LAWRENCE TECHNOLOGICAL UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

MACHINE LEARNING ASSIGMDENT -4(FINAL REPORT)

“Credit Card Default Prediction”

Data set information:

Predictive Model to Identify Default Risk

Introduction & Domain Knowledge

Objective: Predict whether a credit card holder will default on payment in the next month.

Importance: Helps financial institutions manage risk and optimize lending.

Dataset: UCI Default of Credit Card Clients (30,000 records of Taiwanese credit card holders).

Key Features: Includes demographic data, financial behavior, and payment history.

Dataset Analysis

Total Data: 30,000 rows, 25 columns.

Categories:

Demographic: Gender, Age, Education, Marital Status.

Financial: Credit Limit, Payment History, Bill Amounts, Payment Amounts.

Target Variable: Default payment next month (1 = Default, 0 = No Default).

Feature Analysis & Selection

Demographic Features: Weak correlation but potentially valuable in combination.

Payment History (PAY_0 to PAY_6): Strong correlation with default.

Billing and Payment Amounts: Moderate predictive power; provides insights into financial discipline. Conclusion: Payment history and payment amounts are critical predictors.

▪ DETALIED STEPS

IMPORTING USEFUL LIBRARIES

Tools Help

+ Code + Text

```
[1] import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt
```

Start coding or generate with AI.

LOAD THE DATASET

ools Help

+ Code + Text

```
[1] import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt
```

```
# Load the dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/00350/default%20of%20credit%20car
data = pd.read_excel(url, header=1)  # Make sure to check the structure of the dataset
data.head()
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AM
0	1	20000	2	2	1	24	2	2	-1	-1	...	0	
1	2	120000	2	2	2	26	-1	2	0	0	...	3272	34
2	3	90000	2	2	2	34	0	0	0	0	...	14331	149
3	4	50000	2	2	1	37	0	0	0	0	...	28314	289
4	5	50000	1	2	1	57	-1	0	-1	0	...	20940	191

5 rows × 25 columns

DATASET FIRST VIEW

VARIABLES DESCRIPTION

Help

+ Code + Text

[2]

5 rows × 25 columns

```
# Check basic statistics
data.describe()
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.00000
mean	15000.500000	167484.322667	1.603733	1.853133	1.551867	35.485500	-0.01670
std	8660.398374	129747.661567	0.489129	0.790349	0.521970	9.217904	1.12380
min	1.000000	10000.000000	1.000000	0.000000	0.000000	21.000000	-2.00000
25%	7500.750000	50000.000000	1.000000	1.000000	1.000000	28.000000	-1.00000
50%	15000.500000	140000.000000	2.000000	2.000000	2.000000	34.000000	0.00000
75%	22500.250000	240000.000000	2.000000	2.000000	2.000000	41.000000	0.00000
max	30000.000000	1000000.000000	2.000000	6.000000	3.000000	79.000000	8.00000

8 rows × 25 columns

CHECK FOR MISSING VALUES

Tools Help All changes saved

+ Code + Text

[3]

75%	22500.250000	240000.000000	2.000000	2.000000	2.000000	41.000000	0.00000
max	30000.000000	1000000.000000	2.000000	6.000000	3.000000	79.000000	8.00000

8 rows × 25 columns

```
# Check for missing values
missing_values = data.isnull().sum()
```

[5]

```
# Check for duplicates
duplicates = data.duplicated().sum()
```

CHECKING FOR DUPLICATIONS

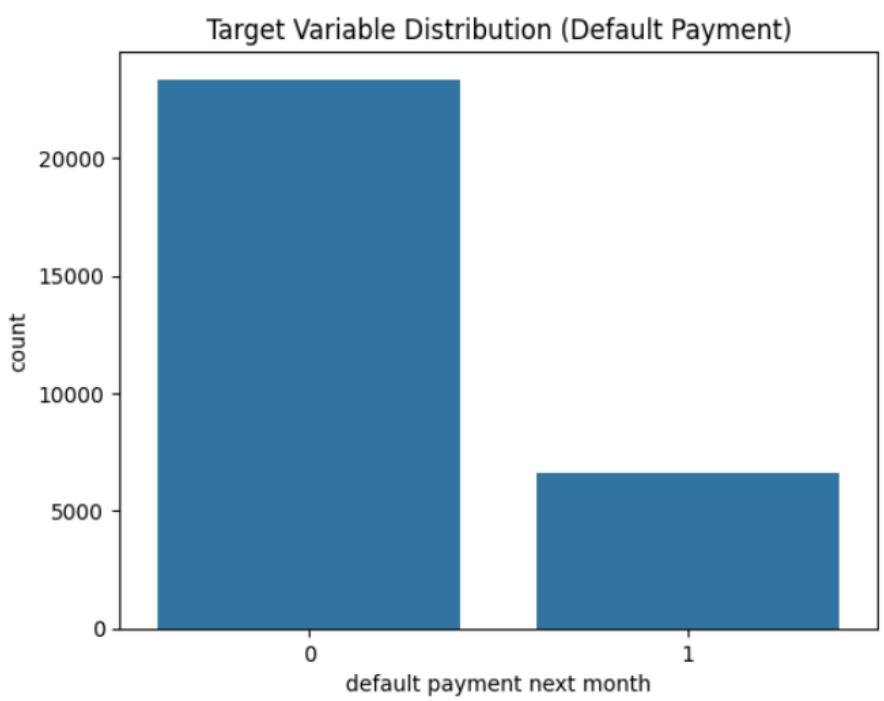
Data Visualization - Target Variable Distribution

Help All changes saved

+ Code + Text

[5]
```
# Check for duplicates
duplicates = data.duplicated().sum()
```

```
# Visualize the target column
sns.countplot(x='default payment next month', data=data)
plt.title("Target Variable Distribution (Default Payment)")
plt.show()
```



Target Imbalance:

Non-Defaults: ~22,000

Defaults: ~5,000

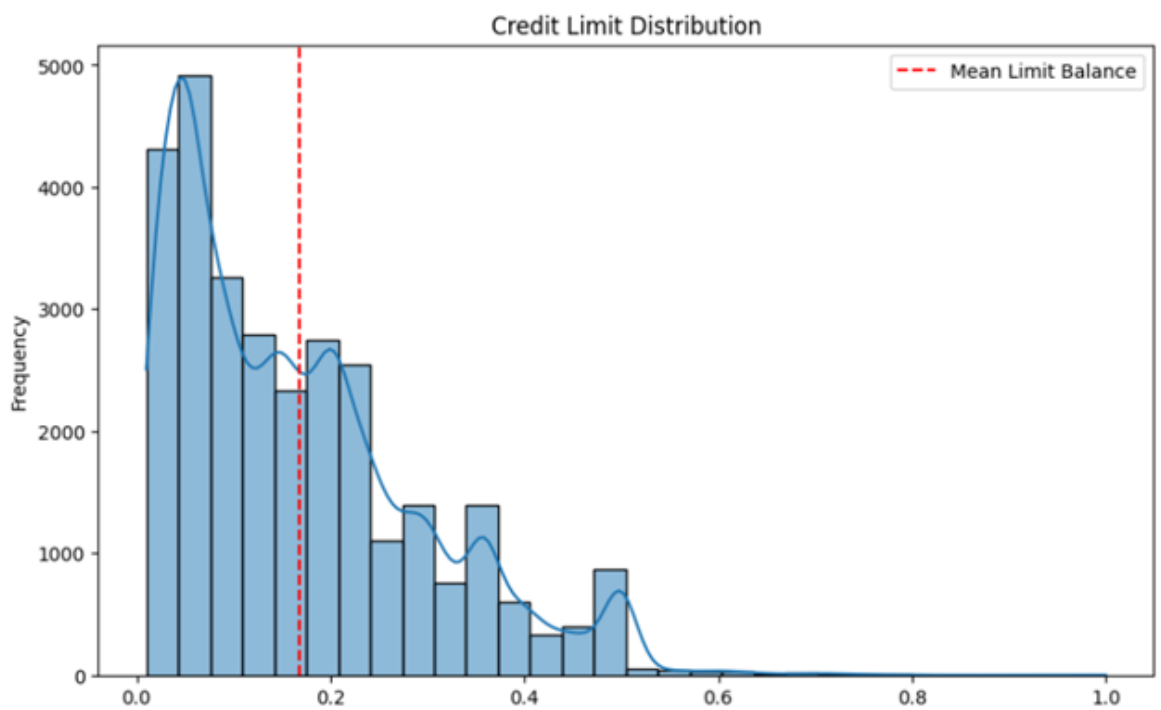
Challenge: Imbalanced data can skew predictions.

Data Visualization - Credit Limit Distribution

ools Help All changes saved

+ Code + Text

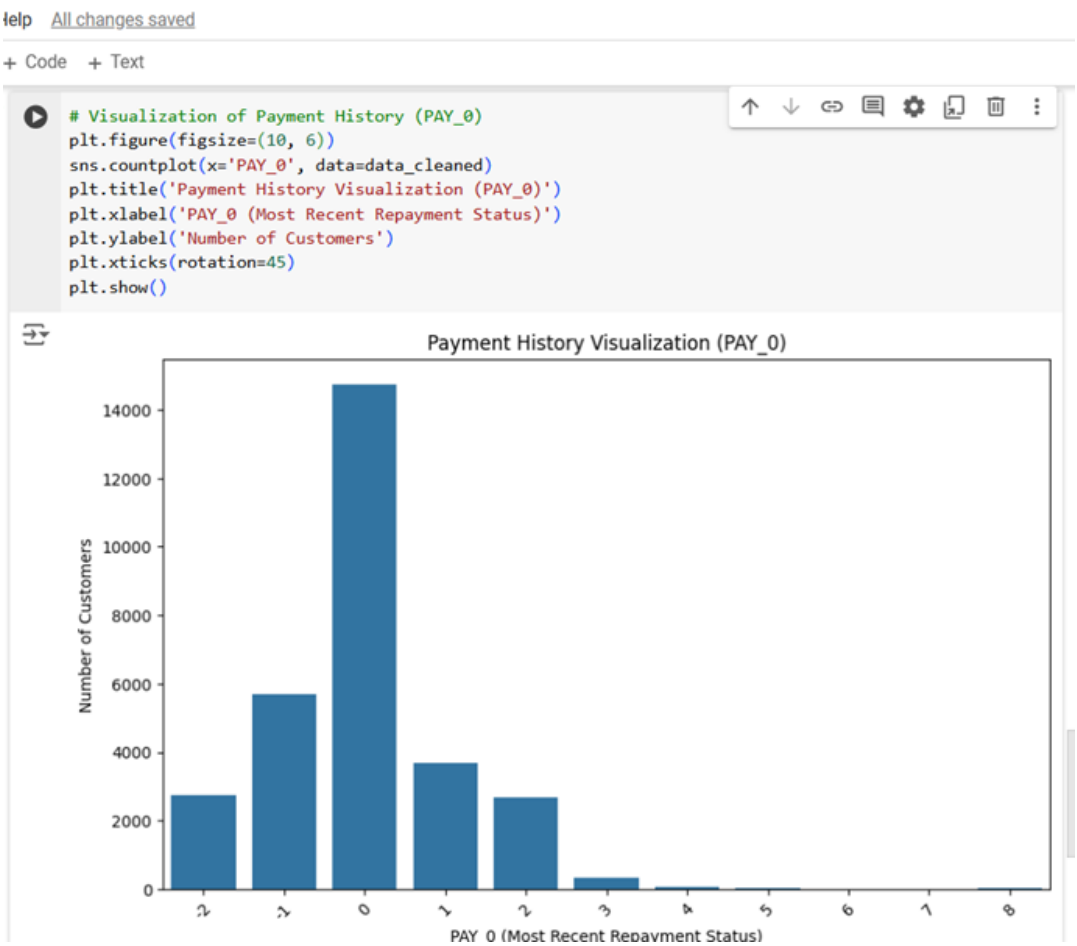
```
# Visualization of Credit Limit Distribution
plt.figure(figsize=(10, 6))
sns.histplot(data_cleaned['LIMIT_BAL'], bins=30, kde=True)
plt.title('Credit Limit Distribution')
plt.xlabel('Limit Balance')
plt.ylabel('Frequency')
plt.axvline(data_cleaned['LIMIT_BAL'].mean(), color='r', linestyle='--', label='Mean Limit Balance')
plt.legend()
plt.show()
```



Observation: Most customers have a credit limit below $200,000.

Mean Limit Balance: Slightly above $200,000.

Data Visualization - Payment History

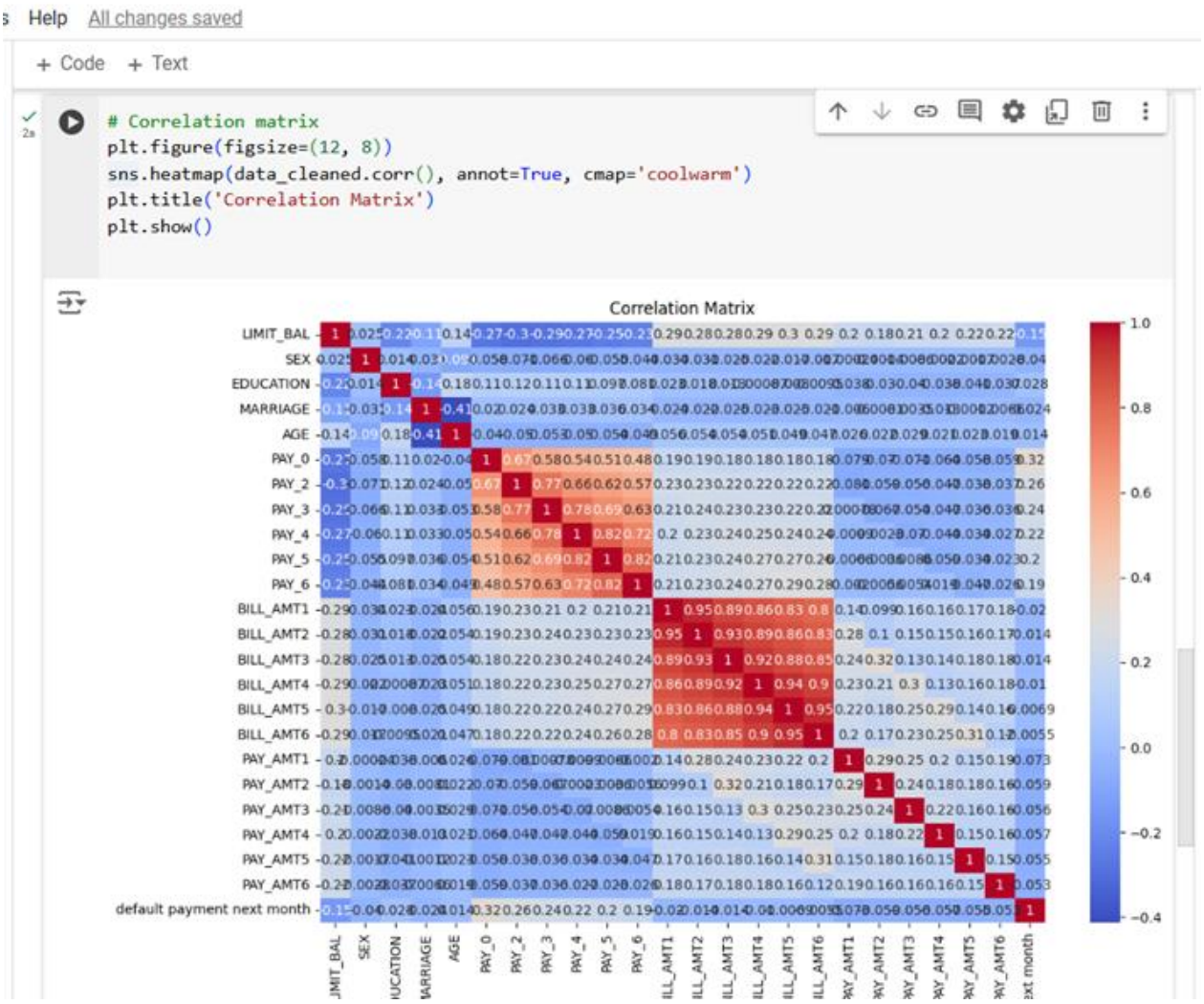


Observation: Majority on-time payments, with fewer delayed payments.

PAY_0: Indicates repayment status (recent).

Visual: (Include the countplot for PAY_0)

Correlation Matrix



Key Correlations:PAY_0 to PAY_6 positively correlate with defaulSt.BILL_AMT1-6 have lower correlations.

Conclusion: Past payment behavior is a strong predictor.

Data Cleaning & Preprocessing

ls Help All changes saved

+ Code + Text

PAY_0 (Most Recent Repayment Status)

```
[13] from sklearn.preprocessing import StandardScaler
```

```
[14] scaler = StandardScaler()
     scaled_features = scaler.fit_transform(data_cleaned.drop('default payment next month', axis=1))

     from sklearn.preprocessing import OneHotEncoder
```

```
# One hot encoding for categorical variables
encoded_data = pd.get_dummies(data_cleaned, columns=['SEX', 'EDUCATION', 'MARRIAGE'], drop_first=Tru
```

Duplicates: Found and removed

Irrelevant Features: Dropped 'ID' column.

Encoding: Categorical variables like SEX, EDUCATION, and MARRIAGE encoded with one-hot encoding.

Scaling: Applied StandardScaler to LIMIT_BAL, BILL_AMT, PAY_AMT columns.

Model Building & Evaluation Plans

Help All changes saved

+ Code + Text

[15]
```
encoded_data = pd.get_dummies(data_cleaned, columns=['SEX', 'EDUCATION', 'MARRIAGE'], drop_f
```

[16]
```
from sklearn.model_selection import train_test_split

# Split the data into train and test sets
X = encoded_data.drop('default payment next month', axis=1)
y = encoded_data['default payment next month']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

[17]
```
# N-fold cross-validation
from sklearn.model_selection import cross_val_score
```

```
# Example for Logistic Regression
from sklearn.linear_model import LogisticRegression

log_reg = LogisticRegression()
cv_scores = cross_val_score(log_reg, X_train, y_train, cv=10)

print("Cross-validation mean score:", cv_scores.mean())
print("Cross-validation standard deviation:", cv_scores.std())
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWar
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT
```

Logistic Regression:

Cross-Validation Score: Mean and Standard Deviation.

Logistic Regression & Random Forest Results

Help

+ Code + Text

```python
# Train Logistic Regression
log_reg.fit(X_train, y_train)
y_pred = log_reg.predict(X_test)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbf
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```python
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      1.00      0.88      4673
           1       0.29      0.00      0.00      1320

    accuracy                           0.78      5993
   macro avg       0.53      0.50      0.44      5993
weighted avg       0.67      0.78      0.68      5993
```

Logistic Regression:

Cross-Validation Score: Mean and Standard Deviation.

Classification Report: Precision, Recall, F1-score.

All changes saved

Code + Text

```
20] from sklearn.metrics import classification_report

    print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      1.00      0.88      4673
           1       0.29      0.00      0.00      1320

    accuracy                           0.78      5993
   macro avg       0.53      0.50      0.44      5993
weighted avg       0.67      0.78      0.68      5993
```

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=100)
rf.fit(X_train, y_train)

y_pred_rf = rf.predict(X_test)

print(classification_report(y_test, y_pred_rf))
```

```
              precision    recall  f1-score   support

           0       0.84      0.94      0.89      4673
           1       0.63      0.34      0.44      1320

    accuracy                           0.81      5993
   macro avg       0.73      0.64      0.67      5993
weighted avg       0.79      0.81      0.79      5993
```

Random Forest:

Classification Report: Precision, Recall, F1-score.Initial observations and performance comparison.

ols Help All changes saved

+ Code + Text

[21]

```
           accuracy                           0.81      5993
          macro avg       0.73      0.64      0.67      5993
       weighted avg       0.79      0.81      0.79      5993
```

[22]

```
from imblearn.over_sampling import SMOTE
```

[23]

```
smote = SMOTE()
X_res, y_res = smote.fit_resample(X_train, y_train)
```

```
# Train model on balanced data
log_reg_smote = LogisticRegression()
log_reg_smote.fit(X_res, y_res)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbf
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression
LogisticRegression()
```

Balancing the Dataset: SMOTE to address class imbalance.

Model Selection: Logistic Regression (interpretability) and Random Forest (complex relationships).

Evaluation Metrics: Precision, Recall, F1-score, ROC-AUC to assess model performance.

INSTALLING GRADIO

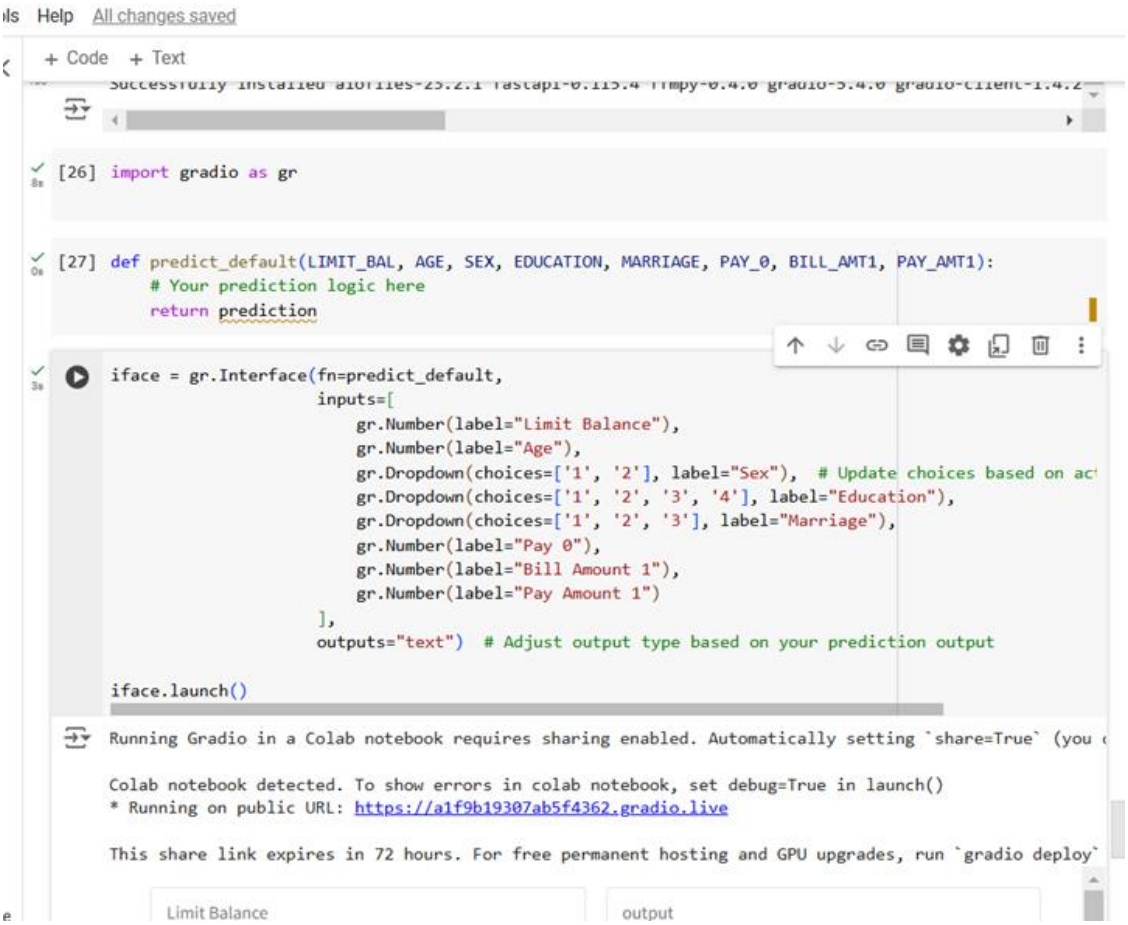
Help All changes saved

+ Code + Text

```
!pip install gradio
```

```
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages (from anyio<5.0
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.10/dist-packages (from anyio<
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyi
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from httpx>=0.24
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.10/dist-packages (from httpx
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.10/dist-packages (from htt
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingfac
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from huggingfac
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.10/dist-packages (from huggin
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pand
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: pydantic-core==2.23.4 in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.10/dist-packages (from typer<
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.10/dist-packages (from typer
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dat
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown
Downloading gradio-5.4.0-py3-none-any.whl (56.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.7/56.7 MB 9.3 MB/s eta 0:00:00
Downloading gradio_client-1.4.2-py3-none-any.whl (319 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 319.8/319.8 kB 19.5 MB/s eta 0:00:00
Downloading python_multipart-0.0.12-py3-none-any.whl (23 kB)
Downloading tomlkit-0.12.0-py3-none-any.whl (37 kB)
Downloading aiofiles-23.2.1-py3-none-any.whl (15 kB)
Downloading fastapi-0.115.4-py3-none-any.whl (94 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 94.7/94.7 kB 6.7 MB/s eta 0:00:00
Downloading huggingface_hub-0.26.2-py3-none-any.whl (447 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 447.5/447.5 kB 25.4 MB/s eta 0:00:00
Downloading MarkupSafe-2.1.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (25 kB)
```

DEPLOYMENT PLAN



Interface: Gradio for model deployment.

User Inputs:Credit Limit, Age, Sex, Education, Marital Status, Recent Payment Status, Bill Amount, Payment Amount.

Output: Prediction of default status.

Help All changes saved

+ Code + Text

```
[27] def predict_default(LIMIT_BAL, AGE, SEX, EDUCATION, MARRIAGE, PAY_0, BILL_AMT1, PAY_AMT1):
         # Your prediction logic here
         return prediction
```

```
iface = gr.Interface(fn=predict_default,
                     inputs=[
                         gr.Number(label="Limit Balance"),
                         gr.Number(label="Age"),
                         gr.Dropdown(choices=['1', '2'], label="Sex"),  # Update choices based on act
                         gr.Dropdown(choices=['1', '2', '3', '4'], label="Education"),
                         gr.Dropdown(choices=['1', '2', '3'], label="Marriage"),
                         gr.Number(label="Pay 0"),
                         gr.Number(label="Bill Amount 1"),
                         gr.Number(label="Pay Amount 1")
                     ],
                     outputs="text")  # Adjust output type based on your prediction output

iface.launch()
```

```
Running Gradio in a Colab notebook requires sharing enabled. Automatically setting `share=True` (you c

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://a1f9b19307ab5f4362.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy`
```

1

Marriage

1

Pay 0

09a8b69bfe1a2e8d54.gradio.live

Finish updat

Limit Balance

0

Age

0

Sex

1

Education

1

Marriage

1

Pay 0

0

Bill Amount 1

0

Pay Amount 1

0

Clear

Submit

output

Flag

Use via API · Built with Gradio

Conclusion

Summary: Explored and preprocessed data, identified key features, balanced the dataset, and built predictive models.

Next Steps: Refine models, test additional algorithms, optimize deployment for practical use.

Github link:

https://github.com/Vaddevinod9573/credit-card-default-prediction.git