

**Logistic Regression Report**

Prepared by:  
 Nima and Arushi

Empirical Finance & Predictive Analysis  
 Assignment 29–31: Payment Status Analysis

**Content**

|  |  |  |
| --- | --- | --- |
| Assignment No: | Name | Page no |
| 29 | Encoding dataset for Logistic Regression | 3-6 |
| 30 | Training and Testing a Logistic Regression Model | 7-12 |
| 31 | Odds Ratio | 13-18 |

**Assignment 29 - Encoding dataset for Logistic Regression**

Task:

Use binary and ordinal encoding to make the variables readable by the logistic regression method

Content:

Logistic regression is a statistical method used for binary classification problems, where the output is either 0 or 1 (e.g., yes/no, spam/not spam, pass/fail). Instead of predicting a continuous value like in linear regression, logistic regression predicts the probability that a given input belongs to a particular category.

### **Encoding Methods**

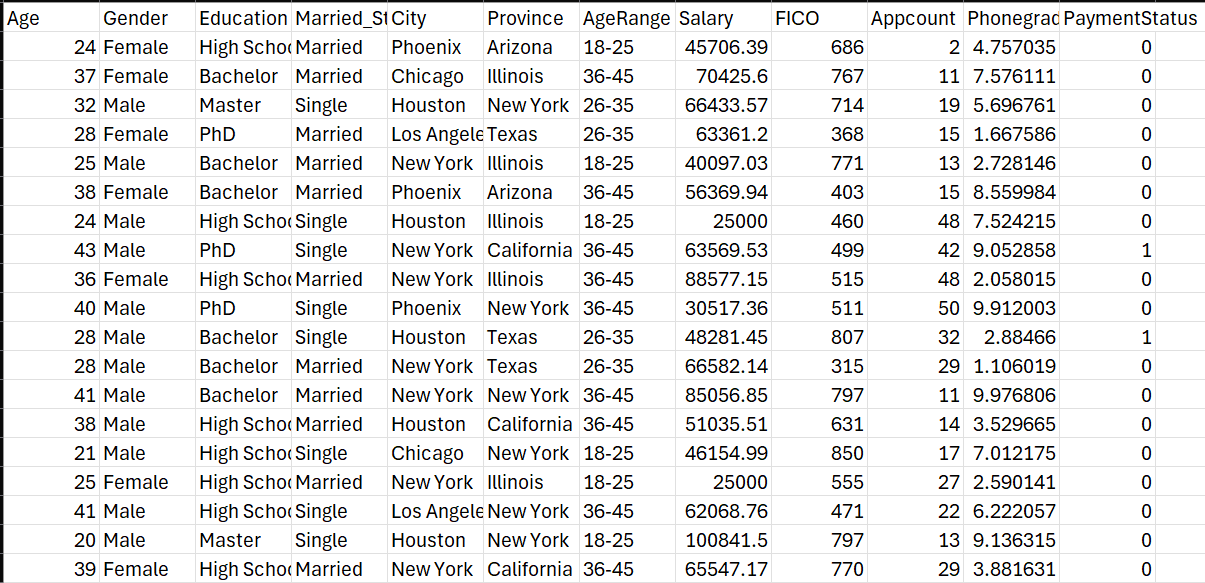
* **Binary Encoding**: Converts categorical values into binary digits and then represents them as separate columns. Useful when dealing with high-cardinality categorical features.
* **Ordinal Encoding**: Assigns integer values based on the order of categories (e.g., Low → 1, Medium → 2, High → 3). Used when categories have a meaningful order.

In preparing the dataset for logistic regression analysis, it is essential to convert categorical variables into numerical formats suitable for the model. For this assignment, the following encoding strategies were employed:

* **Binary Encoding:** Applied to variables with two categories:
  + *Gender:* Encoded as 0 and 1.
  + *Marital Status:* Encoded as 0 and 1.
  + *City:* Encoded according to predefined criteria reflecting city tiers.
  + *Province:* Encoded following a specific ranking system.
* **Ordinal Encoding:** Utilized for variables with multiple categories that have an inherent order:
  + *Education Level:* Encoded based on the hierarchy of educational attainment.
  + *Age Range:* Encoded to reflect sequential age groups.

These encoding methods ensure that categorical variables are transformed into a format that maintains their inherent relationships and is interpretable by the logistic regression model. Numeric variables were utilized in their original form, as they are already suitable for analysis.

Original Dataset in Excel:



Execution in Python:

STEP 1:

Import required libraries

import pandas as pd

import category\_encoders as ce

# Make sure this package is installed (pip install category\_encoders)

STEP 2:

Importing excel dataset to python. Replace the empty string with the actual CSV file path.

df = pd.read\_csv("")

STEP 3:

Ordinal encoding for Education and Age Range.

ordinal\_encoder = ce.OrdinalEncoder(cols=['Education', 'AgeRange'])

df = ordinal\_encoder.fit\_transform(df)

STEP 4:

Binary coding for gender, married status, city and province.

df['Gender'] = df['Gender'].map({'Male': 0, 'Female': 1})

df['Married\_Status'] = df['Married\_Status'].map({'Single': 0, 'Married': 1})

binary\_encoder = ce.BinaryEncoder(cols=['City', 'Province'])

df = binary\_encoder.fit\_transform(df)

STEP 5:

Preview the encoded dataframe (optional)

print(df.head())

STEP 6:

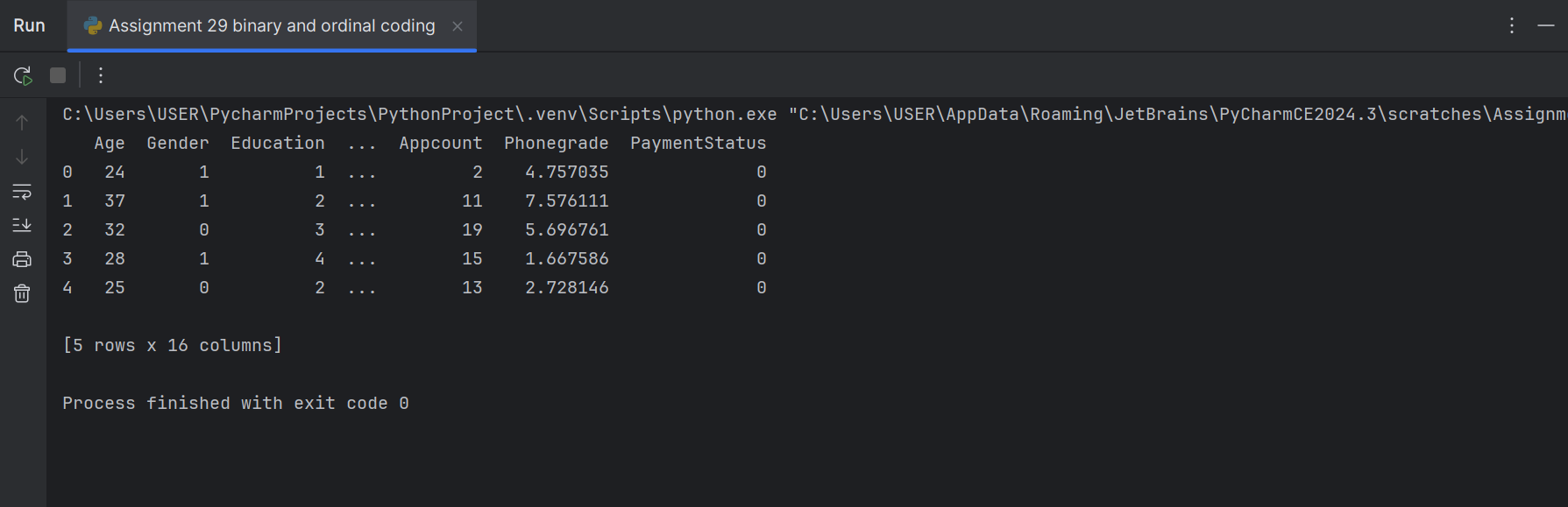
Save the encoded dataset to a new CSV file

# Replace with your desired file path

df.to\_csv("", index=False)

print("Encoded file saved successfully.")

**Python results:**



**Exported to excel:**

A screenshot of a computer

AI-generated content may be incorrect.

Literature and Software References:

A picture containing circle

Description automatically generated with low confidenceAnderson, D., Sweeney, D., Williams, T., Camm, J., Cochran, J. (2014). Statistics for Business and Economics. (12th ed.). Cengage. Page 725-732

A blue circle with a white x in it

Description automatically generatedExcel path: "C:\Users\USER\OneDrive\文档\Emprical Finance & predictive analysis\logistic\_regression\_dataset\_ (Orginal File).csv"

Python file

**Assignment 30 - Training and Testing a Logistic Regression Model**

Task:

Fit a logistic regression model to predict the payment status.

Please use 90% of the data to train the model and 10% of the model to test the model.

Content:

* Training Data (90%):

We will use 90% of our datasets to train the logistic regression model.

We will be learning the relationship between the input features (independent variables) and the target variable (payment status, which might be 0 or 1) through this.

* Testing Data (10%):

The remaining 10% of the data will be set aside and not used during the model training phase.

* This testing set is used to evaluate how well the model generalizes to new, unseen data. We put this data into the model after it’s trained to check how accurate its predictions are.

We are doing logistic regression in order to compare our actual data with the predictions of payment status.

* If the model does well on this unseen data, it suggests that it has generalized well and is likely to perform well when faced with new data in the real world.
* If the model performs poorly on the test data, it might mean that the model has overfitted the training data and might not generalize well to new, unseen data.

And according to our observations, the second statement is true as the model performs poorly because it has overfitted the training data.

Execution in Python:

STEP 1:

Import necessary libraries

import pandas as pd

import statsmodels.api as sm

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

STEP 2:

Load the 29th assignment excel path. Replace the path below with the actual CSV file path

df = pd.read\_csv("")

STEP 3:

Define feature variables (X) and target variable (y).

Assuming 'payment status' is the column to predict.

target\_column = 'PaymentStatus'

X = df.drop(columns=[target\_column])

y = df[target\_column]

STEP 4:

Split the data into train and test sets (90% train, 10% test).

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.10, random\_state=42

)

STEP 5:

Standardize the features (important for logistic regression).

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

STEP 6:

Convert scaled data back to DataFrame to retain column names.

X\_train\_scaled\_df = pd.DataFrame(X\_train\_scaled, columns=X.columns)

X\_train\_scaled\_df = sm.add\_constant(X\_train\_scaled\_df)

# Align indices

y\_train = y\_train.reset\_index(drop=True)

STEP 7:

Fit logistic regression using statsmodels for named summary.

logit\_model = sm.Logit(y\_train, X\_train\_scaled\_df)

result = logit\_model.fit()

STEP 8:

Show detailed regression summary.

print(result.summary())

STEP 9:

Predict and evaluate.

X\_test\_scaled\_df = pd.DataFrame(X\_test\_scaled, columns=X.columns)

X\_test\_scaled\_df = sm.add\_constant(X\_test\_scaled\_df)

y\_pred = result.predict(X\_test\_scaled\_df)

y\_pred\_class = (y\_pred >= 0.5).astype(int)

print("\nConfusion Matrix:")

print(confusion\_matrix(y\_test, y\_pred\_class))

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred\_class))

print("\nAccuracy Score:")

print(accuracy\_score(y\_test, y\_pred\_class))

STEP 10:

Save actual vs. predicted results to a CSV

results\_df = pd.DataFrame({

'Actual': y\_test.values,

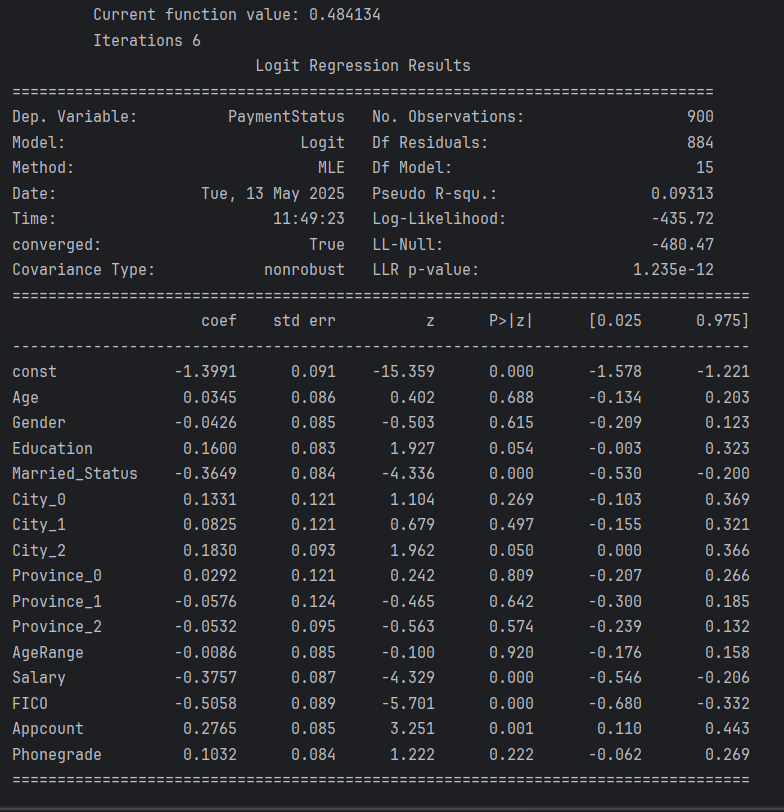
'Predicted': y\_pred\_class

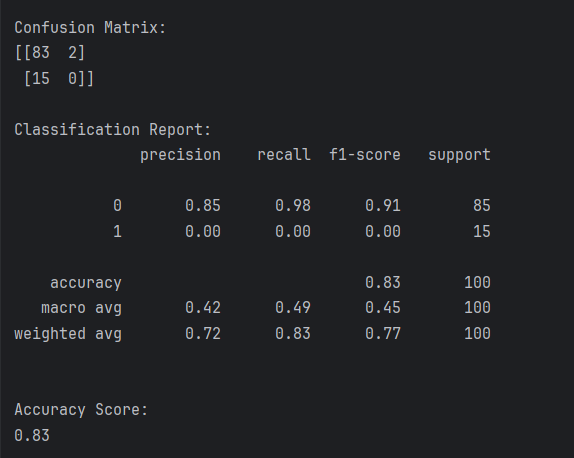
}).reset\_index(drop=True)

output\_path = ("")

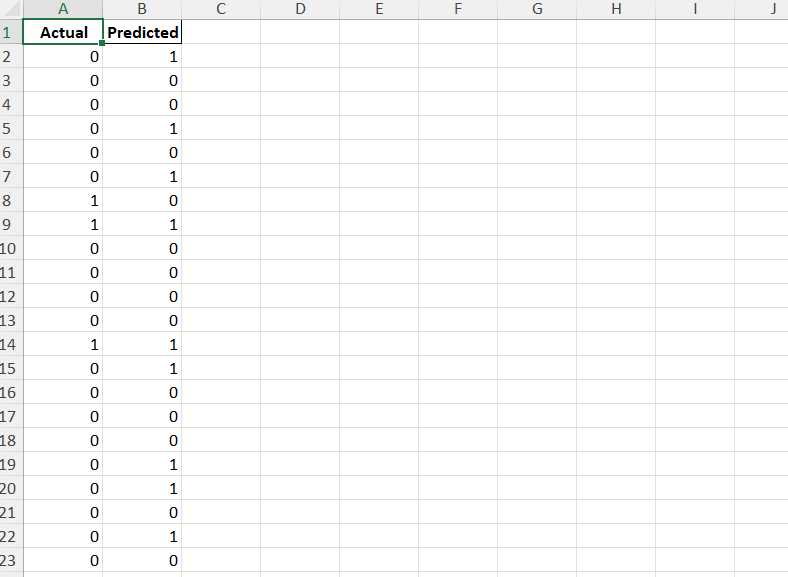
results\_df.to\_csv(output\_path, index=False)

print(f"\nResults saved to: {output\_path}")

**Python results:**



**Exported to excel:**



Literature & software references:

A picture containing circle

Description automatically generated with low confidenceAnderson, D., Sweeney, D., Williams, T., Camm, J., Cochran, J. (2014). Statistics for Business and Economics. (12th ed.). Cengage. Page 725-732

A blue circle with a white x in it

Description automatically generatedExcel path: "C:\Users\USER\OneDrive\文档\Emprical Finance & predictive analysis\Assignment 29.xlsx"

Python file

**Assignment 31 – Odds Ratio**

Task:

Interpret the coefficients using the odds ratio.

Content:

To interpret the impact of each feature on the likelihood of payment status, we calculated the odds ratio by exponentiating the logistic regression coefficients. The odds ratio indicates how a one-unit change in the feature affects the odds of a customer being classified as a non-payer (Payment Status = 1).

The odds are a way to express the likelihood of an event happening compared to it not happening.

Odds=P(Success)/(1-P(Success))

Where: P(Success) is the probability of the event occurring, 1−P(Success) is the probability of the event not occurring.

If P(Success)=0.8,

then: Odds=0.8/ (1−0.8) =0.8/0.2=4

The odds of success are 4 to 1 (meaning success is 4 times more likely than failure).

* Odds ratio measures the impact on the odds of a one-unit increase in only one of the independent variables.

Odds Ratio= Odds1/Odds0

* If odds ratio > 1: the variable **increases** the odds of the outcome
* If odds ratio < 1: the variable **decreases** the odds of the outcome
* If odds ratio = 1: the variable has **no effect**

Execution in Python:

STEP 1:

Import required libraries.

import pandas as pd

import numpy as np

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

STEP 2:

Load the dataset from Assignment 29. Replace the empty string with your actual CSV file path.

df = pd.read\_csv("")

STEP 3:

Define features (X) and target variable (y). 'PaymentStatus' is the binary target we want to predict.

X = df.drop("PaymentStatus", axis=1)

y = df["PaymentStatus"]

STEP 4:

Split the data into training and testing sets (90% train, 10% test)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.10, random\_state=42

)

STEP 5:

Scale the feature data for better model performance.

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

STEP 6:

Fit logistic regression model.

model = LogisticRegression(max\_iter=2000, class\_weight='balanced')

model.fit(X\_train\_scaled, y\_train)

STEP 7:

Interpret model coefficients as odds ratios.

feature\_names = X\_train.columns

coefficients = model.coef\_[0]

# Convert coefficients to odds ratios

odds\_ratios = np.exp(coefficients)

# Create a DataFrame for results

odds\_df = pd.DataFrame({

'Feature': feature\_names,

'Coefficient': coefficients,

'Odds Ratio': odds\_ratios

}).sort\_values(by='Odds Ratio', ascending=False)

print(odds\_df)

STEP 8:

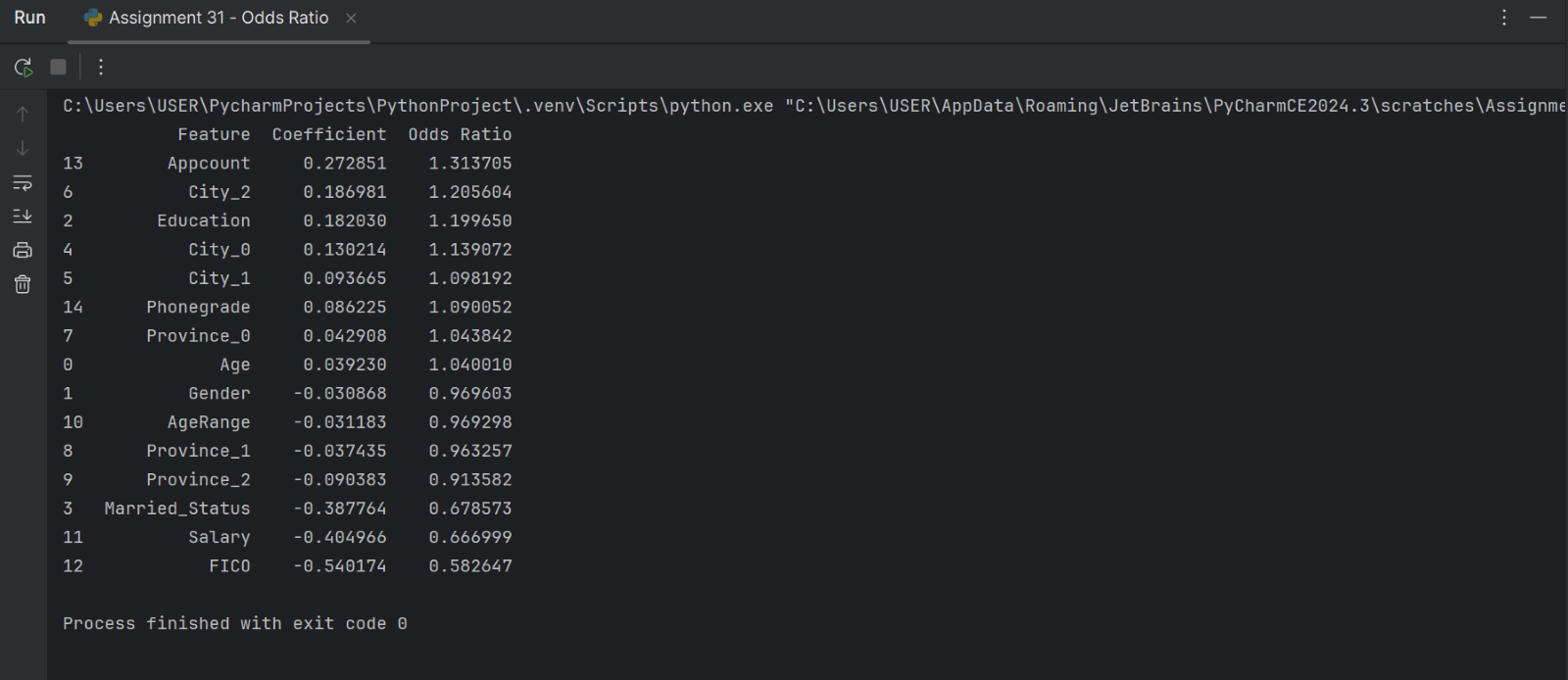
Save odds ratio results to a CSV file for reporting.

# Replace with your desired file path

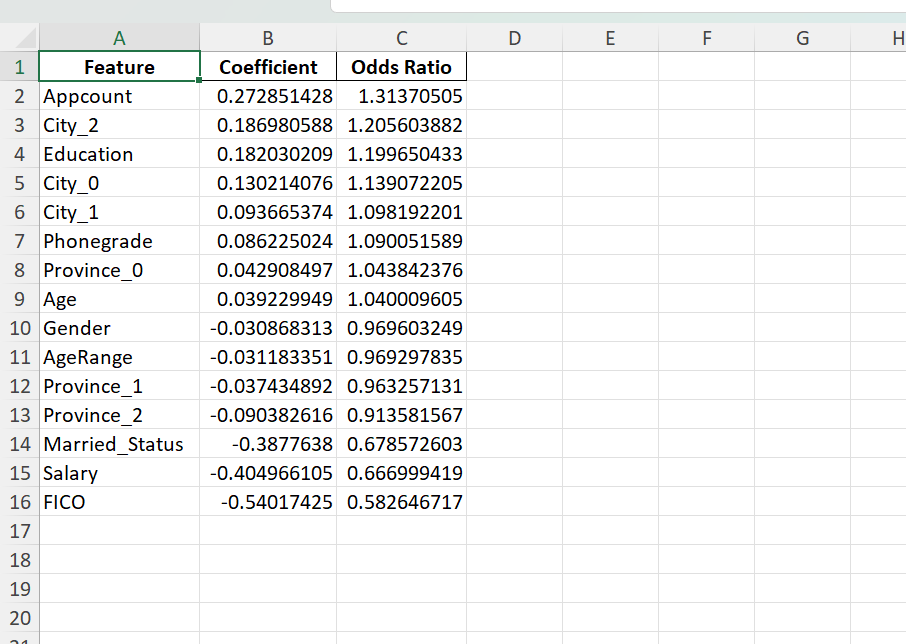
odds\_df.to\_csv("", index=False)

print("Odds ratio results saved successfully.")

**Python results:**



**Exported to excel:**



**Key Findings:**

Features that increase the odds of non-payment:

* Appcount (1.31): Each additional application increases the odds of non-payment by 31%.
* Education (1.20): Higher education level is associated with a 20% increase in odds of non-payment.
* City\_2 (1.21) and City\_0 (1.14): Living in these cities slightly increases the risk of non-payment.

Features that decrease the odds of non-payment:

* FICO Score (0.58): A higher FICO score reduces the odds of non-payment by 42%, making it one of the most influential predictors.
* Salary (0.67): Higher income reduces the odds by approximately 33%.
* Married Status (0.68): Being married appears to lower the risk of non-payment.

Literature & software references:

A picture containing circle

Description automatically generated with low confidenceAnderson, D., Sweeney, D., Williams, T., Camm, J., Cochran, J. (2014). Statistics for Business and Economics. (12th ed.). Cengage. Page 725-732

A blue circle with a white x in it

Description automatically generatedExcel path: "C:\Users\USER\OneDrive\文档\Emprical Finance & predictive analysis\Assignment 29.xlsx"

Python file