SVM_iris_2_columns.ipynb

File Edit View Insert Runtime Tools Help Save in GitHub to keep changes

Files

- ..
- sample_data
- finalized_model.sav
- loan approval.csv

Disk 75.04 GB available

+ Code + Text Copy to Drive

RAM Disk Gemini

[29]
```
from sklearn import svm, datasets
import sklearn.model_selection as model_selection
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# import library sklearn, numpy, matplotlib, pandas
```

[30]
```
from google.colab import files
uploaded = files.upload()
# get data
```

Choose Files loan approval.csv

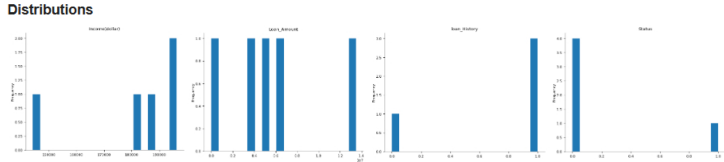
- **loan approval.csv**(text/csv) - 36638 bytes, last modified: 12/29/2024 - 100% done

```
Saving loan approval.csv to loan approval (2).csv
```

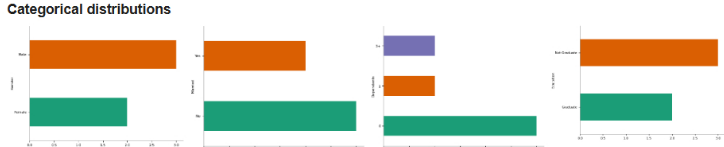
[31]
```
df = pd. read_csv("/content/loan approval.csv")
df['Status'] = df['Status'].map({'Y': 1, 'N': 0})
# read and set new status

df.head()
#show data
```

	Gender	Married	Dependents	Education	Self_Employed	Income(dollar)	Coapplicant	Loan_Amount	Term(month)	loan_History	Area	Status
0	Male	No	0	Not Graduate	No	144200.0	No	3500000	360.0	1.0	Urban	0
1	Female	No	3+	Not Graduate	No	183000.0	No	0	360.0	0.0	Urban	0
2	Male	Yes	1	Graduate	No	188000.0	No	6100000	360.0	NaN	Rural	0
3	Male	Yes	0	Graduate	No	195000.0	Yes	13500000	360.0	1.0	Rural	0
4	Female	No	0	Not Graduate	No	196300.0	No	5300000	360.0	1.0	Semiurban	1

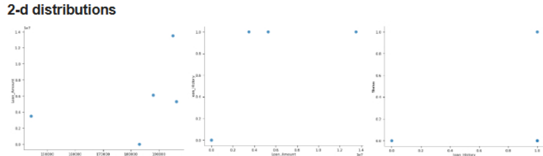
Distributions



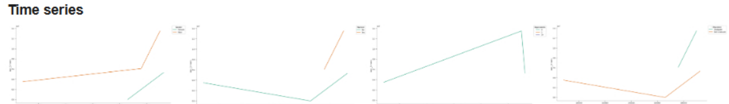
Categorical distributions



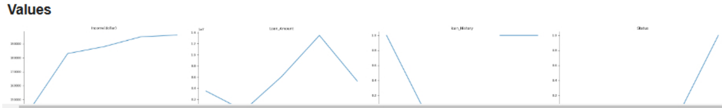
2-d distributions



Time series



Values



[32]
```
df.columns # show column
```

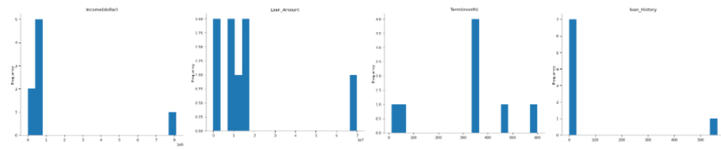
```
Index(['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed',
       'Income(dollar)', 'Coapplicant', 'Loan_Amount', 'Term(month)',
       'loan_History', 'Area', 'Status'],
      dtype='object')
```

[33]
```
df.describe()  # show detail
```

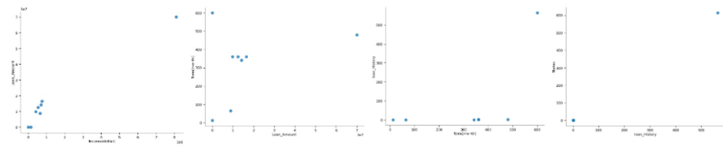
	Income(dollar)	Loan_Amount	Term(month)	loan_History	Status
count	6.140000e+02	6.140000e+02	600.00000	564.000000	614.000000
mean	7.024705e+05	1.414104e+07	342.00000	0.842199	0.687296
std	6.458664e+05	8.815682e+06	65.12041	0.364878	0.463973
min	1.442000e+05	0.000000e+00	12.00000	0.000000	0.000000
25%	4.166000e+05	9.800000e+06	360.00000	1.000000	0.000000
50%	5.416500e+05	1.250000e+07	360.00000	1.000000	1.000000

75%	7.521750e+05	1.647500e+07	360.00000	1.000000	1.000000
max	8.100000e+06	7.000000e+07	480.00000	1.000000	1.000000

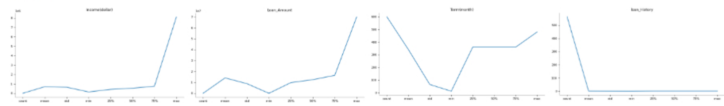
Distributions



2-d distributions

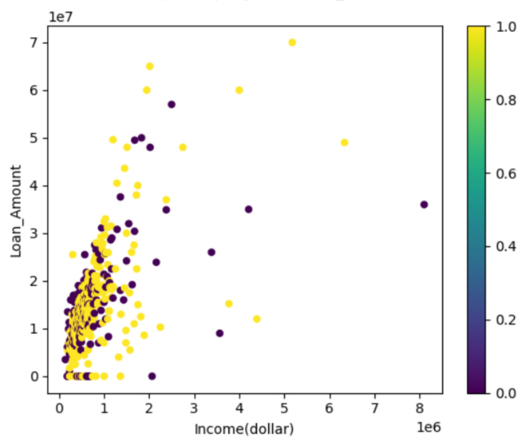


Values



[34]
```
df.plot(kind='scatter', x='Income(dollar)', y='Loan_Amount', c=df['Status'], cmap=plt.cm.viridis) # show graph
```

```
<Axes: xlabel='Income(dollar)', ylabel='Loan_Amount'>
```



[35]
```
y = df['Status'] # set y for train
X = df.drop(columns=['Status','Gender','Married','Dependents','Education','Self_Employed','Coapplicant','Term(month)','loan_History','Area']) # d

scaler = StandardScaler() # ตั้งค่าสเกล

X_scaled = scaler.fit_transform(X)  # เอาสเกลใส่ตัวแปล
X_train, X_test, y_train, y_test = model_selection.train_test_split(X_scaled, y, train_size=0.80, random_state=101) # จำแนกข้อมูล
```

[36]
```
X_train # print ข้อมูลสำหรับ train
```

```
       [-3.61469036e-01, -4.02002848e-01],
       [-3.74795340e-01, -6.85819595e-01],
       [ 5.10784049e-01,  1.32360297e+00],
       [ 3.17279075e+00,  3.84389569e+00],
       [ 1.51128797e-01,  6.65148121e-01],
       [ 1.49398636e+00,  1.51659836e+00],
       [-8.03769954e-02, -9.80989012e-01],
       [ 2.38834473e-01,  7.78674820e-01],
       [-1.51812183e-01, -2.99828819e-01],
       [ 4.88935108e-01,  2.33746666e-01],
       [-1.36471438e-01,  2.11041326e-01],
       [-7.01134831e-01, -5.49587556e-01],
       [-6.29389729e-01, -1.26480576e+00],
       [-1.91791095e-01, -2.43065470e-01],
       [-7.57229273e-01, -4.81471537e-01],
       [-1.89776654e-01, -1.06833431e-01],
       [ 1.12389542e-01,  5.17563413e-01],
       [-5.38894827e-01, -6.97172265e-01],
       [-2.31305136e-01,  3.01862685e-01],
       [-6.10794886e-01,  1.28954496e+00],
       [-5.51911217e-01, -2.20360130e-01],
       [-3.25673963e-01,  2.79157345e-01],
       [ 2.20495970e-02,  4.38094724e-01],
       [-3.86417116e-01, -1.97654790e-01],
       [-4.55992820e-01, -4.70118867e-01],
       [-5.99173109e-01, -4.70118867e-01],
       [-3.54960841e-01, -5.49587556e-01],
       [-5.77169212e-01, -4.36060858e-01],
       [ 1.62926384e+00,  9.75146270e-02],
       [-6.67509157e-01, -9.24225663e-01],
       [-4.43906172e-01, -7.99346294e-01],
       [-4.05476830e-01, -2.43065470e-01],
       [ 1.35344034e+00, -9.80989012e-01],
       [-4.97831216e-01, -3.33886829e-01],
       [-5.28822621e-01, -6.40408916e-01],
       [ 1.04151185e+00, -1.29538771e-01],
       [-3.91220784e-01, -3.22534159e-01],
       [ 7.70956891e-01,  4.02553841e+00],
       [-5.44628237e-01, -7.42582945e-01],
       [ 5.81545835e-02,  8.46790839e-01],
```

```
 [ 5.55203141e-02,  5.17563413e-01],
 [-4.65290241e-01, -3.56592168e-01],
 [ 1.39661978e-01,  1.15331293e+00],
 [ 5.77260612e-01,  1.93664715e+00],
 [-5.39824570e-01, -3.33886829e-01],
 [-9.43231275e-02, -9.69636342e-01],
 [-7.26291442e-02,  2.33746666e-01],
 [-3.07698949e-01, -3.45239499e-01],
 [-2.49125194e-01,  3.01862685e-01],
 [-4.53203594e-01, -1.97654790e-01],
 [-5.85846805e-01, -6.97172265e-01],
 [-3.83472933e-01, -1.60538586e+00],
 [-5.72055630e-01, -8.10698964e-01],
 [-5.52531045e-01, -1.32156911e+00],
 [-4.90548236e-01, -5.15529547e-01],
 [-2.64775853e-01, -6.14227514e-02],
 [-5.74844857e-01, -5.15529547e-01],
 [ 2.54485132e-01,  1.07384424e+00],
```

[45]
```python
# train แบบ linear
linear = svm.SVC(kernel='linear', C=10)
linear.fit(X_train, y_train)

# train แบบ polynomial
poly = svm.SVC(kernel='poly', degree=10, C=100)
poly.fit(X_train, y_train)
```

```
SVC
SVC(C=100, degree=10, kernel='poly')
```

[46]
```python
# print accuracy linear
print("Train set accuracy = " + str(linear.score(X_train, y_train)))
print("Test set accuracy = " + str(linear.score(X_test, y_test)))
```

```
Train set accuracy = 0.6883910386965377
Test set accuracy = 0.6829268292682927
```

[47]
```python
# print accuracy polynomial
print("Train set accuracy = " + str(poly.score(X_train, y_train)))
print("Test set accuracy = " + str(poly.score(X_test, y_test)))
```

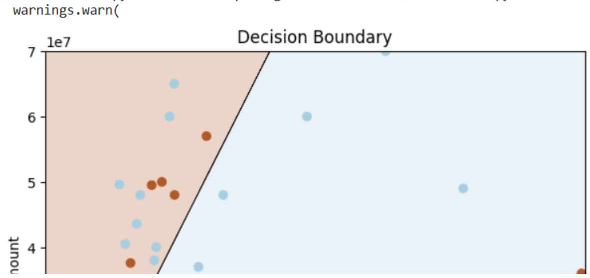
```
Train set accuracy = 0.6985743380855397
Test set accuracy = 0.6910569105691057
```

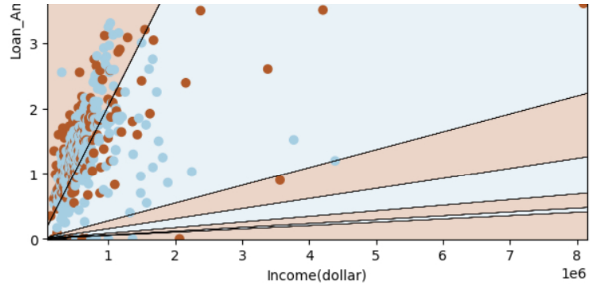
```python
poly_pred = poly.predict(X_test)
comparison_df = pd.DataFrame({'y_test': y_test, 'poly_pred': poly_pred, 'match': y_test == poly_pred})
print(comparison_df) # เทียบ + print ความแม่น โดยจะเทียบระหว่างข้อมูลจริงที่อยู่ใน data set กับข้อมูลที่เรา gen ออกมา
```

```
     y_test  poly_pred  match
216       0          1  False
55        1          1   True
593       0          0   True
438       0          1  False
351       1          1   True
..      ...        ...    ...
437       1          1   True
283       1          1   True
2         0          1  False
355       1          1   True
353       1          1   True

[123 rows x 3 columns]
```

[49]
```python
def plot_decision_boundary(clf, X, y, cmap='Paired_r'):
    h = 5000  # Boundary lines' resolution
    x_min, x_max = X['Income(dollar)'].min() - 10*h, X['Income(dollar)'].max() + 10*h
    y_min, y_max = X['Loan_Amount'].min() - 10*h, X['Loan_Amount'].max() + 10*h
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))
    # use column names for prediction to avoid dimension issues
    Z = clf.predict(pd.DataFrame(np.c_[xx.ravel(), yy.ravel()], columns=['Income(dollar)', 'Loan_Amount']))

    Z = Z.reshape(xx.shape)

    plt.figure(figsize=(7,6))
    plt.contourf(xx, yy, Z, cmap=cmap, alpha=0.25)  # Background
    plt.contour(xx, yy, Z, colors='k', linewidths=0.2)  # Boundary lines

    # Plot the training data points
    plt.scatter(X['Income(dollar)'], X['Loan_Amount'], c=y, cmap=cmap)
    plt.xlabel('Income(dollar)')
    plt.ylabel('Loan_Amount')
    plt.title('Decision Boundary')
    plt.show()


# Call with the training data and polynomial model
plot_decision_boundary(poly, X, y)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:2732: UserWarning: X has feature names, but SVC was fitted without feature nam
  warnings.warn(
```





[50]
```
conf_matrix_polySVM = confusion_matrix(y_test, poly_pred)
precision_polySVM = precision_score(y_test, poly_pred, average="macro")
recall_polySVM = recall_score(y_test, poly_pred, average="macro")
f1_polySVM = f1_score(y_test, poly_pred, average="macro")

print("Polynomial SVM efficiency \n")
print("Precision: ", precision_polySVM)
print("Recall: ", recall_polySVM)
print("F1-Score: ", f1_polySVM)

print("Confusion Matrix:\n", conf_matrix_polySVM)

# print Precision , Recall , F1-Score , Confusion Matrix
```

```
Polynomial SVM efficiency

Precision:  0.6286549707602339
Recall:  0.5402930402930403
F1-Score:  0.5082070707070707
Confusion Matrix:
 [[ 5 34]
 [ 4 80]]
```

```
import joblib

filename = 'finalized_model.sav'
joblib.dump(linear, filename)

# export model
```

```
['finalized_model.sav']
```