PYTHON PROJECT

How can you predict, on the basis of various parameters, whether a visitor will contribute to the website's sales?

- Our dataset
- Chart interpretations
- Prediction models
- Bonuses

OUR DATASET

OUR DATASET

Onlne Shoppers Intention

Overview

- 12,330 sessions represented by feature vectors.
- Each session corresponds to a different user within a 1-year period.

Objective

- Designed to avoid bias towards specific campaigns, user profiles, special days, or periods.
- Ensures a diverse and unbiased representation for comprehensive analysis.

CONTEXT

Objective : Maximize income

In-depth analysis of user behavior to optimize the experience, adjust campaigns, and increase conversion rates to achieve our financial growth goals.

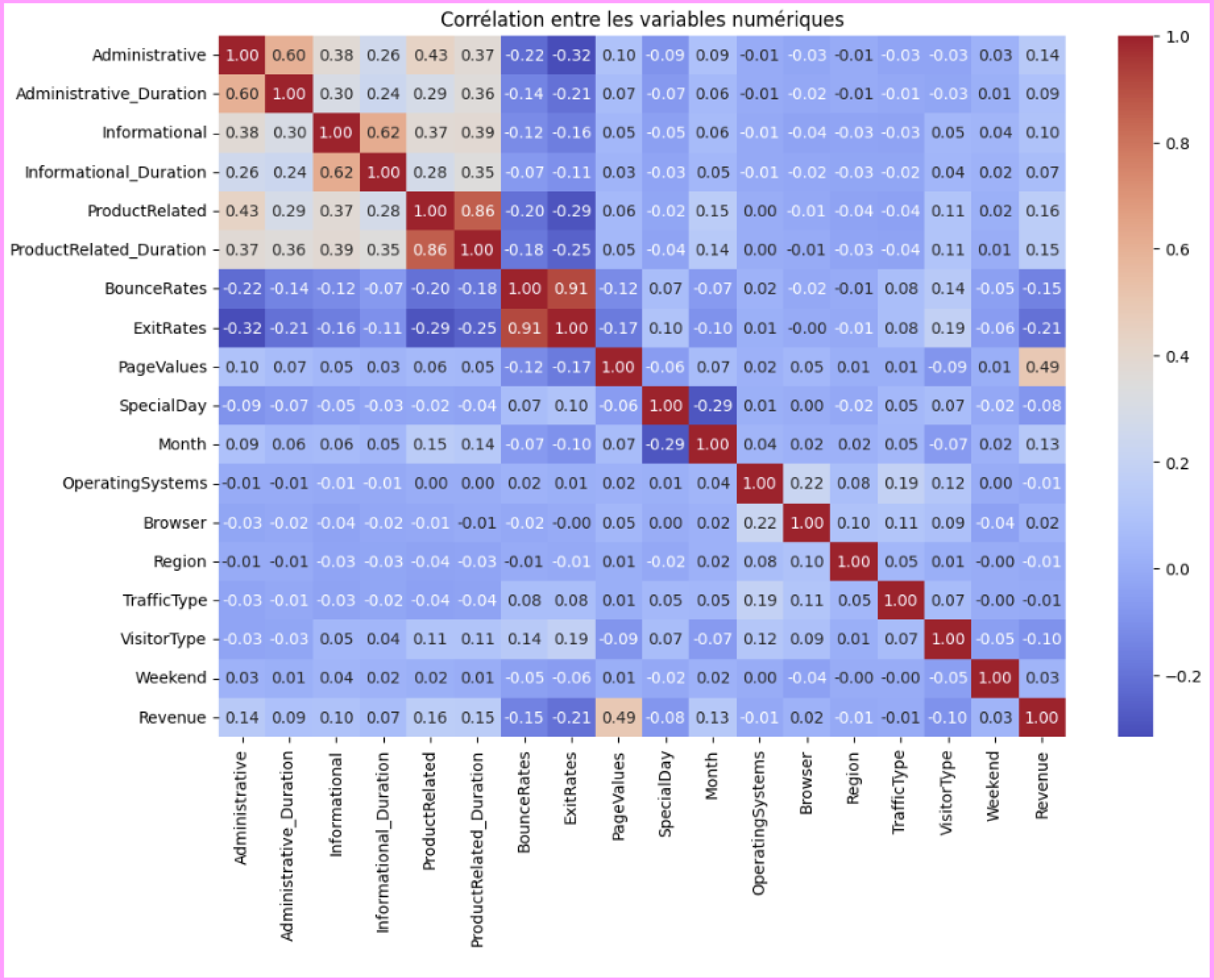
DATA ENCODING

Region : France, Malaysia, Italy, Spain, Germany, England, Poland, Colombia, and Romania

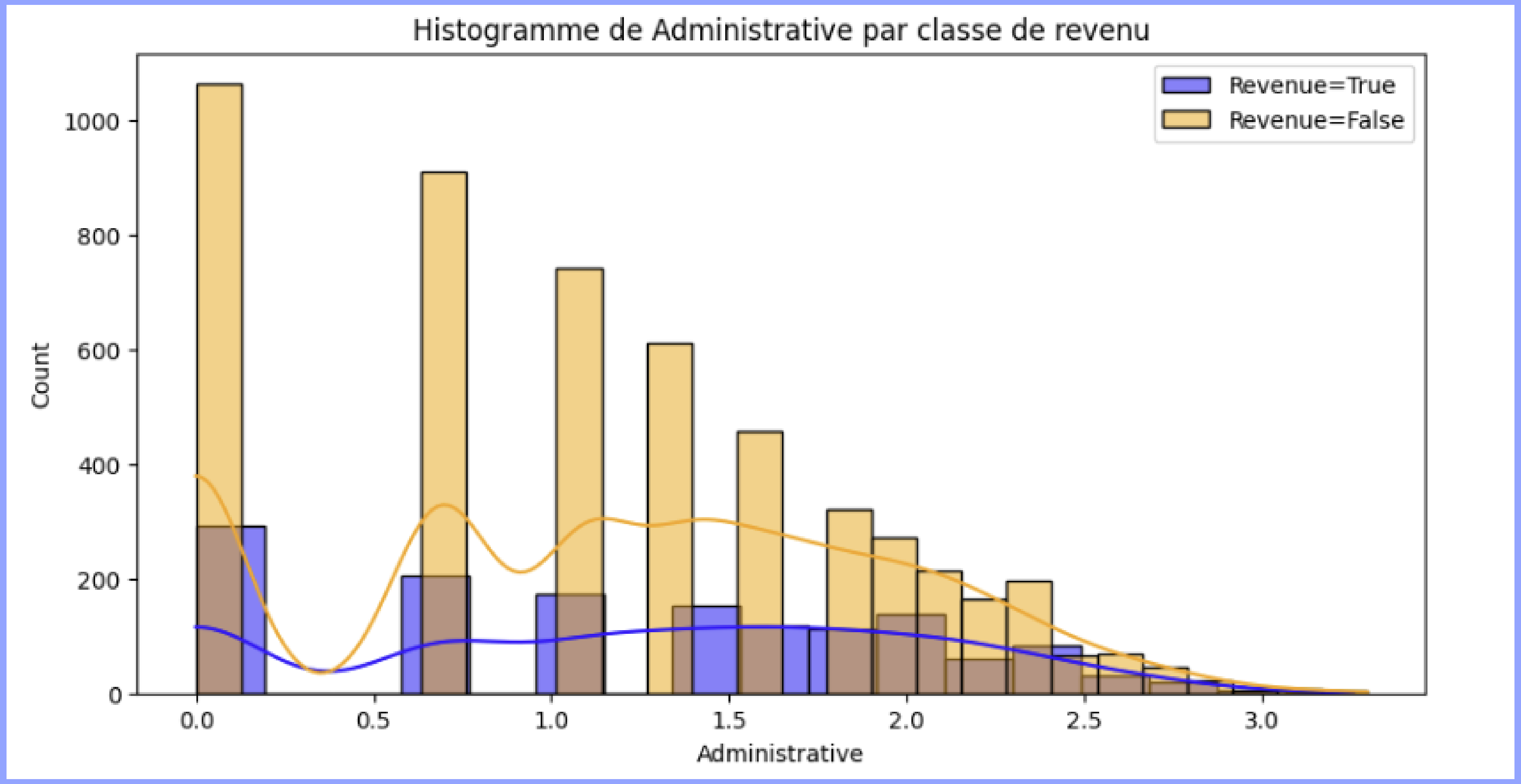
Browser : Chrome, DuckDuckGo, Mozilla Firefox, Microsoft Edge, Safari, Vivaldi, Opera, TOR Browser, Maxthon, Torch Browser, UC Browser, Avast Secure Browser, and Chromium Browser

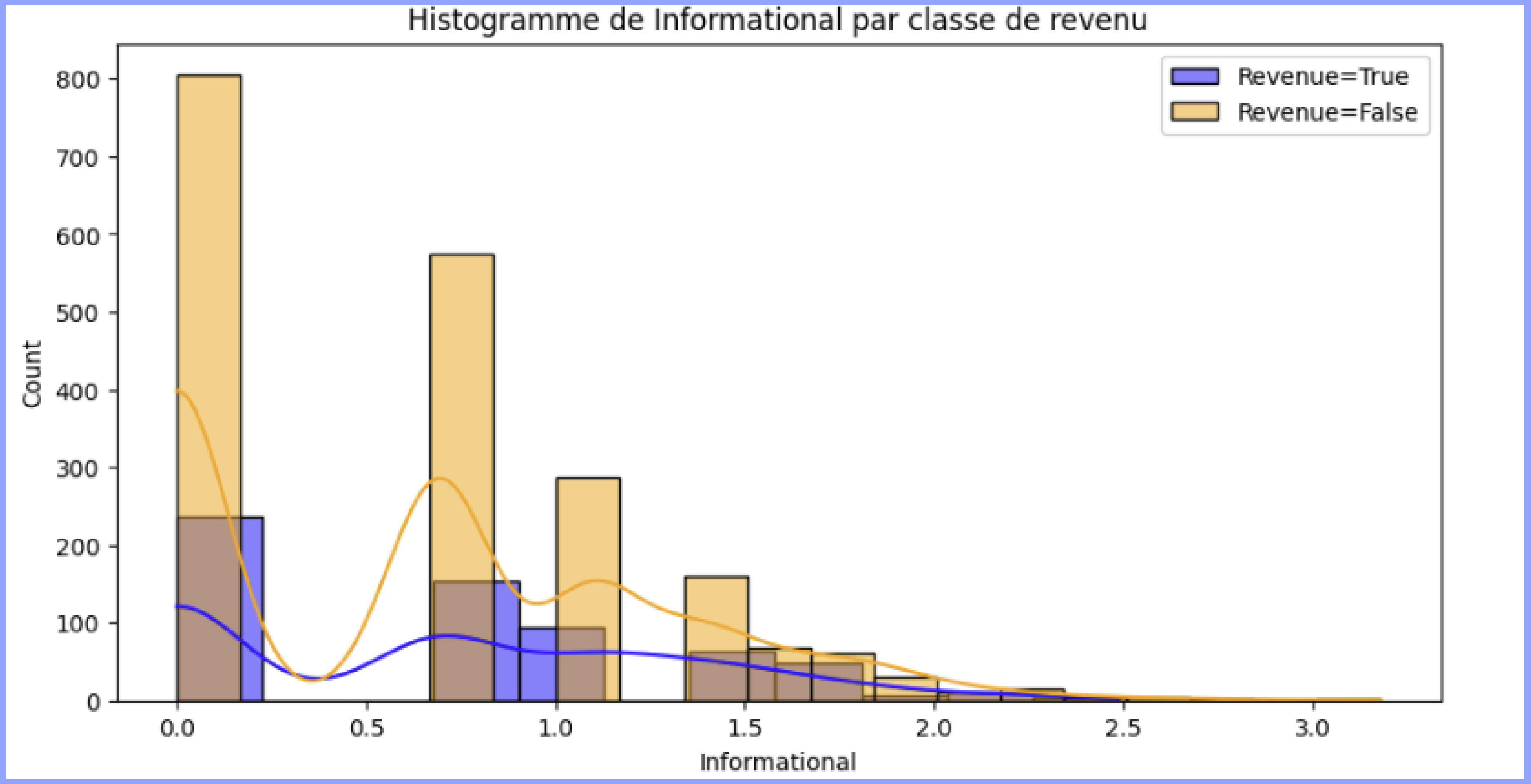
Operating Systems : Windows, MacOS, iOS, Android, GNU, Linux, Unix, and RTX

CHART INTERPRETATIONS

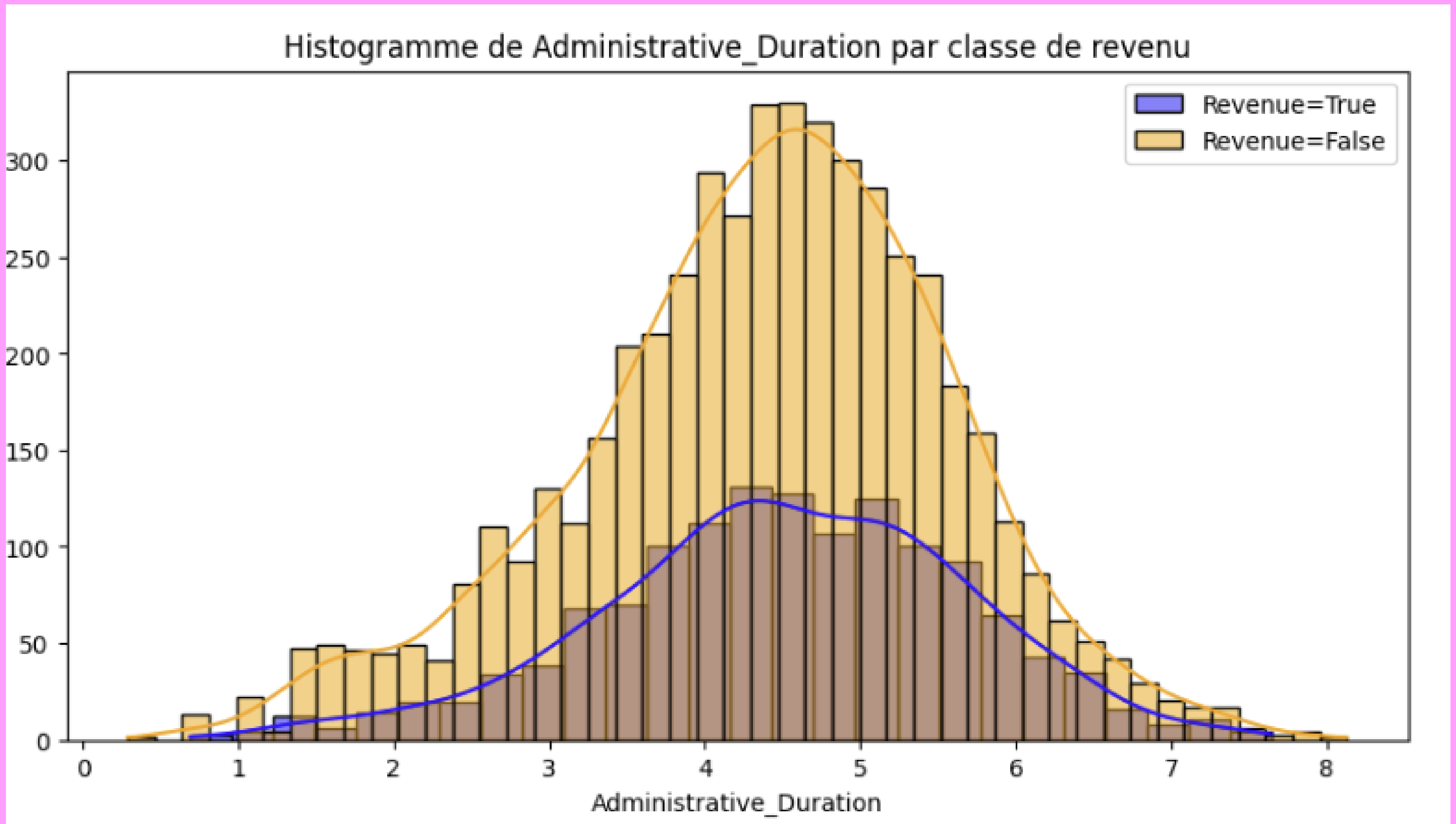


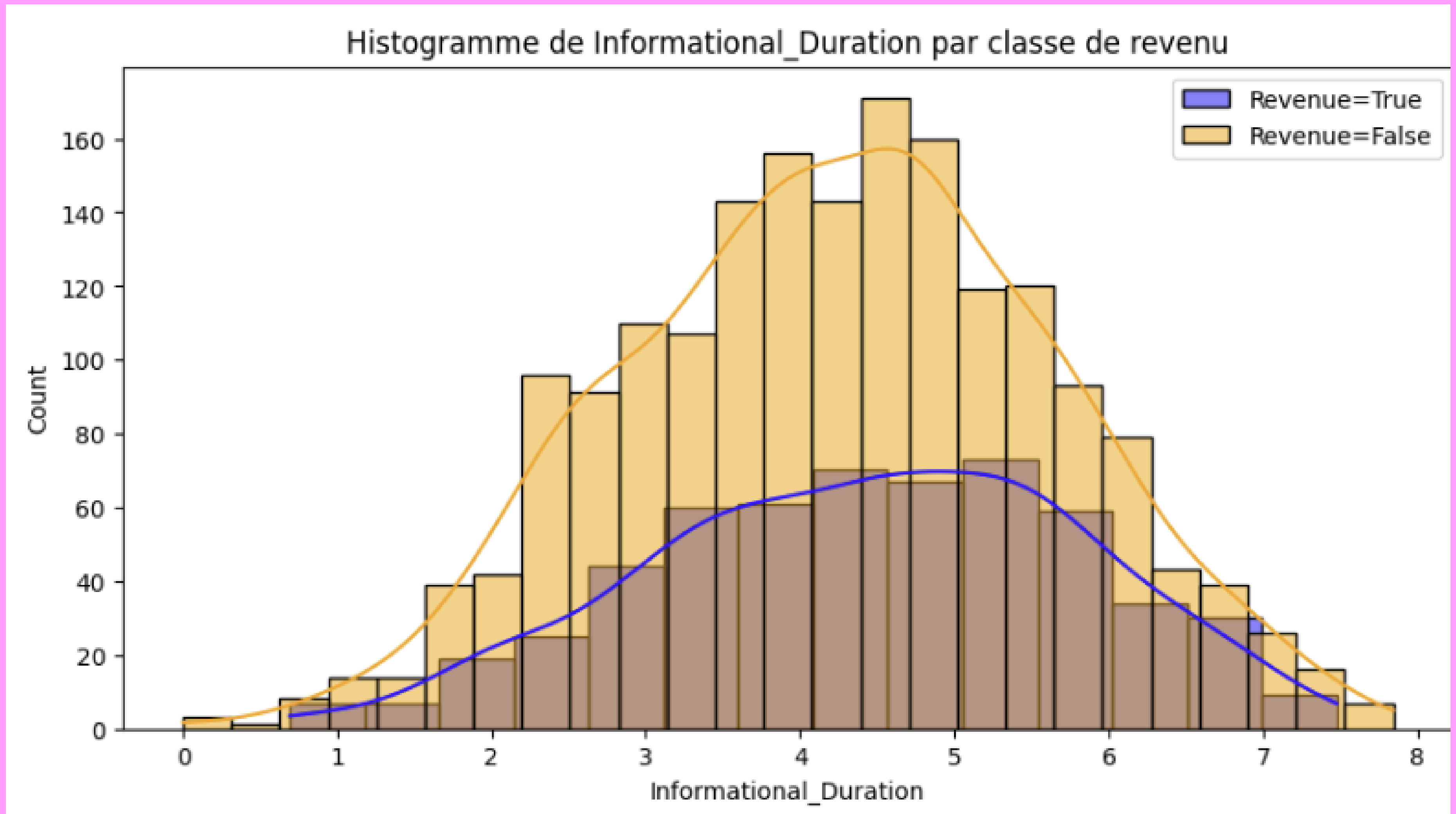
We notice thanks to the correlation matrix that certain parameters such as 'PageValue', 'ExitRates' and the types of pages influence the income



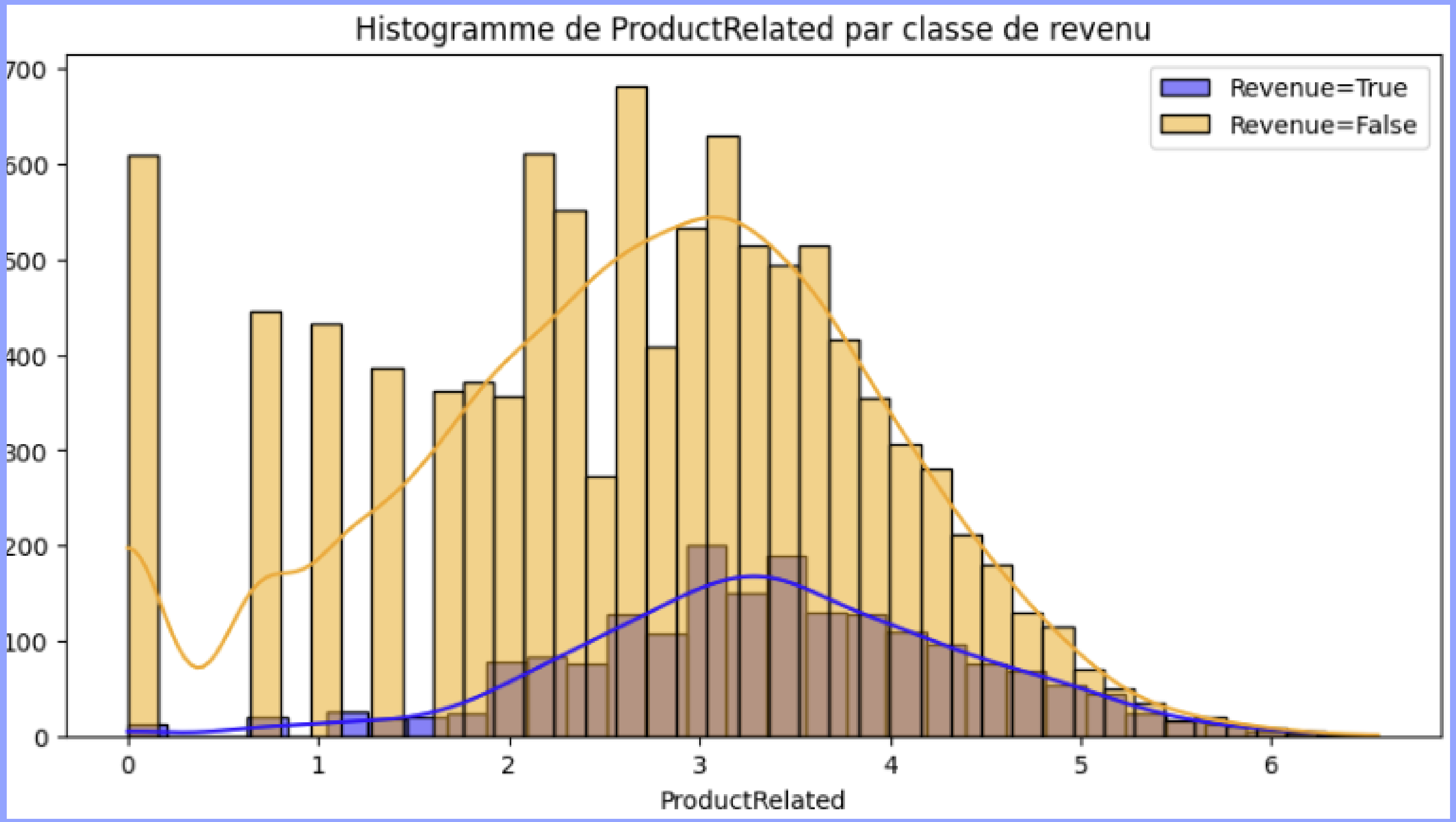


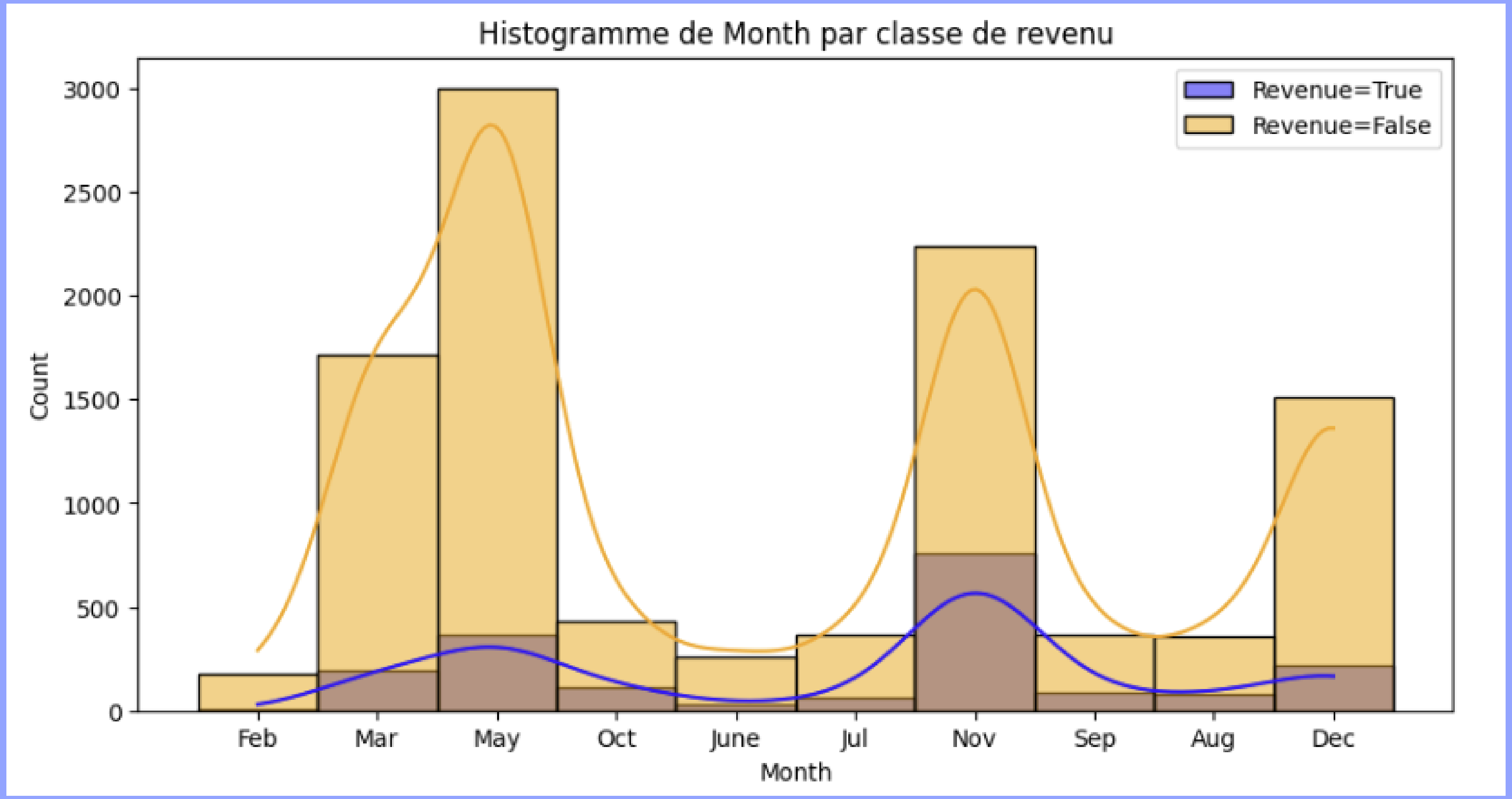
We can think from the histogram that administrative type pages have a greater “chance” of bringing in revenue when they are viewed for a shorter period of time. Indeed, the tallest stick is on the far right at 0. This seems counterintuitive but upon closer inspection we notice that the proportion of revenue = False for the value 0 is 3 or even 4 times higher than that of revenue = True. Finally, we should not rely on the size of the revenue because it is consistent that the reported revenue is more frequent since the number of times when an administrative type page has been consulted very briefly is much higher than when it is consulted more. a long time. On the other hand, if we compare the proportion over the same duration between revenue = True and revenue = False, we see that generally, a visitor will have the longer the visitor stays on the site, the greater the "chance" of contributing positively to income. . For the histogram for the informational page type, we have the same conclusions.





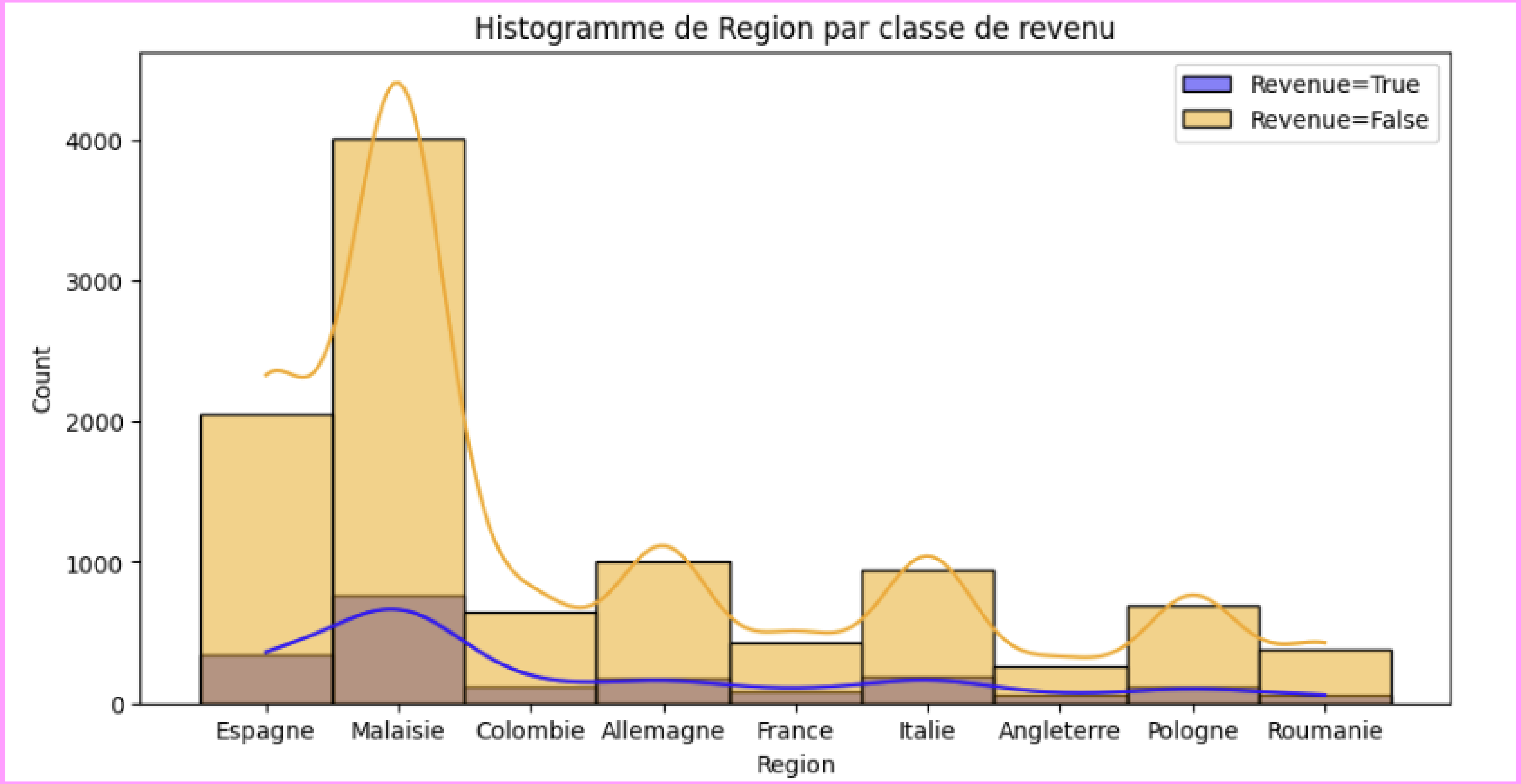
For the histograms for the Administrative_Duration and Informational_Duration page types, we note that in general the probability that a visitor contributes positively to revenue is 50% or a little higher when visitors tend to stay longer.



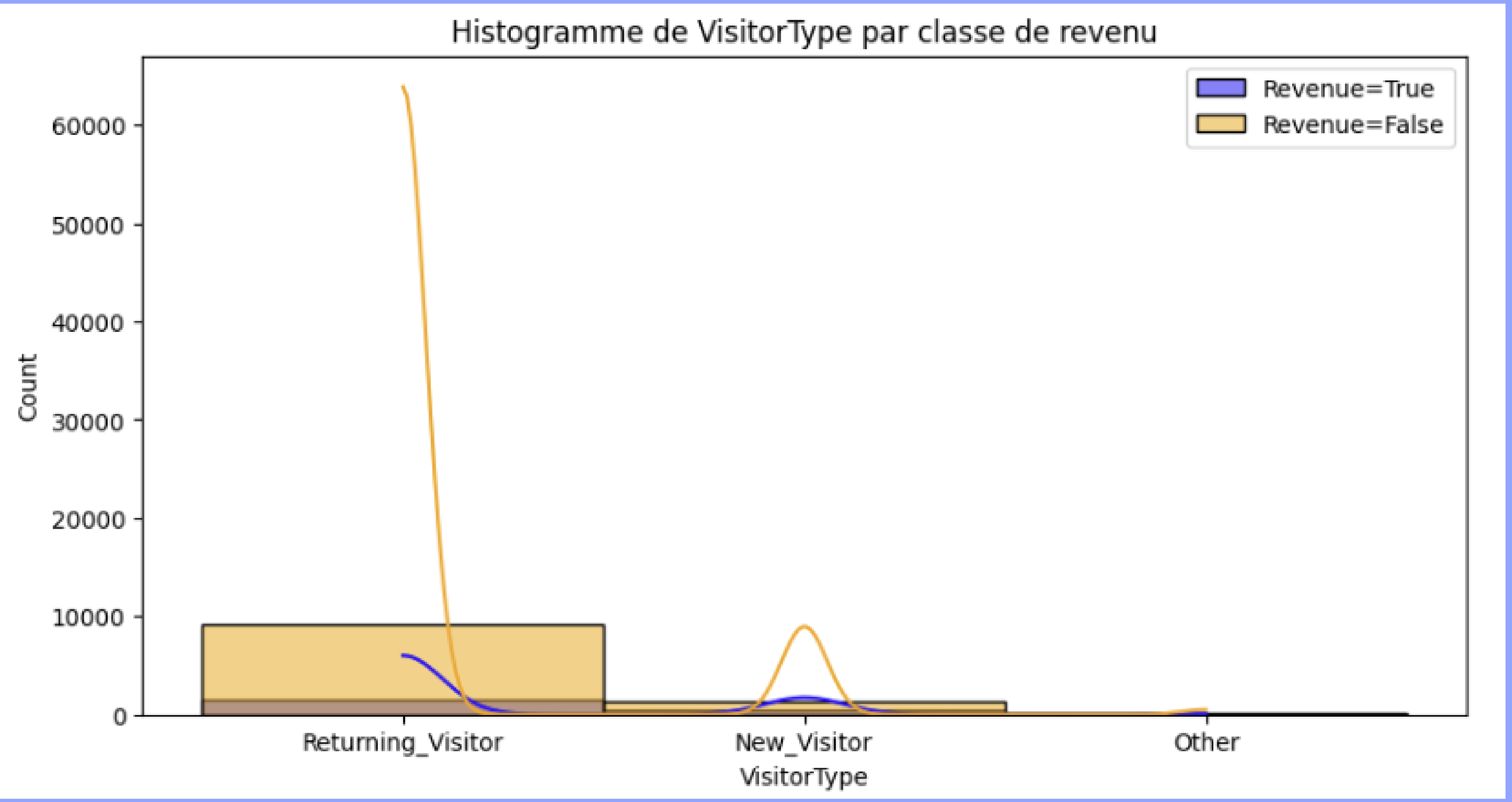


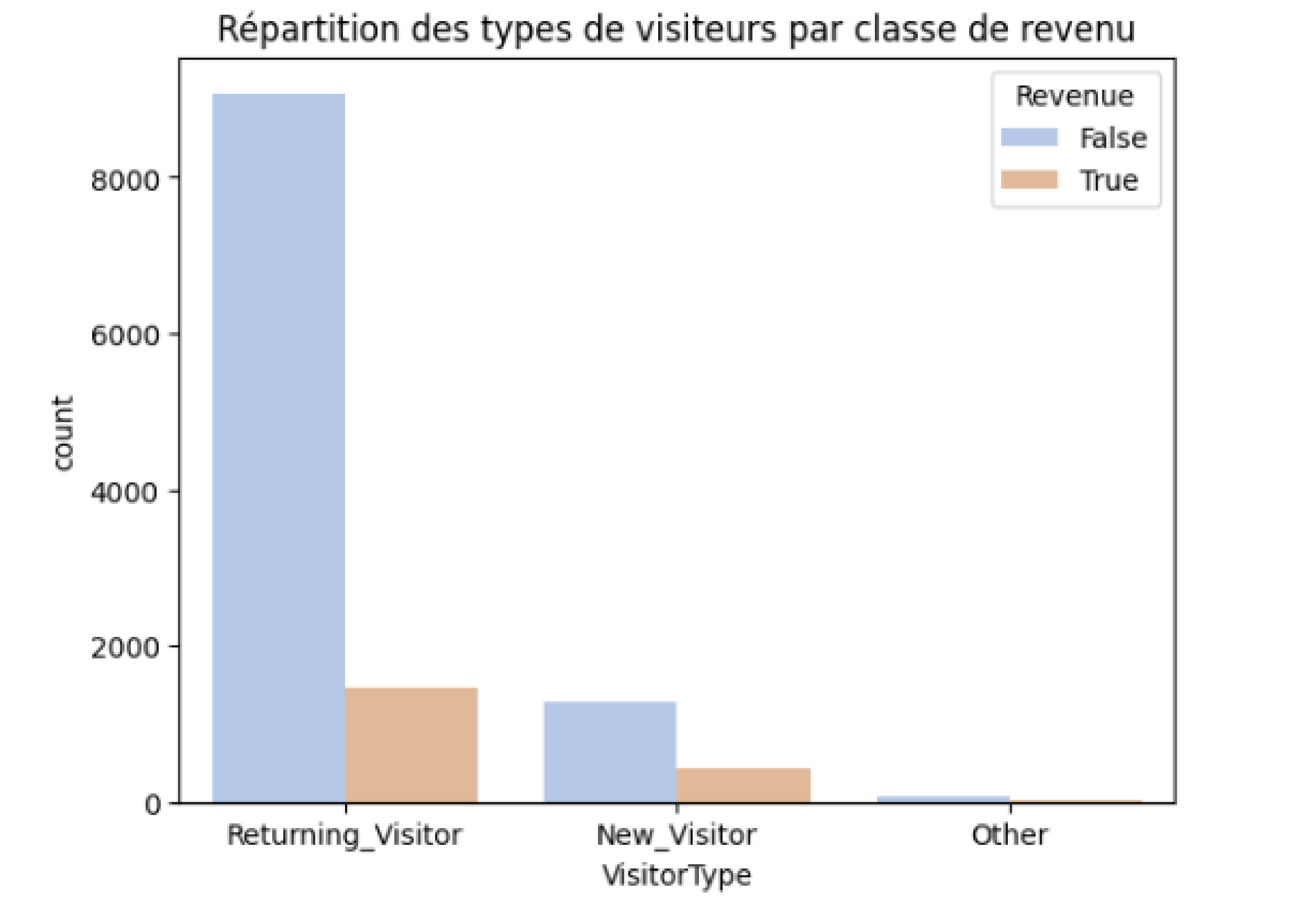
For the histogram of the ProductRelated page type, we see that the more time the visitor spends on the page, the greater the probability of contributing to the income will be, reaching 100% (between 5 and 6), the same for the histogram from ProductRelated.

We notice on the histogram of the months that the month of November is the month which brings in the most income and the proportion of visitors who bring in income out of the total number of visitors for the month is close to 50% unlike the month of March. , May and December which are the second months to bring in income where the ratio is smaller.

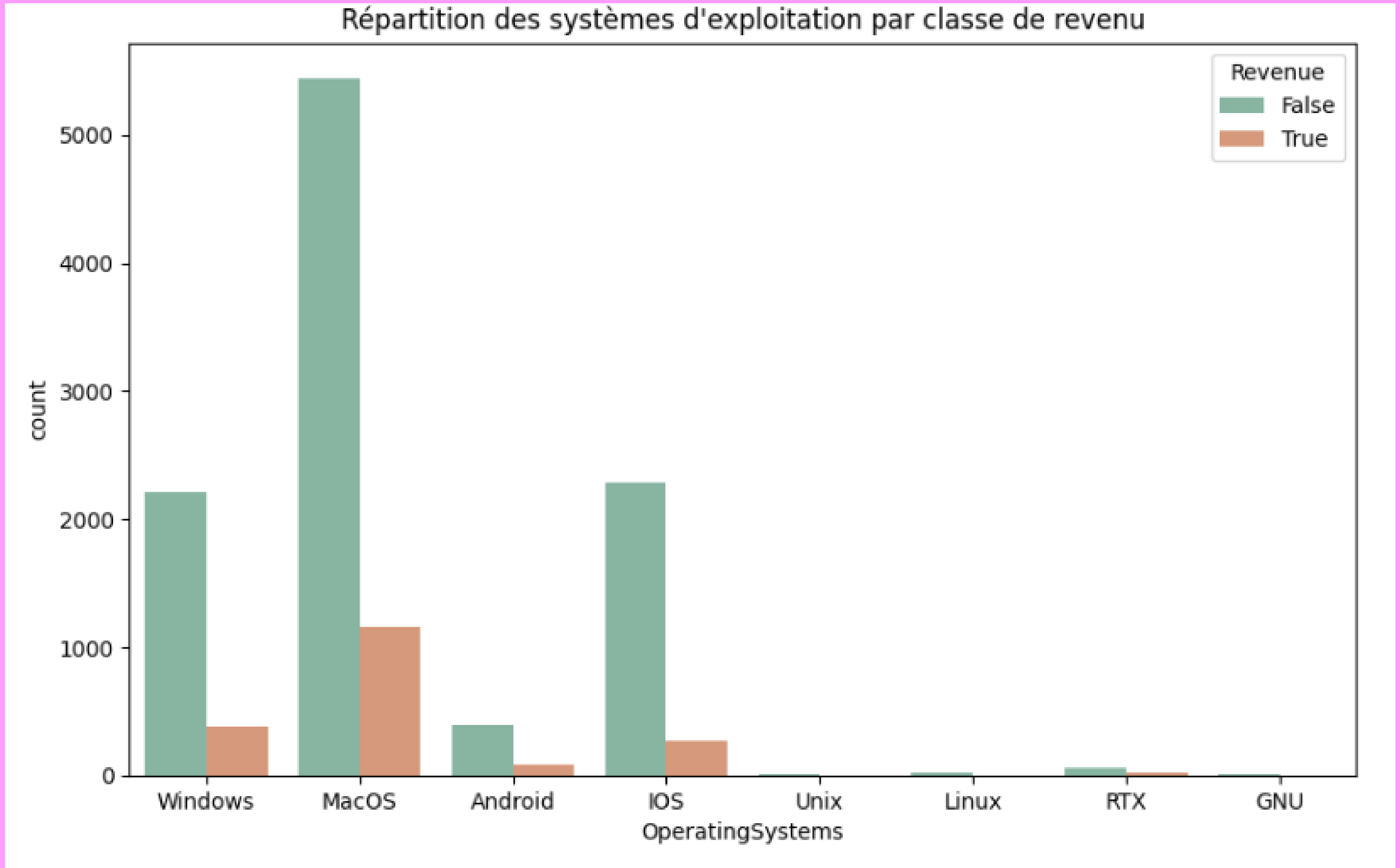


On the histogram of the regions, we see that visitors to Malaysia contribute the most to the income because they are more numerous (just in terms of visitors). We also notice that around 1/4 of visitors contribute to the income, regardless of the country they come from.





We notice that old visitors are those who bring in the most income (twice as much as new visitors) but that the gap between those who bring in income and those who do not bring in income among old visitors is much greater (factor 2 or 3) than that of new visitors. Indeed, it seems that there is a one in two chance that a new visitor will contribute to the income

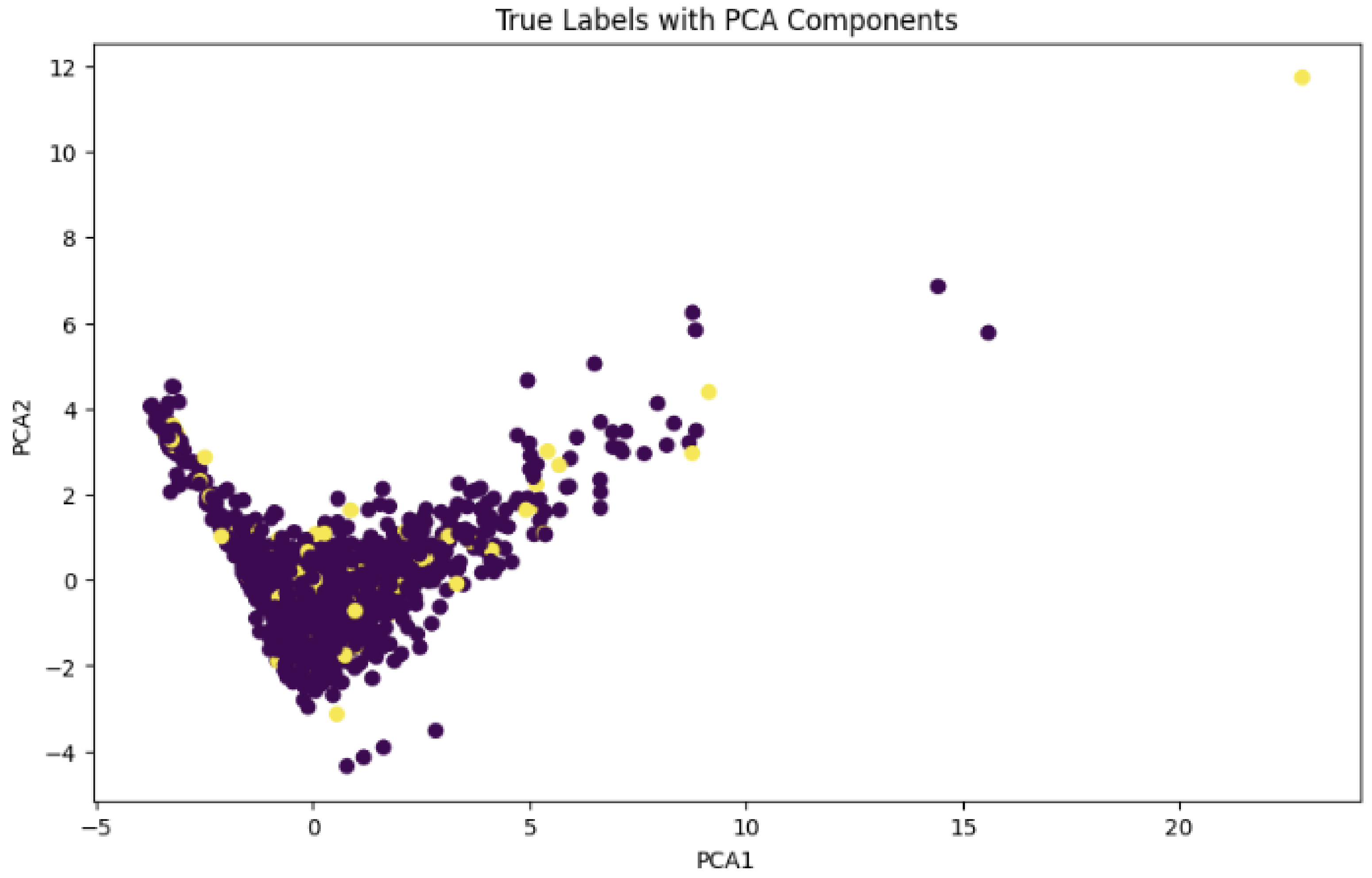


Windows, MacOs and IOS seem to be among the settings that increase revenue. But given that these are the systems used by visitors, it seems intuitive that the diagram would take this turn, so we cannot conclude that using these operating systems would be a factor that would potentially bring in revenue. On the other hand, we can see that the gap between those who report income and those who do not seem to be similar for each operating system (ratio of 4)

PREDICTION MODELS

RandomForestClassifier with PCA

```python
from sklearn.ensemble import RandomForestClassifier

param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

rf_classifier = RandomForestClassifier(random_state=42)

grid_search = GridSearchCV(rf_classifier, param_grid, cv=5)
grid_search.fit(X_train, y_train)

best_rf_model = grid_search.best_estimator_

y_pred = best_rf_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

pca_df = pd.DataFrame(X_test, columns=['PCA1', 'PCA2'])
pca_df['True_Labels'] = y_train

plt.figure(figsize=(10, 6))
scatter = plt.scatter(pca_df['PCA1'], pca_df['PCA2'], c=pca_df['True_Labels'], cmap='viridis')
plt.title('True Labels with PCA Components')
plt.xlabel('PCA1')
plt.ylabel('PCA2')
#plt.legend(*scatter.legend_elements(), title='Labels') # à changer, ça fait bugger
plt.show()


print(f'Best Hyperparameters: {grid_search.best_params_}')

accuracy_rf = accuracy_score(y_test, y_pred)
print(f'Accuracy (Random Forest): {accuracy_rf}')

conf_matrix_rf = confusion_matrix(y_test, y_pred)
print(f'Confusion Matrix (Random Forest):\n{conf_matrix_rf}')

error_rate_rf = 1 - accuracy_rf
print(f'Error Rate (Random Forest): {error_rate_rf}')

class_report_rf = classification_report(y_test, y_pred)
print(f'Classification Report (Random Forest):\n{class_report_rf}')
```




```
Best Hyperparameters: {'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 100}
Accuracy (Random Forest): 0.8422546634225466
Confusion Matrix (Random Forest):
[[2033   22]
 [ 367   44]]
Error Rate (Random Forest): 0.15774533657745338
Classification Report (Random Forest):
              precision    recall  f1-score   support

           0       0.85      0.99      0.91      2055
           1       0.67      0.11      0.18       411

    accuracy                           0.84      2466
   macro avg       0.76      0.55      0.55      2466
weighted avg       0.82      0.84      0.79      2466
```

```
Accuracy (Random Forest): 0.8422546634225466
Confusion Matrix (Random Forest):
[[2033   22]
 [ 367   44]]
```

With RandomForestClassifer, we see that the precision is 0.84 overall but if we look at the confusion matrix, we notice that this model seems to have some difficulty in correctly predicting the visitors who will contribute to the revenue

RandomForestClassifier without PCA

```python
param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

rf_classifier = RandomForestClassifier(random_state=42)

grid_search = GridSearchCV(rf_classifier, param_grid, cv=5)
grid_search.fit(X_train2, y_train2)

best_rf_model = grid_search.best_estimator_

y_pred = best_rf_model.predict(X_test2)


print(f'Best Hyperparameters: {grid_search.best_params_}')

accuracy_rf = accuracy_score(y_test2, y_pred)
print(f'Accuracy (Random Forest): {accuracy_rf}')

conf_matrix_rf = confusion_matrix(y_test2, y_pred)
print(f'Confusion Matrix (Random Forest):\n{conf_matrix_rf}')

error_rate_rf = 1 - accuracy_rf
print(f'Error Rate (Random Forest): {error_rate_rf}')

class_report_rf = classification_report(y_test2, y_pred)
print(f'Classification Report (Random Forest):\n{class_report_rf}')
```

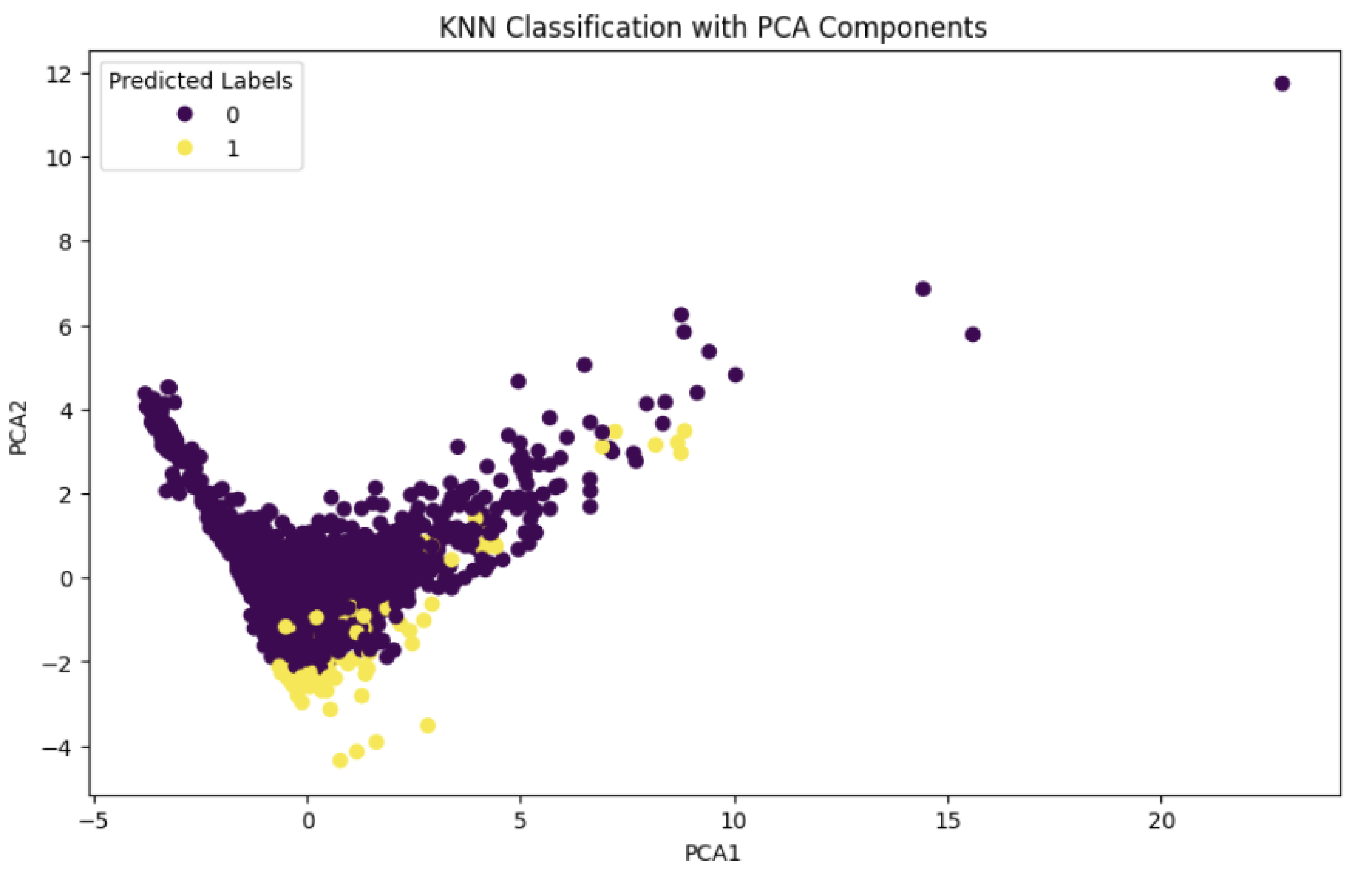

```
Accuracy (Random Forest): 0.8913219789132197
Confusion Matrix (Random Forest):
[[1986   69]
 [ 199  212]]
```


```
Best Hyperparameters: {'max_depth': 10, 'min_samples_leaf':
Accuracy (Random Forest): 0.8913219789132197
Confusion Matrix (Random Forest):
[[1986   69]
 [ 199  212]]
Error Rate (Random Forest): 0.10867802108678026
Classification Report (Random Forest):
              precision    recall  f1-score   support

           0       0.91      0.97      0.94      2055
           1       0.75      0.52      0.61       411

    accuracy                           0.89      2466
   macro avg       0.83      0.74      0.77      2466
weighted avg       0.88      0.89      0.88      2466
```

Unsurprisingly the prediction is better without PCA but this is not sufficient the error is a little less than 1/2 to classify the true negatives

KNeighborsClassifier with PCA

```
from sklearn.neighbors import KNeighborsClassifier

n_neighbors_values = [3, 5, 7, 9]

param_grid = {'n_neighbors': n_neighbors_values}

knn = KNeighborsClassifier()

grid_search = GridSearchCV(knn, param_grid, cv=5)
grid_search.fit(X_train, y_train)

best_knn_model = grid_search.best_estimator_

best_knn_labels = best_knn_model.predict(X_test)

pca_df = pd.DataFrame(X_test, columns=['PCA1', 'PCA2'])
pca_df['Predicted_Labels'] = best_knn_labels

plt.figure(figsize=(10, 6))
scatter = plt.scatter(pca_df['PCA1'], pca_df['PCA2'], c=pca_df['Predicted_Labels'], cmap='viridis')
plt.title('KNN Classification with PCA Components')
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.legend(*scatter.legend_elements(), title='Predicted Labels')
plt.show()


print(f'Best Hyperparameters: {grid_search.best_params_}')

accuracy_knn = accuracy_score(y_test, best_knn_labels)
print(f'Accuracy (KNN): {accuracy_knn}')

conf_matrix_knn = confusion_matrix(y_test, best_knn_labels)
print(f'Confusion Matrix (KNN):\n{conf_matrix_knn}')

error_rate_knn = 1 - accuracy_knn
print(f'Error Rate (KNN): {error_rate_knn}')

class_report_knn = classification_report(y_test, best_knn_labels)
print(f'Classification Report (KNN):\n{class_report_knn}')
```



```
Best Hyperparameters: {'n_neighbors': 9}
Accuracy (KNN): 0.8398215733982157
Confusion Matrix (KNN):
[[1996   59]
 [ 336   75]]
Error Rate (KNN): 0.16017842660178427
Classification Report (KNN):
              precision    recall  f1-score   support

           0       0.86      0.97      0.91      2055
           1       0.56      0.18      0.28       411

    accuracy                           0.84      2466
   macro avg       0.71      0.58      0.59      2466
weighted avg       0.81      0.84      0.80      2466
```

This model seems to be a little less efficient than RandomForestClassifier, in fact its accuracy is a little less great and there is also the same problem in correctly predicting visitors contributing to the revenue

KNeighborsClassifier without PCA

```python
n_neighbors_values = [i for i in range (1,11)]

param_grid = {'n_neighbors': n_neighbors_values}

knn = KNeighborsClassifier()

grid_search = GridSearchCV(knn, param_grid, cv=5)
grid_search.fit(X_train2, y_train2)

best_knn_model = grid_search.best_estimator_

best_knn_labels = best_knn_model.predict(X_test2)


print(f'Best Hyperparameters: {grid_search.best_params_}')

accuracy_knn = accuracy_score(y_test2, best_knn_labels)
print(f'Accuracy (KNN): {accuracy_knn}')

conf_matrix_knn = confusion_matrix(y_test2, best_knn_labels)
print(f'Confusion Matrix (KNN):\n{conf_matrix_knn}')

error_rate_knn = 1 - accuracy_knn
print(f'Error Rate (KNN): {error_rate_knn}')

class_report_knn = classification_report(y_test2, best_knn_labels)
print(f'Classification Report (KNN):\n{class_report_knn}')
```

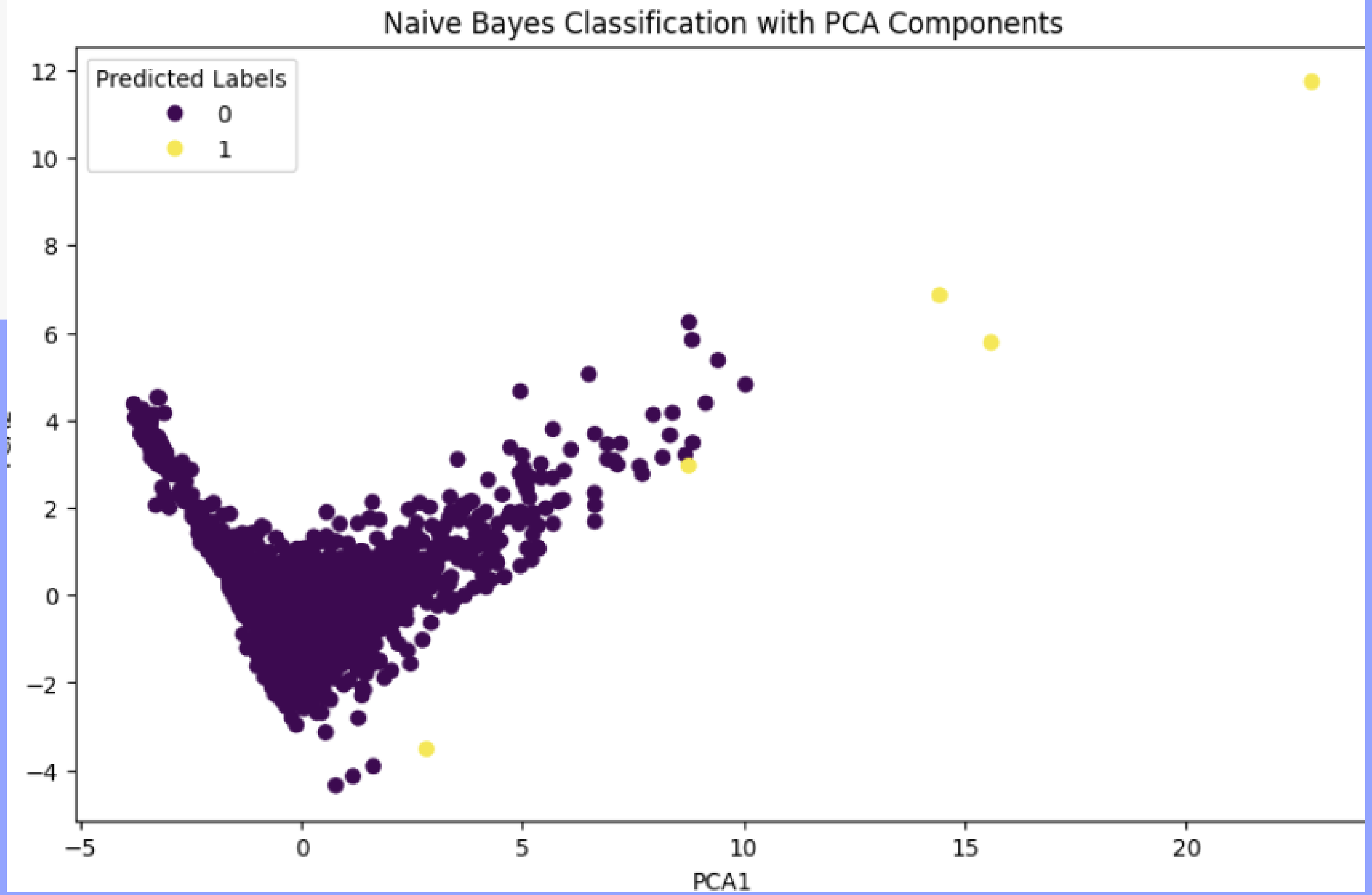
```
Best Hyperparameters: {'n_neighbors': 9}
Accuracy (KNN): 0.8682076236820763
Confusion Matrix (KNN):
[[2008   47]
 [ 278  133]]
Error Rate (KNN): 0.13179237631792373
Classification Report (KNN):
              precision    recall  f1-score   support

           0       0.88      0.98      0.93      2055
           1       0.74      0.32      0.45       411

    accuracy                           0.87      2466
   macro avg       0.81      0.65      0.69      2466
weighted avg       0.86      0.87      0.85      2466
```

Without PCA unsurprisingly, the model becomes more accurate but it remains less efficient than RandomForestClassifier and still struggles for true negatives

GaussianNB with PCA

```python
from sklearn.naive_bayes import GaussianNB

naive_bayes = GaussianNB()

naive_bayes.fit(X_train, y_train)

predicted_labels = naive_bayes.predict(X_test)

pca_df = pd.DataFrame(X_test, columns=['PCA1', 'PCA2'])
pca_df['Predicted_Labels'] = predicted_labels

plt.figure(figsize=(10, 6))
scatter = plt.scatter(pca_df['PCA1'], pca_df['PCA2'], c=pca_df['Predicted_Labels'], cmap='viridis')
plt.title('Naive Bayes Classification with PCA Components')
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.legend(*scatter.legend_elements(), title='Predicted Labels')
plt.show()


accuracy_nb = accuracy_score(y_test, predicted_labels)
print(f'Accuracy (Naive Bayes): {accuracy_nb}')

conf_matrix_nb = confusion_matrix(y_test, predicted_labels)
print(f'Confusion Matrix (Naive Bayes):\n{conf_matrix_nb}')
error_rate_nb = 1 - accuracy_nb
print(f'Error Rate (Naive Bayes): {error_rate_nb}')

class_report_nb = classification_report(y_test, predicted_labels)
print(f'Classification Report (Naive Bayes):\n{class_report_nb}')
```

```
Accuracy (Naive Bayes): 0.8337388483373885
Confusion Matrix (Naive Bayes):
[[2053    2]
 [ 408    3]]
Error Rate (Naive Bayes): 0.1662611516626115146
Classification Report (Naive Bayes):
              precision    recall  f1-score   support

           0       0.83      1.00      0.91      2055
           1       0.60      0.01      0.01       411

    accuracy                           0.83      2466
   macro avg       0.72      0.50      0.46      2466
weighted avg       0.80      0.83      0.76      2466
```



Again, the model makes a lot of false prediction for visitors who contribute to revenue, only 3 correct answers...

GaussianNB without PCA

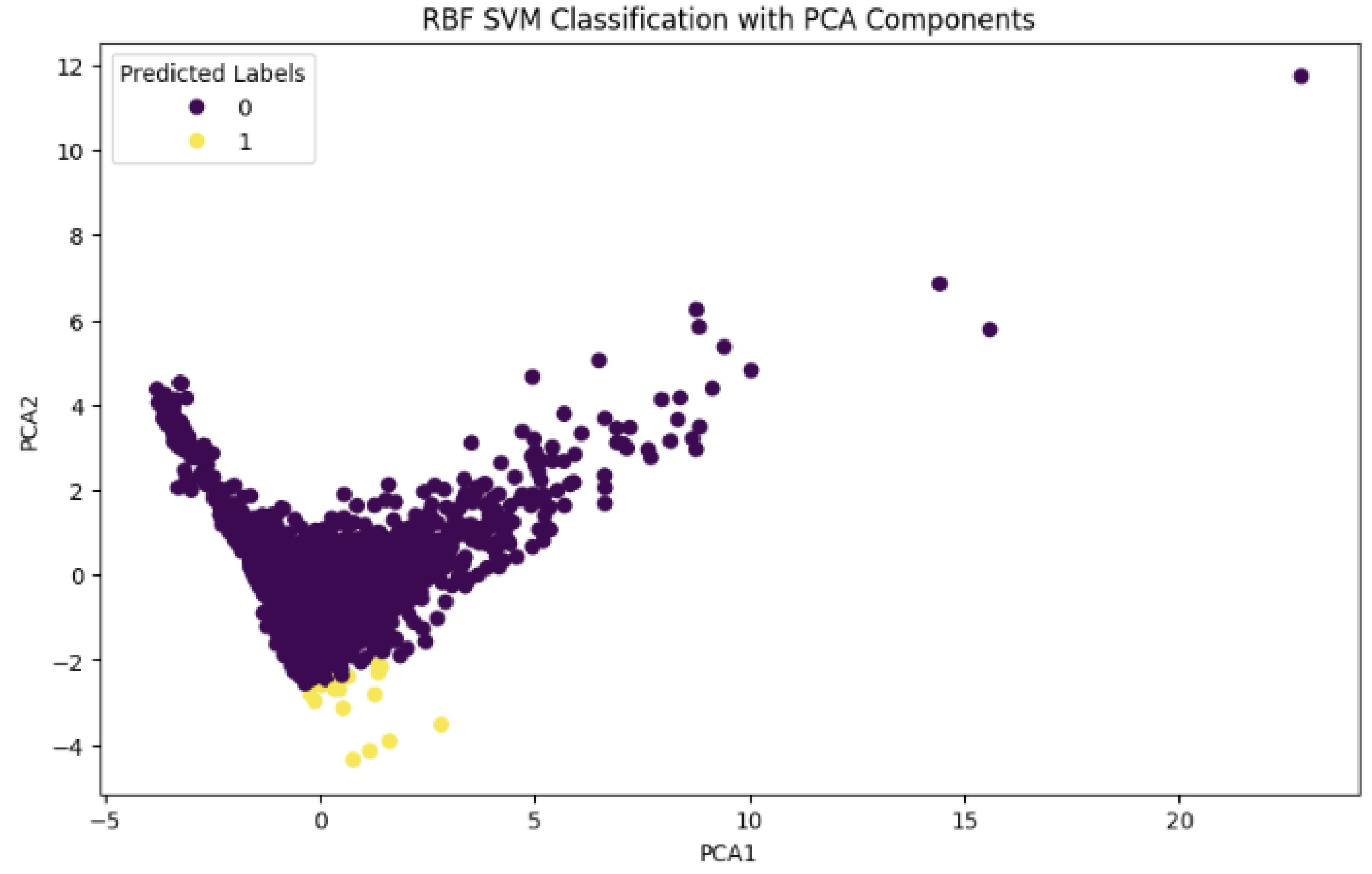
```python
from sklearn.naive_bayes import GaussianNB

naive_bayes = GaussianNB()

naive_bayes.fit(X_train2, y_train2)

predicted_labels = naive_bayes.predict(X_test2)


accuracy_nb = accuracy_score(y_test2, predicted_labels)
print(f'Accuracy (Naive Bayes): {accuracy_nb}')

conf_matrix_nb = confusion_matrix(y_test2, predicted_labels)
print(f'Confusion Matrix (Naive Bayes):\n{conf_matrix_nb}')

error_rate_nb = 1 - accuracy_nb
print(f'Error Rate (Naive Bayes): {error_rate_nb}')

class_report_nb = classification_report(y_test2, predicted_labels)
print(f'Classification Report (Naive Bayes):\n{class_report_nb}')
```

```
Accuracy (Naive Bayes): 0.7850770478507705
Confusion Matrix (Naive Bayes):
[[1654  401]
 [ 129  282]]
Error Rate (Naive Bayes): 0.21492295214922952
Classification Report (Naive Bayes):
              precision    recall  f1-score   support

           0       0.93      0.80      0.86      2055
           1       0.41      0.69      0.52       411

    accuracy                           0.79      2466
   macro avg       0.67      0.75      0.69      2466
weighted avg       0.84      0.79      0.80      2466
```

We have much less prediction error for visitors who will contribute to the income, but on the other hand, we have much more errors for those who will not contribute.

SVC with PCA

```python
from sklearn.svm import SVC

svm_model = SVC(kernel='rbf')

svm_model.fit(X_train, y_train)

predicted_labels = svm_model.predict(X_test)

pca_df = pd.DataFrame(X_test, columns=['PCA1', 'PCA2'])
pca_df['Predicted_Labels'] = predicted_labels

plt.figure(figsize=(10, 6))
scatter = plt.scatter(pca_df['PCA1'], pca_df['PCA2'], c=pca_df['Predicted_Labels'], cmap='viridis')
plt.title('RBF SVM Classification with PCA Components')
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.legend(*scatter.legend_elements(), title='Predicted Labels')
plt.show()


accuracy_svm = accuracy_score(y_test, predicted_labels)
print(f'Accuracy (SVM): {accuracy_svm}')

conf_matrix_svm = confusion_matrix(y_test, predicted_labels)
print(f'Confusion Matrix (SVM):\n{conf_matrix_svm}')

error_rate_svm = 1 - accuracy_svm
print(f'Error Rate (SVM): {error_rate_svm}')

class_report_svm = classification_report(y_test, predicted_labels)
print(f'Classification Report (SVM):\n{class_report_svm}')
```



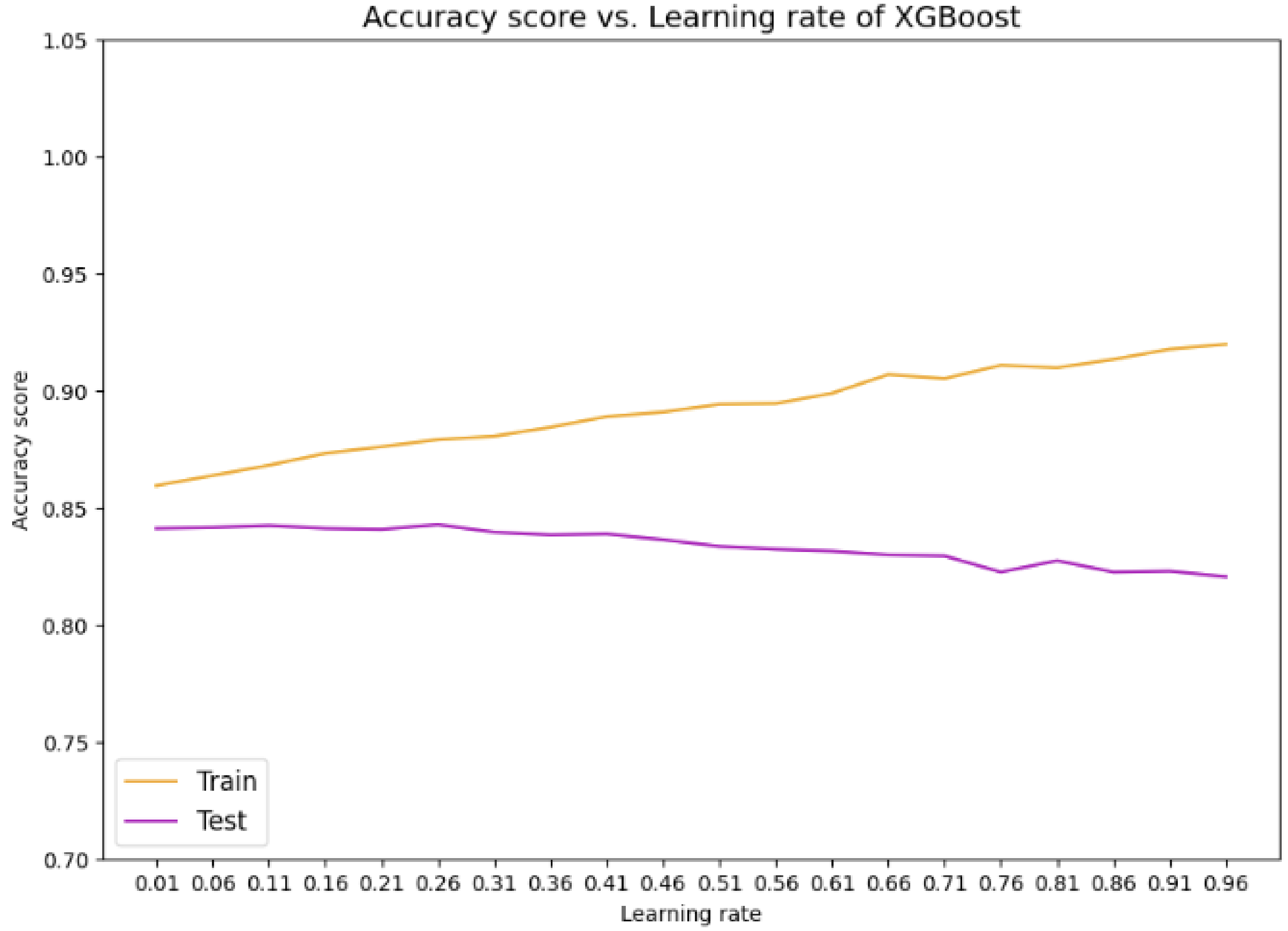

```
Accuracy (SVM): 0.8381995133819952
Confusion Matrix (SVM):
[[2053    2]
 [ 397   14]]
Error Rate (SVM): 0.16180048661800484
Classification Report (SVM):
              precision    recall  f1-score   support

           0       0.84      1.00      0.91      2055
           1       0.88      0.03      0.07       411

    accuracy                           0.84      2466
   macro avg       0.86      0.52      0.49      2466
weighted avg       0.84      0.84      0.77      2466
```

SVC without PCA

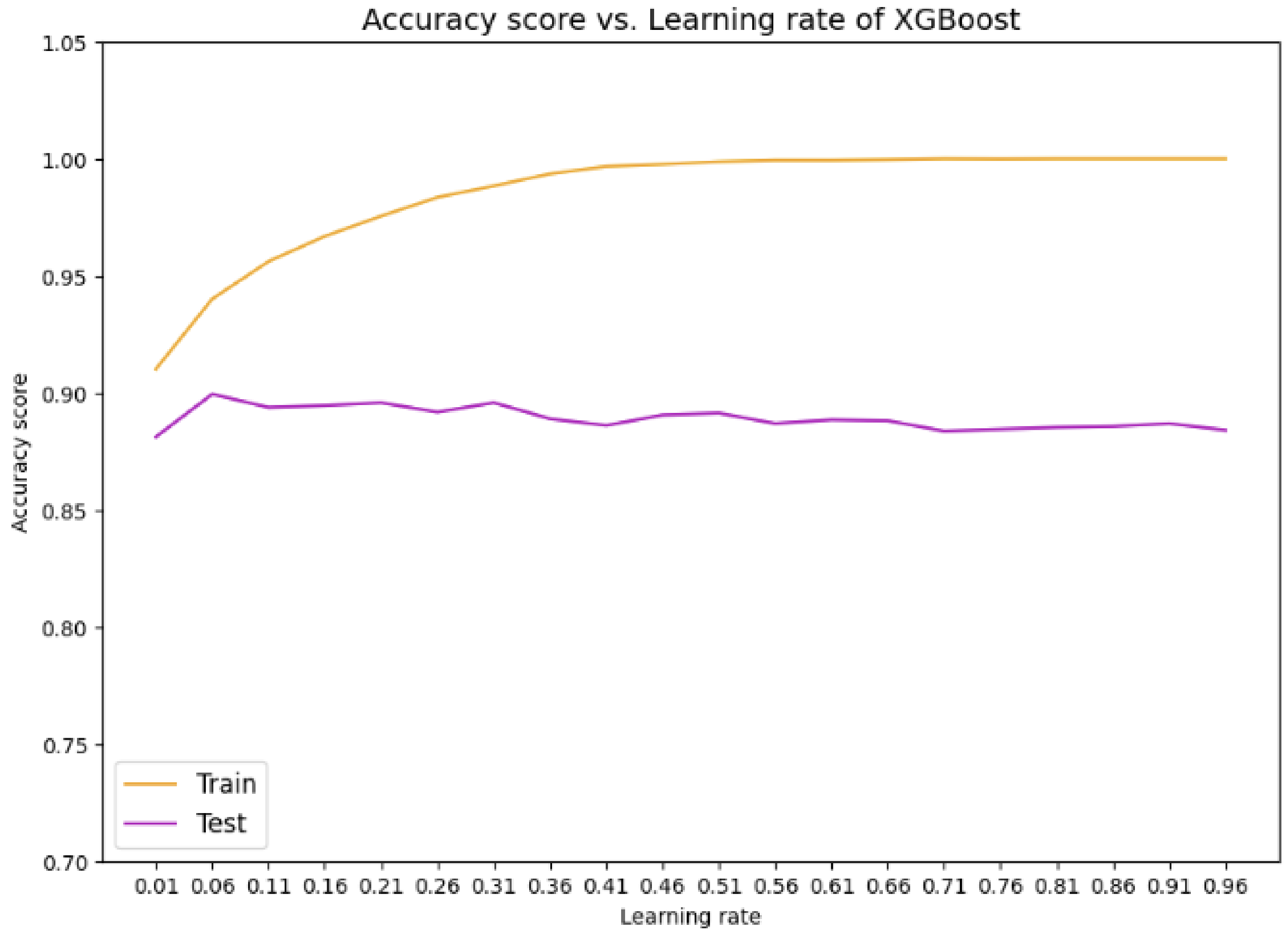
```
from sklearn.svm import SVC

svm_model = SVC(kernel='rbf')

svm_model.fit(X_train2, y_train2)

predicted_labels = svm_model.predict(X_test2)


accuracy_svm = accuracy_score(y_test2, predicted_labels)
print(f'Accuracy (SVM): {accuracy_svm}')

conf_matrix_svm = confusion_matrix(y_test2, predicted_labels)
print(f'Confusion Matrix (SVM):\n{conf_matrix_svm}')

error_rate_svm = 1 - accuracy_svm
print(f'Error Rate (SVM): {error_rate_svm}')

class_report_svm = classification_report(y_test2, predicted_labels)
print(f'Classification Report (SVM):\n{class_report_svm}')
```

```
Accuracy (SVM): 0.878345498783455
Confusion Matrix (SVM):
[[1993   62]
 [ 238  173]]
Error Rate (SVM): 0.12165450121654497
Classification Report (SVM):
              precision    recall  f1-score   support

           0       0.89      0.97      0.93      2055
           1       0.74      0.42      0.54       411

    accuracy                           0.88      2466
   macro avg       0.81      0.70      0.73      2466
weighted avg       0.87      0.88      0.86      2466
```

XGBClassifier with PCA and without PCA




```
Accuracy: 0.8426601784266018
Confusion Matrix:
[[2015   40]
 [ 348   63]]
Error Rate: 0.1573398215733982
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.98      0.91      2055
           1       0.61      0.15      0.25       411

    accuracy                           0.84      2466
   macro avg       0.73      0.57      0.58      2466
weighted avg       0.81      0.84      0.80      2466
```




```
Accuracy: 0.8994322789943228
Confusion Matrix:
[[1972   83]
 [ 165  246]]
Error Rate: 0.10056772100567724
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.96      0.94      2055
           1       0.75      0.60      0.66       411

    accuracy                           0.90      2466
   macro avg       0.84      0.78      0.80      2466
weighted avg       0.89      0.90      0.89      2466
```

CONCLUSION

We have seen that several factors such as page types, months, etc. can have an influence on whether or not a visitor will contribute to revenue.

For the prediction part, we did not find a sufficiently precise model overall. On the other hand, some models such as KNeighborsClassifier for example are effective in predicting with few errors those who will not contribute to the income while GaussianNB without PCA is the most effective model among those tested to predict visitors without a lot of errors which will contribute to income, even if there are frequent errors.

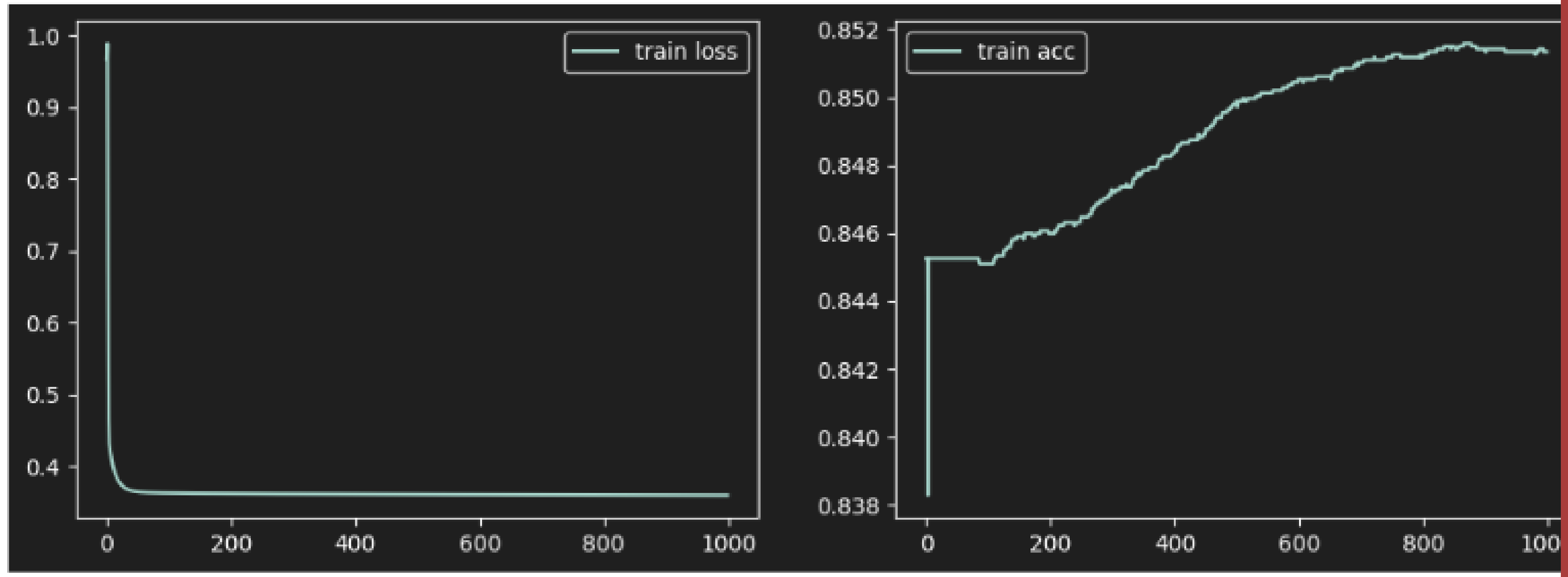
Thus, depending on the needs and our priorities, we will choose the appropriate model.

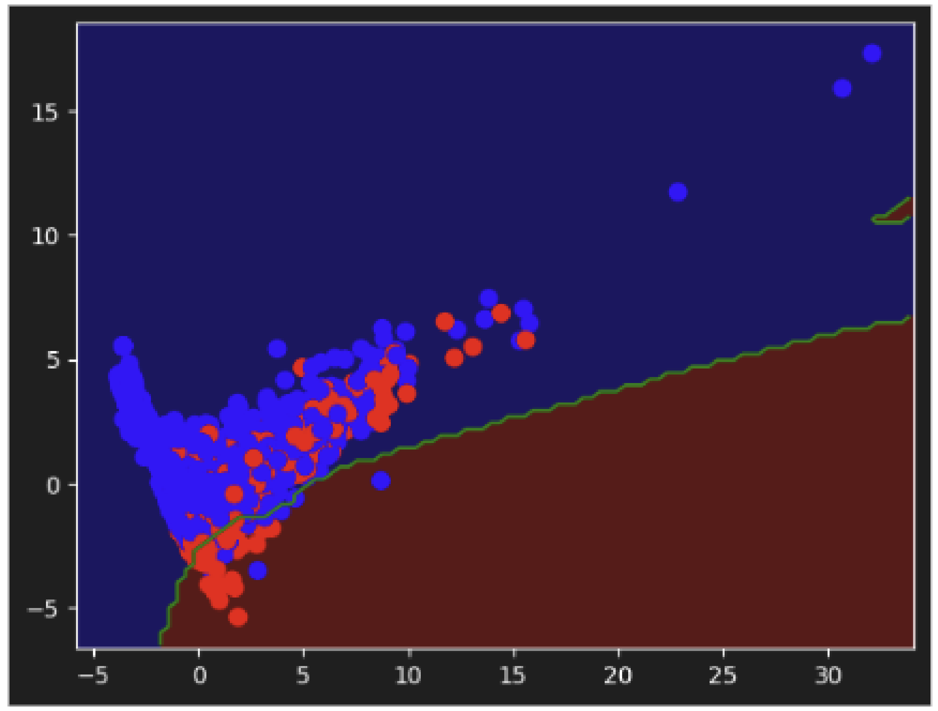
Regression Logistic sans PCA

```
Accuracy: 0.8690186536901865
Confusion Matrix:
 [[2006   49]
 [ 274  137]]
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.98      0.93      2055
           1       0.74      0.33      0.46       411

    accuracy                           0.87      2466
   macro avg       0.81      0.65      0.69      2466
weighted avg       0.86      0.87      0.85      2466

Mean Squared Error:
 0.13098134630981345
```

BONUS

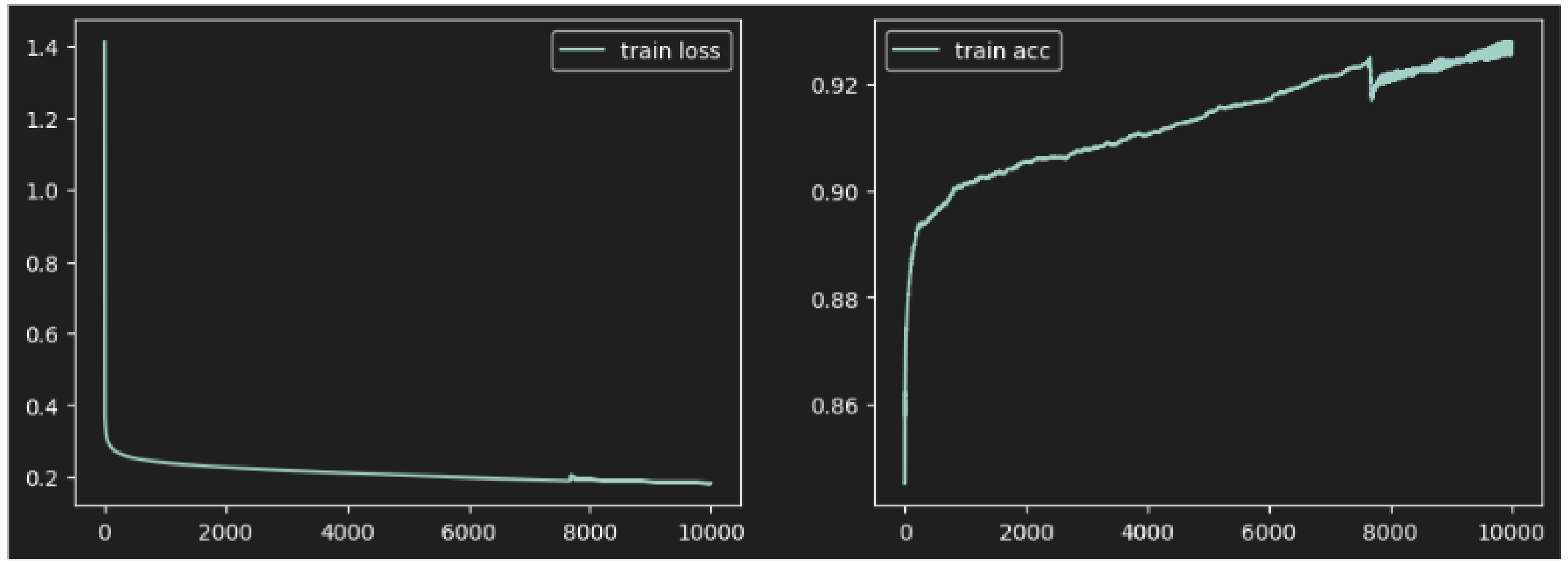
```
100%|██████████████████████████████████████████████████| 1000/1000 [00:41<00:00, 24.09it/s]
```





```
Accuracy: 0.851338199513382
Confusion Matrix:
[[10389    33]
 [ 1800   108]]
Error Rate: 0.14866180048661803
Classification Report:
              precision    recall  f1-score   support

           0       0.85      1.00      0.92     10422
           1       0.77      0.06      0.11      1908

    accuracy                           0.85     12330
   macro avg       0.81      0.53      0.51     12330
weighted avg       0.84      0.85      0.79     12330
```

```
100%|██████████████████████████████████████████████████| 10000/10000 [05:29<00:00, 30.39it/s]
```




```
Accuracy: 0.9278183292781833
Confusion Matrix:
[[9999  423]
 [ 467 1441]]
Error Rate: 0.0721816707218167
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96     10422
           1       0.77      0.76      0.76      1908

    accuracy                           0.93     12330
   macro avg       0.86      0.86      0.86     12330
weighted avg       0.93      0.93      0.93     12330
```

Conclusion, mon réseau de neurones sans PCA remporte la bataille avec :

Accuracy: 0.928

Confusion Matrix:
[[9999 423]
[467 1441]]

Error Rate: 0.072

Classification Report:

	precision	recall	f1-score	support
0	0.96	0.96	0.96	10422
1	0.77	0.76	0.76	1908
accuracy			0.93	12330
macro avg	0.86	0.86	0.86	12330
weighted avg	0.93	0.93	0.93	12330

This latest model seems to be much more efficient than the previous ones.