

這是關於kaggle上Credit Card Fraud Detection的logestic regression問題
prompt1 :

幫我用python解決logestic regression的credit card fraud detection問題(one problem on kaggle contest)，且務必要包含以下重點:

- 1.按照CRISP-DM框架的六個階段
- 2.要能夠get data from web
- 3.需要有confusion matrix
- 4.accuracy要高(over 60% at least)

由於kaggle上的public score是0.69263, 所以我要求accuracy至少要有0.6

Public Score ⓘ

0.69263

prompt2 :

幫我用python解決logestic regression的credit card fraud detection問題(one problem on kaggle contest)，且務必要包含以下重點:

- 1.按照CRISP-DM框架的六個階段
- 2.要能夠get data from local
- 3.需要有confusion matrix
- 4.accuracy要高(over 60% at least)

我先將網路上的data下載下來(分別是train.csv/test.csv), 並讓程式直接從電腦的file path中讀取這兩個data

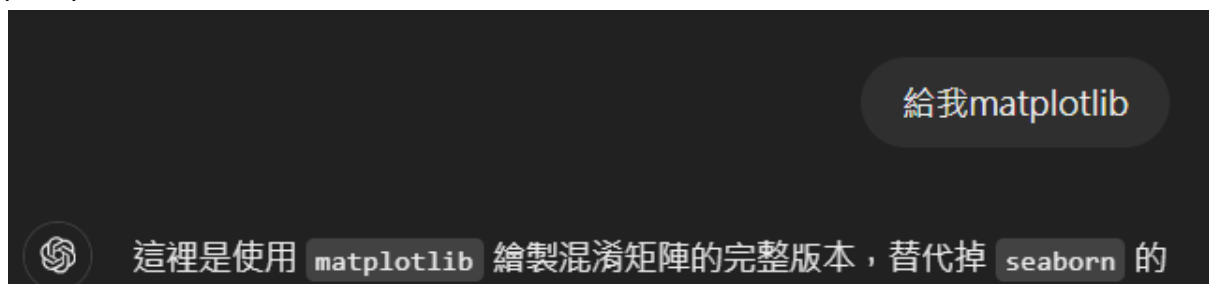
prompt3 :

幫我用python解決logestic regression的credit card fraud detection問題(one problem on kaggle contest)，且務必要包含以下重點:

- 1.按照CRISP-DM框架的六個階段
- 2.要能夠get data from local
- 3.需要有confusion matrix
- 4.accuracy要高(over 60% at least)
- 5.Submission File:For each id in the test set, you must predict a probability for the isFraud variable.

prompt2出來要的結果不是kaggle要求的格式，所以將kaggle上的submission file request再次prompt給chatgpt

prompt4 :



code中用到了"seaborn"套件，我沒安裝此套件所以改用matplotlib，matplotlib主要是用來繪製confusion matrix.

prompt5 :

幫我將輸出的submission.csv和Confusion Matrix儲存在跟train.csv和test.csv同一個file

將submission.csv和confusion matrix圖片存在和train.csv和test.csv同一資料夾

Result :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150000 entries, 0 to 149999
Data columns (total 32 columns):
#   Column                Non-Null Count  Dtype  
---  -
0   id                    150000 non-null  int64   
1   Time                  150000 non-null  float64  
2   feat1                 150000 non-null  float64  
3   feat2                 150000 non-null  float64  
4   feat3                 150000 non-null  float64  
5   feat4                 150000 non-null  float64  
6   feat5                 150000 non-null  float64  
7   feat6                 150000 non-null  float64  
8   feat7                 150000 non-null  float64  
9   feat8                 150000 non-null  float64  
10  feat9                 150000 non-null  float64  
11  feat10                150000 non-null  float64  
12  feat11                150000 non-null  float64  
13  feat12                150000 non-null  float64  
14  feat13                150000 non-null  float64  
15  feat14                150000 non-null  float64  
16  feat15                150000 non-null  float64  
17  feat16                150000 non-null  float64  
18  feat17                150000 non-null  float64  
19  feat18                150000 non-null  float64  
20  feat19                150000 non-null  float64  
21  feat20                150000 non-null  float64  
22  feat21                150000 non-null  float64  
23  feat22                150000 non-null  float64  
24  feat23                150000 non-null  float64  
25  feat24                150000 non-null  float64  
26  feat25                150000 non-null  float64  
27  feat26                150000 non-null  float64  
28  feat27                150000 non-null  float64  
29  feat28                150000 non-null  float64  
30  Transaction_Amount    150000 non-null  float64  
31  IsFraud               150000 non-null  int64   
dtypes: float64(30), int64(2)
memory usage: 36.6 MB
None
Confusion Matrix:
[[44910   2]
 [   86   2]]
Accuracy: 99.80%
ROC AUC Score: 0.74
Submission file saved to: D:/vs code/AIoT_Project/LogisticRegression/submission.csv
Confusion matrix image saved to: D:/vs code/AIoT_Project/LogisticRegression/confusion_matrix.png
```



Credit Card Fraud Detection

Can you detect fraud from customer transactions?



Overview Data Code Models Discussion **Leaderboard** Rules Team Submissions

Leaderboard

[Raw Data](#)[Refresh](#)

YOUR RECENT SUBMISSION

**submission.csv**

Submitted by zhanweiS · Submitted 30 minutes ago

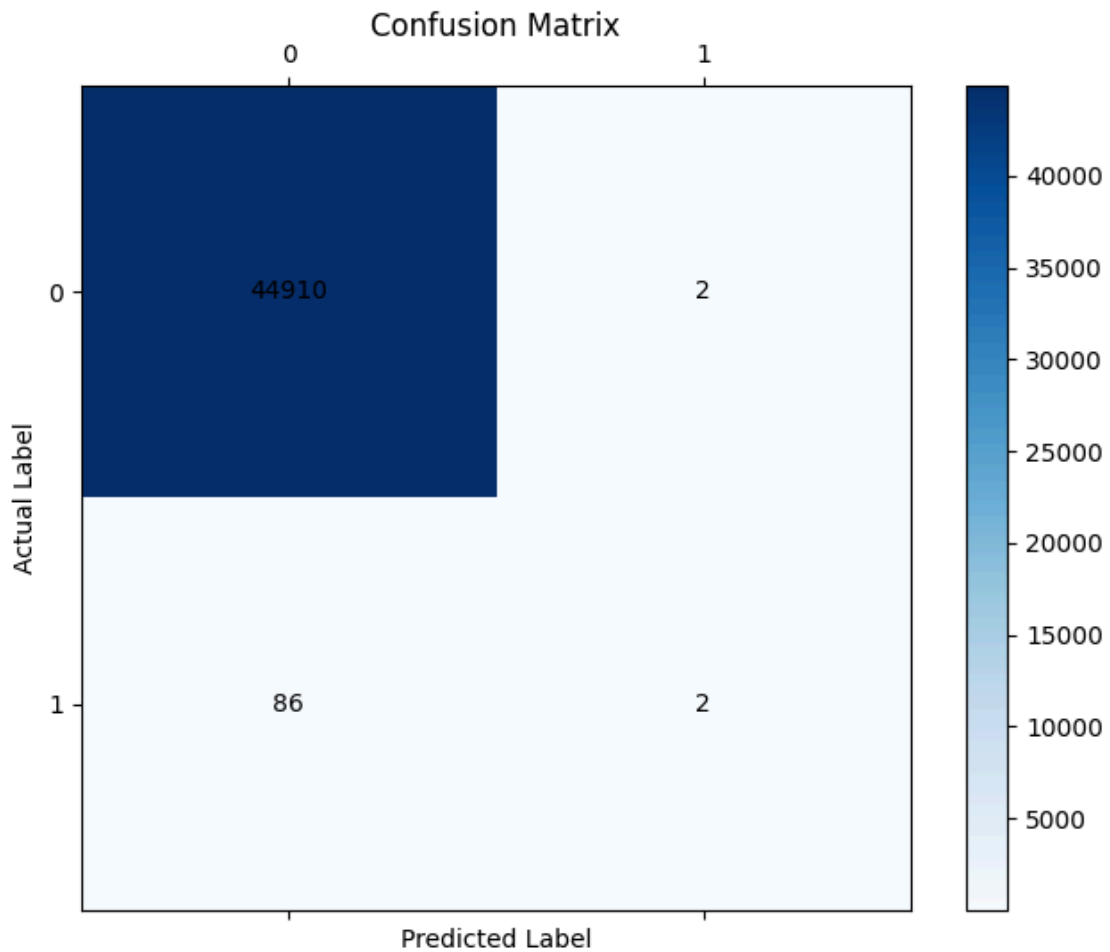
Score: 0.60039

Public score: 0.69263

[Jump to your leaderboard position](#)

49	▼ 5	AGNIVA BHATTACHARYA		0.62201	11	7mo
50	—	ansh Joseph		0.57961	8	7mo

由於這項contest已經結束，這是屬於late submission所以不能參與leader board的排名，但大概是50名左右。



```
Confusion Matrix:  
[[44910    2]  
 [    86    2]]  
Accuracy: 99.80%  
ROC AUC Score: 0.74
```

我的Accuracy高達99.8%，不確定是否正確，於是詢問chatgpt

你所獲得的高準確率（99.8%）很可能是因為數據集中正例（詐欺交易）和負例（正常交易）的不平衡所導致的，也就是說大部分交易是正常交易，而詐欺交易非常少見。因此，模型傾向於將大部分交易預測為正常交易，從而獲得看似很高的準確率，但實際上模型可能對詐欺交易的識別效果並不好。

從你提供的混淆矩陣來看，數據不平衡導致的結果是合理的，但並不代表模型的表現足夠好，尤其是在檢測少數類別（即詐欺交易）時。這樣的混淆矩陣在不平衡數據集中很常見。

讓我們具體分析這個混淆矩陣：

lua

複製程式碼

```
[[44910, 2],  
 [ 86, 2]]
```

- **44910**：模型正確預測了 44910 筆正常交易（真正類別為 0 的正確預測，True Negatives）。
- **2**：模型正確預測了 2 筆詐欺交易（真正類別為 1 的正確預測，True Positives）。
- **86**：模型錯誤地將 86 筆詐欺交易預測為正常交易（真正類別為 1 的錯誤預測，False Negatives）。
- **2**：模型錯誤地將 2 筆正常交易預測為詐欺交易（真正類別為 0 的錯誤預測，False Positives）。

理解混淆矩陣中的問題：

1. 高準確率但低檢測率：混淆矩陣顯示模型對大部分正常交易的預測是正確的（44910/44912），因此你得到了很高的準確率（99.8%）。但模型只正確預測了2筆詐欺交易，漏掉了86筆詐欺交易，這表明模型在檢測詐欺交易上效果不佳。
2. 不平衡數據問題：詐欺交易（少數類別）在數據集中非常少，因此模型傾向於預測為正常交易。這導致模型的高準確率掩蓋了在少數類別上的差劣表現。

改善建議：

- 混淆矩陣是合理的：這是因為不平衡數據集的特性導致的，但它顯示了模型對詐欺交易檢測的低效。這種高準確率主要是由於模型預測了大量的正常交易（多數類別）。
- 進一步改進：你可以考慮處理數據不平衡問題（例如過採樣、下採樣或使用加權模型）來提高模型對詐欺交易的檢測能力。使用精確率、召回率和 F1 分數進行模型評估也會更具體地反映模型在不平衡數據上的性能。

總結來說，雖然混淆矩陣是合理的，但它顯示出模型在檢測詐欺交易（少數類別）上的不足。

雖然confusion matrix合理，但此分析模型對於想要檢測的數據量少的特定數據可能較為不足，若是增加更多的是別feature或是修改recall的話，可能會更好。

```
Confusion Matrix:  
[[44910    2]  
 [   86    2]]  
Accuracy: 99.80%  
ROC AUC Score: 0.74
```

我的ROC AUC Score是0.74

你的模型的 **ROC AUC Score = 0.74**，這表示模型能夠在約 74% 的時間內正確區分正類和負類樣本。這是一個不錯的結果，尤其是在處理不平衡數據集時，能夠表明模型有一定的區分能力，但仍有改進空間。

總結來說，ROC AUC Score = 0.74 表示你的模型有一定的區分能力，但仍可以通過數據處理、模型優化等方法來進一步提升。