



TELECOM CHURN DATASET

BY GROUP 4



01

INFORMATION



BALIGH MNASSRI

- Data Scientist - Machine Learning Engineer
- France

SOURCE DATA : <https://www.kaggle.com/code/mnassrib/customer-churn-prediction-telecom-churn-dataset/notebook>

02

DATA : 3333 ROWS X 20 COLUMNS

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Total eve charge	Total night minutes	Total night calls	Total night charge	Total int'l minutes	Total int'l calls	Total int'l charge	Customer service calls	Churn
0	LA	117	408	No	No	0	184.5	97	31.37	351.6	80	29.89	215.8	90	9.71	8.7	4	2.35	1	False
1	IN	65	415	No	No	0	129.1	137	21.95	228.5	83	19.42	208.8	111	9.40	12.7	6	3.43	4	True
2	NY	161	415	No	No	0	332.9	67	56.59	317.8	97	27.01	160.6	128	7.23	5.4	9	1.46	4	True
3	SC	111	415	No	No	0	110.4	103	18.77	137.3	102	11.67	189.6	105	8.53	7.7	6	2.08	2	False
4	HI	49	510	No	No	0	119.3	117	20.28	215.1	109	18.28	178.7	90	8.04	11.1	1	3.00	1	False
...	
662	WI	114	415	No	Yes	26	137.1	88	23.31	155.7	125	13.23	247.6	94	11.14	11.5	7	3.11	2	False
663	AL	106	408	No	Yes	29	83.6	131	14.21	203.9	131	17.33	229.5	73	10.33	8.1	3	2.19	1	False
664	VT	60	415	No	No	0	193.9	118	32.96	85.0	110	7.23	210.1	134	9.45	13.2	8	3.56	3	False
665	WV	159	415	No	No	0	169.8	114	28.87	197.7	105	16.80	193.7	82	8.72	11.6	4	3.13	1	False
666	CT	184	510	Yes	No	0	213.8	105	36.35	159.6	84	13.57	139.2	137	6.26	5.0	10	1.35	2	False

Target Variable : Columns Churn

03 ABOUT DATESET

ชุดข้อมูลเกี่ยวกับ

Customer churn
prediction :
Telecom Churn
Dataset

จุดประสงค์

เพื่อคาดการณ์
การเปลี่ยนใจของลูกค้า
ในบริษัทโทรศัพท์มือถือ
Churn

04

DATA DESCRIPTION

- 'State' = รัฐ
- 'Account length' = จำนวนวันที่ลูกค้ามีบัญชีใช้
- 'Area code' = รหัสพื้นที่
- 'International plan' = บริการโทรศัพท์ระหว่างประเทศ
- 'Voice mail plan' = บริการข้อความเสียง
- 'Number vmail messages' = จำนวนข้อความในกล่องจดหมายเสียง
- 'Total day minutes' = จำนวนนาทีที่โทรศัพท์ในต่อเนื่องเข้าก็งหมด
- 'Total day calls' = จำนวนครั้งที่โทรศัพท์ช่วงเช้าก็งหมด
- 'Total day charge' = ค่าบริการรวมก็งวัน
- 'Total eve minutes' = จำนวนนาทีของวันก่อนวันก็งหมด
- 'Total eve calls' = จำนวนก็งหมดที่โทรศัพท์ในช่วงเย็น
- 'Total eve charge' = ค่าบริการในช่วงเย็นก็งหมด
- 'Total night minutes' = จำนวนนาทีก็งหมดที่ใช้ในช่วงกลางคืน
- 'Total night calls' = จำนวนก็งหมดที่โทรศัพท์ในช่วงกลางคืน
- 'Total night charge' = ค่าบริการตอนกลางคืนก็งหมด
- 'Total intl minutes' = จำนวนนาทีที่ใช้ในการโทรศัพท์ต่างประเทศก็งหมด
- 'Total intl calls' = การโทรศัพท์ระหว่างประเทศก็งหมด
- 'Total intl charge' = ค่าบริการโทรศัพท์ระหว่างประเทศ
- 'Customer service calls' = การโทรศัพท์ต่อผู้ดูแลลูกค้า (สอบถามข้อมูลหรือร้องเรียน)
- 'Churn' = คาดการณ์ว่าลูกค้ายกเลิกบริการหรือไม่

05

DATA MINING

Classification

Decision Trees KNN Naive_bayes

1. ชุดข้อมูลประกอบด้วยข้อมูล
เกี่ยวกับการเปลี่ยนใจลูกค้าใน
บริษัทโทรศัพท์ตามช่วงมีเป้าหมาย
เพื่อคาดการณ์ว่าลูกค้ามีแนวโน้ม
ที่จะเลิกใช้งานหรือไม่ ?

2. แบ่งประเภทเพื่อกำหนาย
การเลิกใช้งานและระบุตัวกำหนด
ที่สำคัญที่สุด

Association

ชุดข้อมูลมีข้อมูลเกี่ยวกับรูปแบบ
การใช้งานของลูกค้าและบริการ
ที่สมัครซึ่งสามารถใช้เพื่อบรูปแบบ^{แบบ}
ของการซื้อขายตัวอย่าง เช่น
'ลูกค้าที่ใช้เวลาสนทนากันทั้งวัน
[รวมเช้า , กลางวัน, เย็น]
ในช่วง 501 - 900 นาทีต่อเดือน
มีแนวโน้มที่จะเลิกใช้งาน'

Clustering

K-means

ชุดข้อมูลสามารถใช้สำหรับการ
วิเคราะห์การจัดกลุ่มซึ่งเป้าหมายคือ^{คือ}
การ จัดกลุ่มลูกค้าตามความคล้ายคลึง^{คล้ายคลึงกัน}
กันในแต่ละช่วงของรูปแบบการใช้งาน
[จำนวนที่ในโทรศัพท์ กลางวัน เย็น]
สิ่งนี้สามารถช่วยระบุกลุ่มลูกค้าอยู่ใน^{อยู่ใน}
กลุ่มที่ยกเลิกการบริการหรือใหม่

06 IMPORT DATASET

```
[ ] import pandas as pd  
from google.colab import drive  
  
[ ] import os  
  
[ ] drive.mount('/content/drive')  
path = '/content/drive/MyDrive/datamining/Project'  
Mounted at /content/drive  
  
[ ] data1_train = pd.read_csv(os.path.join(path,'churn-bigml-80.csv')) #Os.path.join = path แต่ลักษณ์ไม่เหมือนกัน = เชื่อมต่อกันแบบมีส่วนหน้า,หลัง ; Encoding = ภาษาไทยเข้าใจง่าย  
data2_test = pd.read_csv(os.path.join(path,'churn-bigml-20.csv')) #Os.path.join = path แต่ลักษณ์ไม่เหมือนกัน = เชื่อมต่อกันแบบมีส่วนหน้า,หลัง ; Encoding = ภาษาไทยเข้าใจง่าย  
  
[ ] data = pd.concat([data1_train, data2_test], axis=0)  
• รวมไฟล์ทั้ง 2 แบบไม่ห้องไว้เลย  
  
[ ] data = data.reset_index(drop=True)  
data.index = data.index + 1
```

1

```
[1] 1 import pandas as pd  
2 from google.colab import drive
```

2

```
[2] 1 import os
```

3

```
[3] 1 drive.mount('/content/drive')  
2 path = '/content/drive/MyDrive/datamining/Project'  
Mounted at /content/drive
```

4 import 2 file

```
[42] data1_train = pd.read_csv(os.path.join(path,'churn-bigml-80.csv')) #Os.path.join = path แต่ลักษณ์ไม่เหมือนกัน = เชื่อมต่อกันแบบมีส่วนหน้า,หลัง ; Encoding = ภาษาไทยเข้าใจง่าย  
data2_test = pd.read_csv(os.path.join(path,'churn-bigml-20.csv')) #Os.path.join = path แต่ลักษณ์ไม่เหมือนกัน = เชื่อมต่อกันแบบมีส่วนหน้า,หลัง ; Encoding = ภาษาไทยเข้าใจง่าย
```

5

```
[ ] data = pd.concat([data1_train, data2_test], axis=0)
```

6

รีเซ็กลำดับทั้งหมด

concat file เพิ่มข้อมูลในแกน X

07

PREPARE DATASET

= 3333 row x 20 col

```
[ ] data1_en_train = pd.get_dummies(data1_en_train_SL, columns=['State','International plan','Voice mail plan'])
# Convert 'True' values to 1 and 'False' values to 0
data1_en_train['Churn'] = data1_en_train['Churn'].astype(int)
```

```
[ ] data2_en_test = pd.get_dummies(data2_en_test_SL, columns=['State','International plan','Voice mail plan'])
# Convert 'True' values to 1 and 'False' values to 0
data2_en_test['Churn'] = data2_en_test['Churn'].astype(int)
```

```
[ ] data1_en_train = data1_en_train[['Account length', 'Area code', 'Number vmail messages',
'Total day minutes', 'Total day calls', 'Total day charge',
'Total eve minutes', 'Total eve calls', 'Total eve charge',
'Total night minutes', 'Total night calls', 'Total night charge',
'Total intl minutes', 'Total intl calls', 'Total intl charge',
'Customer service calls', 'State_AK', 'State_AL', 'State_AR',
'State_AZ', 'State_CA', 'State_CO', 'State_CT', 'State_DC', 'State_DE',
'State_FL', 'State_GA', 'State_HI', 'State_IA', 'State_ID', 'State_IL',
'State_IN', 'State_KS', 'State_KY', 'State_LA', 'State_MA', 'State_MD',
'State_ME', 'State_MI', 'State_MN', 'State_MO', 'State_MS', 'State_MT',
'State_NC', 'State_ND', 'State_NE', 'State_NH', 'State_NJ', 'State_NM',
'State_NV', 'State_NY', 'State_OH', 'State_OK', 'State_OR', 'State_PA',
'State_RI', 'State_SC', 'State_SD', 'State_TN', 'State_TX', 'State_UT',
'State_VA', 'State_VT', 'State_WA', 'State_WI', 'State_WV', 'State_WY',
'International plan_No', 'International plan_Yes', 'Voice mail plan_No',
'Voice mail plan_Yes','Churn']]
```

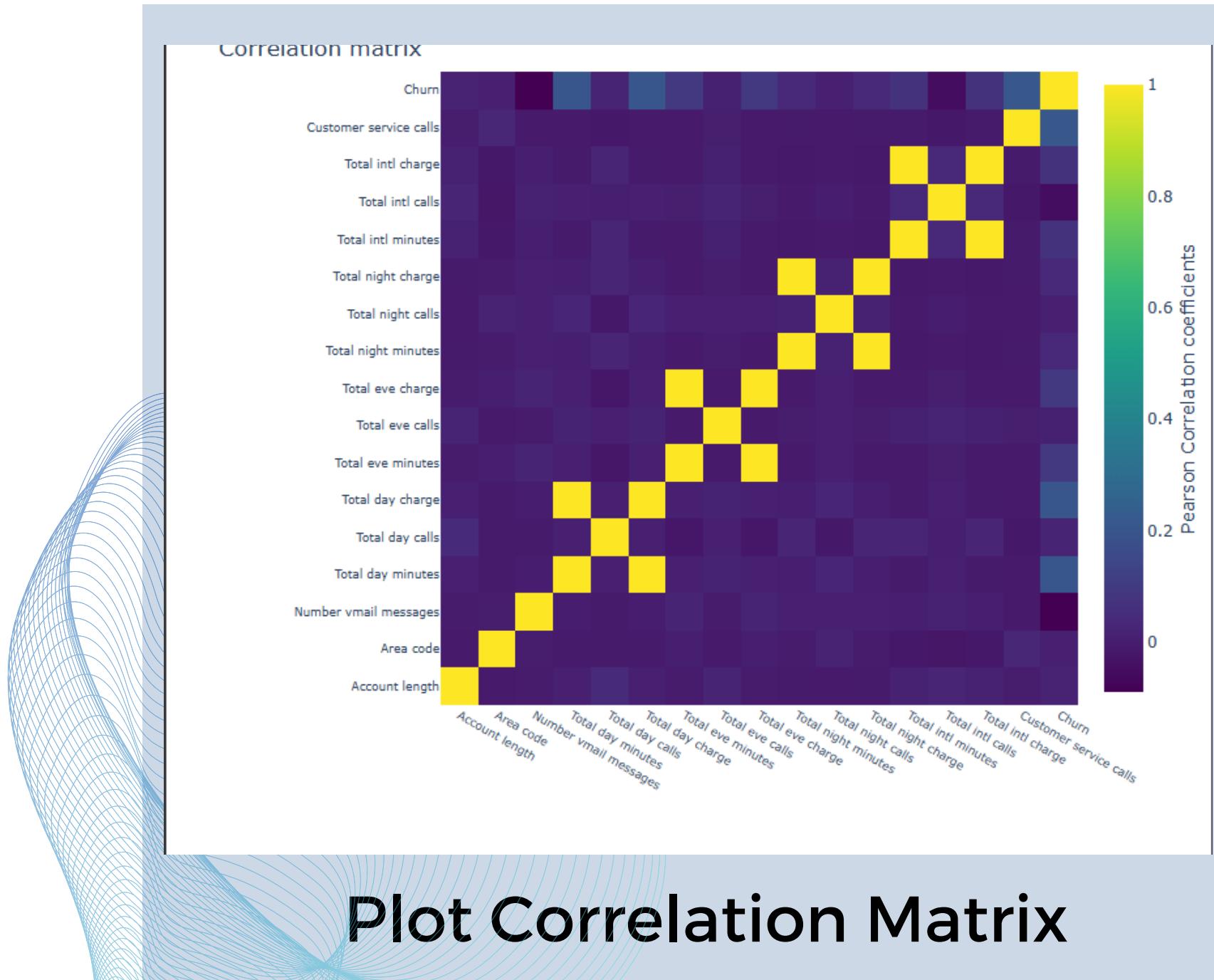
data1_en_train.head()

- Data 1_en = data1_en_train_SL ที่ไว้เทนุ 80% ทำ Dummie
- Data 2_en = data2_en_test_SL ที่ไว้เทส 20% ทำ Dummie
- Dummie Data เพราะ ข้อมูลในคอลัมน์ ['State', 'International plan', 'Voice mail plan'] เป็น ข้อมูล String ซึ่งเราไม่สามารถใช้ในการทำโมเดลได้ เราต้องใช้ข้อมูลที่เป็น Numeric
- เปลี่ยนค่าข้อมูลในคอลัมน์ ['Churn'] โดยให้ True, False ให้เป็น 1, 0 ตามลำดับ เพื่อเหมาะสมสำหรับการทำ โมเดล
- หลังจากทำ Dummie คอลัมน์จะเพิ่มขึ้นจาก 20 เป็น 72 คอลัมน์
- สลับคอลัมน์ให้ 'Churn' อยู่สุดท้ายง่ายต่อการมอง

```
[ ] data2_en_test = data2_en_test[['Account length', 'Area code', 'Number vmail messages',
'Total day minutes', 'Total day calls', 'Total day charge',
'Total eve minutes', 'Total eve calls', 'Total eve charge',
'Total night minutes', 'Total night calls', 'Total night charge',
'Total intl minutes', 'Total intl calls', 'Total intl charge',
'Customer service calls', 'State_AK', 'State_AL', 'State_AR',
'State_AZ', 'State_CA', 'State_CO', 'State_CT', 'State_DC', 'State_DE',
'State_FL', 'State_GA', 'State_HI', 'State_IA', 'State_ID', 'State_IL',
'State_IN', 'State_KS', 'State_KY', 'State_LA', 'State_MA', 'State_MD',
'State_ME', 'State_MI', 'State_MN', 'State_MO', 'State_MS', 'State_MT',
'State_NC', 'State_ND', 'State_NE', 'State_NH', 'State_NJ', 'State_NM',
'State_NV', 'State_NY', 'State_OH', 'State_OK', 'State_OR', 'State_PA',
'State_RI', 'State_SC', 'State_SD', 'State_TN', 'State_TX', 'State_UT',
'State_VA', 'State_VT', 'State_WA', 'State_WI', 'State_WV', 'State_WY',
'International plan_No', 'International plan_Yes', 'Voice mail plan_No',
'Voice mail plan_Yes','Churn']]
```

data2_en_test.head()

[ต่อ]



plot corr ของ data ก่อนจะทำ Dummie
เพื่อดูว่าแต่ละคอลัมน์มีความสัมพันธ์ระหว่างคอลัมน์ด้วย
กันเองเท่าไหร่ เพื่อใช้ในการเลือกคอลัมน์ในการทำ Model

จากภาพ จะเห็นว่าคอลัมน์
['Total day minute']
['Total Day Charge']
['Customer Service Call']

มีค่า col. ประมาณ 0.2 มากกว่าคอลัมน์
ถ้าต้องเลือกคอลัมน์ ให้เหลือน้อยสุด เพื่อกำหนาย
ก็ควรเลือก 3 คอลัมน์นี้

08

CLASSIFICATION

8.1 Decision Trees

```
[26] X_train = data1_en_train.iloc[:,71:] #ดึงข้อมูล 19 คอลัมน์แรกออกมานา  
y_train = data1_en_train.iloc[:,71:] #ใช้คอลัมน์สุดท้ายเป็น y  
  
[27] X_test = data2_en_test.iloc[:,71:] #ดึงข้อมูล 19 คอลัมน์แรกออกมานา  
y_test = data2_en_test.iloc[:,71:] #ใช้คอลัมน์สุดท้ายเป็น y
```

- 80% train = data1_en_train มี 72 columns
 - เราใช้ 71 columns = X_train
 - Columns 20 = y_train
- 20% test = data2_en_test มี 72 columns
 - เราใช้ 71 columns = X_test
 - Columns 72 = y_test

```
from sklearn.tree import DecisionTreeClassifier  
from sklearn.model_selection import cross_val_score, KFold  
from sklearn.metrics import classification_report  
from sklearn.metrics import accuracy_score  
  
ทดลองใช้ random_state ที่หลายเลขเพื่อเช็คค่าที่ดีที่สุด  
  
# Create Decision Tree classifier object  
clf0 = DecisionTreeClassifier(random_state=2,max_depth=7)  
  
clf1= DecisionTreeClassifier(random_state=1,max_depth=7)  
clf2= DecisionTreeClassifier(random_state=2,max_depth=7)  
clf3= DecisionTreeClassifier(random_state=3,max_depth=7)  
clf4= DecisionTreeClassifier(random_state=4,max_depth=7)  
clf5= DecisionTreeClassifier(random_state=5,max_depth=7)  
clf6= DecisionTreeClassifier(random_state=6,max_depth=7)  
clf7 = DecisionTreeClassifier(random_state=7,max_depth=7)  
clf8= DecisionTreeClassifier(random_state=8,max_depth=7)  
clf9 = DecisionTreeClassifier(random_state=9,max_depth=7)  
clf10 = DecisionTreeClassifier(random_state=10,max_depth=7)
```

```
clf1.fit(X_train, y_train)  
clf2.fit(X_train, y_train)  
clf3.fit(X_train, y_train)  
clf4.fit(X_train, y_train)  
clf5.fit(X_train, y_train)  
clf6.fit(X_train, y_train)  
clf7.fit(X_train, y_train)  
clf8.fit(X_train, y_train)  
clf9.fit(X_train, y_train)  
clf10.fit(X_train, y_train)
```

```
DecisionTreeClassifier  
DecisionTreeClassifier(max_depth=7, random_state=10)
```

Train

import package แผนภาพต้นไม้ และ Cross_val_Score , K_Fold ไว้คำนาย
ทดลองสร้างโมเดลที่มี random_state แตกต่างกัน เพื่อดูว่าตัวไหนให้ค่าคำนายนี้ดีสุด

8.1 Decision Trees

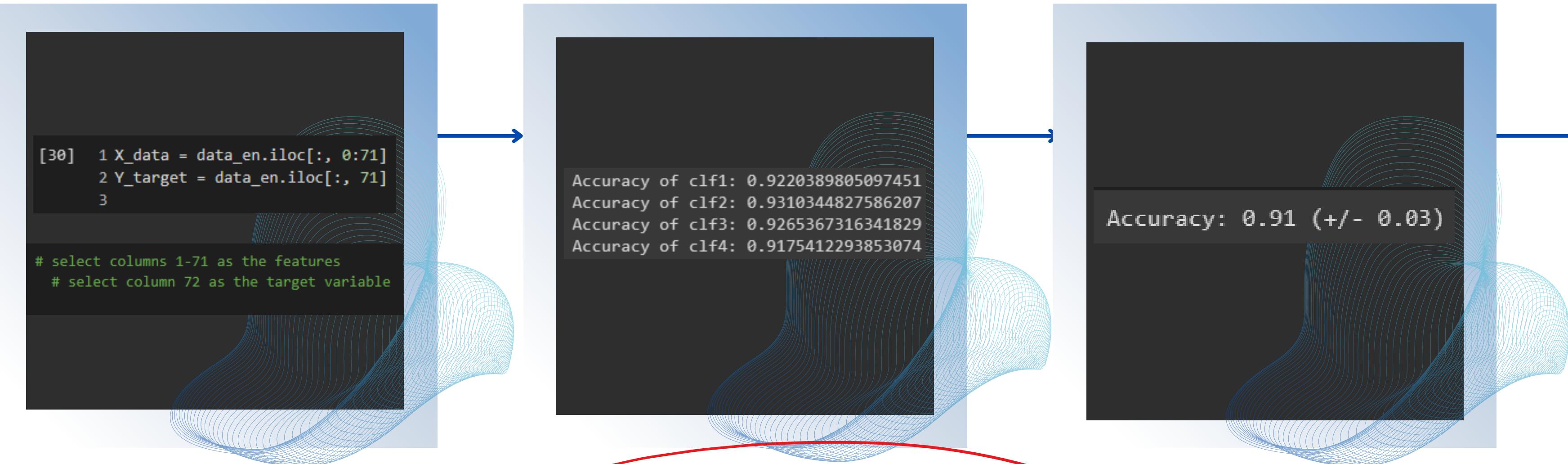
```
Predict  
[33]: y_pred_1 = clf1.predict(X_test)  
y_pred_2 = clf2.predict(X_test)  
y_pred_3 = clf3.predict(X_test)  
y_pred_4 = clf4.predict(X_test)  
y_pred_5 = clf5.predict(X_test)  
y_pred_6 = clf6.predict(X_test)  
y_pred_7 = clf7.predict(X_test)  
y_pred_8 = clf8.predict(X_test)  
y_pred_9 = clf9.predict(X_test)  
y_pred_10 = clf10.predict(X_test)  
  
Evaluate  
▶ print(f"Accuracy of clf1: {accuracy_score(y_test, y_pred_1)}") #This model is da best Accuracy of clf1: 0.9460269865067467  
print(f"Accuracy of clf2: {accuracy_score(y_test, y_pred_2)}")  
print(f"Accuracy of clf3: {accuracy_score(y_test, y_pred_3)}")  
print(f"Accuracy of clf4: {accuracy_score(y_test, y_pred_4)}")  
print(f"Accuracy of clf5: {accuracy_score(y_test, y_pred_5)}")  
print(f"Accuracy of clf6: {accuracy_score(y_test, y_pred_6)}")  
print(f"Accuracy of clf7: {accuracy_score(y_test, y_pred_7)}")  
print(f"Accuracy of clf8: {accuracy_score(y_test, y_pred_8)}")  
print(f"Accuracy of clf9: {accuracy_score(y_test, y_pred_9)}")  
print(f"Accuracy of clf10: {accuracy_score(y_test, y_pred_10)}")
```

This model is da best Accuracy of clf1: 0.94602698650674

```
Accuracy of clf1: 0.9460269865067467  
Accuracy of clf2: 0.9415292353823088  
Accuracy of clf3: 0.9415292353823088  
Accuracy of clf4: 0.9430284857571214  
Accuracy of clf5: 0.9445277361319341  
Accuracy of clf6: 0.9430284857571214  
Accuracy of clf7: 0.9430284857571214  
Accuracy of clf8: 0.9445277361319341  
Accuracy of clf9: 0.9430284857571214  
Accuracy of clf10: 0.9430284857571214  
  
▶ DT_score = accuracy_score(y_test, y_pred_1)  
]  
▶ print(f'Accuracy DecisionTree', DT_score)  
  
Accuracy DecisionTree 0.9460269865067467
```

Accuracy DecisionTree 0.9460269865067467 = 94%

8.1 Decision Trees



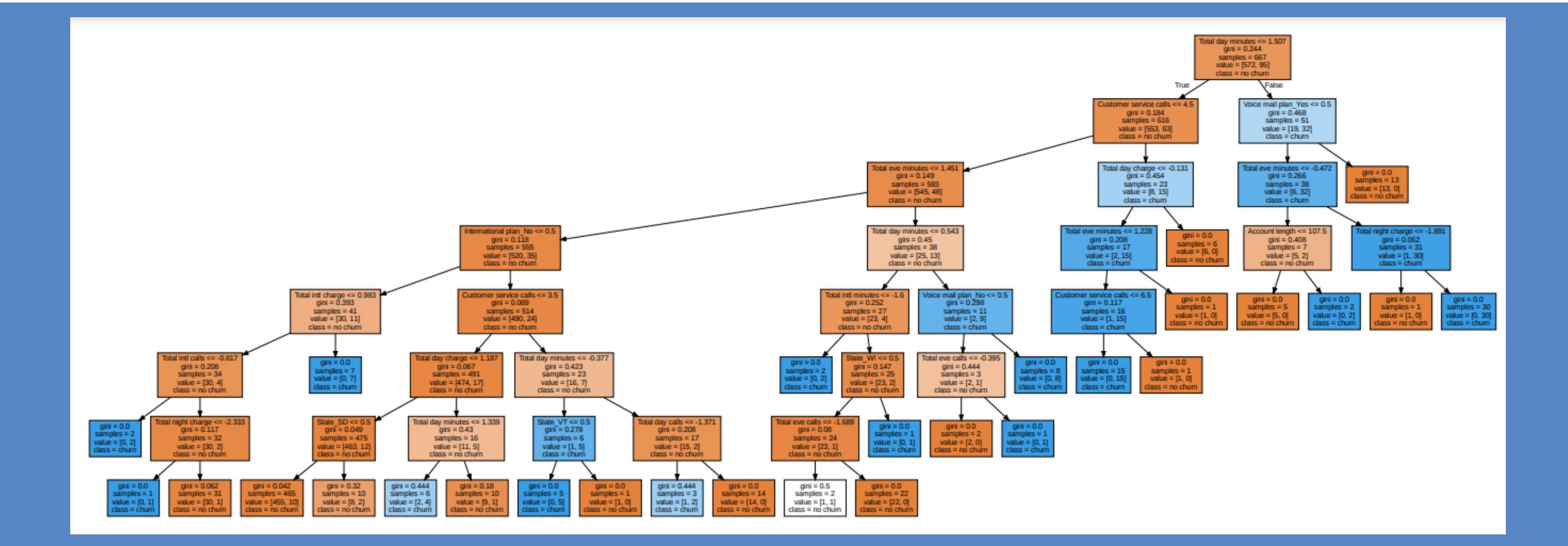
เลือกคอลัมน์ใน X_data = คอลัมน์ 1 - 71 ให้เป็น data สำหรับ
เลือกคอลัมน์ใน Y_data = คอลัมน์ 72 ให้เป็น target variable

`clf2 = DecisionTreeClassifier(random_state=2)`
มีค่าแม่นยำมาสุดจึงเลือกใช้ [random_state = 2] = 0.93

ทำ cross validation โดยใช้ KFold แบ่งชุดข้อมูลเป็น 10 ส่วน
ใช้โมเดล DecisionTreeClassifier

จะได้ว่า โมเดล Decision Tree มีความแม่นยำเฉลี่ยที่ 0.91 [91%]
โดยค่าความแปรปรวน [standard deviation] คือ 0.03

8.1 Decision Trees



08

CLASSIFICATION

8.2 KNN

```
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.metrics import accuracy_score  
from sklearn.model_selection import cross_val_score  
import numpy as np
```

Evaluate with test set

```
▶ knn1 = KNeighborsClassifier(n_neighbors=9, weights='distance')  
knn1.fit(X_train,y_train)  
yknn_pred1 = knn1.predict(X_test)
```

```
[42] from sklearn.metrics import accuracy_score  
# Assume y_true and y_pred are the true and predicted labels, respectively  
knn_score = accuracy_score(y_test, yknn_pred1)  
# Print the accuracy score  
print(" Model KNN n_neighbors = 9 มีความแม่นยำที่โมเดลท่านาย Churn ของ X_test =", knn_score )
```

Model KNN n_neighbors = 9 มีความแม่นยำที่โมเดลท่านาย Churn ของ X_test = 0.8590704647676162

- โค้ดนี้เป็นการฝึกและใช้งานโมเดล KNN ด้วยการกำหนด hyperparameters = 9
- ใช้ X_train และ y_train เพื่อฝึกโมเดล
- ทดสอบโมเดลด้วยชุดข้อมูล X_test และ y_test
- แสดงผลลัพธ์เป็น yknn_pred1 ซึ่งเป็นค่าที่โมเดลท่านาย Churn ของ X_test ออกมาได้

08

CLASSIFICATION

8.3 Naive_bayes

```
from sklearn.naive_bayes import MultinomialNB, GaussianNB
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import f1_score, recall_score

gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred0 = gnb.predict(X_test)
gnb_score0 = gnb.score(X_test, y_test)
gnb_score0
f1 = f1_score(y_test, y_pred0)
recall = recall_score(y_test, y_pred0)

print("โมเดล GaussianNB ได้ค่าทำนาย =", gnb_score0)
print("F1 ได้ค่าเฉลี่ยของความแม่นยำของการทำนาย =", f1) #(มีค่าระหว่าง 0 ถึง 1 ค่าที่มากกว่าแสดงว่าโมเดลสามารถทำนายได้มากขึ้น)
print("Recall ได้ค่าสัมประสิทธิ์การทำนายความถูกต้องของโมเดล =", recall) #(มีค่าระหว่าง 0 ถึง 1 ค่าที่มากกว่าแสดงว่าโมเดลสามารถทำนายได้มากขึ้น)
```

โมเดล GaussianNB ได้ค่าทำนาย = 0.5922038980509745 F1 ได้ค่าเฉลี่ยของความแม่นยำของการทำนาย
= 0.26881720430107525 Recall ได้ค่าสัมประสิทธิ์การทำนายความถูกต้องของโมเดล = 0.5263157894736842

ทำนายโมเดล Naive_bays ครั้งที่ 1 ได้ค่าทำนายโมเดล

```
โมเดล GaussianNB ได้ค่าทำนาย = 0.6356821589205397
F1 ได้ค่าเฉลี่ยของความแม่นยำของการทำนาย = 0.3154929577464789
Recall ได้ค่าสัมประสิทธิ์การทำนายความถูกต้องของโมเดล = 0.5714285714285714
```

- เนื่องจากค่าทำนายจากโมเดล มีค่าน้อยมาก = 0.636, 63%
จึงต้องปรับปรุงโมเดลแล้วทำใหม่เพื่อให้ได้ค่าดีขึ้น

8.3 Naive_bayes

	Total day minutes	Total day charge	Customer service calls	Churn
1	184.5	31.37	1	0
2	129.1	21.95	4	1
3	332.9	56.59	4	1
4	110.4	18.77	2	0
5	119.3	20.28	1	0
...
3329	134.7	22.90	2	0
3330	156.2	26.55	2	0
3331	231.1	39.29	3	0
3332	180.8	30.74	2	0
3333	234.4	39.85	0	0

3333 rows × 4 columns

- วิเคราะห์สหสัมพันธ์ : แต่ละรายการกับตัวแปรเป้าหมาย = corr
- มองหา features ที่มี corr. กับ target สูงๆ ใช้กำหนดในโมเดลนี้ = ['Total day minute'], ['Total Day Charge'], ['Customer Service Call']

คำนำหน้า Naive_bays ครั้งที่ 2 ได้คำนำหน้าโมเดล

โมเดล GaussianNB ได้ค่าทำนาย = 0.8740629685157422
F1 ได้ค่าเฉลี่ยของความแม่นยำของการทำนาย = 0.46835443037974683
Recall ได้ค่าสัมประสิทธิ์การทำนายความถูกต้องของโมเดล = 0.3894736842105263

จากโมเดล NaviyBay ที่ปรับปรุงได้คำนำหน้ามากขึ้นจาก 0.63% เป็น 0.874 %

F1 ได้ค่าเฉลี่ยของความแม่นยำของการทำนาย = 0.4
(มีค่าระหว่าง 0 ถึง 1 ค่าที่มากกว่าแสดงว่าโมเดลสามารถทำนายได้ดีมากขึ้น)

Recall ได้ค่าสัมประสิทธิ์การทำนายความถูกต้องของโมเดล = 0.3
(มีค่าระหว่าง 0 ถึง 1 ค่าที่มากกว่าแสดงว่าโมเดลสามารถทำนายได้ดีมากขึ้น)

สรุปการทำ Classifications

```
# Initialize figure
fig, ax = plt.subplots()
fig.set_figheight(7)
fig.set_figwidth(14)
fig.set_facecolor('white')

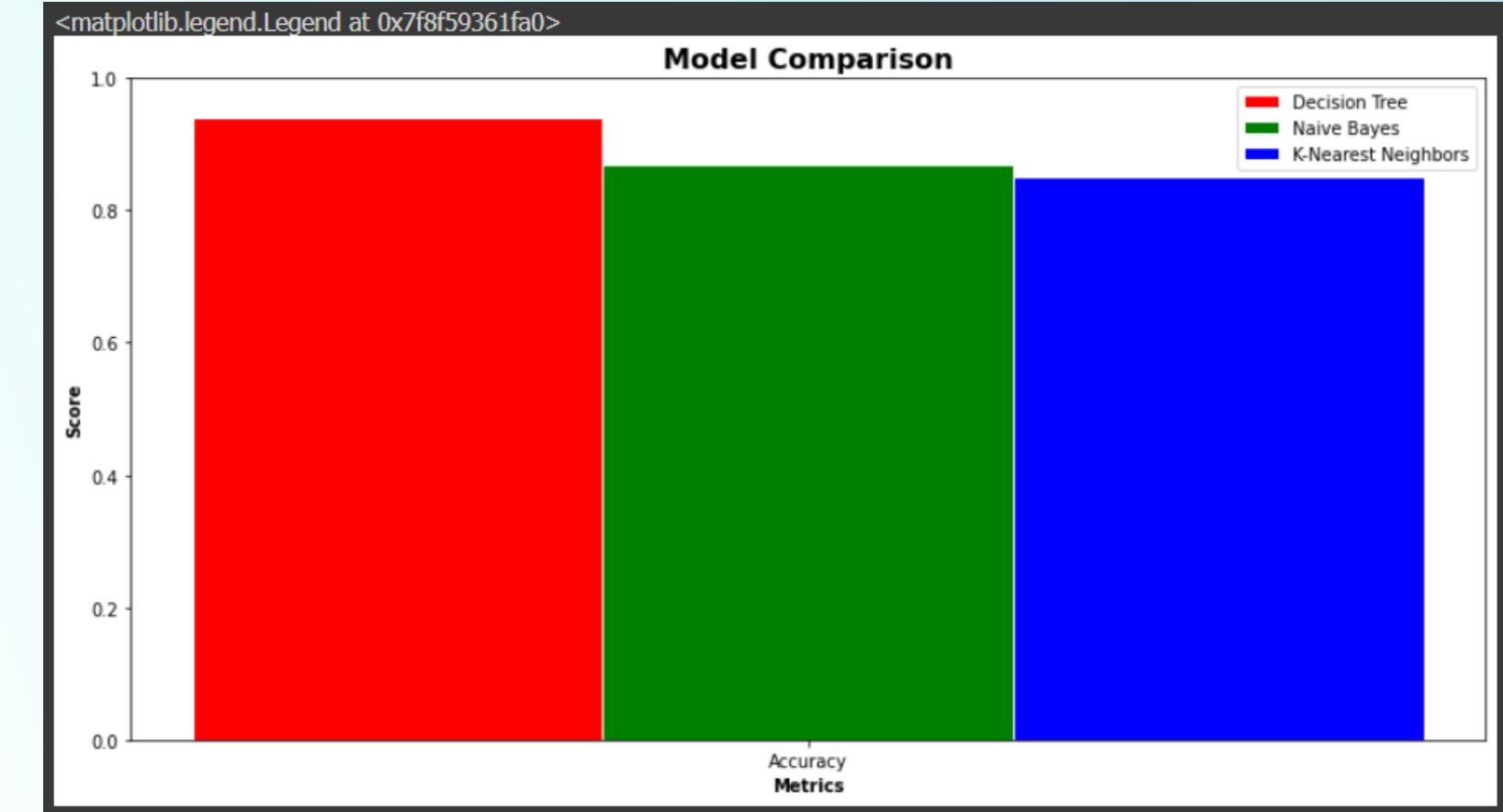
# Set bar size
barWidth = 0.2
DT_score = 0.94 # replace with your Decision Tree accuracy
gnb_score = 0.87 # replace with your Naive Bayes accuracy
knn_score = 0.85 # replace with your K-Nearest Neighbors accuracy

# Set position of bar on X axis
r1 = np.arange(1)
r2 = [x + barWidth for x in r1]
r3 = [x + barWidth for x in r2]

# Make the plot
ax.bar(r1, [DT_score], width=barWidth, edgecolor='white', color='red')
ax.bar(r2, [gnb_score], width=barWidth, edgecolor='white', color='green')
ax.bar(r3, [knn_score], width=barWidth, edgecolor='white', color='blue')

# Configure x and y axis
ax.set_xlabel('Metrics', fontweight='bold')
labels = ['Accuracy']
ax.set_xticks([r + (barWidth * 1) for r in range(1)], )
ax.set_xticklabels(labels)
ax.set_ylabel('Score', fontweight='bold')
ax.set_ylim(0, 1)

# Create legend & title
ax.set_title('Model Comparison', fontsize=16, fontweight='bold')
ax.legend(['Decision Tree', 'Naive Bayes', 'K-Nearest Neighbors'], loc='upper right')
```



Model

DecisionTree ได้ค่า Accuracy เท่ากับ **0.94**

KNeighbors ได้ค่า parameter ที่ดีที่สุดคือ K=9, weighted='distance'
ได้ค่า Accuracy เท่ากับ **0.85**

NaiveBayes ได้ค่า Accuracy เท่ากับ **0.87**

จึงสรุปว่าเลือกใช้โมเดล DecisionTree
เนื่องจากมีค่า Accuracy มากที่สุดเมื่อใช้ Data test ในการทดสอบเหมือนกัน

09

ASSOCIATION RULES

PREPARE DATASET

```
!pip install apyori
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
```

```
data["Churn"] = data["Churn"].replace({True: 1, False: 0})
dataAZ1 = data.copy()
dataAZ1['total_call_all_day'] = dataAZ1['Total day minutes'] + dataAZ1['Total eve minutes'] + dataAZ1['Total night minutes']
dataAZ1[['total_call_all_day','Churn']]
dataAZ1.head()
```

```
# create a new column 'total_call_all_day_range'
dataAZ1['total_call_all_day_range'] = pd.cut(dataAZ1['total_call_all_day'], [200, 500, 900], labels=['total_call_all_day 201-500', 'total_call_all_day 501-900'])
# create dummies
dummies = pd.get_dummies(dataAZ1['total_call_all_day_range'])
# concatenate dummies with original dataframe
dataAZ1 = pd.concat([dataAZ1, dummies], axis=1)
dataAZ1 = dataAZ1.drop(['total_call_all_day', 'total_call_all_day_range'], axis=1)
dataAZ1 = dataAZ1[['total_call_all_day 201-500', 'total_call_all_day 501-900','Churn']]
```

```
# ทำ one-hot encoding กับคอลัมน์ 'total_call_all_day'
onehot_data = pd.get_dummies(dataAZ1)

# สร้างกู้โดยใช้ apriori algorithm
frequent_itemsets = apriori(onehot_data, min_support=0.1, use_colnames=True)

# สร้างกู้ด้วย association rule
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=0)

# แสดงผลลัพธ์
rules
```

Install

เลือก colum น

1. **antecedents** คือ สิ่งที่เป็นก่อนหน้า [antecedent] ใน association rule นั้นๆ ซึ่งอาจเป็นสินค้าหรือคุณลักษณะที่ต้องการตรวจสอบว่าเมื่อมีการเลือกหนึ่งสิ่งนั้น สิ่งอื่นๆ จะเกิดขึ้นตามมา
2. **consequents** คือ สิ่งที่ตามมา [consequent] ใน association rule นั้นๆ ซึ่งอาจเป็นสินค้าหรือคุณลักษณะที่เกิดขึ้นตามจากสิ่งที่เป็นก่อนหน้านั้น
3. **antecedent support** คือ ความน่าจะเป็นของข้อมูลในชุดข้อมูลทั้งหมดที่มีคุณลักษณะตรงกับ antecedents
4. **consequent support** คือ ความน่าจะเป็นของข้อมูลในชุดข้อมูลทั้งหมดที่มีคุณลักษณะตรงกับ consequents
5. **support** คือ ความน่าจะเป็นของข้อมูลในชุดข้อมูลทั้งหมดที่มีคุณลักษณะตรงกับ antecedents และ consequents พร้อมกัน
6. **confidence** คือ ความน่าจะเป็นที่ consequents จะเกิดขึ้น โดยที่ antecedents เป็นจริง คือค่าเฉลี่ยกัน [support] ของ antecedents และ consequents ที่เกิดขึ้นพร้อมกัน [support] หารด้วยค่า support ของ antecedents
7. **lift** จะเห็นได้ว่าโอกาสที่ลูกค้าที่มี total_call_all_day อยู่ในช่วง 501-900 จะเป็นลูกค้าที่ยกเลิกบริการ [Churn_1] คือประมาณ 18.40%
8. **leverage** จะเห็นได้ว่าโอกาสที่ลูกค้าที่มี total_call_all_day อยู่ในช่วง 501-900 จะเป็นลูกค้าที่ใช้บริการต่อ [Churn_0] Churn_0 คือประมาณ 81.60%
9. **conviction**

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(total_call_all_day 501-900)	(Churn)	0.816082	0.144914	0.122712	0.150368	1.03763	0.00445	1.006418
1		(Churn) (total_call_all_day 501-900)	0.144914	0.816082	0.122712	0.846791	1.03763	0.00445	1.200440

10

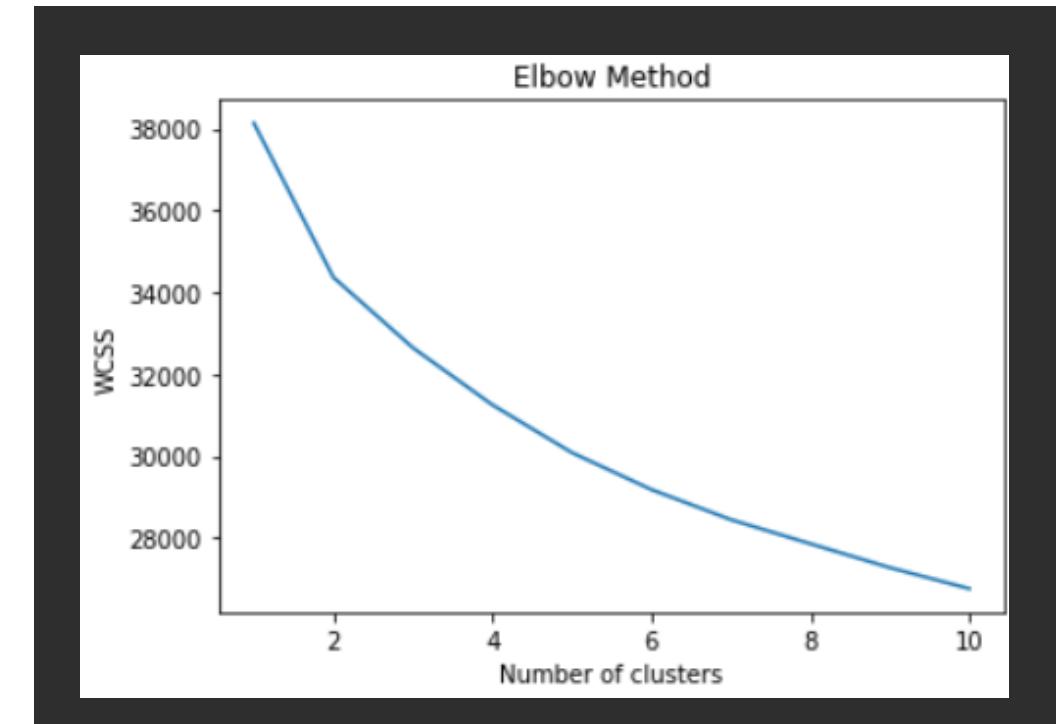
CLUSTERING K-MEAN

PREPARE DATASET



Plot PCA สร้างขึ้นเพื่อที่จะแสดงการกระจายของข้อมูลในรูป 2 มิติ หลังจากปรับค่าใน Columns ลงเป็น 2 องค์ประกอบหลัก

โดยใช้ PCA ในพล็อตแต่ละจุดแสดงถึง ลูกค้า สืบของจุดนั้นแสดงว่า ลูกค้ายกเลิกหรือไม่? [แดง:ยกเลิก, 粉:ไม่ยกเลิก] พล็อตนี้ช่วยให้เห็นภาพการแยกของทั้งสองกลุ่มง่ายมากขึ้นด้วยสีของจุด



Elbow Method ในการหาจำนวน cluster ที่เหมาะสมในการแบ่งกลุ่ม ข้อมูลจะทดสอบ k-means models ตั้งแต่ $k=1$ จนถึง $k=10$ และเก็บค่าจากการแบ่งกลุ่มแต่ละครั้งมาเก็บไว้แล้ว รูปกราฟ WCSS ต่อจำนวน cluster และดูจุดที่เป็น 'Elbow' เพื่อเลือกจำนวน cluster ในที่นี้คือ $K=2$ ที่เหมาะสมที่สุดสำหรับข้อมูลในการแบ่งกลุ่ม

```
pd.crosstab(df_telecom['Churn'], kmeans.labels_) #ท่านาย ว่า 0 ตรง 833 แต่ผิดพลาดไป 2017, ท่านายว่า 1 ตรง 403 ผิดไป 80
```

col_0	0	1
Churn		
0	2017	833
1	403	80

อันนี้ด้วยyyiy หรือบายให้หน่อยค้าบบ

10

CLUSTERING K-MEAN

```

1 # Initialize KMeans object with desired number of clusters
2 kmeans = KMeans(n_clusters=2)
3
4 # Fit the model to your data
5 kmeans.fit(df_telecom)
6 # kmeans1.fit(features)
7 # Predict the cluster labels for each data point
8 labels = kmeans.predict(df_telecom)
9 # labels1 = kmeans.predict(features)
10
11 # Get the coordinates of the cluster centroids
12 centroids = kmeans.cluster_centers_
13 # centroids1 = kmeans.cluster_centers_

1 # Select the features for clustering
2 X = df_telecom[[i for i in df_telecom.columns if i not in target_col]]
3
4 # Perform k-means clustering with 2 clusters
5 kmeans = KMeans(n_clusters=2, random_state=0)
6 clusters = kmeans.fit_predict(X)
7
8 # Add the cluster labels to the data
9 df_telecom['Cluster'] = clusters

```



- สร้างโมเดล clustering ด้วย KMeans โดยกำหนดจำนวน cluster เป็น **n_clusters = 2**
- fit Model KMeans ด้วยข้อมูล **df_telecom**
- predict cluster labels สำหรับแต่ละ data point และเก็บไว้ในตัวแปร **labels**
- เมื่อได้ cluster labels แล้ว ทำการคำนวณ **centroid** ของแต่ละ **cluster** เก็บไว้ในตัวแปร **centroids**
- เพิ่มคอลัมน์ ['Cluster'] ที่มีค่า centroid ที่ได้จาก Model

```

1 pd.crosstab(df_telecom['Churn'], kmeans.labels_)

col_0      0      1
Churn
0        2017   833
1         403    80

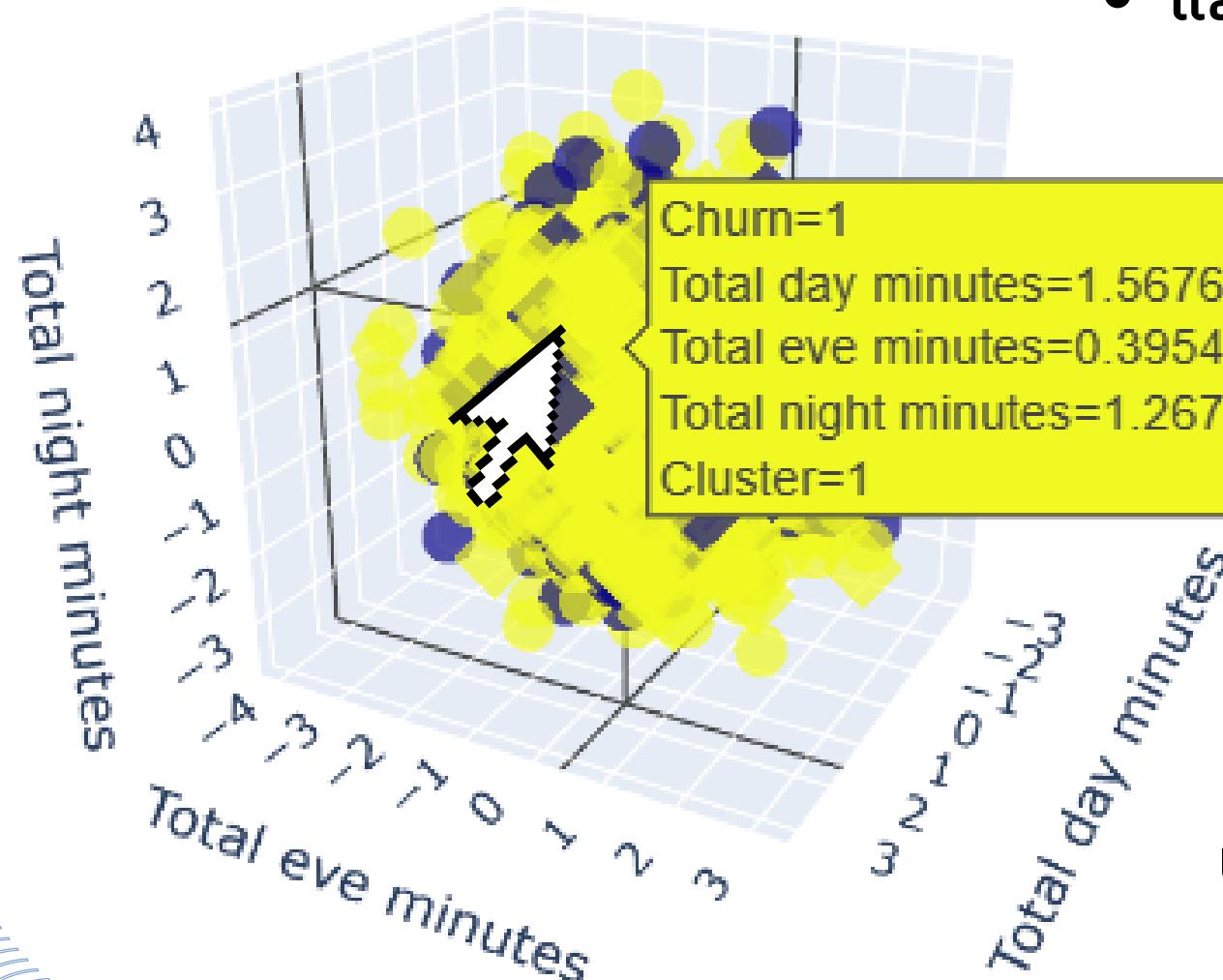
```



Confusion matrix

หมายความว่า ไม่ยกเลิกการใช้บริการ ตรง 833 แต่ผิดพลาดไป 2017
 หมายความว่า ยกเลิกการใช้บริการ ตรง 403 แต่ผิดพลาดไป 80

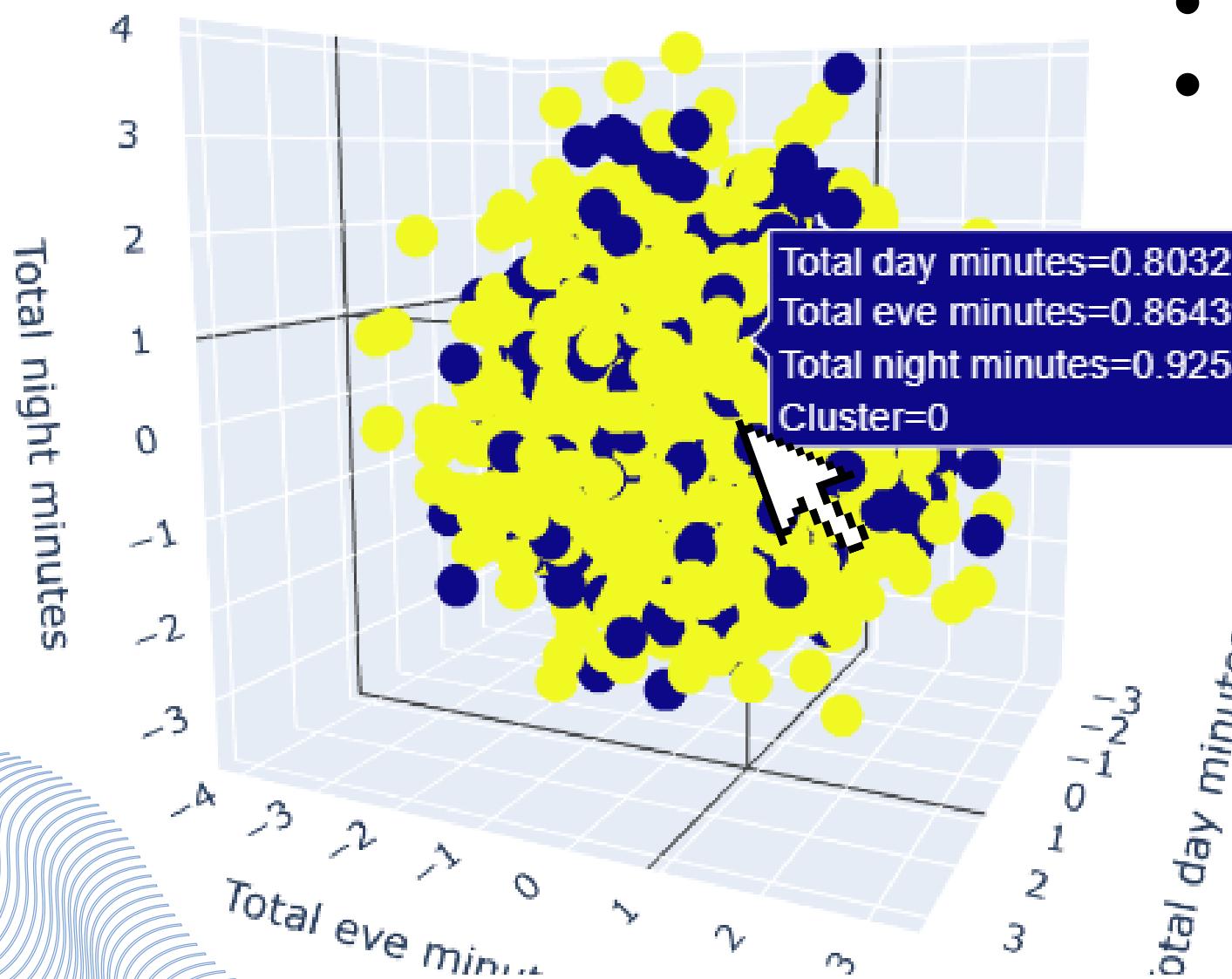
- Plot กราฟแบบ 3 มิติโดย
- กำหนดแกน x,y,z ของกราฟด้วย
- ค่า 'Total day minutes', 'Total eve minutes', 'Total night minutes' ตามลำดับ
- และกำหนดให้สีของจุดในกราฟตาม Cluster [0,1]
- และสัญลักษณ์ตาม Churn และกำหนดความโปร่งแสงของจุดในกราฟด้วย opacity=0.7



- ค่า 'Total day minutes', 'Total eve minutes', 'Total night minutes'
- มีค่า = **1.56, 0.39, 1.26**
- จะอยู่ในกลุ่มของ Churn=1

- จากกราฟทำให้ทราบว่าลูกค้าอาจมีความน่าจะเป็นที่จะยกเลิกบริการในอนาคต'
- ค่า 3 คอลัมน์ที่ plot อาจจะเป็นตัวชี้วัดที่ โดยค่าเหล่านี้อาจมีความสัมพันธ์กับการใช้บริการของลูกค้าในแต่ละช่วงเวลาของวัน [กลางวัน, เย็น, กลางคืน]
- ซึ่งอาจส่งผลต่อความพึงพอใจและการตัดสินใจของลูกค้าในการใช้บริการในอนาคต

- Plot กราฟแบบ 3 มิติโดย
- กำหนดแกน x,y,z ของกราฟด้วย
- ค่า 'Total day minutes', 'Total eve minutes', 'Total night minutes' ตามลำดับ
- และกำหนดให้สีของจุดในกราฟตาม Cluster [0,1]
- และสัญลักษณ์ตาม Churn และกำหนดความโปร่งแสงของจุดในกราฟด้วย opacity=0.7



- ค่า 'Total day minutes', 'Total eve minutes', 'Total night minutes'
- มีค่า = **0.80, 0.86, 0.92**
- จะอยู่ในกลุ่มของ Churn=0

THANK YOU



นายณัฐกร ณ พวงแก้ว 633021015-9



นางสาววิทรดี นาดี 633021022-2



นางสาวลุจิรา มั่นหาท้าว 633021025-6



นางสาวศศิวิมล วิลาชัย 633021023-0



นางสาวอรัญญา จันทร์ 633021028-0