\_applicant,present\_residence=present\_residence,real\_estate=real\_estate,other\_installment=other\_installment,no\_of\_credits=no\_of\_credits,job=job,no\_of\_dependents=no\_of\_dependents,phone=phone,purpose=purpose)          return render\_template('index.html')    if \_\_name\_\_ == '\_\_main\_\_':      app.run(debug=False) |

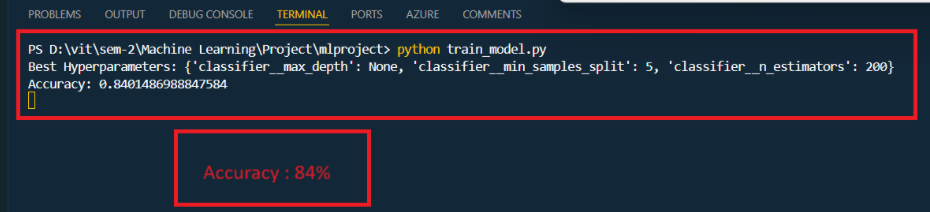
**Model.py**

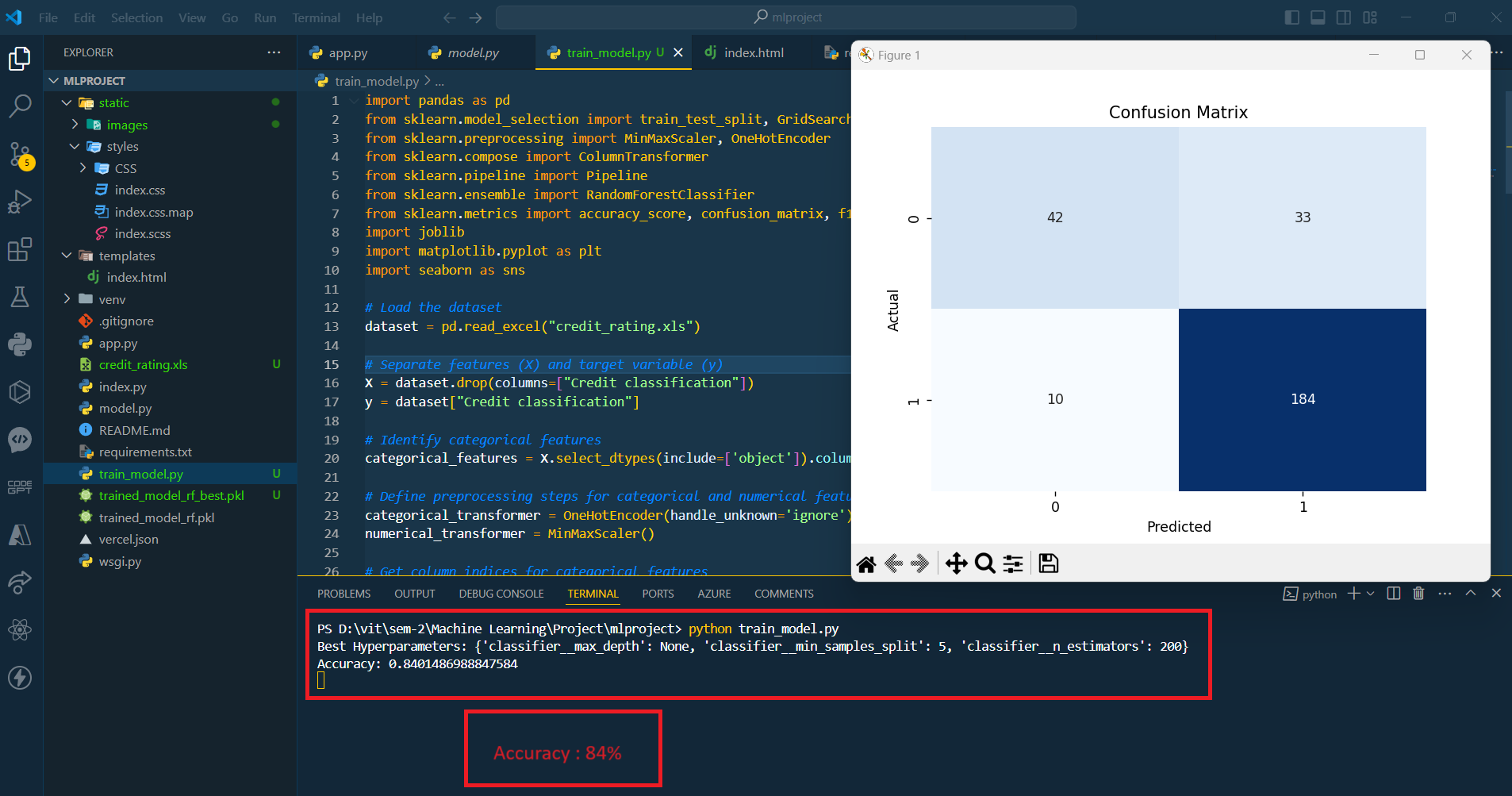
|  |
| --- |
| *#model.py*  import joblib  *# Load the trained model*  def load\_model(model\_path="trained\_model\_rf.pkl"): *# Update the model path to the one saved with Random Forest*      return joblib.load(model\_path)  *# Function to predict credit classification*  def predict\_credit\_classification(model, input\_data):      try:          prediction = model.predict(input\_data)          return prediction      except Exception as e:          return str(e) |

**CONFUSION MATRIX & ACCURACY**

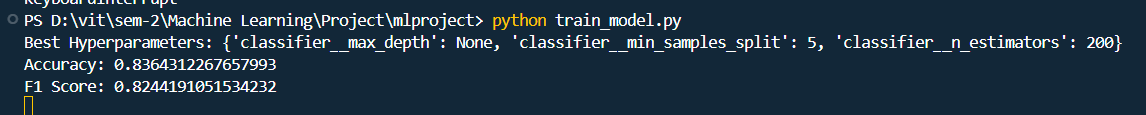
****

**ACCURACY**

****

****

**F1 SCORE**

****