hyperparameter-tuning

December 9, 2023

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import f1_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix, accuracy_score, recall_score,␣
 ↪precision_score
from sklearn.metrics import classification_report
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A
NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy
(detected version 1.24.3
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

[2]:
```
df = pd.read_csv('/kaggle/input/bankloan/train.csv')
df.head()
```

[2]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	\
0	LP001002	Male	No	0	Graduate	No	
1	LP001003	Male	Yes	1	Graduate	No	
2	LP001005	Male	Yes	0	Graduate	Yes	
3	LP001006	Male	Yes	0	Not Graduate	No	
4	LP001008	Male	No	0	Graduate	No	

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	\
0	5849	0.0	NaN	360.0	
1	4583	1508.0	128.0	360.0	
2	3000	0.0	66.0	360.0	
3	2583	2358.0	120.0	360.0	
4	6000	0.0	141.0	360.0	

	Credit_History	Property_Area	Loan_Status

```
0             1.0         Urban           Y
1             1.0         Rural           N
2             1.0         Urban           Y
3             1.0         Urban           Y
4             1.0         Urban           Y
```

[3]:
```
df = df.rename(columns=str.lower)
```

[4]:
```
df.shape
```

[4]:
```
(614, 13)
```

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   loan_id            614 non-null    object
 1   gender             601 non-null    object
 2   married            611 non-null    object
 3   dependents         599 non-null    object
 4   education          614 non-null    object
 5   self_employed      582 non-null    object
 6   applicantincome    614 non-null    int64
 7   coapplicantincome  614 non-null    float64
 8   loanamount         592 non-null    float64
 9   loan_amount_term   600 non-null    float64
 10  credit_history     564 non-null    float64
 11  property_area      614 non-null    object
 12  loan_status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

[6]:
```
en = LabelEncoder()
cat =  ['gender','married','education',␣
 ↪'self_employed','property_area','loan_status']
for cols in cat:
    df[cols] = en.fit_transform(df[cols])
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:605:
FutureWarning: is_sparse is deprecated and will be removed in a future version.
Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:614:
FutureWarning: is_sparse is deprecated and will be removed in a future version.
```

```
Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:605:
FutureWarning: is_sparse is deprecated and will be removed in a future version.
Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:614:
FutureWarning: is_sparse is deprecated and will be removed in a future version.
Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:605:
FutureWarning: is_sparse is deprecated and will be removed in a future version.
Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:614:
FutureWarning: is_sparse is deprecated and will be removed in a future version.
Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:605:
FutureWarning: is_sparse is deprecated and will be removed in a future version.
Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:614:
FutureWarning: is_sparse is deprecated and will be removed in a future version.
Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:605:
FutureWarning: is_sparse is deprecated and will be removed in a future version.
Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:614:
FutureWarning: is_sparse is deprecated and will be removed in a future version.
Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:605:
FutureWarning: is_sparse is deprecated and will be removed in a future version.
Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:614:
FutureWarning: is_sparse is deprecated and will be removed in a future version.
Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
```

[7]:
```
df['dependents'].replace('3+',3,inplace=True)
```

[8]:
```
df.head()
```

[8]:

	loan_id	gender	married	dependents	education	self_employed
0	LP001002	1	0	0	0	0
1	LP001003	1	1	1	0	0
2	LP001005	1	1	0	0	1
3	LP001006	1	1	0	1	0
4	LP001008	1	0	0	0	0

	applicantincome	coapplicantincome	loanamount	loan_amount_term
0	5849	0.0	NaN	360.0
1	4583	1508.0	128.0	360.0
2	3000	0.0	66.0	360.0
3	2583	2358.0	120.0	360.0
4	6000	0.0	141.0	360.0

	credit_history	property_area	loan_status
0	1.0	2	1
1	1.0	0	0
2	1.0	2	1
3	1.0	2	1
4	1.0	2	1

[9]:
```
df.isna().sum()
```

[9]:
```
loan_id               0
gender                0
married               0
dependents           15
education             0
self_employed         0
applicantincome       0
coapplicantincome     0
loanamount           22
loan_amount_term     14
credit_history       50
property_area         0
loan_status           0
dtype: int64
```

[10]:
```
df_clean = df
df_clean.drop('loan_id', axis=1,inplace=True)
```

[11]:
```
from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors=3)
df_clean = pd.DataFrame(imputer.fit_transform(df),columns = df_clean.columns)
df_clean.isnull().sum()
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:767:
```

```
FutureWarning: is_sparse is deprecated and will be removed in a future version.
Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:605:
FutureWarning: is_sparse is deprecated and will be removed in a future version.
Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:614:
FutureWarning: is_sparse is deprecated and will be removed in a future version.
Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:767:
FutureWarning: is_sparse is deprecated and will be removed in a future version.
Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:605:
FutureWarning: is_sparse is deprecated and will be removed in a future version.
Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:614:
FutureWarning: is_sparse is deprecated and will be removed in a future version.
Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
```

[11]:
```
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
```

[12]:
```
X = df_clean.drop(columns=['loan_status']).values
y = df_clean['loan_status'].values
```

[13]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,␣
↪random_state = 0)
```

[14]:
```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

[15]:
```
rfc = RandomForestClassifier(criterion = 'entropy', random_state = 42)
rfc.fit(X_train, y_train)
```

[15]:
```
RandomForestClassifier(criterion='entropy', random_state=42)
```

[16]:
```
y_pred = rfc.predict(X_test)
```

[17]:
```
from sklearn import metrics
metrix = metrics.accuracy_score(y_test, y_pred)
print('Accuracy', metrix)
```

```
Accuracy 0.8051948051948052
```

[18]:
```
from pprint import pprint
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.ensemble import RandomForestClassifier
```

[19]:
```
rf_classifier = RandomForestClassifier()
print("Best Parameters:")
pprint(rf_classifier.get_params())
```

```
Best Parameters:
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'sqrt',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

[20]:
```
param_grid = {
    'n_estimators': [50, 100, 150, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2'],
```

```
    'bootstrap': [True, False],
    'criterion': ['gini', 'entropy']
}
```

[21]:
```
grid_search = GridSearchCV(estimator=rf_classifier, param_grid=param_grid,␣
↪cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)
print("Grid Search Completed.")
```

```
Grid Search Completed.
```

[22]:
```
# Train the model with the best parameters
best_rf_model = grid_search.best_estimator_
best_rf_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = best_rf_model.predict(X_test)
```

[23]:
```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8311688311688312
```