Project Title

Credit Risk Analysis Using Python, Logistic Regression, and Power BI

Submitted By: **Submission Date:**28/07/2025

Name: Yadav Manish Hukumdev

Project Overview:

This internship project focuses on building a simple yet practical Credit Risk Prediction system. The main goal is to predict whether a customer is likely to default on a loan based on their financial and demographic details. The project covers the complete pipeline, starting from data cleaning and preparation using Python libraries to training a logistic regression model. The trained model is then deployed as a user-friendly web application using Flask. Finally, a Power BI dashboard is created to visualize key insights from the predictions, helping stakeholders make better lending decisions.

Objective:

To predict whether a customer is likely to default on a loan or not, using logistic regression, and to deploy the model as a simple web application.

Tools & Technologies Used:

Part	Tools
Data Cleaning	Python, Pandas
Model Building	Scikit-learn (Logistic Regression)
Web Application	Flask
Frontend	HTML, CSS, JavaScript
Dashboard	Power BI
IDE	Jupyter Notebook, VS Code

Methodology:

Step 1: Data Collection

- Collected historical customer loan data (bankloans.csv).

Key columns:

- Age, Education Level, Years Employed, Years at Address, Income, Debt-to-Income, Credit Debt, Other Debt

Step 2: Data Cleaning

- Loaded data in Pandas.
- Checked for nulls & duplicates.
- Filled missing numerical values with median.
- Removed duplicate rows.
- Verified data types.

```python
#Explore the Data
df.info()
df.describe()
df.isnull().sum()
df['default'].value_counts() #checking the balance count of targeterd variable
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1150 entries, 0 to 1149
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1150 non-null   int64
 1   ed        1150 non-null   int64
 2   employ    1150 non-null   int64
 3   address   1150 non-null   int64
 4   income    1150 non-null   int64
 5   debtinc   1150 non-null   float64
 6   creddebt  1150 non-null   float64
 7   othdebt   1150 non-null   float64
 8   default   700 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 81.0 KB

default
0.0    517
1.0    183
Name: count, dtype: int64
```

```python
#Clean the data
df = df.dropna()
#removes duplicate
df = df.drop_duplicates()
```

```python
df.shape
df.head()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 700 entries, 0 to 699
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       700 non-null    int64
 1   ed        700 non-null    int64
 2   employ    700 non-null    int64
 3   address   700 non-null    int64
 4   income    700 non-null    int64
 5   debtinc   700 non-null    float64
 6   creddebt  700 non-null    float64
 7   othdebt   700 non-null    float64
 8   default   700 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 54.7 KB
```

```python
df.to_csv("DataCredit_Cleaned.csv",index=False)
```

```python
df.head()
```

	age	ed	employ	address	income	debtinc	creddebt	othdebt	default
0	41	3	17	12	176	9.3	11.359392	5.008608	1.0
1	27	1	10	6	31	17.3	1.362202	4.000798	0.0
2	40	1	15	14	55	5.5	0.856075	2.168925	0.0
3	41	1	15	14	120	2.9	2.658720	0.821280	0.0
4	24	2	2	0	28	17.3	1.787436	3.056564	1.0

Step 3: Feature Selection

- Selected important columns:
- Age, Education Level, Years Employed, Years at Address, Income, Debt-to-Income, Credit Debt, Other Debt

Step 4: Model Building

- Split data: training (80%) & testing (20%).
- Trained Logistic Regression using scikit-learn.
- Predicted on test data.

Evaluated using:

- Accuracy score
- Confusion matrix
- Classification report

```python
#Split Data into X (features) and y (target)
x = df.drop('default' , axis=1) #All column except 'default'
y = df['default']  #Targeted column
```

```python
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

```python
#Train a Model (Logistics Regression)
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train, y_train)
```

```
▾ LogisticRegression ①②

LogisticRegression()
```

```python
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
y_pred = model.predict(x_test)
print("Accuracy", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy 0.85
              precision    recall  f1-score   support

         0.0       0.86      0.95      0.90       102
         1.0       0.81      0.58      0.68        38

    accuracy                           0.85       140
   macro avg       0.84      0.76      0.79       140
weighted avg       0.85      0.85      0.84       140

[[97  5]
 [16 22]]
```

Step 5: Model Saving

- Saved trained model using Joblib for reuse in Flask app.

Step 6: Flask Web App

- Built app.py with:
- Login page: user authentication.
- Dashboard page: links to Prediction & Logout.
- Prediction page: form for user to enter details.
- Shows: Default / Not Default, Risk Level, Confidence.

Step 7: Frontend

- Created pages using HTML.
- Added styles with CSS.
- Used JavaScript for form validation (optional).
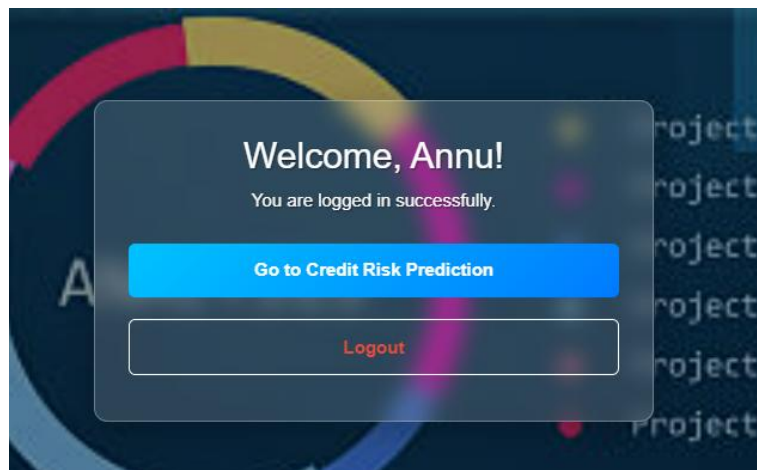
Step 8: Predictions

- User fills in:
- Age, Education, Employment Years, Address Years, Credit Debt, Debt-to-Income Ratio, Income.
- Backend sends data to model → model predicts:

Output:

- Default or Not Default.
- Risk Range: High / Medium / Low.
- Confidence Score: Probability of default.

Step 9: Power BI Dashboard

- **E**xported Prediction data to csv.
- Loaded into Power Bi
- Created charts:
- Credit Card Debt by Age Group and Actual Default Status.
- Application by Age Group and Default Status.
- Total Applications by Education Level.
- Overall Default Status.
- Also used visuals like Table, KPI Cards, Slicers.
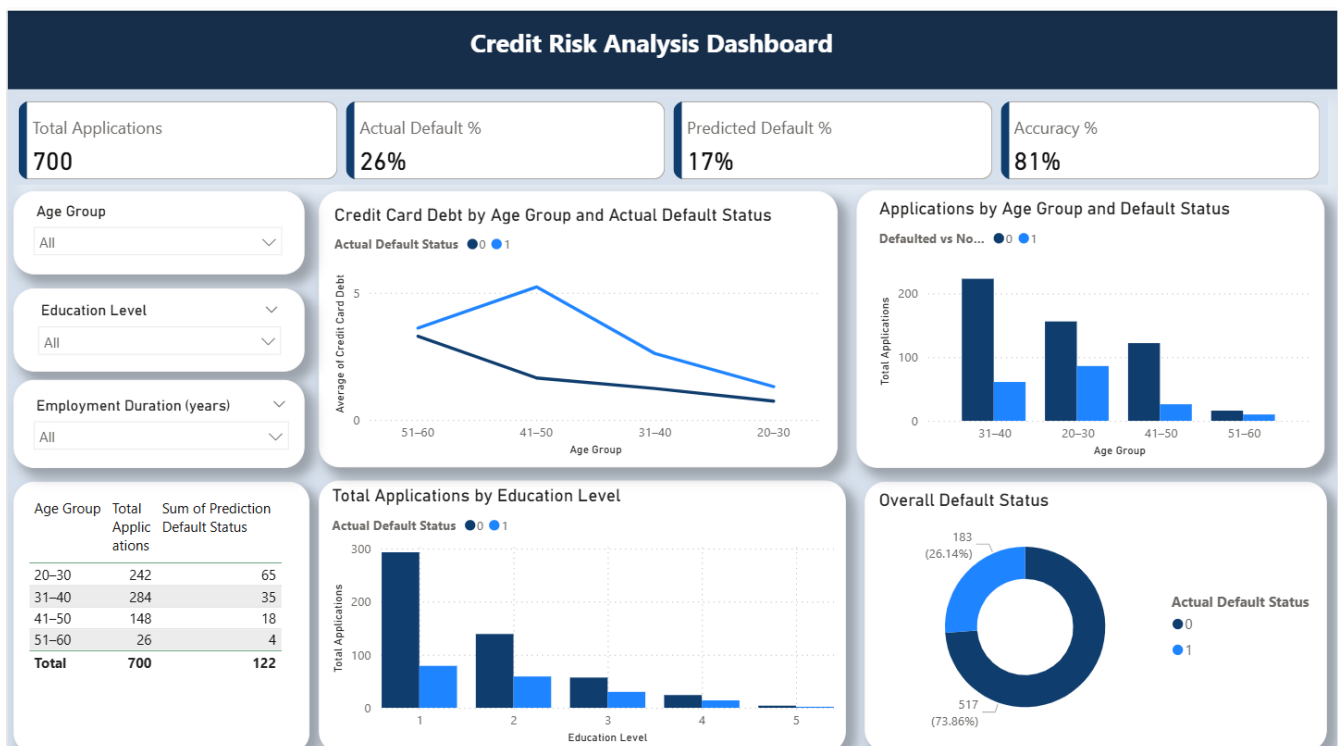
Step 10: Testing

- Tested web app for:
- Correct login flow.
- Accurate predictions.
- Proper logout.
- Checked different scenarios for high and low risk.

Problems & Challenges Faced:

During the development of this project, several challenges were encountered. The raw credit data had missing and duplicate values, which needed careful cleaning to ensure accurate predictions. Another major challenge was handling the imbalance in the dataset between

default and non-default cases, which could affect model performance. Deploying the logistic regression model into a Flask web application required proper integration between the backend and frontend, which was initially complex. Additionally, it was important to make the prediction output clear and easy to understand for non-technical users.

Solutions & Strategies Implemented:

To overcome these challenges, various strategies and tools were used. The Pandas library helped in cleaning the data by removing null values and duplicates. The logistic regression model was chosen for its simplicity and effectiveness in binary classification tasks. To handle the deployment, Flask was used to build a secure and accessible web application. HTML, CSS, and JavaScript were used to design simple and responsive pages for a better user experience. Finally, Power BI was used to create interactive dashboards to present the prediction results and trends in a clear and visual manner.

Industry Research:

Credit risk prediction is an important part of the FinTech industry and financial institutions worldwide. Many companies offer similar solutions for credit scoring and risk analysis. For example, Experian is a global credit reporting agency that provides credit scoring services to help lenders assess customer risk. Equifax is another leading company that offers consumer credit reports and risk management solutions. TransUnion is also well-known for its credit monitoring and data analytics services to support better financial decision-making. This project is a small-scale attempt to replicate similar industry practices using basic tools and techniques.

Testing Evidence & Validation:

The entire credit risk prediction system was tested thoroughly to make sure that each part works correctly. The login page accepts valid usernames and passwords. After logging in, the dashboard and prediction pages open without any errors. Different test inputs were given in the prediction form and the results (Default / Not Default) were displayed correctly along with risk range and confidence level. This proves that the Flask web app, trained model, and user interface are all working as expected.

Conclusion:

This project successfully predicts whether a customer is likely to default on a loan using logistic regression. It covers complete steps — from data cleaning and model building to deploying a simple web application. The Power BI dashboard shows clear visual insights about customer risk levels and default trends. This helps banks and financial companies to make better decisions about lending money and managing credit risk.

Recommendations / Next Steps:

- Try other advanced models like Decision Tree or Random Forest to improve prediction accuracy.
- Deploy the Flask web app on a live cloud server (like Heroku or Render) so that users can access it online.
- Connect the prediction system to a real-time bank database to make it more useful for actual credit departments.
- Add extra features like PDF report download and automatic email alerts for high-risk customers.
- Use more detailed data in the future to cover other risk factors such as credit history or payment behavior.