# 21162101007_KSHITIJGUPTA

## PR5

```
[ ] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns

    from sklearn.model_selection import train_test_split

    from sklearn.preprocessing import StandardScaler
    from sklearn.linear_model import LogisticRegression
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.metrics import confusion_matrix, precision_recall_fscore_support, log_loss, classification_report, accuracy_score
```

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

```
Choose files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving UCI_Credit_Card.csv to UCI_Credit_Card.csv
```

```
[ ] df = pd.read_csv(io.BytesIO(uploaded['UCI_Credit_Card.csv']))
    print(df.shape)
    df.head()
    df
```

```
(30000, 25)
```

| | ID | LIMIT_BAL | SEX | EDUCATION | MARRIAGE | AGE | PAY_0 | PAY_2 | PAY_3 | PAY_4 | ... | BILL_AMT4 | BILL_AMT5 | BILL_AMT6 | PAY_AMT1 | PAY_AMT2 | PAY_AMT3 | PAY_AMT4 | PAY_AMT5 | PAY_AMT6 | default.payment.next.month |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 20000.0 | 2 | 2 | 1 | 24 | 2 | 2 | -1 | -1 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 689.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1 |
| **1** | 2 | 120000.0 | 2 | 2 | 2 | 26 | -1 | 2 | 0 | 0 | ... | 3272.0 | 3455.0 | 3261.0 | 0.0 | 1000.0 | 1000.0 | 1000.0 | 0.0 | 2000.0 | 1 |
| **2** | 3 | 90000.0 | 2 | 2 | 2 | 34 | 0 | 0 | 0 | 0 | ... | 14331.0 | 14948.0 | 15549.0 | 1518.0 | 1500.0 | 1000.0 | 1000.0 | 1000.0 | 5000.0 | 0 |
| **3** | 4 | 50000.0 | 2 | 2 | 1 | 37 | 0 | 0 | 0 | 0 | ... | 28314.0 | 28959.0 | 29547.0 | 2000.0 | 2019.0 | 1200.0 | 1100.0 | 1069.0 | 1000.0 | 0 |
| **4** | 5 | 50000.0 | 1 | 2 | 1 | 57 | -1 | 0 | -1 | 0 | ... | 20940.0 | 19146.0 | 19131.0 | 2000.0 | 36681.0 | 10000.0 | 9000.0 | 689.0 | 679.0 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **29995** | 29996 | 220000.0 | 1 | 3 | 1 | 39 | 0 | 0 | 0 | 0 | ... | 88004.0 | 31237.0 | 15980.0 | 8500.0 | 20000.0 | 5003.0 | 3047.0 | 5000.0 | 1000.0 | 0 |
| **29996** | 29997 | 150000.0 | 1 | 3 | 2 | 43 | -1 | -1 | -1 | -1 | ... | 8979.0 | 5190.0 | 0.0 | 1837.0 | 3526.0 | 8998.0 | 129.0 | 0.0 | 0.0 | 0 |
| **29997** | 29998 | 30000.0 | 1 | 2 | 2 | 37 | 4 | 3 | 2 | -1 | ... | 20878.0 | 20582.0 | 19357.0 | 0.0 | 0.0 | 22000.0 | 4200.0 | 2000.0 | 3100.0 | 1 |
| **29998** | 29999 | 80000.0 | 1 | 3 | 1 | 41 | 1 | -1 | 0 | 0 | ... | 52774.0 | 11855.0 | 48944.0 | 85900.0 | 3409.0 | 1178.0 | 1926.0 | 52964.0 | 1804.0 | 1 |
| **29999** | 30000 | 50000.0 | 1 | 2 | 1 | 46 | 0 | 0 | 0 | 0 | ... | 36535.0 | 32428.0 | 15313.0 | 2078.0 | 1800.0 | 1430.0 | 1000.0 | 1000.0 | 1000.0 | 1 |

30000 rows × 25 columns

```
[ ] print(df.shape)
    print(df.head())
    print(df.columns)
```

```
(30000, 25)
   ID  LIMIT_BAL  SEX  EDUCATION  MARRIAGE  AGE  PAY_0  PAY_2  PAY_3  PAY_4  \
0   1    20000.0    2          2         1   24      2      2     -1     -1
1   2   120000.0    2          2         2   26     -1      2      0      0
2   3    90000.0    2          2         2   34      0      0      0      0
3   4    50000.0    2          2         1   37      0      0      0      0
4   5    50000.0    1          2         1   57     -1      0     -1      0

   ...  BILL_AMT4  BILL_AMT5  BILL_AMT6  PAY_AMT1  PAY_AMT2  PAY_AMT3  \
0  ...        0.0        0.0        0.0       0.0     689.0       0.0
1  ...     3272.0     3455.0     3261.0       0.0    1000.0    1000.0
2  ...    14331.0    14948.0    15549.0    1518.0    1500.0    1000.0
3  ...    28314.0    28959.0    29547.0    2000.0    2019.0    1200.0
4  ...    20940.0    19146.0    19131.0    2000.0   36681.0   10000.0

   PAY_AMT4  PAY_AMT5  PAY_AMT6  default.payment.next.month
0       0.0       0.0       0.0                          1
1    1000.0       0.0    2000.0                          1
2    1000.0    1000.0    5000.0                          0
3    1100.0    1069.0    1000.0                          0
4    9000.0     689.0     679.0                          0

[5 rows x 25 columns]
Index(['ID', 'LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE', 'PAY_0',
       'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6', 'BILL_AMT1', 'BILL_AMT2',
       'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1',
       'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6',
       'default.payment.next.month'],
      dtype='object')
```

```
[ ] # Check for missing values
    print(df.isnull().sum())
```

```
ID                          0
LIMIT_BAL                   0
SEX                         0
EDUCATION                   0
MARRIAGE                    0
AGE                         0
PAY_0                       0
PAY_2                       0
PAY_3                       0
PAY_4                       0
PAY_5                       0
PAY_6                       0
BILL_AMT1                   0
BILL_AMT2                   0
BILL_AMT3                   0
BILL_AMT4                   0
BILL_AMT5                   0
BILL_AMT6                   0
PAY_AMT1                    0
PAY_AMT2                    0
PAY_AMT3                    0
PAY_AMT4                    0
```

```
PAY_AMT5                    0
PAY_AMT6                    0
default.payment.next.month  0
dtype: int64
```

```
# Split the data into features (X) and target (y)
X = df.drop('default.payment.next.month', axis=1)
y = df[['default.payment.next.month']]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
sc = StandardScaler()
```

```
X_train_scaled = sc.fit_transform(X_train)
X_test_scaled = sc.fit_transform(X_test)
```

```
# Logistic Regression Model
LR = LogisticRegression()
LR.fit(X_train_scaled,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1183: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for
  y = column_or_1d(y, warn=True)
▾ LogisticRegression
LogisticRegression()
```

```
y_pred_lr = LR.predict(X_test_scaled)
y_pred_lr
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
lr_cm = confusion_matrix(y_test,y_pred_lr)
lr_cm
```

```
array([[4540,  147],
       [ 994,  319]])
```

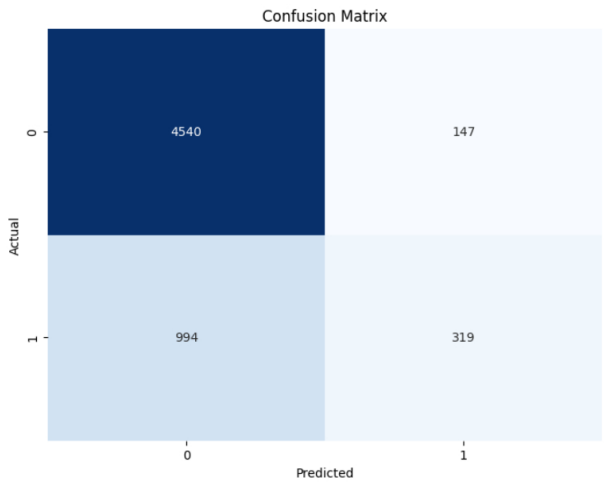
```
lr_acc_score = accuracy_score(y_test,y_pred_lr)
```

```
# Calculate precision, recall, f-score, and support
precision, recall, fscore, support = precision_recall_fscore_support(y_test, y_pred_lr, average='binary')
```

```
y_prob_lr = LR.predict_proba(X_test)[:, 1]  # Probability of class 1 (default)
logloss = log_loss(y_test, y_prob_lr)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:458: UserWarning: X has feature names, but LogisticRegression was fitted without feature names
  warnings.warn(
```

```
plt.figure(figsize=(8, 6))
sns.heatmap(lr_cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```




```
print(f'Precision: {precision:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F-score: {fscore:.4f}')
print(f'Support: {support}')
print(f'Log Loss: {logloss:.4f}')
```

```
Precision: 0.6845
Recall: 0.2430
F-score: 0.3586
Support: None
Log Loss: 7.8876
```

## KNN Classification Model

```
k = 5  # Number of neighbors
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train_scaled, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/neighbors/_classification.py:233: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,
  return self._fit(X, y)
▾ KNeighborsClassifier
```

```
KNeighborsClassifier()
```

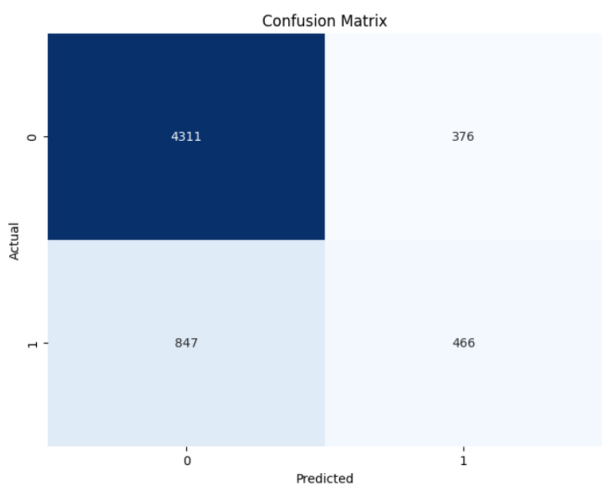
```
y_pred_knn = knn.predict(X_test_scaled)
```

```
knn_cm = confusion_matrix(y_test, y_pred_knn)
```

```
knn_acc_score = accuracy_score(y_test,y_pred_knn)
```

```
knn_class_report = classification_report(y_test, y_pred_knn, target_names=['Non-Default', 'Default'])
```

```
plt.figure(figsize=(8, 6))
sns.heatmap(knn_cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```




```
print(knn_class_report)
```

```
              precision    recall  f1-score   support

 Non-Default       0.84      0.92      0.88      4687
     Default       0.55      0.35      0.43      1313

    accuracy                           0.80      6000
   macro avg       0.69      0.64      0.65      6000
weighted avg       0.77      0.80      0.78      6000
```

## DecisionTree Classifier

```
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train_scaled, y_train)
```

```
DecisionTreeClassifier(random_state=42)
```

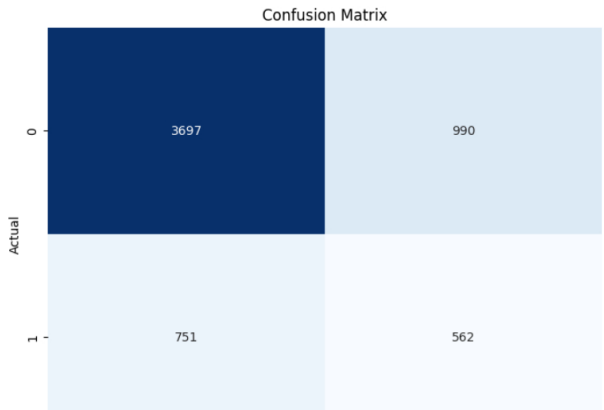
```
y_pred_dt = dt.predict(X_test_scaled)
```

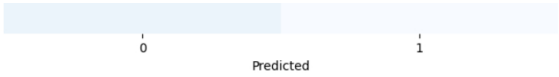
```
dt_cm = confusion_matrix(y_test, y_pred_dt)
```

```
dt_acc_score = accuracy_score(y_test, y_pred_dt)
```

```
dt_class_report = classification_report(y_test, y_pred_dt, target_names=['Non-Default', 'Default'])
```

```
plt.figure(figsize=(8, 6))
sns.heatmap(dt_cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```






```
[ ] print(dt_class_report)
```

```
              precision    recall  f1-score   support

  Non-Default       0.83      0.79      0.81      4687
      Default       0.36      0.43      0.39      1313

     accuracy                           0.71      6000
    macro avg       0.60      0.61      0.60      6000
 weighted avg       0.73      0.71      0.72      6000
```

```
[ ] print("----------ACCURACY SCORE REPORT----------")
    print(f'Accuracy score of Logistic Regression = {round(lr_acc_score,4)}')
    print(f'Accuracy score of KNeighbors Classifier = {round(knn_acc_score,4)}')
    print(f'Accuracy score of DesicionTree Classifier = {round(dt_acc_score,4)}')
```

```
----------ACCURACY SCORE REPORT----------
Accuracy score of Logistic Regression = 0.8098
Accuracy score of KNeighbors Classifier = 0.7962
Accuracy score of DesicionTree Classifier = 0.7098
```

```
[ ] Start coding or generate with AI.
```

Colab paid products - Cancel contracts here