ex1

July 10, 2024

```
[ ]: import pandas as pd
     from sklearn.preprocessing import LabelEncoder
     from sklearn.model_selection import train_test_split, KFold, StratifiedKFold,␣
      ↪cross_val_score
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
      ↪f1_score
```

```
[ ]: df = pd.read_csv('datasets/loan_data.csv')
     df.head()
```

```
[ ]:     Loan_ID Gender Married Dependents     Education Self_Employed  \
     0  LP001003   Male     Yes          1      Graduate            No
     1  LP001005   Male     Yes          0      Graduate           Yes
     2  LP001006   Male     Yes          0  Not Graduate            No
     3  LP001008   Male      No          0      Graduate            No
     4  LP001013   Male     Yes          0  Not Graduate            No

        ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
     0             4583             1508.0       128.0             360.0
     1             3000                0.0        66.0             360.0
     2             2583             2358.0       120.0             360.0
     3             6000                0.0       141.0             360.0
     4             2333             1516.0        95.0             360.0

        Credit_History Property_Area Loan_Status
     0             1.0         Rural           N
     1             1.0         Urban           Y
     2             1.0         Urban           Y
     3             1.0         Urban           Y
     4             1.0         Urban           Y
```

```
[ ]: df.dtypes
```

```
[ ]: Loan_ID               object
     Gender                object
     Married               object
     Dependents            object
     Education              object
     Self_Employed          object
     ApplicantIncome         int64
     CoapplicantIncome     float64
     LoanAmount            float64
     Loan_Amount_Term      float64
     Credit_History        float64
     Property_Area          object
     Loan_Status            object
     dtype: object
```

```
[ ]: df.isnull().sum()
```

```
[ ]: Loan_ID               0
     Gender                5
     Married               0
     Dependents            8
     Education             0
     Self_Employed        21
     ApplicantIncome       0
     CoapplicantIncome     0
     LoanAmount            0
     Loan_Amount_Term     11
     Credit_History       30
     Property_Area         0
     Loan_Status           0
     dtype: int64
```

```
[ ]: df.dropna(inplace=True)
     df.isnull().sum()
```

```
[ ]: Loan_ID              0
     Gender               0
     Married              0
     Dependents           0
     Education            0
     Self_Employed        0
     ApplicantIncome      0
     CoapplicantIncome    0
     LoanAmount           0
     Loan_Amount_Term     0
     Credit_History       0
     Property_Area        0
     Loan_Status          0
     dtype: int64
```

```
[ ]: df.drop('Loan_ID', axis=1, inplace=True)
```

[ ]:

```
le = LabelEncoder()

df.dropna(inplace=True)
df['LoanAmount'] = df['LoanAmount'].astype(int)
df['Loan_Amount_Term'] = df['Loan_Amount_Term'].astype(int)
df['Credit_History'] = df['Credit_History'].astype(int)
df['Gender'] = le.fit_transform(df['Gender'])
df['Married'] = le.fit_transform(df['Married'])
df['Dependents'] = le.fit_transform(df['Dependents'])
df['Education'] = le.fit_transform(df['Education'])
df['Self_Employed'] = le.fit_transform(df['Self_Employed'])
df['Property_Area'] = le.fit_transform(df['Property_Area'])

X=df.drop('Loan_Status', axis=1)
y=df['Loan_Status']
```

[ ]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Initialize the Decision Tree classifier
model = DecisionTreeClassifier(random_state=42)

# Train the model on the training set
model.fit(X_train, y_train)
```

[ ]:

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

[ ]:

```
y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print("Performance Metrics Before Cross-Validation:")
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")
```

```
Performance Metrics Before Cross-Validation:
Accuracy: 0.7312
Precision: 0.7289
Recall: 0.7312
F1 Score: 0.7300
```

[ ]:

```python
kf = KFold(n_splits=5, shuffle=True, random_state=42)
cv_accuracy = cross_val_score(model, X, y, cv=kf, scoring='accuracy')
cv_precision = cross_val_score(model, X, y, cv=kf, scoring='precision_weighted')
cv_recall = cross_val_score(model, X, y, cv=kf, scoring='recall_weighted')
cv_f1 = cross_val_score(model, X, y, cv=kf, scoring='f1_weighted')

print("\nPerformance Metrics After K-fold Cross-Validation:")
print(f"Accuracy: {cv_accuracy.mean():.4f} (+/- {cv_accuracy.std():.4f})")
print(f"Precision: {cv_precision.mean():.4f} (+/- {cv_precision.std():.4f})")
print(f"Recall: {cv_recall.mean():.4f} (+/- {cv_recall.std():.4f})")
print(f"F1 Score: {cv_f1.mean():.4f} (+/- {cv_f1.std():.4f})")
```

```
Performance Metrics After K-fold Cross-Validation:
Accuracy: 0.7598 (+/- 0.0116)
Precision: 0.7670 (+/- 0.0235)
Recall: 0.7598 (+/- 0.0116)
F1 Score: 0.7584 (+/- 0.0206)
```

```
Accuracy: 0.7598 (+/- 0.0116)
Precision: 0.7670 (+/- 0.0235)
Recall: 0.7598 (+/- 0.0116)
F1 Score: 0.7584 (+/- 0.0206)
```

[ ]:

```python
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_accuracy_strat = cross_val_score(model, X, y, cv=skf, scoring='accuracy')
cv_precision_strat = cross_val_score(model, X, y, cv=skf,␣
 ↪scoring='precision_weighted')
cv_recall_strat = cross_val_score(model, X, y, cv=skf,␣
 ↪scoring='recall_weighted')
cv_f1_strat = cross_val_score(model, X, y, cv=skf, scoring='f1_weighted')

print("\nPerformance Metrics After Stratified K-fold Cross-Validation:")
print(f"Accuracy: {cv_accuracy_strat.mean():.4f} (+/- {cv_accuracy_strat.std():.
 ↪4f})")
print(f"Precision: {cv_precision_strat.mean():.4f} (+/- {cv_precision_strat.
 ↪std():.4f})")
print(f"Recall: {cv_recall_strat.mean():.4f} (+/- {cv_recall_strat.std():.4f})")
print(f"F1 Score: {cv_f1_strat.mean():.4f} (+/- {cv_f1_strat.std():.4f})")
```

```
Performance Metrics After Stratified K-fold Cross-Validation:
Accuracy: 0.7498 (+/- 0.0542)
Precision: 0.7608 (+/- 0.0397)
Recall: 0.7498 (+/- 0.0542)
F1 Score: 0.7521 (+/- 0.0502)
```