Teerapat_Kotanart_Loan_Approval_SVM_KNN

File Edit View Insert Runtime Tools Help All changes saved

Share

Files

..

sample_data

loan_approval.csv

Disk 75.21 GB available

+ Code + Text

RAM Disk

Gemini

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files  loan_approval.csv
• loan_approval.csv(text/csv) - 36638 bytes, last modified: 1/11/2025 - 100% done
Saving loan_approval.csv to loan_approval.csv
```

```
[54] import pandas as pd
     df = pd.read_csv('loan_approval.csv')
```

```
[55] df.head() # View the first 5 rows
     df.info() # Check for missing values and data types
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Gender           601 non-null    object
 1   Married          611 non-null    object
 2   Dependents       599 non-null    object
 3   Education        614 non-null    object
 4   Self_Employed    582 non-null    object
 5   Income(dollar)   614 non-null    float64
 6   Coapplicant      610 non-null    object
 7   Loan_Amount      614 non-null    int64
 8   Term(month)      600 non-null    float64
 9   loan_History     564 non-null    float64
 10  Area             614 non-null    object
 11  Status           614 non-null    object
dtypes: float64(3), int64(1), object(8)
memory usage: 57.7+ KB
```

```
[58] numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns

     df[numerical_cols] = df[numerical_cols].fillna(df[numerical_cols].mean())
```

```
[72] from sklearn.preprocessing import LabelEncoder

     le = LabelEncoder() # สร้าง LabelEncoder

     categorical_cols = df.select_dtypes(include=['object']).columns # เลือกคอลัมน์ที่เป็นข้อความ (object type)

     # แปลงข้อมูลในคอลัมน์ที่เป็นข้อความด้วย LabelEncoder
     for col in categorical_cols:
         df[col] = le.fit_transform(df[col])

     print(df.head()) # ดูผลลัพธ์
```

```
   Gender  Married  Dependents  Education  Self_Employed  Income(dollar)  \
0       1        1           0          1              0        144200.0
1       0        1           3          1              0        183000.0
2       1        2           1          0              0        188000.0
3       1        2           0          0              0        195000.0
4       0        1           0          1              0        196300.0

   Coapplicant  Loan_Amount  Term(month)  loan_History  Area  Status
0            0      3500000        360.0      1.000000     2       0
1            0            0        360.0      0.000000     2       0
2            0      6100000        360.0      0.842199     0       0
3            1     13500000        360.0      1.000000     0       0
4            0      5300000        360.0      1.000000     1       1
```

```
[73] # กำหนดว่า Target เป็นคอลัมน์สุดท้าย
     X = df.iloc[:, :-1] # เลือกทุกคอลัมน์ยกเว้นคอลัมน์สุดท้าย
     y = df.iloc[:, -1] # เลือกคอลัมน์สุดท้ายเป็น Target
```

```
[74] from sklearn.model_selection import train_test_split

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=42) # แบ่งข้อมูล X (Features) และ y (Target) เป็นชุดข้อมูลฝึก (training) และชุดข้อมูลทดสอบ
```

```
[75] from sklearn.svm import SVC
     svm_model = SVC() # สร้างโมเดล SVM (Support Vector Machine) โดยใช้คลาส SVC
     svm_model.fit(X_train, y_train) # ฝึกโมเดล SVM ด้วยข้อมูล train (X_train, y_train)
```

```
SVC
SVC()
```

```
[76] y_pred_svm = svm_model.predict(X_test) # ใช้โมเดล SVM ที่ฝึกแล้ว (svm_model) เพื่อทำนายค่าของ Target (y) โดยใช้ข้อมูล Features ในชุดทดสอบ (X_test).
```

```
[77] from sklearn.neighbors import KNeighborsClassifier
     knn_model = KNeighborsClassifier() # # สร้างโมเดล KNN
     knn_model.fit(X_train, y_train) # ฝึกโมเดล KNN ด้วยข้อมูลฝึก (X_train, y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier()
```

```
[78] y_pred_knn = knn_model.predict(X_test) # ใช้โมเดล KNN ที่ฝึกแล้ว (knn_model) เพื่อทำนายค่าของ Target (y) โดยใช้ข้อมูล Features ในชุดทดสอบ X_test
```

```
[79] from sklearn.metrics import accuracy_score, precision_score,recall_score, f1_score, confusion_matrix
```

```
[80] # Example for SVM
     print("SVM Metrics:")
     print("Accuracy:", accuracy_score(y_test, y_pred_svm))
     print("Precision:", precision_score(y_test, y_pred_svm,
     average='binary'))
     print("Recall:", recall_score(y_test, y_pred_svm,
     average='binary'))
     print("F1-Score:", f1_score(y_test, y_pred_svm,
     average='binary'))
     print("Confusion Matrix:\n", confusion_matrix(y_test,
     y_pred_svm))
```

```
SVM Metrics:
Accuracy: 0.7235772357723578
Precision: 0.7235772357723578
Recall: 1.0
F1-Score: 0.839622641509434
Confusion Matrix:
 [[ 0 34]
 [ 0 89]]
```