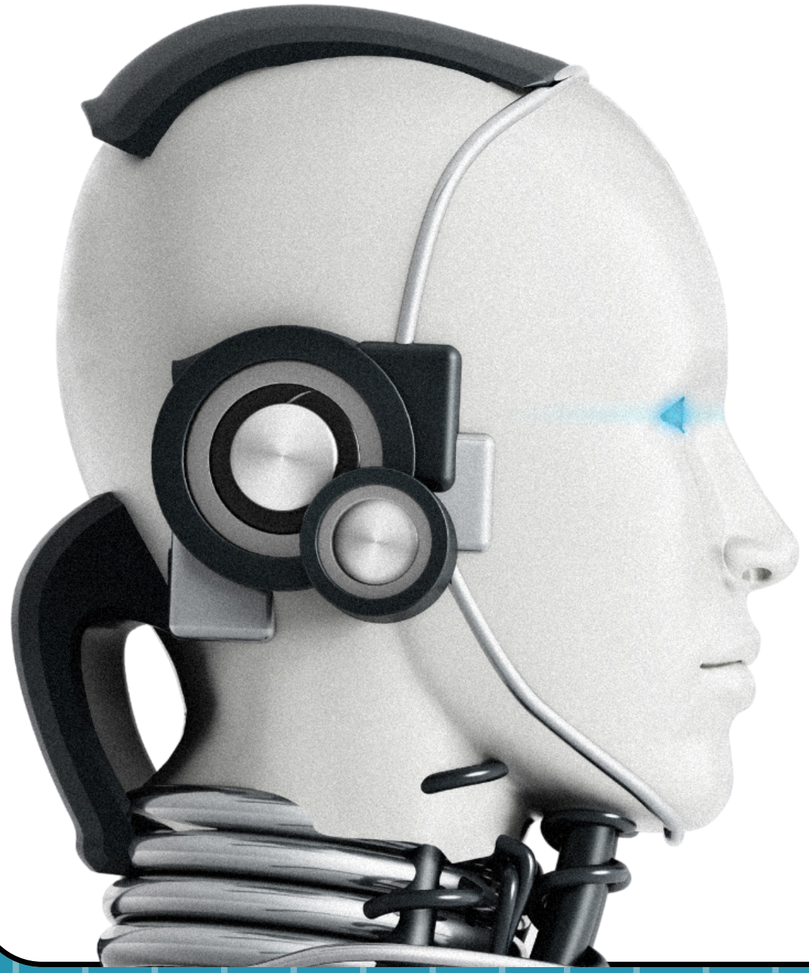


SIMULATE2 ALGORITHM PRECISTION

G.Starpopo

AI for robotic

วัตถุประสงค์

เป็นการจำลองการคาดการณ์ว่าผลลัพธ์จากข้อมูลที่มีอยู่ผ่านattribute ของข้อมูล ลูกค้าธนาคารที่เลิกใช้บริการไปแล้วหรือยังใช้อยู่ โดยเรายังไม่ได้ใช้ข้อมูลของ นนร. มาสร้างโมเดล
โดยวัตถุประสงค์ที่ว่าหาก เรามีข้อมูลคุณลักษณะของ นนร. แต่ละคนตำแหน่งอะไรบ้าง ในปีถัดไป เราก็อาจจะสามารถคาดการณ์ ตำแหน่ง นักเรียนบังคับบัญชาได้

ANGORITHM

เหตุผลที่เลือกใช้ Decision Tree และ KNN:

1. Decision Tree:
   - ทนทานต่อข้อมูลที่มีลักษณะเฉพาะ: Decision Tree สามารถจัดการกับข้อมูลที่มีลักษณะเฉพาะ (non-linear relationships) ได้ดีเพราะมันแบ่งข้อมูลตามเงื่อนไขต่างๆ
   - ตีความง่าย: การตีความผลลัพธ์จาก Decision Tree ง่ายกว่า เพราะสามารถแสดงเป็นกราฟที่เข้าใจได้ง่าย
   - ไม่ต้องการการปรับมาตรวัดระยะห่าง: ไม่มีความจำเป็นในการเลือกหรือปรับมาตรวัดระยะห่าง เช่น Euclidean distance
   - ข้อดี: เข้าใจง่าย เพราะมันเหมือนต้นไม้ที่แสดงการตัดสินใจทีละขั้นตอน
   - ข้อเสีย: อาจทำงานได้ไม่ดีถ้าข้อมูลมีความซับซ้อนมากหรือมีข้อมูลเสียง (noise)
2. KNN:
   - การเรียนรู้ที่ไม่ต้องฝึก (Lazy Learning): KNN ไม่ต้องใช้เวลาในการฝึก (training) มาก เพียงแต่ใช้เวลาในการคำนวณระยะห่างในระหว่างการทำนาย
   - ยืดหยุ่นสูง: สามารถปรับการทำงานตามลักษณะของข้อมูลได้โดยการเลือกค่า k และมาตรวัดระยะห่างที่เหมาะสม
   - ไม่ต้องสร้างโมเดล: ไม่มีการสร้างโมเดลที่ต้องจัดเก็บ การทำนายจะอิงตามข้อมูลที่มีอยู่ทั้งหมด
   - ข้อดี: ทำงานได้ดีเมื่อข้อมูลมีความชัดเจนและมีรูปแบบที่เรียบง่าย
   - ข้อเสีย: อาจช้าและใช้ทรัพยากรมากเมื่อต้องทำงานกับข้อมูลจำนวนมาก

DATA

ข้อมูล ลูกค้าธนาคารที่เลิกใช้บริการไปแล้วหรือยังใช้อยู่

HEADER ของข้อมูล

1	RowNumb	CustomerI	Surname	CreditScor	Geography	Gender	Age	Tenure	Balance	NumOfPro	HasCrCard	IsActiveMe	EstimatedS	Exited
2	1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101348.9	1
3	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.6	0
4	3	15619304	Onio	502	France	Female	42	8	159660.8	3	1	0	113931.6	1
5	4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.63	0
6	5	15737888	Mitchell	850	Spain	Female	43	2	125510.8	1	1	1	79084.1	0
7	6	15574012	Chu	645	Spain	Male	44	8	113755.8	2	1	0	149756.7	1
8	7	15592531	Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0
9	8	15656148	Obinna	376	Germany	Female	29	4	115046.7	4	1	0	119346.9	1
10	9	15792365	He	501	France	Male	44	4	142051.1	2	0	1	74940.5	0
11	10	15592389	H?	684	France	Male	27	2	134603.9	1	1	1	71725.73	0
12	11	15767821	Bearce	528	France	Male	31	6	102016.7	2	0	0	80181.12	0
13	12	15737173	Andrews	497	Spain	Male	24	3	0	2	1	0	76390.01	0
14	13	15632264	Kay	476	France	Female	34	10	0	2	1	0	26260.98	0
15	14	15691483	Chin	549	France	Female	25	5	0	2	0	0	190857.8	0

DECISIONTREECLASSIFIE

DecisionTreeClassifier เป็นโมเดลการเรียนรู้ด้วยเครื่อง (Machine Learning) ที่ใช้สำหรับการจัดหมวดหมู่ข้อมูล (Classification) โดยสร้างโมเดลในรูปแบบของโครงสร้างต้นไม้ (Tree Structure)

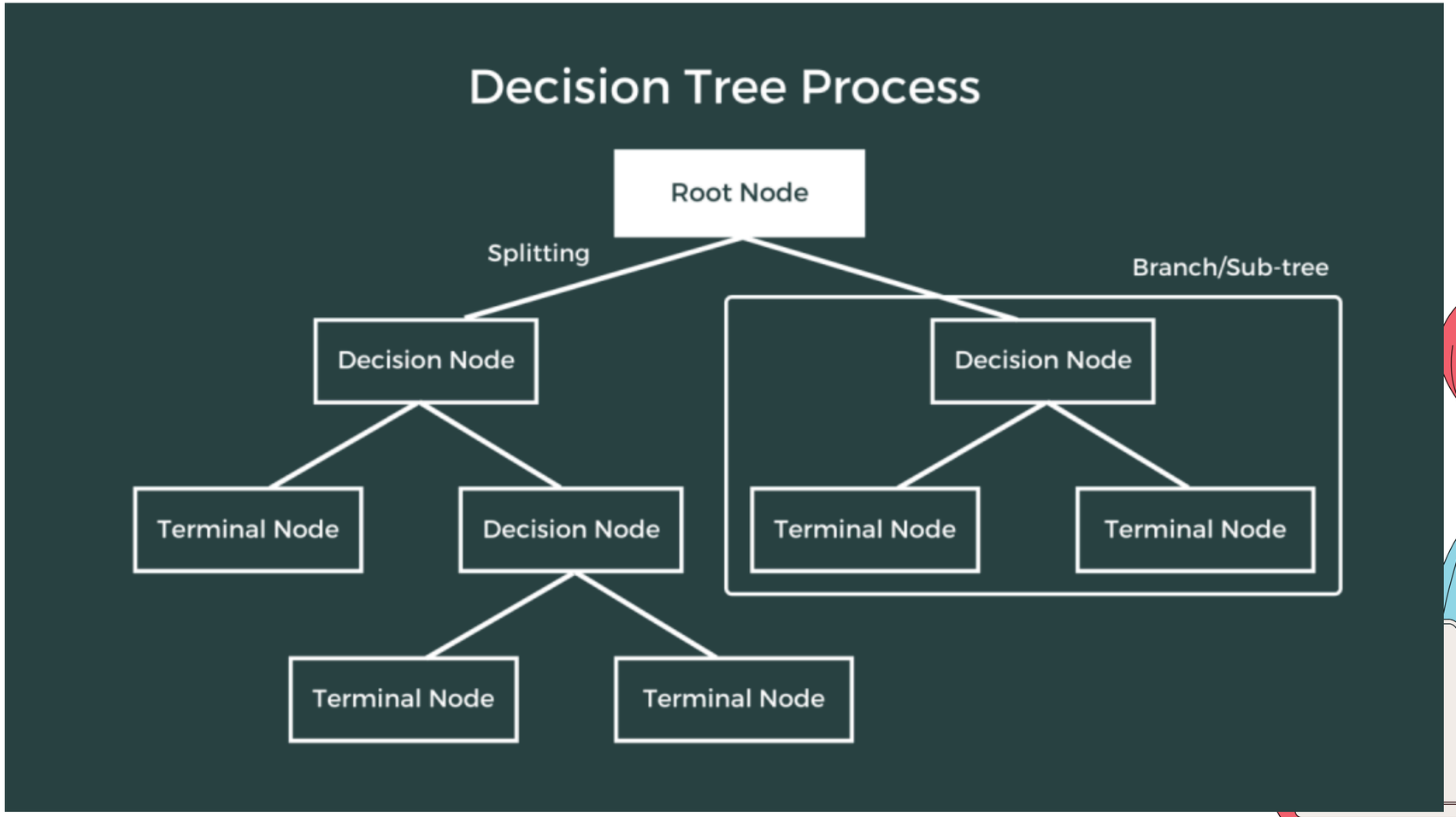
- **Root Node (โหนดราก)**: เป็นโหนดเริ่มต้นของต้นไม้การตัดสินใจ มันคือจุดที่เริ่มการแบ่งข้อมูลครั้งแรก โดยใช้คุณสมบัติที่ดีที่สุดในการแบ่งข้อมูลออกเป็นกลุ่มย่อยๆ

- **Internal Node (โหนดภายใน)**: โหนดภายในคือตำแหน่งที่ทำการแบ่งข้อมูลเพิ่มเติมตามคุณสมบัติอื่นๆ โหนดเหล่านี้จะมีเส้นทางออกไปยังโหนดลูก (Child Node) หลายๆ อัน ขึ้นอยู่กับการตัดสินใจของโหนดนั้นๆ

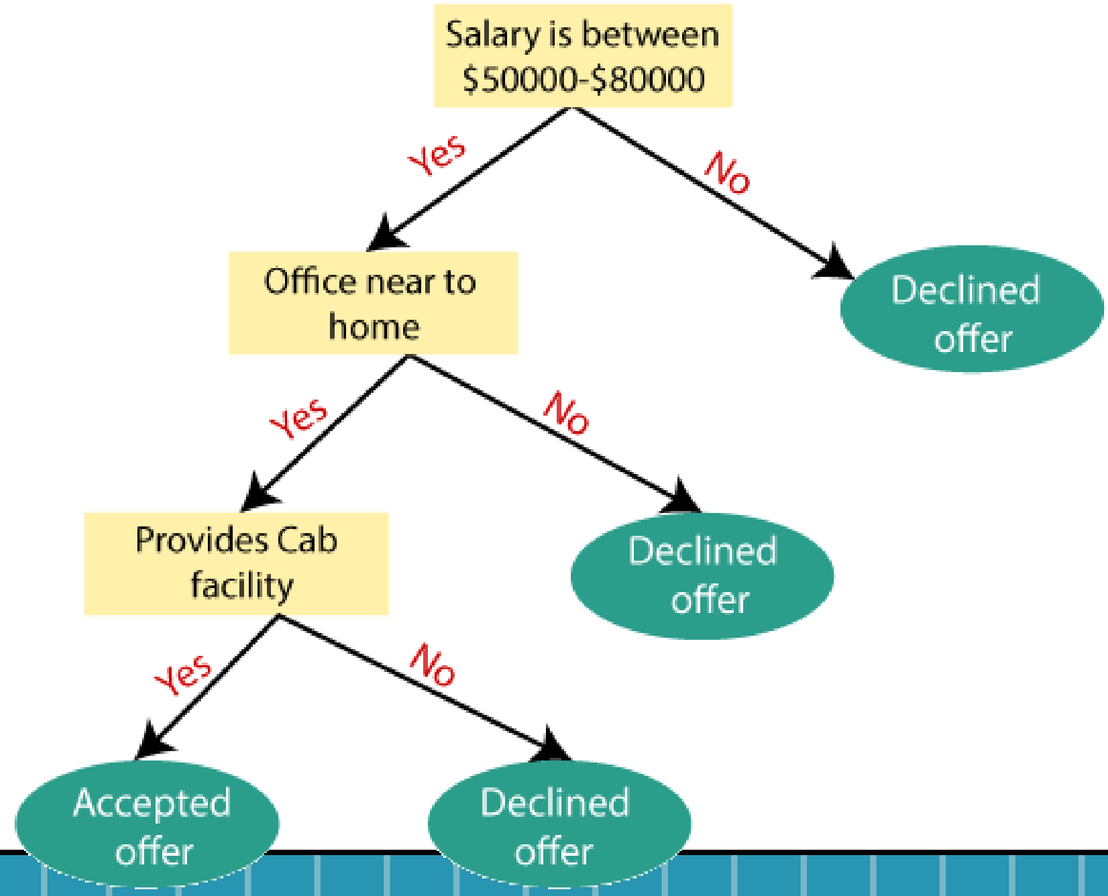
- **Leaf Node (โหนดใบ)**: โหนดใบคือจุดสิ้นสุดของเส้นทางในต้นไม้การตัดสินใจ และจะให้ผลลัพธ์สุดท้ายหรือการจำแนกประเภทนั้นๆ ไม่มีการแบ่งข้อมูลต่อจากโหนดใบนี้

- **Decision Node (โหนดการตัดสินใจ)**: โหนดการตัดสินใจเป็นโหนดที่มีการเลือกเส้นทางตามเกณฑ์การตัดสินใจ เช่น การเปรียบเทียบค่าของคุณสมบัติ โดยโหนดการตัดสินใจจะพิจารณาจากข้อมูลที่มีอยู่และเลือกเส้นทางที่เหมาะสมในการดำเนินการต่อ

DECISIONTREECLASSIFIE



DECISIONTREECLASSIFIE



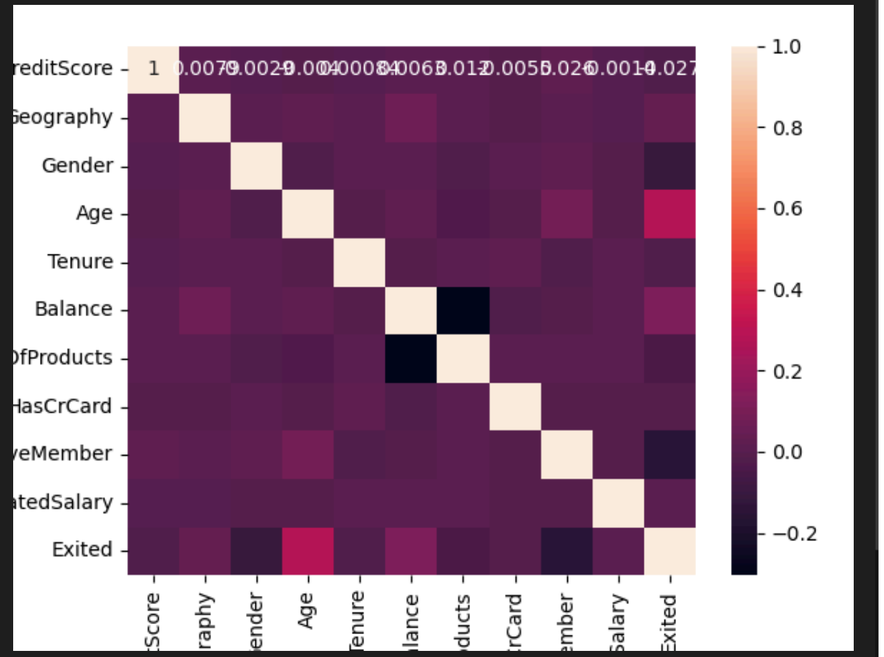
CODE

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay
from sklearn.tree import DecisionTreeClassifier, plot_tree

```

```
# Load the dataset
csv = pd.read_csv("Churn Modeling.csv")
df = pd.DataFrame(csv)

df = df.drop('RowNumber', axis=1)
df = df.drop('CustomerId', axis=1)
df = df.drop('Surname', axis=1)

print(df.info())

# Initialize LabelEncoder
le = LabelEncoder()

# Encode categorical variables
df['Geography'] = le.fit_transform(df['Geography'])
df['Gender'] = le.fit_transform(df['Gender'])

# Check for NaN values
nan_count = df.isnull().sum().sum()
print(f"Number of NaN: {nan_count}")
```

CODE

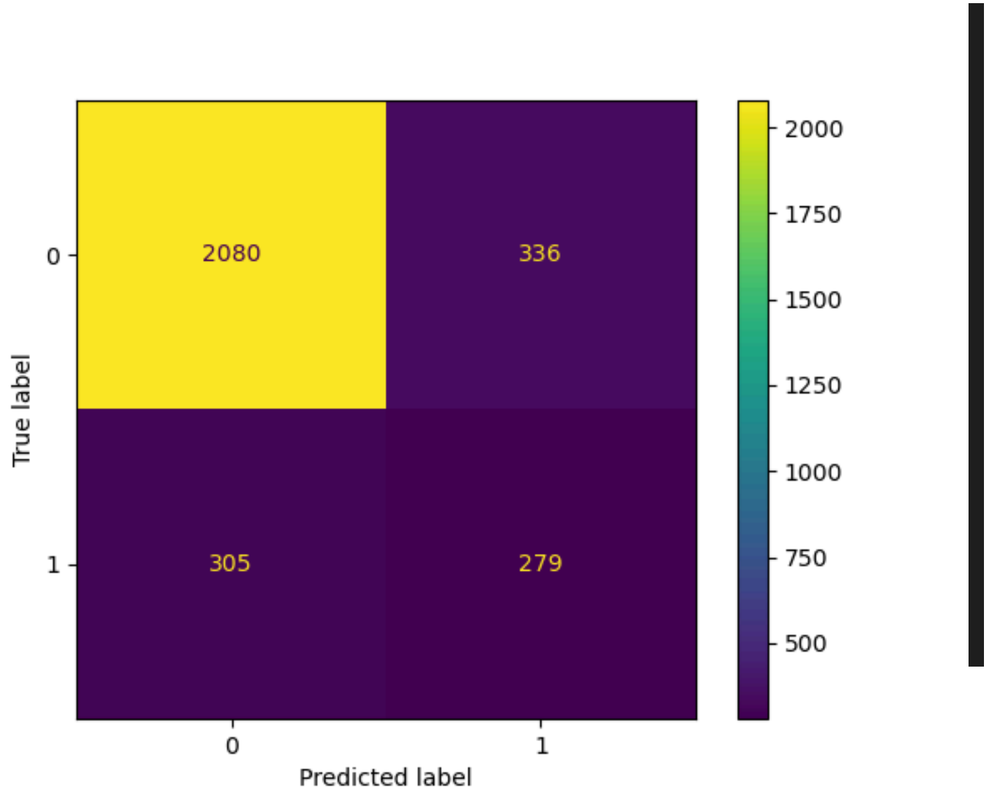
```python
# Fill NaN values with the mean of the column
df = df.fillna(df.mean())

# Plot correlation matrix
corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True)
plt.show()

# Prepare data for modeling
X = df.drop('Exited', axis=1)
y = df['Exited']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Binning certain columns in training data
X_train['Age'] = pd.cut(X_train['Age'],
                        bins=[0, 30, 40, 50, 60, 70, 80, 90, 100],
                        labels=['0', '30', '40', '50', '60', '70', '80', '90'])

X_train['Balance'] = pd.cut(X_train['Balance'],
                            bins=[0, 50000, 100000, 150000, 200000, 250000],
                            labels=['0', '50000', '100000', '150000', '200000'])

X_train['EstimatedSalary'] = pd.cut(X_train['EstimatedSalary'],
                                    bins=[0, 50000, 100000, 150000, 200000, 250000],
                                    labels=['0', '50000', '100000', '150000', '200000'])

print(X_train[:100])
```



CODE

```python
# Train a decision tree model
model = DecisionTreeClassifier()
model.fit(X_train, y_train)

# Predict and evaluate the model
y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print(f"Score: {accuracy_score(y_test, y_pred)}")

# Plot confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

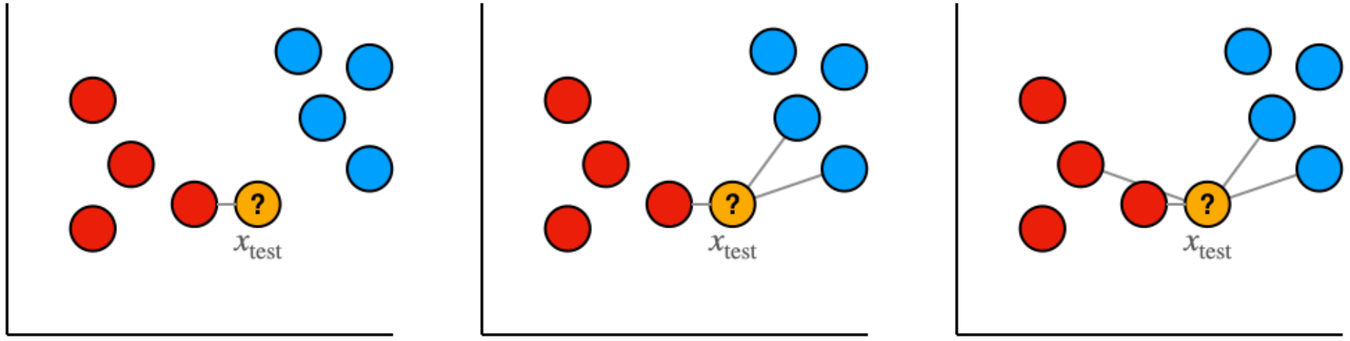
```
Score: 0.7863333333333333
```



PRECISION

KNN

KNN (K-Nearest Neighbors) คือ อัลกอริธึมการเรียนรู้ของเครื่องที่ใช้สำหรับการจำแนกประเภท (classification) หรือการคาดการณ์ค่า (regression) โดยวิธีการทำงานหลักของ KNN คือการหาค่า "K" ที่ใกล้เคียงที่สุด (nearest neighbors) จากจุดข้อมูลที่ต้องการคาดการณ์หรือจำแนกประเภท และใช้ข้อมูลของเพื่อนบ้าน(ข้อมูลที่อยู่ใกล้กัน)เหล่านั้นในการตัดสินใจ



$k = 1$

Nearest point is red, so x_{test} classified as red

$k = 3$

Nearest points are {red, blue, blue} so x_{test} classified as blue

$k = 4$

Nearest points are {red, red, blue, blue} so classification of x_{test} is not properly defined

KNN

หลักการทำงานของ KNN

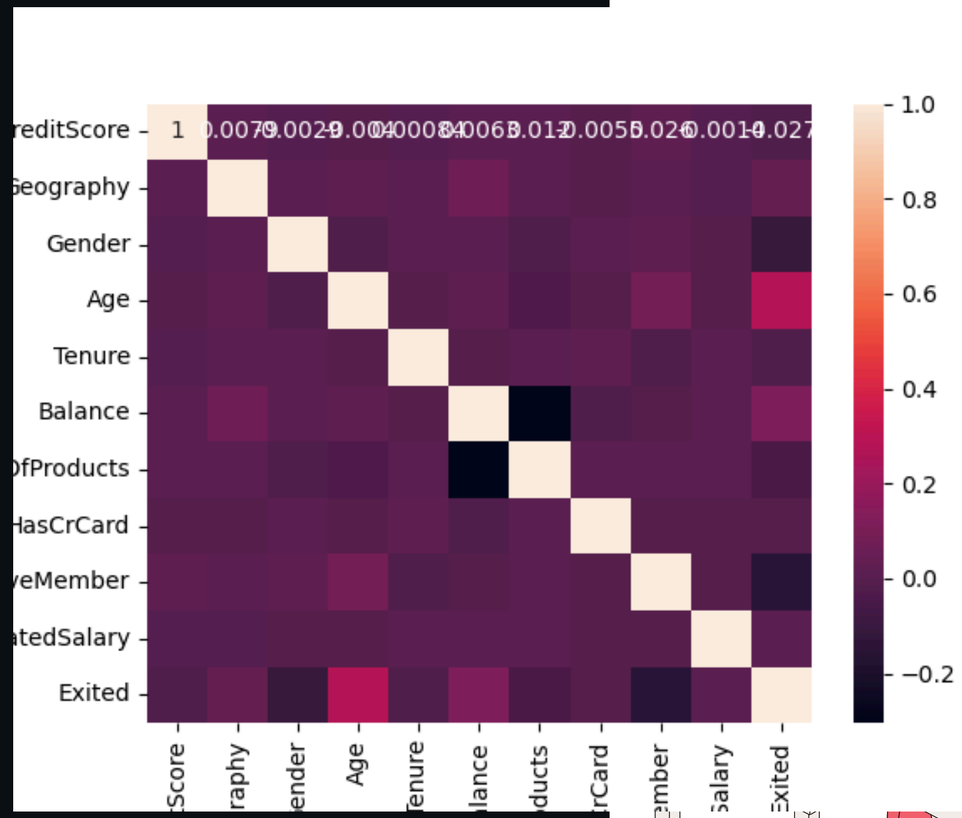
1. การเลือกค่า K: เลือกจำนวน K ที่เป็นจำนวนเพื่อนบ้านที่ใกล้เคียงที่สุดที่ใช้ในการคำนวณ
2. การคำนวณระยะทาง: คำนวณระยะทางระหว่างจุดข้อมูลที่ต้องการคาดการณ์กับจุดข้อมูลในชุดข้อมูลที่มีอยู่ โดยปกติจะใช้ระยะทาง Euclidean
3. การเลือก K จุดที่ใกล้ที่สุด: หาค่า K จุดที่มีระยะทางใกล้เคียงที่สุด
4. การจำแนกประเภทหรือคาดการณ์ค่า:
    - สำหรับการจำแนกประเภท: นับจำนวนประเภทของจุดเพื่อนบ้านที่ใกล้ที่สุด และกำหนดประเภทที่พบมากที่สุดให้กับจุดที่ต้องการคาดการณ์
    - สำหรับการคาดการณ์ค่า: คำนวณค่าเฉลี่ย (หรือค่าอื่น ๆ) ของค่าใน K จุดที่ใกล้ที่สุดและใช้ค่าเฉลี่ยนี้เป็นการคาดการณ์

CODE

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler

# Load the dataset
csv = pd.read_csv("Churn Modeling.csv")
df = pd.DataFrame(csv)

df = df.drop('RowNumber', axis=1)
df = df.drop('CustomerId', axis=1)
df = df.drop('Surname', axis=1)

print(df.info())

```

CODE

```python
# Initialize LabelEncoder
le = LabelEncoder()

# Encode categorical variables
df['Geography'] = le.fit_transform(df['Geography'])
df['Gender'] = le.fit_transform(df['Gender'])

# Check for NaN values
nan_count = df.isnull().sum().sum()
print(f"Number of NaN: {nan_count}")

# Fill NaN values with the mean of the column
df = df.fillna(df.mean())

# Plot correlation matrix
corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True)
plt.show()

```



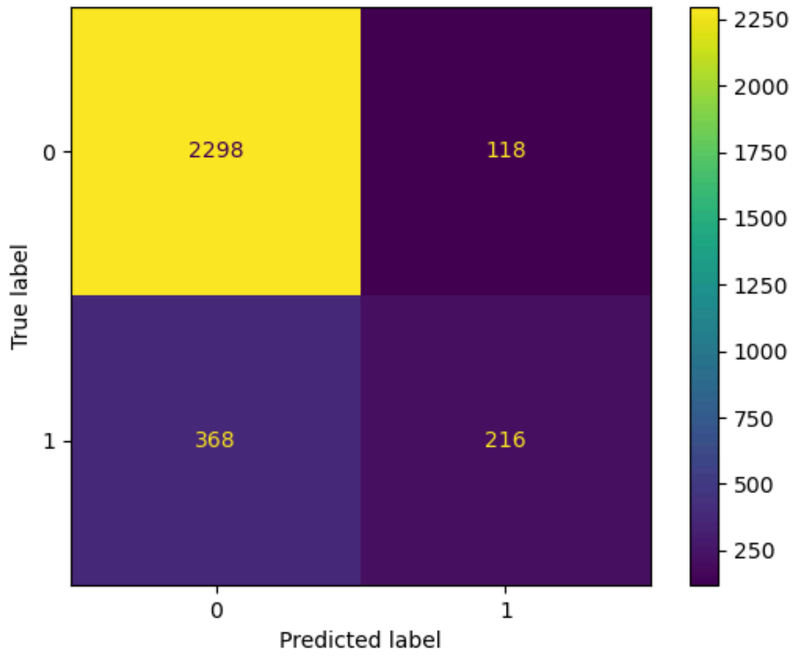
CODE

```
39      # Prepare data for modeling
40      X = df.drop('Exited', axis=1)
41      y = df['Exited']
42      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
43
44      # Standardize the data
45      scaler = StandardScaler()
46      X_train = scaler.fit_transform(X_train)
47      X_test = scaler.transform(X_test)
48
49      # Train a KNN model
50      knn = KNeighborsClassifier(n_neighbors=5)
51      knn.fit(X_train, y_train)
52
53      # Predict and evaluate the model
54      y_pred = knn.predict(X_test)
55      cm = confusion_matrix(y_test, y_pred)
56      print(f"Score: {accuracy_score(y_test, y_pred)}")
57
```

CODE

```
58      # Plot confusion matrix
59      disp = ConfusionMatrixDisplay(confusion_matrix=cm)
60      disp.plot()
61      plt.show()
```

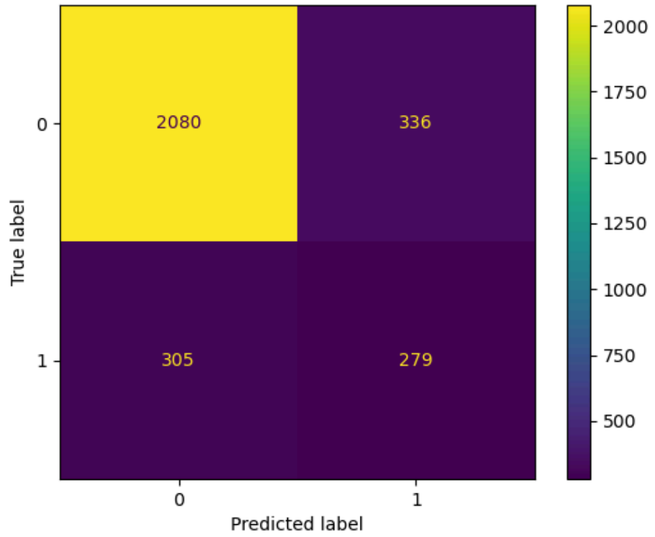
```
None
Number of NaN: 0
Score: 0.838
```



PRECISION

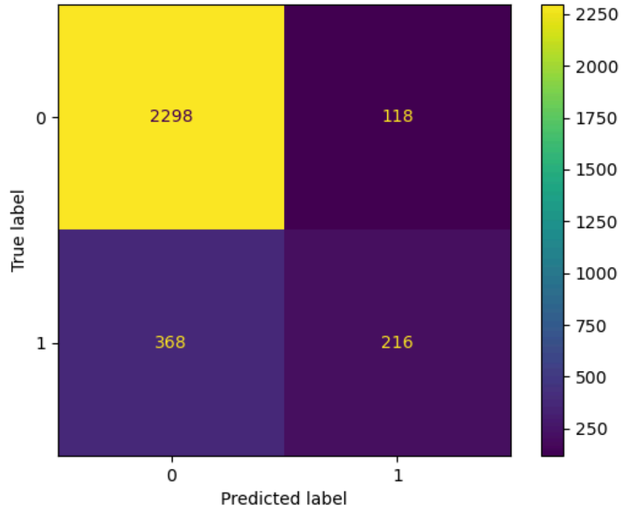
SUMMARY

DecisionTreeClassifie




```
Score: 0.7863333333333333
```

KNN




```
Score: 0.838
```

THANK YOU