Moussadeq_DSC550_Week9

May 11, 2024

DSC 550

Week 9 Assignment

Said Moussadeq

1. Import the dataset

[2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

[5]:
```
# loading the data
df_loan = pd.read_csv(r'Loan_train.csv')
df_loan.head(5)
```

[5]:
```
    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
```

2. **Prepare the data for modeling by performing the required steps:**
   - **Drop the column "Load_ID."**

[5]:
```
# dropping the "Load_ID" column
df_loan = df_loan.drop(['Loan_ID'], axis=1)

df_loan.head(3)
```

[5]:
```
  Gender Married Dependents Education Self_Employed  ApplicantIncome  \
0   Male      No          0  Graduate            No             5849
1   Male     Yes          1  Graduate            No             4583
2   Male     Yes          0  Graduate           Yes             3000

   CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
0                0.0         NaN             360.0             1.0
1             1508.0       128.0             360.0             1.0
2                0.0        66.0             360.0             1.0

  Property_Area Loan_Status
0         Urban           Y
1         Rural           N
2         Urban           Y
```

   - **Drop any rows with missing data.**

[8]:
```
# checking for NaN values
df_loan.isnull().sum()
```

[8]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

[9]:
```
# dropping NaN values
df_loanV2 = df_loan.dropna()
df_loanV2.isnull().sum()
```

[9]:
```
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
```

- **Convert the categorical features into dummy variables.**

[10]:
```python
# Identifying categorical columns
categorical_cols = df_loanV2.columns[df_loanV2.dtypes == 'object']

# Creating dummy variables for the categorical columns
dummy_vars = pd.get_dummies(df_loanV2[categorical_cols])

# Drop original categorical columns from the DataFrame
df_loanV2 = df_loanV2.drop(categorical_cols, axis=1)

# Concatenate the dummy variables to the original DataFrame
df_loanV2 = pd.concat([df_loanV2, dummy_vars], axis=1)
```

[12]:
```python
# Checking the shape of the DataFrame 'df_loanV3'
df_loanV2.shape
```

[12]: (480, 502)

[14]:
```python
df_loanV2.head(3)
```

[14]:
```
   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
1             4583             1508.0       128.0             360.0
2             3000                0.0        66.0             360.0
3             2583             2358.0       120.0             360.0

   Credit_History  Loan_ID_LP001003  Loan_ID_LP001005  Loan_ID_LP001006  \
1             1.0              True             False             False
2             1.0             False              True             False
3             1.0             False             False              True

   Loan_ID_LP001008  Loan_ID_LP001011  …  Dependents_3+  Education_Graduate  \
1             False             False  …          False                True
```

```
2          False          False  ...          False          True
3          False          False  ...          False          False

   Education_Not Graduate  Self_Employed_No  Self_Employed_Yes  \
1                   False              True              False
2                   False             False               True
3                    True              True              False

   Property_Area_Rural  Property_Area_Semiurban  Property_Area_Urban  \
1                 True                    False                False
2                False                    False                 True
3                False                    False                 True

   Loan_Status_N  Loan_Status_Y
1           True          False
2          False           True
3          False           True

[3 rows x 502 columns]
```

[37]:

```
for col in df_loanV2:
    print(col)
```

```
ApplicantIncome
CoapplicantIncome
LoanAmount
Loan_Amount_Term
Credit_History
Loan_ID_LP001003
Loan_ID_LP001005
Loan_ID_LP001006
Loan_ID_LP001008
Loan_ID_LP001011
Loan_ID_LP001013
Loan_ID_LP001014
Loan_ID_LP001018
Loan_ID_LP001020
Loan_ID_LP001024
Loan_ID_LP001028
Loan_ID_LP001029
Loan_ID_LP001030
Loan_ID_LP001032
Loan_ID_LP001036
Loan_ID_LP001038
Loan_ID_LP001043
Loan_ID_LP001046
Loan_ID_LP001047
Loan_ID_LP001066
```

Loan_ID_LP001068
Loan_ID_LP001073
Loan_ID_LP001086
Loan_ID_LP001095
Loan_ID_LP001097
Loan_ID_LP001098
Loan_ID_LP001100
Loan_ID_LP001112
Loan_ID_LP001114
Loan_ID_LP001116
Loan_ID_LP001119
Loan_ID_LP001120
Loan_ID_LP001131
Loan_ID_LP001138
Loan_ID_LP001144
Loan_ID_LP001146
Loan_ID_LP001151
Loan_ID_LP001155
Loan_ID_LP001157
Loan_ID_LP001164
Loan_ID_LP001179
Loan_ID_LP001186
Loan_ID_LP001194
Loan_ID_LP001195
Loan_ID_LP001197
Loan_ID_LP001198
Loan_ID_LP001199
Loan_ID_LP001205
Loan_ID_LP001206
Loan_ID_LP001207
Loan_ID_LP001222
Loan_ID_LP001225
Loan_ID_LP001228
Loan_ID_LP001233
Loan_ID_LP001238
Loan_ID_LP001241
Loan_ID_LP001243
Loan_ID_LP001245
Loan_ID_LP001248
Loan_ID_LP001253
Loan_ID_LP001255
Loan_ID_LP001256
Loan_ID_LP001259
Loan_ID_LP001263
Loan_ID_LP001265
Loan_ID_LP001267
Loan_ID_LP001275
Loan_ID_LP001279

Loan_ID_LP001282
Loan_ID_LP001289
Loan_ID_LP001310
Loan_ID_LP001316
Loan_ID_LP001318
Loan_ID_LP001319
Loan_ID_LP001322
Loan_ID_LP001325
Loan_ID_LP001327
Loan_ID_LP001333
Loan_ID_LP001334
Loan_ID_LP001343
Loan_ID_LP001345
Loan_ID_LP001349
Loan_ID_LP001367
Loan_ID_LP001369
Loan_ID_LP001379
Loan_ID_LP001384
Loan_ID_LP001385
Loan_ID_LP001401
Loan_ID_LP001404
Loan_ID_LP001421
Loan_ID_LP001422
Loan_ID_LP001430
Loan_ID_LP001431
Loan_ID_LP001432
Loan_ID_LP001439
Loan_ID_LP001451
Loan_ID_LP001473
Loan_ID_LP001478
Loan_ID_LP001482
Loan_ID_LP001487
Loan_ID_LP001488
Loan_ID_LP001489
Loan_ID_LP001491
Loan_ID_LP001492
Loan_ID_LP001493
Loan_ID_LP001497
Loan_ID_LP001498
Loan_ID_LP001504
Loan_ID_LP001507
Loan_ID_LP001508
Loan_ID_LP001514
Loan_ID_LP001516
Loan_ID_LP001518
Loan_ID_LP001519
Loan_ID_LP001520
Loan_ID_LP001528

Loan_ID_LP001529
Loan_ID_LP001531
Loan_ID_LP001532
Loan_ID_LP001535
Loan_ID_LP001536
Loan_ID_LP001543
Loan_ID_LP001552
Loan_ID_LP001560
Loan_ID_LP001562
Loan_ID_LP001565
Loan_ID_LP001570
Loan_ID_LP001572
Loan_ID_LP001577
Loan_ID_LP001578
Loan_ID_LP001579
Loan_ID_LP001580
Loan_ID_LP001586
Loan_ID_LP001594
Loan_ID_LP001603
Loan_ID_LP001606
Loan_ID_LP001608
Loan_ID_LP001610
Loan_ID_LP001616
Loan_ID_LP001630
Loan_ID_LP001633
Loan_ID_LP001636
Loan_ID_LP001637
Loan_ID_LP001639
Loan_ID_LP001640
Loan_ID_LP001641
Loan_ID_LP001647
Loan_ID_LP001653
Loan_ID_LP001656
Loan_ID_LP001657
Loan_ID_LP001658
Loan_ID_LP001664
Loan_ID_LP001665
Loan_ID_LP001666
Loan_ID_LP001673
Loan_ID_LP001674
Loan_ID_LP001677
Loan_ID_LP001688
Loan_ID_LP001691
Loan_ID_LP001692
Loan_ID_LP001693
Loan_ID_LP001698
Loan_ID_LP001699
Loan_ID_LP001702

Loan_ID_LP001708
Loan_ID_LP001711
Loan_ID_LP001713
Loan_ID_LP001715
Loan_ID_LP001716
Loan_ID_LP001720
Loan_ID_LP001722
Loan_ID_LP001726
Loan_ID_LP001736
Loan_ID_LP001743
Loan_ID_LP001744
Loan_ID_LP001750
Loan_ID_LP001751
Loan_ID_LP001758
Loan_ID_LP001761
Loan_ID_LP001765
Loan_ID_LP001776
Loan_ID_LP001778
Loan_ID_LP001784
Loan_ID_LP001790
Loan_ID_LP001792
Loan_ID_LP001798
Loan_ID_LP001800
Loan_ID_LP001806
Loan_ID_LP001807
Loan_ID_LP001811
Loan_ID_LP001813
Loan_ID_LP001814
Loan_ID_LP001819
Loan_ID_LP001824
Loan_ID_LP001825
Loan_ID_LP001835
Loan_ID_LP001836
Loan_ID_LP001841
Loan_ID_LP001843
Loan_ID_LP001844
Loan_ID_LP001846
Loan_ID_LP001849
Loan_ID_LP001854
Loan_ID_LP001859
Loan_ID_LP001868
Loan_ID_LP001870
Loan_ID_LP001871
Loan_ID_LP001872
Loan_ID_LP001875
Loan_ID_LP001877
Loan_ID_LP001882
Loan_ID_LP001884

Loan_ID_LP001888
Loan_ID_LP001891
Loan_ID_LP001892
Loan_ID_LP001894
Loan_ID_LP001896
Loan_ID_LP001900
Loan_ID_LP001903
Loan_ID_LP001904
Loan_ID_LP001907
Loan_ID_LP001910
Loan_ID_LP001914
Loan_ID_LP001915
Loan_ID_LP001917
Loan_ID_LP001924
Loan_ID_LP001925
Loan_ID_LP001926
Loan_ID_LP001931
Loan_ID_LP001935
Loan_ID_LP001936
Loan_ID_LP001938
Loan_ID_LP001940
Loan_ID_LP001947
Loan_ID_LP001953
Loan_ID_LP001954
Loan_ID_LP001955
Loan_ID_LP001963
Loan_ID_LP001964
Loan_ID_LP001974
Loan_ID_LP001977
Loan_ID_LP001978
Loan_ID_LP001993
Loan_ID_LP001994
Loan_ID_LP001996
Loan_ID_LP002002
Loan_ID_LP002004
Loan_ID_LP002006
Loan_ID_LP002031
Loan_ID_LP002035
Loan_ID_LP002050
Loan_ID_LP002051
Loan_ID_LP002053
Loan_ID_LP002065
Loan_ID_LP002067
Loan_ID_LP002068
Loan_ID_LP002082
Loan_ID_LP002086
Loan_ID_LP002087
Loan_ID_LP002097

Loan_ID_LP002098
Loan_ID_LP002112
Loan_ID_LP002114
Loan_ID_LP002115
Loan_ID_LP002116
Loan_ID_LP002119
Loan_ID_LP002126
Loan_ID_LP002129
Loan_ID_LP002131
Loan_ID_LP002138
Loan_ID_LP002139
Loan_ID_LP002140
Loan_ID_LP002141
Loan_ID_LP002142
Loan_ID_LP002143
Loan_ID_LP002149
Loan_ID_LP002151
Loan_ID_LP002158
Loan_ID_LP002160
Loan_ID_LP002161
Loan_ID_LP002170
Loan_ID_LP002175
Loan_ID_LP002180
Loan_ID_LP002181
Loan_ID_LP002187
Loan_ID_LP002190
Loan_ID_LP002191
Loan_ID_LP002194
Loan_ID_LP002197
Loan_ID_LP002201
Loan_ID_LP002205
Loan_ID_LP002211
Loan_ID_LP002219
Loan_ID_LP002224
Loan_ID_LP002225
Loan_ID_LP002229
Loan_ID_LP002231
Loan_ID_LP002234
Loan_ID_LP002236
Loan_ID_LP002239
Loan_ID_LP002244
Loan_ID_LP002250
Loan_ID_LP002255
Loan_ID_LP002262
Loan_ID_LP002265
Loan_ID_LP002266
Loan_ID_LP002277
Loan_ID_LP002281

Loan_ID_LP002284
Loan_ID_LP002287
Loan_ID_LP002288
Loan_ID_LP002296
Loan_ID_LP002297
Loan_ID_LP002300
Loan_ID_LP002301
Loan_ID_LP002305
Loan_ID_LP002308
Loan_ID_LP002314
Loan_ID_LP002315
Loan_ID_LP002317
Loan_ID_LP002318
Loan_ID_LP002328
Loan_ID_LP002332
Loan_ID_LP002335
Loan_ID_LP002337
Loan_ID_LP002341
Loan_ID_LP002342
Loan_ID_LP002345
Loan_ID_LP002347
Loan_ID_LP002348
Loan_ID_LP002361
Loan_ID_LP002364
Loan_ID_LP002366
Loan_ID_LP002367
Loan_ID_LP002368
Loan_ID_LP002369
Loan_ID_LP002370
Loan_ID_LP002377
Loan_ID_LP002379
Loan_ID_LP002387
Loan_ID_LP002390
Loan_ID_LP002398
Loan_ID_LP002403
Loan_ID_LP002407
Loan_ID_LP002408
Loan_ID_LP002409
Loan_ID_LP002418
Loan_ID_LP002422
Loan_ID_LP002429
Loan_ID_LP002434
Loan_ID_LP002443
Loan_ID_LP002446
Loan_ID_LP002448
Loan_ID_LP002449
Loan_ID_LP002453
Loan_ID_LP002455

Loan_ID_LP002459
Loan_ID_LP002467
Loan_ID_LP002472
Loan_ID_LP002473
Loan_ID_LP002484
Loan_ID_LP002487
Loan_ID_LP002493
Loan_ID_LP002494
Loan_ID_LP002500
Loan_ID_LP002505
Loan_ID_LP002515
Loan_ID_LP002517
Loan_ID_LP002519
Loan_ID_LP002524
Loan_ID_LP002527
Loan_ID_LP002529
Loan_ID_LP002531
Loan_ID_LP002534
Loan_ID_LP002536
Loan_ID_LP002537
Loan_ID_LP002541
Loan_ID_LP002543
Loan_ID_LP002544
Loan_ID_LP002545
Loan_ID_LP002547
Loan_ID_LP002555
Loan_ID_LP002556
Loan_ID_LP002571
Loan_ID_LP002582
Loan_ID_LP002585
Loan_ID_LP002586
Loan_ID_LP002587
Loan_ID_LP002600
Loan_ID_LP002602
Loan_ID_LP002603
Loan_ID_LP002606
Loan_ID_LP002615
Loan_ID_LP002619
Loan_ID_LP002622
Loan_ID_LP002626
Loan_ID_LP002634
Loan_ID_LP002637
Loan_ID_LP002640
Loan_ID_LP002643
Loan_ID_LP002648
Loan_ID_LP002652
Loan_ID_LP002659
Loan_ID_LP002670

Loan_ID_LP002683
Loan_ID_LP002684
Loan_ID_LP002689
Loan_ID_LP002690
Loan_ID_LP002692
Loan_ID_LP002693
Loan_ID_LP002699
Loan_ID_LP002705
Loan_ID_LP002706
Loan_ID_LP002714
Loan_ID_LP002716
Loan_ID_LP002720
Loan_ID_LP002723
Loan_ID_LP002731
Loan_ID_LP002734
Loan_ID_LP002738
Loan_ID_LP002739
Loan_ID_LP002740
Loan_ID_LP002741
Loan_ID_LP002743
Loan_ID_LP002755
Loan_ID_LP002767
Loan_ID_LP002768
Loan_ID_LP002772
Loan_ID_LP002776
Loan_ID_LP002777
Loan_ID_LP002785
Loan_ID_LP002788
Loan_ID_LP002789
Loan_ID_LP002792
Loan_ID_LP002795
Loan_ID_LP002798
Loan_ID_LP002804
Loan_ID_LP002807
Loan_ID_LP002813
Loan_ID_LP002820
Loan_ID_LP002821
Loan_ID_LP002832
Loan_ID_LP002836
Loan_ID_LP002837
Loan_ID_LP002840
Loan_ID_LP002841
Loan_ID_LP002842
Loan_ID_LP002855
Loan_ID_LP002862
Loan_ID_LP002863
Loan_ID_LP002868
Loan_ID_LP002874

```
Loan_ID_LP002877
Loan_ID_LP002892
Loan_ID_LP002893
Loan_ID_LP002894
Loan_ID_LP002911
Loan_ID_LP002912
Loan_ID_LP002916
Loan_ID_LP002917
Loan_ID_LP002926
Loan_ID_LP002928
Loan_ID_LP002931
Loan_ID_LP002936
Loan_ID_LP002938
Loan_ID_LP002940
Loan_ID_LP002941
Loan_ID_LP002945
Loan_ID_LP002948
Loan_ID_LP002953
Loan_ID_LP002958
Loan_ID_LP002959
Loan_ID_LP002961
Loan_ID_LP002964
Loan_ID_LP002974
Loan_ID_LP002978
Loan_ID_LP002979
Loan_ID_LP002983
Loan_ID_LP002984
Loan_ID_LP002990
Gender_Female
Gender_Male
Married_No
Married_Yes
Dependents_0
Dependents_1
Dependents_2
Dependents_3+
Education_Graduate
Education_Not Graduate
Self_Employed_No
Self_Employed_Yes
Property_Area_Rural
Property_Area_Semiurban
Property_Area_Urban
Loan_Status_N
Loan_Status_Y
```

3. Split the data into a training and test set, where the "Loan_Status" column is the target.

[38]:
```
from sklearn.model_selection import train_test_split

# separate features and target
X = df_loanV2.drop(['Loan_Status_Y', 'Loan_Status_N'], axis=1) # features
y = df_loanV2['Loan_Status_Y'] # target
```

[39]:
```
# split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

4. Create a pipeline with a min-max scaler and a KNN classifier (see section 15.3 in the Machine Learning with Python Cookbook).

[40]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.model_selection import GridSearchCV
```

[41]:
```
# defining the pipeline
pipeline = Pipeline([
    ('min_max_scaler', MinMaxScaler()),  # Step 1: Min-Max Scaler
    ('knn_classifier', KNeighborsClassifier())  # Step 2: KNN Classifier
])
```

5. Fit a default KNN classifier to the data with this pipeline. Report the model accuracy on the test set. Note: Fitting a pipeline model works just like fitting a regular model.

[42]:
```
# fiting the pipeline to the training data sets
pipeline.fit(X_train, y_train)
```

[42]:
```
Pipeline(steps=[('min_max_scaler', MinMaxScaler()),
                ('knn_classifier', KNeighborsClassifier())])
```

[43]:
```
# predicting the target value for the test set
y_pred = pipeline.predict(X_test)
```

[44]:
```
# calculating accuracy
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.71
```

6. Create a search space for your KNN classifier where your "n__neighbors" parameter varies from 1 to 10. (see section 15.3 in the Machine Learning with Python Cookbook).

[45]:
```
# define the parameter grid
param_grid = {
    'knn_classifier__n_neighbors': list(range(1, 11))  # 1 to 10
}

# creating a GridSearchCV object
grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='accuracy')
```

7. Fit a grid search with your pipeline, search space, and 5-fold cross-validation to find the best value for the "n__neighbors" parameter.

[46]:
```
# fit the GridSearchCV object to the training data
grid_search.fit(X_train, y_train)
```

[46]:
```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('min_max_scaler', MinMaxScaler()),
                                       ('knn_classifier',
                                        KNeighborsClassifier())]),
             param_grid={'knn_classifier__n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8,
                                                         9, 10]},
             scoring='accuracy')
```

[47]:
```
# Best parameters found
print("Best parameters found: ", grid_search.best_params_)

# Best score (accuracy) from GridSearchCV
print("Best score (accuracy): ", grid_search.best_score_)
```

```
Best parameters found:  {'knn_classifier__n_neighbors': 4}
Best score (accuracy):  0.7134996582365003
```

8. Find the accuracy of the grid search best model on the test set. Note: It is possible that this will not be an improvement over the default model, but likely it will be.

[48]:
```
# Evaluate the best model found by the grid search on the test set
test_accuracy = grid_search.score(X_test, y_test)

print(f"Accuracy of the best model on the test set: {test_accuracy:.2f}")
```

```
Accuracy of the best model on the test set: 0.73
```

9. Now, repeat steps 6 and 7 with the same pipeline, but expand your search space to include logistic regression and random forest models with the hyperparameter values in section 12.3 of the Machine Learning with Python Cookbook.

Step 6

[49]:
```
# Create a pipeline with a generic 'classifier' step
pipeline = Pipeline([
    ('min_max_scaler', MinMaxScaler()),
    ('classifier', None)  # Placeholder for the classifier
])
```

[50]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import numpy as np
```

[51]:
```
# creating the expanded search space
search_space = [
    # KNN Classifier parameters
    {"classifier": [KNeighborsClassifier()],
     "classifier__n_neighbors": list(range(1, 11))},  # 1 to 10
    # Logistic Regression parameters
    {"classifier": [LogisticRegression(max_iter=500, solver='liblinear')],
     "classifier__penalty": ['l1', 'l2'],
     "classifier__C": np.logspace(0, 4, 10)},
    # Random Forest parameters
    {"classifier": [RandomForestClassifier()],
     "classifier__n_estimators": [10, 100, 1000],
     "classifier__max_features": [1, 2, 3]}
]
```

[52]:
```
# Create a GridSearchCV object
grid_search = GridSearchCV(pipeline, search_space, cv=5, scoring='accuracy',␣
 ↪verbose=1)
```

Step 7

[53]:
```
# Fit the GridSearchCV object to the training data
grid_search.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 39 candidates, totalling 195 fits
```

[53]:
```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('min_max_scaler', MinMaxScaler()),
                                       ('classifier', None)]),
             param_grid=[{'classifier': [KNeighborsClassifier()],
                          'classifier__n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9,
                                                      10]},
                         {'classifier': [LogisticRegression(C=7.742636826811269,
                                                            max_iter=500,
                                                            penalty='l1',
solver='liblinear')],
                          'classifier__C': array([1.00000000e+00,
2.78255940e+00, 7.74263683e+00, 2.15443469e+01,
```

```
5.99484250e+01, 1.66810054e+02, 4.64158883e+02, 1.29154967e+03,
3.59381366e+03, 1.00000000e+04]),
                     'classifier__penalty': ['l1', 'l2']},
                    {'classifier': [RandomForestClassifier()],
                     'classifier__max_features': [1, 2, 3],
                     'classifier__n_estimators': [10, 100, 1000]}],
             scoring='accuracy', verbose=1)
```

[54]:
```
# Evaluate the best model found by the grid search on the test set
test_accuracy = grid_search.score(X_test, y_test)

print(f"Best model parameters: {grid_search.best_params_}")
print(f"Accuracy of the best model on the test set: {test_accuracy:.2f}")
```

```
Best model parameters: {'classifier': LogisticRegression(C=7.742636826811269,
max_iter=500, penalty='l1',
                   solver='liblinear'), 'classifier__C': 7.742636826811269,
'classifier__penalty': 'l1'}
Accuracy of the best model on the test set: 0.81
```

10. What are the best model and hyperparameters found in the grid search? Find the accuracy of this model on the test set.

[35]:
```
# evaluating the best model found by the grid search on the test set
best_model_accuracy = grid_search.score(X_test, y_test)

print(f"Accuracy of the Logistic Regression model on the test set:␣
↪{best_model_accuracy:.2f}")
```

```
Accuracy of the Logistic Regression model on the test set: 0.80
```

The best model and hyperparameters identified through the grid search were from the Logistic Regression model.

11. Summarize your results.

Initially, we applied a KNN model, which achieved an accuracy of 67%. After testing additional models, the Logistic Regression emerged as the superior choice, improving accuracy to 79%. This underscores the significance of careful model selection and hyperparameter tuning in the modeling process. These steps are crucial for identifying the most effective model and adjusting its parameters to enhance performance.