HomeNet

HOME-NET

Project name : Classification of Home Network Users .

Presented by : winners

Date : 10 april 2023

Our Team

Marwa Ayed

Alaa Kilani

Eya Ben Jemaa

Nassim TOUMI

Summary

HomeNet

Introduction

The Classification of Home Network Users to Improve User Experience project is a study that aims to improve the user experience for home network users. With the increase in the use of home networks, it has become increasingly important to understand user habits and needs to provide a high quality experience.

The project focuses on using classification techniques to identify different types of home network users, Once the different types of users have been identified, the goal is to offer personalized solutions to meet the needs of each type of user, thus improving the overall experience

Bussiness objectives

- Improve the quality of the network connection for users for a better user experience
- Optimize network performance to minimize packet loss and improve connection reliability
- Evaluate network performance at different times of the day to better understand bandwidth demands and user behaviors

Data science objectives

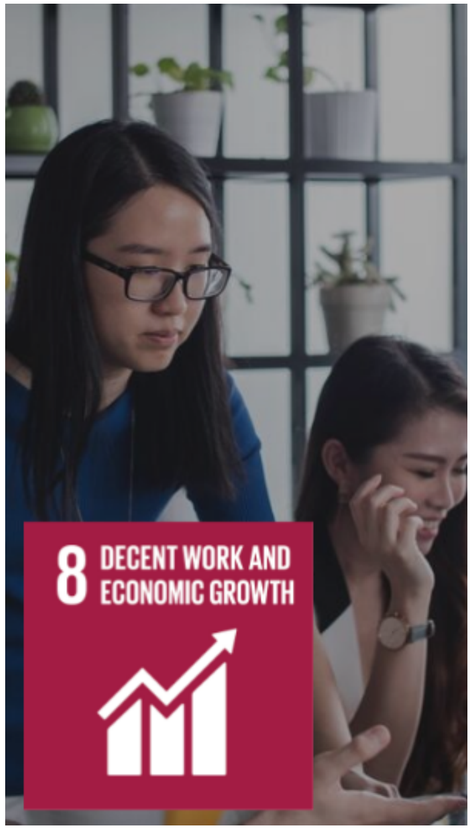
- Improve the quality of the network connection
- Optimize network performance
- Evaluate network performance

The data source

we would like to express our sincere gratitude to Orange Tunisia for providing us with access to their database for our machine learning project.
Working with real-world data is an invaluable experience for anyone pursuing a career in data science, and we are grateful for the opportunity to learn and grow from this experience.
We appreciate Orange Tunisia's willingness to share their resources and support our learning journey, and we hope to continue collaborating with them in the future .

Sustainable development objectives

Decent work and economic growth



we can enable users to work efficiently remotely, which can contribute to economic growth

Quality education

we can facilitate access to online education for users, which can improve the quality of education

Futur vision

- Improve prediction accuracy by using more advanced machine learning algorithms and including more variables relevant to customer satisfaction .
- Implement natural language processing (NLP) techniques to analyze customer feedback and understand the reasons for their dissatisfaction or satisfaction .
- Develop interactive data visualization tools to help analysts and managers understand trends and correlations between variables

Data comprehension

Understanding information through analysis and interpretation.

Satisfied clients

	day	hour	specifictime	SYN-ACK-recepteur-to-ACK-emitter	SYN-ACK-emitter-to-ACK-recepteur	Latence de connexion	Délai de réponse	Latence coté serveur	Latence coté client	Taux retransmission downlink	Taux retransmission uplink	target
0	2021-06-10	18	2021-06-10 18:14:59	46.00	910.50	3.50	47.67	48.0	52.0	0.2745	0.0750	1
1	2021-06-10	18	2021-06-10 18:14:59	0.00	0.00	0.00	0.00	0.0	0.0	0.0000	0.4286	1
2	2021-06-10	18	2021-06-10 18:19:59	5.00	6.00	2.67	4.67	0.0	4.0	0.0000	0.0000	1
3	2021-06-10	18	2021-06-10 18:19:59	46.00	1971.00	7.33	46.00	7.0	46.0	0.1000	0.0000	1
4	2021-06-10	18	2021-06-10 18:24:59	25.00	3.00	9.00	25.00	3.0	48.0	0.0000	0.0000	1
...	...	...	...	...	...	...	...	...	...	...	...	...
16732	2021-06-16	23	2021-06-16 23:24:59	13.00	1.00	1.00	7.00	1.0	12.0	0.0000	0.0000	1
16733	2021-06-16	23	2021-06-16 23:39:59	17.00	5.00	2.33	17.33	2.0	17.0	0.0000	0.0000	1
16734	2021-06-16	23	2021-06-16 23:44:59	45.00	3.00	2.00	45.00	0.0	45.0	0.0000	0.0000	1
16735	2021-06-16	23	2021-06-16 23:44:59	5.33	1.33	1.00	4.67	2.0	14.0	0.0000	0.0000	1
16736	2021-06-16	23	2021-06-16 23:59:59	18.00	37.33	3.33	18.33	3.0	18.0	0.0000	0.0000	1

1824669 rows × 12 columns

here is the table of the data set of satisfied customers by adding the target column with the value 1

Unsatisfied clients

	day	hour	specifictime	SYN-ACK-recepteur-to-ACK-emitter	SYN-ACK-emitter-to-ACK-recepteur	Latence de connexion	Délai de réponse	Latence coté serveur	Latence coté client	Taux retransmission downlink	Taux retransmission uplink
0	2021-06-10	18	2021-06-10 18:09:59	45.0	36.0	3.0	45.0	229.0	45.0	0.2000	0.0000
1	2021-06-10	18	2021-06-10 18:09:59	33.0	3.0	0.0	33.0	2.0	33.0	0.0000	0.0000
2	2021-06-10	18	2021-06-10 18:09:59	24.0	2.0	5.0	24.0	2.0	24.0	0.0000	0.0000
3	2021-06-10	18	2021-06-10 18:09:59	5.0	1.0	2.0	25.5	300.0	18.0	0.3404	0.1429
4	2021-06-10	18	2021-06-10 18:09:59	5.0	3.0	5.0	5.0	2.0	5.0	0.0000	0.2400
...	...	...	...	...	...	...	...	...	...	...	...
3520	2021-06-16	23	2021-06-16 23:39:59	0.0	0.0	0.0	0.0	185.0	24.0	0.0000	0.0000
3521	2021-06-16	23	2021-06-16 23:44:59	0.0	0.0	0.0	0.0	233.0	27.0	0.0000	0.0000
3522	2021-06-16	23	2021-06-16 23:49:59	0.0	0.0	0.0	0.0	223.0	30.0	0.0000	0.0000
3523	2021-06-16	23	2021-06-16 23:54:59	0.0	0.0	0.0	0.0	289.0	24.0	0.0000	0.0000
3524	2021-06-16	23	2021-06-16 23:59:59	0.0	0.0	0.0	0.0	228.0	25.0	0.1250	0.0000

1862703 rows × 11 columns

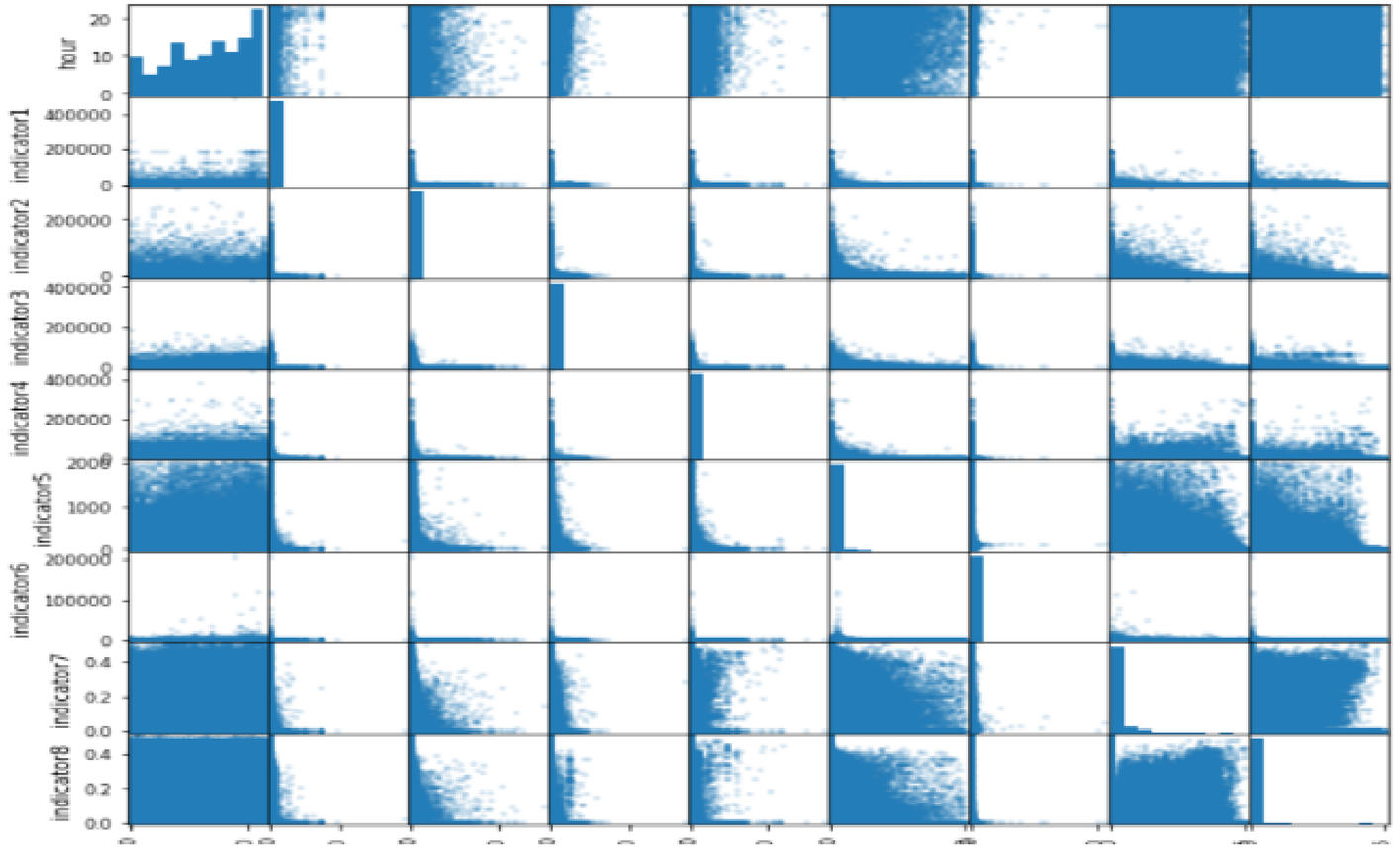
here is the table of the data set of satisfied customers by adding the target column with the value 0

data description for df1

	hour	SYN-ACK-recepteur-to-ACK-emitter	SYN-ACK-emitter-to-ACK-recepteur	Latence de connexion	Délai de réponse	Latence coté serveur	Latence coté client	Taux retransmission downlink	Taux retransmission uplink	target
count	1.824669e+06	1.824669e+06	1.824669e+06	1.824669e+06	1.824669e+06	1.824669e+06	1.824669e+06	1.824669e+06	1.824669e+06	1824669.0
mean	1.340716e+01	7.167716e+01	2.161264e+02	1.390619e+02	1.738866e+02	4.680445e+01	2.172535e+01	2.341425e-02	1.801746e-02	1.0
std	6.795130e+00	1.466963e+03	2.338637e+03	1.716952e+03	3.047865e+03	1.002573e+02	2.933425e+02	5.719918e-02	6.860591e-02	0.0
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.0
25%	8.000000e+00	4.500000e+00	3.000000e+00	6.700000e-01	4.630000e+00	5.000000e+00	8.000000e+00	0.000000e+00	0.000000e+00	1.0
50%	1.400000e+01	1.153000e+01	7.000000e+00	2.670000e+00	1.270000e+01	1.800000e+01	1.500000e+01	0.000000e+00	0.000000e+00	1.0
75%	2.000000e+01	2.250000e+01	3.900000e+01	8.250000e+00	2.400000e+01	4.800000e+01	2.500000e+01	1.590000e-02	0.000000e+00	1.0
max	2.300000e+01	4.874515e+05	3.036850e+05	4.327250e+05	4.421220e+05	2.000000e+03	2.130030e+05	5.000000e-01	5.000000e-01	1.0

here we have the measurements that we will use in the boxplot below

Cloud points



according to this figure we have no strong correlation because the distribution of the points is not in linear form .

missing values for the satisfied clients

```
for col in df1.columns:
    print('Variable: ', col)
    print('Nombre de valeurs manquantes: ', df1[col].isnull().sum())
    print('\n')
```

```
Variable:  day
Nombre de valeurs manquantes:  0


Variable:  hour
Nombre de valeurs manquantes:  0


Variable:  specifictime
Nombre de valeurs manquantes:  0


Variable:  SYN-ACK-recepteur-to-ACK-emitter
Nombre de valeurs manquantes:  0


Variable:  SYN-ACK-emitter-to-ACK-recepteur
Nombre de valeurs manquantes:  0


Variable:  Latence de connexion
Nombre de valeurs manquantes:  0


Variable:  Délai de réponse
Nombre de valeurs manquantes:  0


Variable:  Latence coté serveur
Nombre de valeurs manquantes:  0
```

we can see that there are no missing values in all the features

Normalization for df1

Entrée [22]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
df1_normalized = scaler.fit_transform(df1)
```

Entrée [23]:
```
df1
```

Out[23]:

	day	hour	specifictime	indicator1	indicator2	indicator3	indicator4	indicator5	indicator6	indicator7	indicator8	target
0	1.623283e+09	18	1.623349e+09	46.00	910.50	3.50	47.67	48.0	52.0	0.2745	0.0750	1
1	1.623283e+09	18	1.623349e+09	0.00	0.00	0.00	0.00	0.0	0.0	0.0000	0.4286	1
2	1.623283e+09	18	1.623349e+09	5.00	6.00	2.67	4.67	0.0	4.0	0.0000	0.0000	1
3	1.623283e+09	18	1.623349e+09	46.00	1971.00	7.33	46.00	7.0	46.0	0.1000	0.0000	1
4	1.623283e+09	18	1.623349e+09	25.00	3.00	9.00	25.00	3.0	48.0	0.0000	0.0000	1
...	...	...	...	...	...	...	...	...	...	...	...	...
16732	1.623802e+09	23	1.623886e+09	13.00	1.00	1.00	7.00	1.0	12.0	0.0000	0.0000	1
16733	1.623802e+09	23	1.623887e+09	17.00	5.00	2.33	17.33	2.0	17.0	0.0000	0.0000	1
16734	1.623802e+09	23	1.623887e+09	45.00	3.00	2.00	45.00	0.0	45.0	0.0000	0.0000	1
16735	1.623802e+09	23	1.623887e+09	5.33	1.33	1.00	4.67	2.0	14.0	0.0000	0.0000	1
16736	1.623802e+09	23	1.623888e+09	18.00	37.33	3.33	18.33	3.0	18.0	0.0000	0.0000	1

1824669 rows × 12 columns

we did the normalization for the df1 data frame of satisfied customers importedMinMaxscalar and we initialize the scaler finally we do the normalization

Correlation

```
df.corr()
```

	hour	indicator1	indicator2	indicator3	indicator4	indicator5	indicator6	indicator7
hour	1.000000	0.020750	0.044260	-0.001184	0.023392	0.031834	-0.001937	0.020638
indicator1	0.020750	1.000000	0.130312	0.200820	0.924852	0.014380	0.782011	-0.006647
indicator2	0.044260	0.130312	1.000000	0.108060	0.122486	0.460600	0.032786	0.103808
indicator3	-0.001184	0.200820	0.108060	1.000000	0.189020	0.043852	0.130018	0.066919
indicator4	0.023392	0.924852	0.122486	0.189020	1.000000	0.031581	0.772109	0.011981
indicator5	0.031834	0.014380	0.460600	0.043852	0.031581	1.000000	0.054040	0.174256
indicator6	-0.001937	0.782011	0.032786	0.130018	0.772109	0.054040	1.000000	-0.015220
indicator7	0.020638	-0.006647	0.103808	0.066919	0.011981	0.174256	-0.015220	1.000000
target	0.004629	-0.005869	0.002944	0.007358	-0.004102	-0.021396	-0.000852	0.057240

we see that indicators 7, 2 and 3 have values close to 1, so they are the indicators that are linked to satisfaction .
we see that indicators 1,4,5 and 6 have values close to 0 so they are linked to non-satisfaction .

Data preparation

Organizing and transforming data into a suitable format for analysis.

Concatenation

```
df = pd.concat([df1, df2])
```

```
df
```

	day	hour	specifictime	SYN-ACK-recepteur-to-ACK-emitter	SYN-ACK-emitter-to-ACK-recepteur	Latence de connexion	Délai de réponse	Latence coté serveur	Latence coté client	Taux retransmission downlink	Taux retransmission uplink	target
0	2021-06-10	18	2021-06-10 18:14:59	46.0	910.5	3.50	47.67	48.0	52.0	0.2745	0.0750	1
1	2021-06-10	18	2021-06-10 18:14:59	0.0	0.0	0.00	0.00	0.0	0.0	0.0000	0.4286	1
2	2021-06-10	18	2021-06-10 18:19:59	5.0	6.0	2.67	4.67	0.0	4.0	0.0000	0.0000	1
3	2021-06-10	18	2021-06-10 18:19:59	46.0	1971.0	7.33	46.00	7.0	46.0	0.1000	0.0000	1
4	2021-06-10	18	2021-06-10 18:24:59	25.0	3.0	9.00	25.00	3.0	48.0	0.0000	0.0000	1
...	...	...	...	...	...	...	...	...	...	...	...	...
3520	2021-06-16	23	2021-06-16 23:39:59	0.0	0.0	0.00	0.00	185.0	24.0	0.0000	0.0000	0
3521	2021-06-16	23	2021-06-16 23:44:59	0.0	0.0	0.00	0.00	233.0	27.0	0.0000	0.0000	0
3522	2021-06-16	23	2021-06-16 23:49:59	0.0	0.0	0.00	0.00	223.0	30.0	0.0000	0.0000	0
3523	2021-06-16	23	2021-06-16 23:54:59	0.0	0.0	0.00	0.00	289.0	24.0	0.0000	0.0000	0
3524	2021-06-16	23	2021-06-16 23:59:59	0.0	0.0	0.00	0.00	228.0	25.0	0.1250	0.0000	0

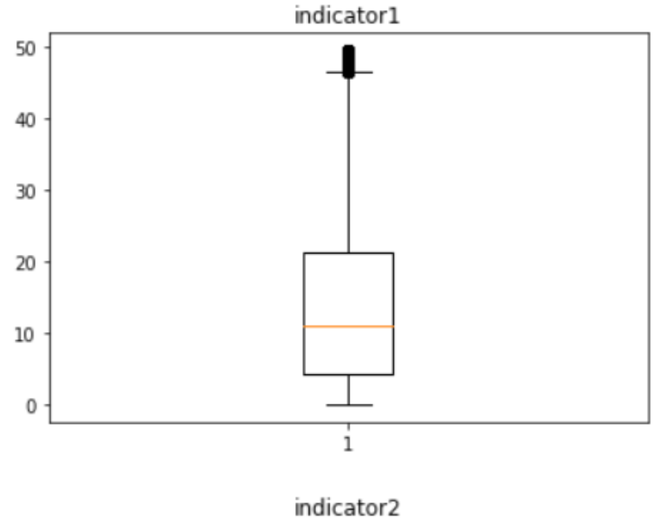
3687372 rows × 12 columns

- we have concatenated the 2 data sets the satisfied customers and the unsatisfied customers and I added a target value 1 for satisfied clients and value 0 for unsatisfied clients

Boxplot before cleaning data

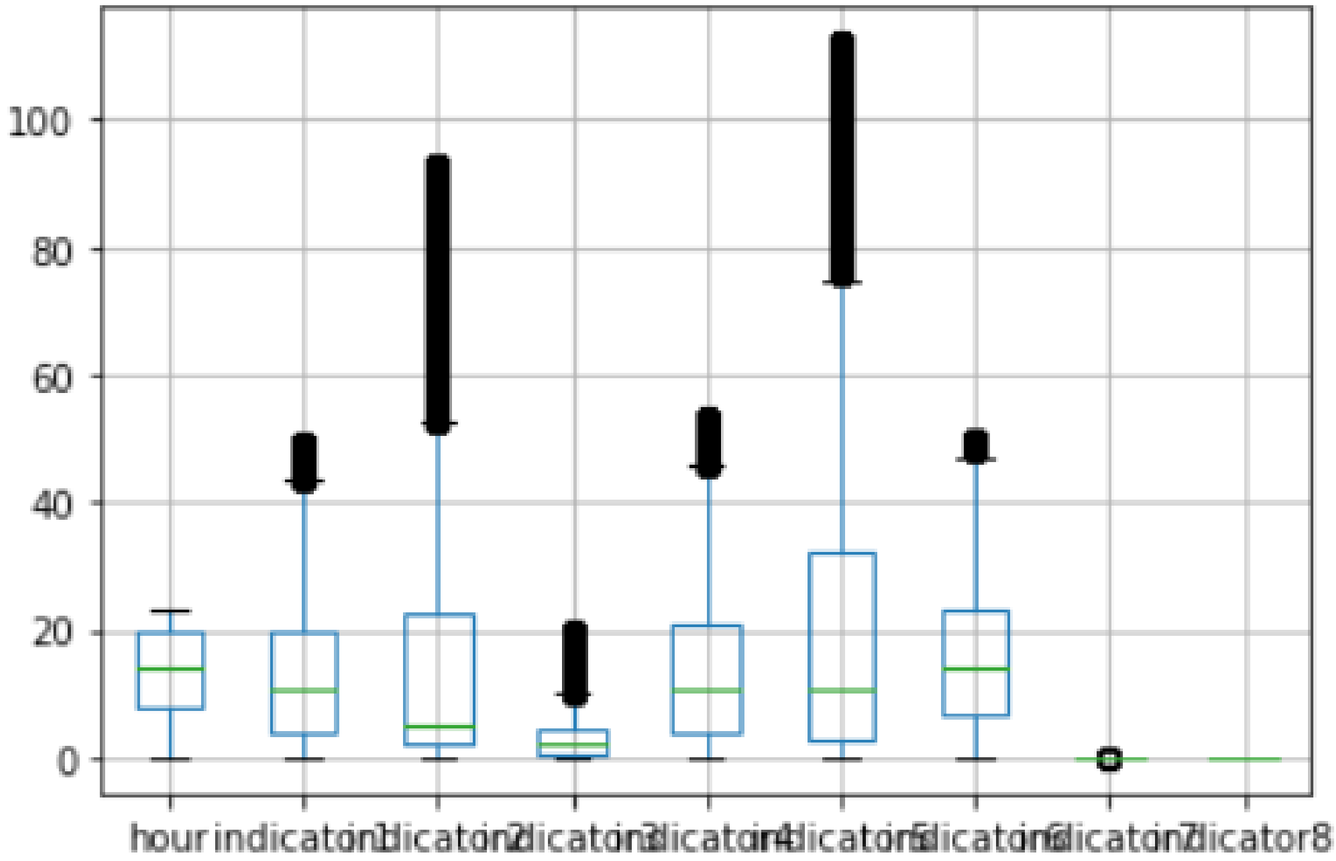
```python
# Calcul de la plage interquartile pour chaque colonne numérique
Q1 = df1.select_dtypes(include='number').quantile(0.25)
Q3 = df1.select_dtypes(include='number').quantile(0.75)
IQR = Q3 - Q1

# Définition des limites pour les boxplots
lower_limit = Q1 - 1.5 * IQR
upper_limit = Q3 + 1.5 * IQR

# Affichage des boxplots pour chaque colonne numérique
for col in df1.select_dtypes(include='number').columns:
    plt.figure()
    plt.title(col)
    plt.boxplot(df1[col][(df1[col] >= lower_limit[col]) & (df1[col] <= upper_limit[col])])
    plt.show()
```

1



- we define the limits for the boxplot by the lower limit which represents the 1st percentile and upper-limit which represents the 3rd percentile limit

Boxplot after cleaning the data



• For indicator 1 we see that the value of the 3rd percentile has decreased from 22.5 to 19.5 which means that 75% of all the values of indicator 1 require a maximum value of 19.5 to establish the connection and the value of the 1st percentile has to remain constant at 4.5 which says that 25% of all values of indicator 1 require 4.5 as a minimum value to establish the connection

Balance of data

```
counts = df['target'].value_counts()

print(counts)
```

```
0     1862703
1     1824669
Name: target, dtype: int64
```

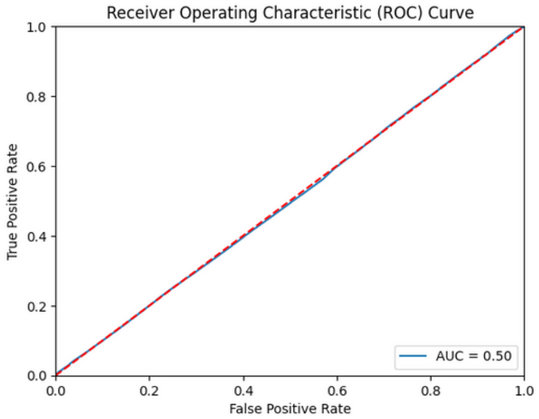
Modeling

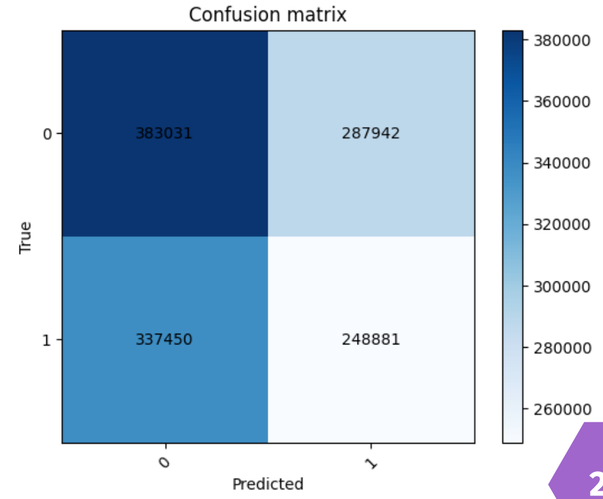
Constructing a mathematical or computational representation of a system or phenomenon to make predictions or gain insights.

Xg boost

```
import pandas as pd
precision = precision_score(Y_test, pred_test, average='weighted')
recall = recall_score(Y_test, pred_test, average='weighted')
f1 = f1_score(Y_test, pred_test, average='weighted')
accuracy = accuracy_score(Y_test, pred_test)
print('Precision', precision)
print('Recall', recall)
print('F1-Score', f1)
print('Accuracy', accuracy)
```

```
Precision 0.4999147095525761 6
Recall 0.5025928494620235
F1-Score 0.5004790201212881
Accuracy 0.5025928494620235
```

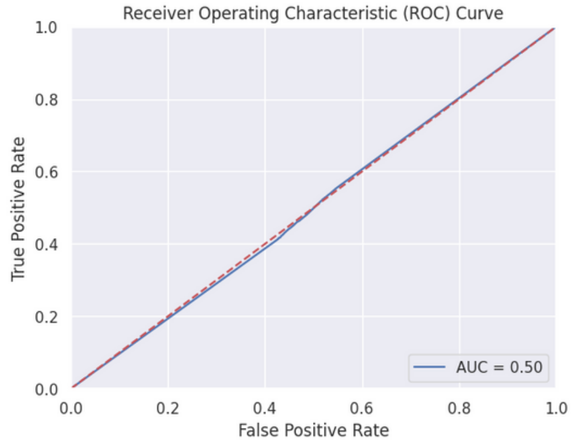


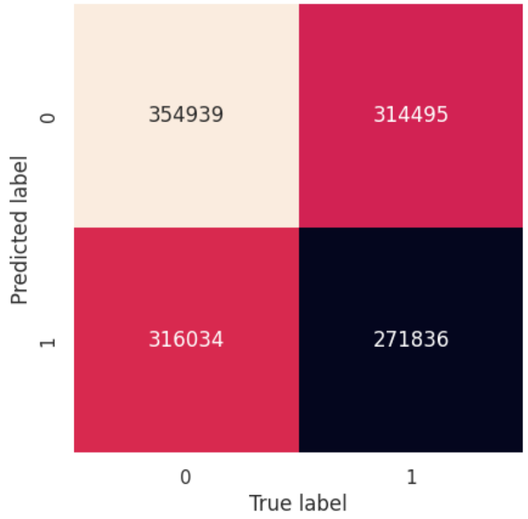


Decision Tree

```
from sklearn.metrics import*
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import confusion_matrix
#from sklearn.metrics import plot_confusion_matrix
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
pred_test10 = clf_DecisionTreeClassifier.predict((X_test))
print(classification_report(Y_test, pred_test10))
```

```
              precision    recall  f1-score   support

           0       0.53      0.53      0.53    670973
           1       0.46      0.46      0.46    586331

    accuracy                           0.50   1257304
   macro avg       0.50      0.50      0.50   1257304
weighted avg       0.50      0.50      0.50   1257304
```

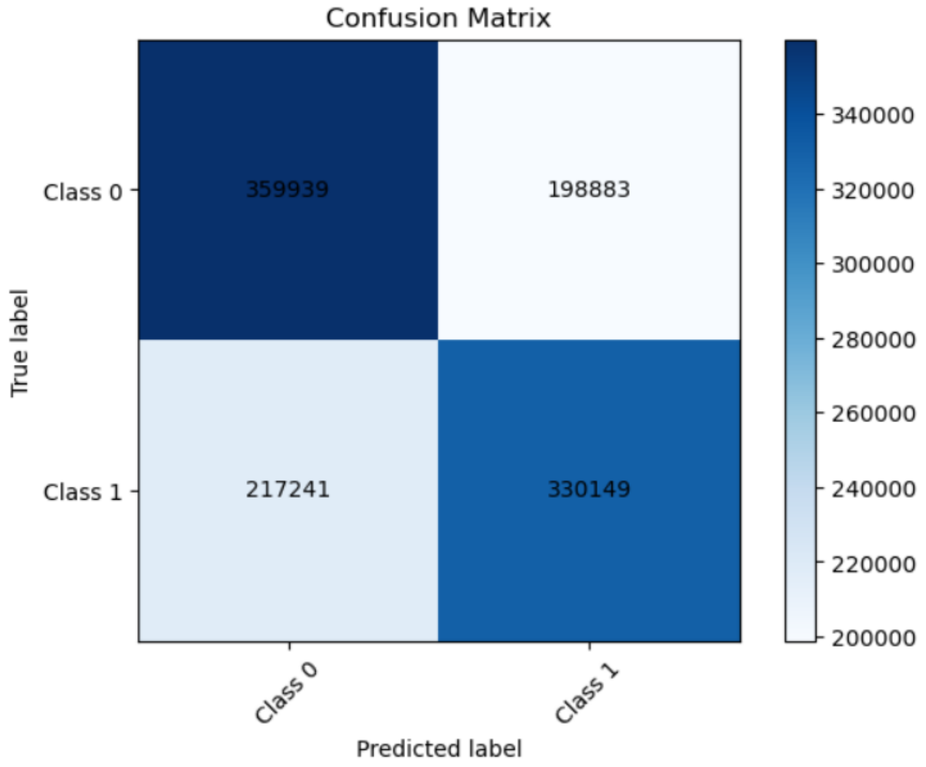




Random Forest

```
rf = RandomForestClassifier(n_estimators=100, random_state=42)

# Entraîner le modèle sur l'ensemble d'entraînement
rf.fit(X_train, y_train)

# Prédire les valeurs de la cible sur l'ensemble de test
y_pred_rf = rf.predict(X_test)

y_pred_rf = rf.predict(X_test)

# Évaluer les performances du modèle
precision_rf = precision_score(y_test, y_pred_rf, average='weighted')
recall_rf = recall_score(y_test, y_pred_rf, average='weighted')
f1_rf = f1_score(y_test, y_pred_rf, average='weighted')
accuracy_rf = accuracy_score(y_test, y_pred_rf)

print('Precision:', precision_rf)
print('Recall:', recall_rf)
print('F1-Score:', f1_rf)
print('Accuracy:', accuracy_rf)
```

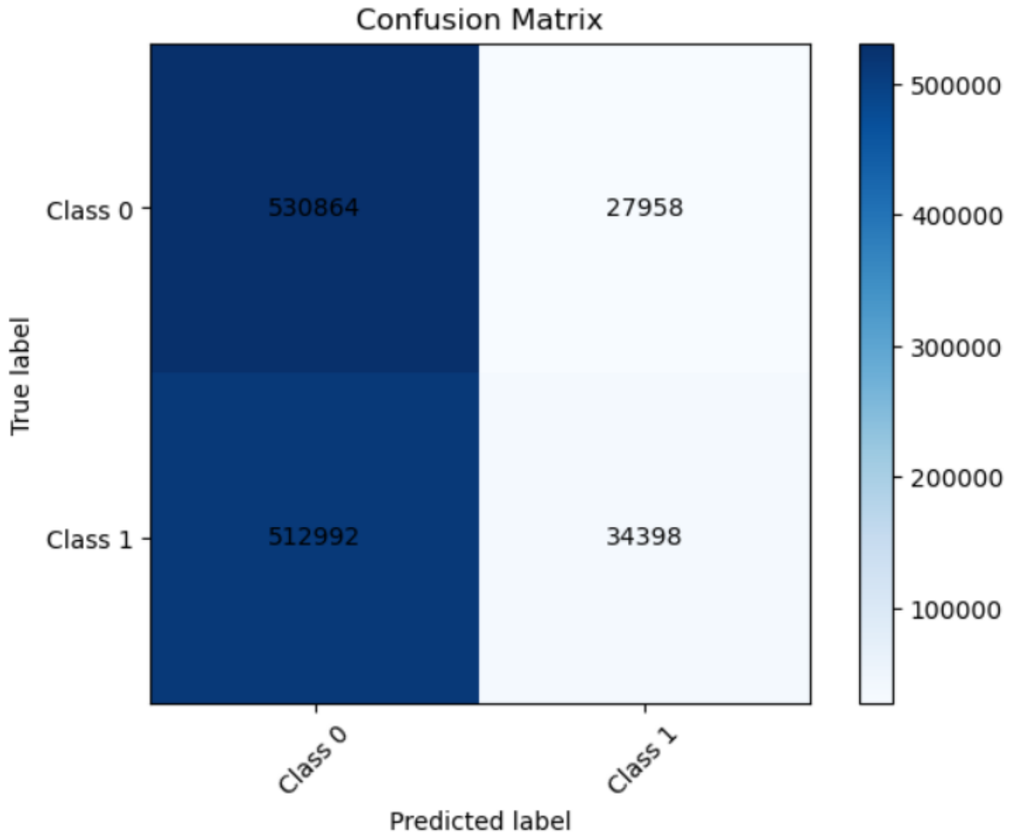
```
Precision: 0.6238371900076821
Recall: 0.6238297903114413
F1-Score: 0.6236615544382463
Accuracy: 0.6238297903114413
```



Gaussian NB

```python
gnb = GaussianNB()

# Entraîner le modèle sur l'ensemble d'entraînement
gnb.fit(X_train, y_train)

# Prédire les valeurs de la cible sur l'ensemble de test
y_pred_gnb = gnb.predict(X_test)

# Évaluer les performances du modèle
precision_gnb = precision_score(y_test, y_pred_gnb, average='weighted')
recall_gnb = recall_score(y_test, y_pred_gnb, average='weighted')
f1_gnb = f1_score(y_test, y_pred_gnb, average='weighted')
accuracy_gnb = accuracy_score(y_test, y_pred_gnb)

print('Precision:', precision_gnb)
print('Recall:', recall_gnb)
print('F1-Score:', f1_gnb)
print('Accuracy:', accuracy_gnb)
```

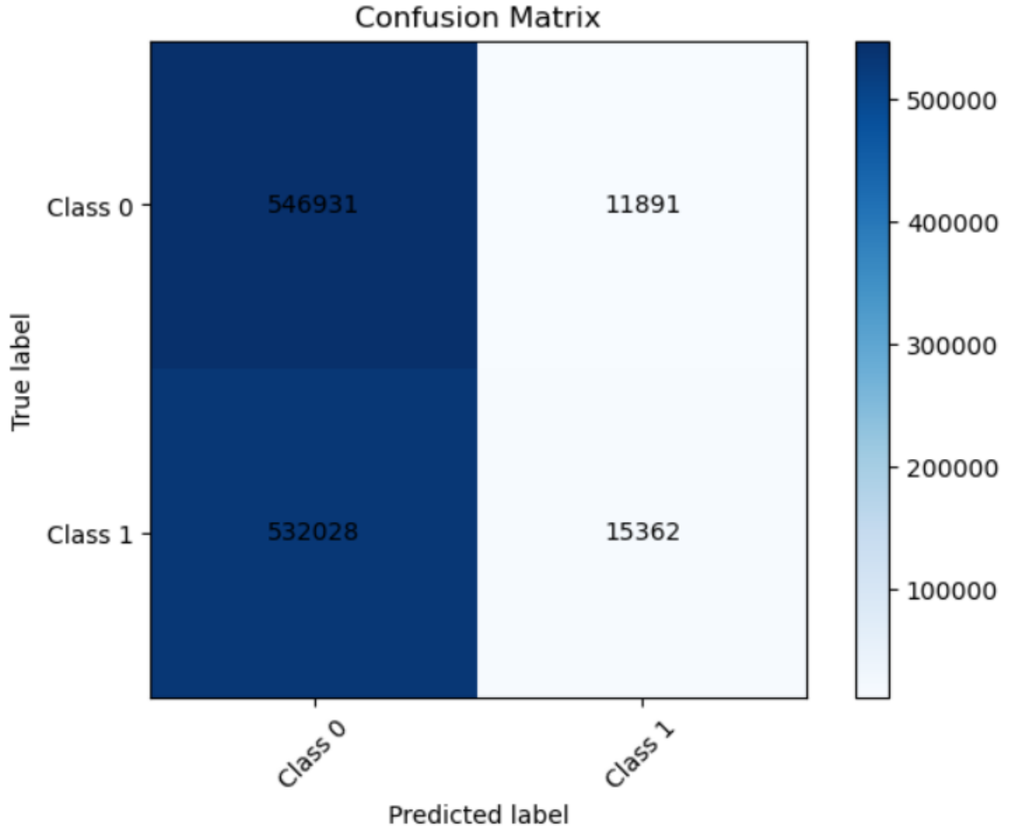
```
Precision: 0.5298771779450945
Recall: 0.5109888520464432
F1-Score: 0.3904893592280493
Accuracy: 0.5109888520464432
```



Logistic Regression

```
logreg = LogisticRegression(random_state=42)

# Entraîner le modèle sur l'ensemble d'entraînement
logreg.fit(X_train, y_train)

# Prédire les valeurs de la cible sur l'ensemble de test
y_pred_logreg = logreg.predict(X_test)

# Évaluer les performances du modèle
precision_logreg = precision_score(y_test, y_pred_logreg, average='weighted')
recall_logreg = recall_score(y_test, y_pred_logreg, average='weighted')
f1_logreg = f1_score(y_test, y_pred_logreg, average='weighted')
accuracy_logreg = accuracy_score(y_test, y_pred_logreg)

print('Precision:', precision_logreg)
print('Recall:', recall_logreg)
print('F1-Score:', f1_logreg)
print('Accuracy:', accuracy_logreg)
```

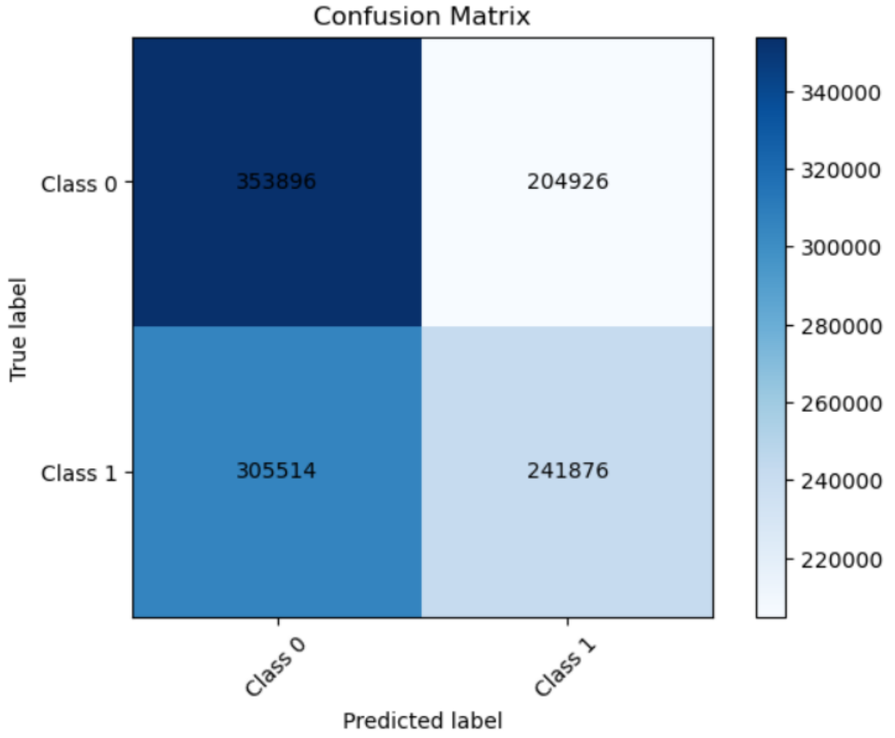
```
Precision: 0.5350002620928909
Recall: 0.5083049180446424
F1-Score: 0.36385433156083447
Accuracy: 0.5083049180446424
```



Catboost : Pseudo Deep Learning Model

```python
catboost_model = CatBoostClassifier(iterations=100, learning_rate=0.1, depth=2, random_seed=42, verbos

# Entraîner le modèle sur l'ensemble d'entraînement
catboost_model.fit(X_train, y_train)

# Prédire les valeurs de la cible sur l'ensemble de test
y_pred_catboost_model = catboost_model.predict(X_test)

# Évaluer les performances du modèle
precision_catboost_model = precision_score(y_test, y_pred_catboost_model, average='weighted')
recall_catboost_model = recall_score(y_test, y_pred_catboost_model, average='weighted')
f1_catboost_model = f1_score(y_test, y_pred_catboost_model, average='weighted')
accuracy_catboost_model = accuracy_score(y_test, y_pred_catboost_model)

print('Precision:', precision_catboost_model)
print('Recall:', recall_catboost_model)
print('F1-Score:', f1_catboost_model)
print('Accuracy:', accuracy_catboost_model)
```

```
97:     learn: 0.6880235        total: 9.88s    remaining: 202ms
98:     learn: 0.6879798        total: 9.97s    remaining: 101ms
99:     learn: 0.6879510        total: 10.1s    remaining: 0us
Precision: 0.5389935210183066
Recall: 0.5385694604650827
F1-Score: 0.534276589755784
Accuracy: 0.5385694604650827
```

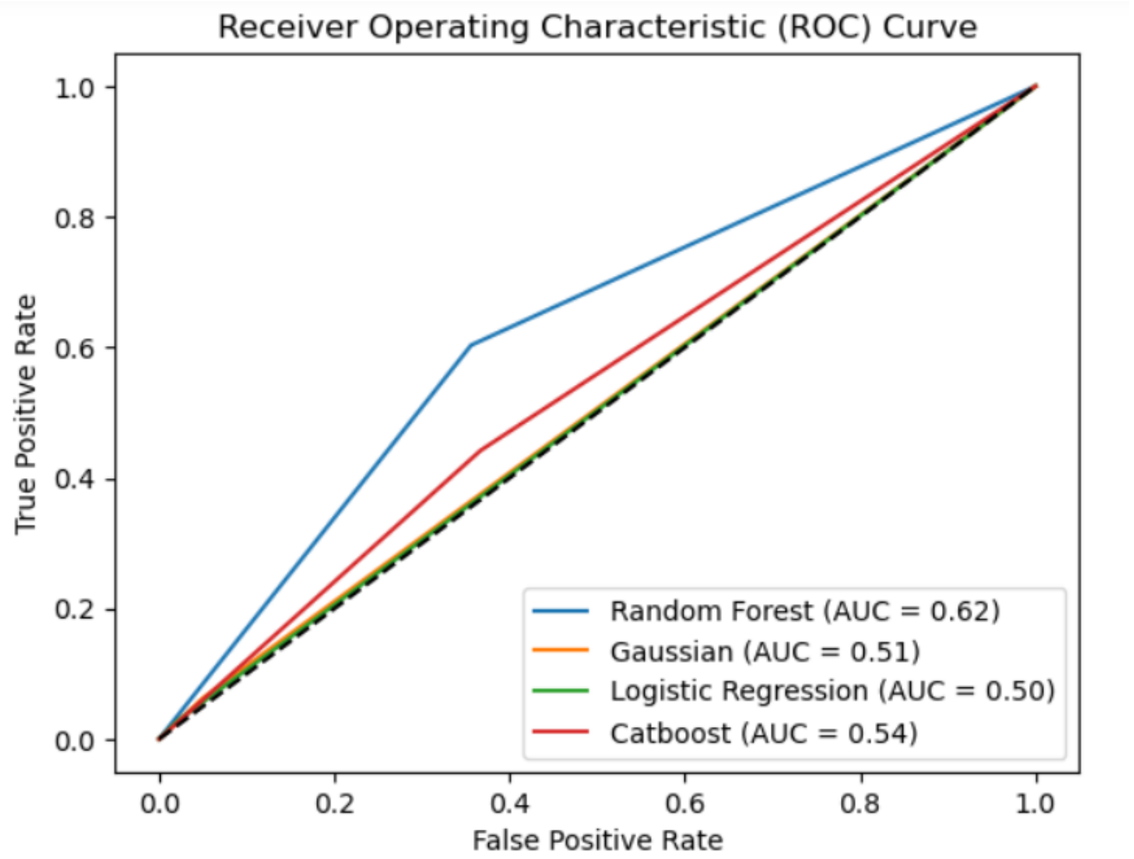


Evaluation

Assessing the performance and accuracy of a model .

Curve ROC

```python
from sklearn.metrics import roc_curve, roc_auc_score

pred = [y_pred_rf,y_pred_gnb,y_pred_logreg,y_pred_catboost_model]
name = ['Random Forest', 'Gaussian','Logistic Regression','Catboost']
for p in range(len(pred)) :
    fpr, tpr, thresholds = roc_curve(y_test, pred[p])
    auc_score = roc_auc_score(y_test, pred[p])

    # Plot the ROC curve
    plt.plot(fpr, tpr, label='%s (AUC = %0.2f)' % (name[p], auc_score))

# Plot the random guessing line
plt.plot([0, 1], [0, 1], 'k--')

# Add labels and legend to the plot
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")

plt.show()
```



Deployment

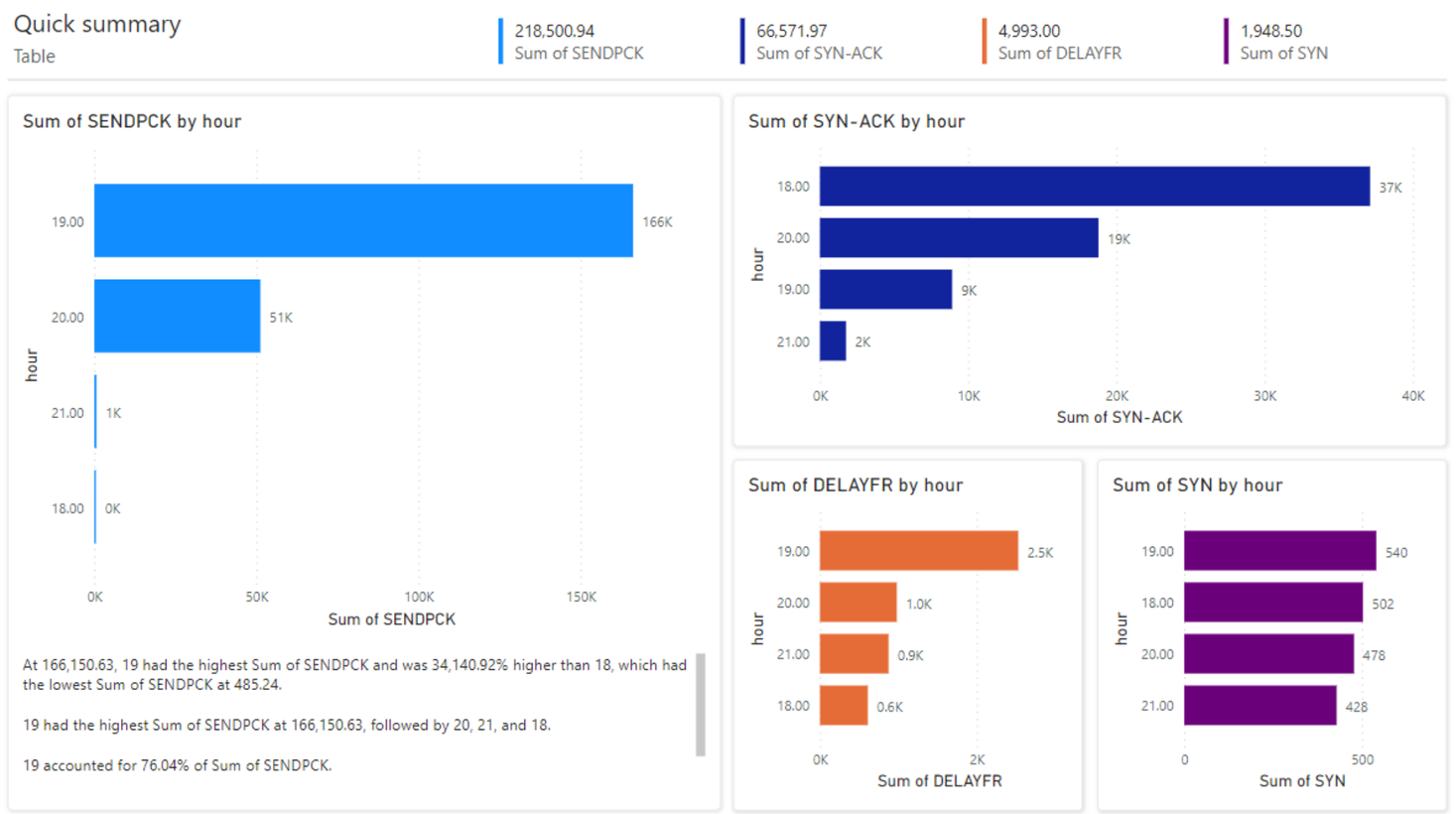
The process of making a model or software application available for use in a production environment.

Dashboard

We choose PowerBI

```
from powerbiclient import QuickVisualize, get_dataset_config, Report
from powerbiclient.authentication import DeviceCodeLoginAuthentication
device_auth = DeviceCodeLoginAuthentication()

import pandas as pd
```

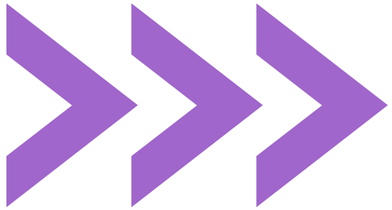
✓ 1m 1.3s



Web applications

We choose Flask
why ?

Flask provides a simple and flexible way to create web applications and APIs, and allows you to easily integrate your machine learning model into a web application



Web applications

```
import joblib
joblib.dump(rf, 'model.pkl')
```

model.pkl	03/05/2023 12:16 AM	Fichier PKL	8 561 884 Ko

```
from flask import Flask, request, jsonify, render_template,redirect
import pickle
import numpy as np
```

```
# Load the machine learning model
with open('model.pkl', 'rb') as f:
    model = pickle.load(f)
```

Web applications

Enter your features:

SYN:

SYN-ACK:

ACK:

SENDPCK:

DELAYFR:

DELAYFDNS:

RETRAND:

RETANSU:

Hour:

Day of Month:

Month:

Year:

Predict

This user have a bad connection 👎

HomeNet

Thanks !

any questions ?