Week5_Lakshimi

February 18, 2024

[47]:
```
import pandas as pd
from pycaret.classification import setup, compare_models, predict_model,␣
↪save_model, load_model
import pickle
from IPython.display import Code
```

[17]:
```
df = pd.read_csv("preped_churn_data.csv")
```

0.1 Initalize auto ML environment

[18]:
```
automl_setup = setup(df, target='Churn')
```

```
<pandas.io.formats.style.Styler at 0x7fa8b4ed6d10>
```

The output is a comprehensive summary of the configuration and transformations applied during the setup phase using PyCaret's setup function for the binary classification task on the churn dataset. The session ID `8517` serves as a unique identifier for the current session. The target variable for classification is identified as `Churn` and the dataset initially had 7043 rows and 13 columns.

After transformations, both the transformed data and the training and test sets maintain the same shape as the original dataset. The configuration involves handling 12 numeric features, employing preprocessing steps, using simple imputation with mean for numeric features, and employing mode imputation for categorical features. Stratified K-Fold cross-validation with 10 folds is used, and the setup utilizes available CPU cores (-1) without GPU acceleration. The experiment is not logged, and the default name for the experiment is `clf-default-name.` The Unique Session ID (USI) is `ebba,` providing a unique identifier for tracking and logging purposes. This detailed summary offers insights into the specific settings and transformations applied to the dataset in preparation for the subsequent model comparison and selection steps.

0.2 Compare various classification models and select the best-performng one

[19]:
```
best_model = compare_models()
```

```
<IPython.core.display.HTML object>
```

```
<pandas.io.formats.style.Styler at 0x7fa8bf3e0c50>
```

```
<IPython.core.display.HTML object>
```

The output represents the results of the model comparison process, where various classification models were evaluated based on multiple performance metrics. The LogisticRegression (LR) emerges as the best-performing model with an accuracy of 0.7982, an AUC of 0.8350, a recall of 0.5298, precision of 0.6459, F1 score of 0.5812, Kappa of 0.4502, and MCC of 0.4546. These metrics collectively suggest that LR achieves a well-balanced performance across different aspects, making it the top choice among the models evaluated.

The table also presents other models and their respective performance metrics, allowing for a comparative analysis. LogisticRegression outperforms models such as LightBM, Gradient Boosting Classifier (gbc), Ada Boost Classifier (ada), Ridge Classifier (ridge), Linear Discriminant Analysis (lda), Random Forest Classifier (rf), e.t.c.

[25]:
```
best_model.get_params()
```

[25]:
```
{'C': 1.0,
 'class_weight': None,
 'dual': False,
 'fit_intercept': True,
 'intercept_scaling': 1,
 'l1_ratio': None,
 'max_iter': 1000,
 'multi_class': 'auto',
 'n_jobs': None,
 'penalty': 'l2',
 'random_state': 8517,
 'solver': 'lbfgs',
 'tol': 0.0001,
 'verbose': 0,
 'warm_start': False}
```

[23]:
```
selected_rows = df.iloc[20:32]
selected_rows
```

[23]:
```
    tenure  PhoneService  MonthlyCharges  TotalCharges  Churn  \
20       1             0           39.65         39.65      1
21      12             1           19.80        202.25      0
22       1             1           20.15         20.15      1
23      58             1           59.90       3505.10      0
24      49             1           59.60       2970.30      0
25      30             1           55.30       1530.60      0
26      47             1           99.35       4749.15      1
27       1             0           30.20         30.20      1
28      72             1           90.25       6369.45      0
29      17             1           64.70       1093.10      1
30      71             1           96.35       6766.95      0
31       2             1           95.50        181.65      0

    TotalCharges_to_MonthlyCharges_ratio  \
```

20	1.000000
21	10.214646
22	1.000000
23	58.515860
24	49.837248
25	27.678119
26	47.802214
27	1.000000
28	70.575623
29	16.894900
30	70.233005
31	1.902094

	PaymentMethod_Bank transfer (automatic) \
20	0
21	1
22	0
23	0
24	0
25	1
26	0
27	0
28	0
29	0
30	0
31	0

	PaymentMethod_Credit card (automatic)	PaymentMethod_Electronic check \
20	0	0
21	0	1
22	0	1
23	1	1
24	1	1
25	0	1
26	0	0
27	0	0
28	1	1
29	0	1
30	1	1
31	1	1

	PaymentMethod_Mailed check	Contract_Month-to-month	Contract_One year \
20	0	0	0
21	0	1	1
22	1	0	0
23	0	1	0
24	0	0	0

```
25                            0                        0                  0
26                            0                        0                  0
27                            0                        0                  0
28                            0                        1                  0
29                            1                        0                  0
30                            0                        1                  0
31                            0                        0                  0

    Contract_Two year
20                  0
21                  0
22                  0
23                  1
24                  0
25                  0
26                  0
27                  0
28                  1
29                  0
30                  1
31                  0
```

0.3 predict churn for the selected rows using best_model

[24]:
```
predict_model(best_model, selected_rows)
```

<pandas.io.formats.style.Styler at 0x7fa8ae9f8350>

[24]:
```
    tenure  PhoneService  MonthlyCharges  TotalCharges  \
20       1             0       39.650002     39.650002
21      12             1       19.799999    202.250000
22       1             1       20.150000     20.150000
23      58             1       59.900002   3505.100098
24      49             1       59.599998   2970.300049
25      30             1       55.299999   1530.599976
26      47             1       99.349998   4749.149902
27       1             0       30.200001     30.200001
28      72             1       90.250000   6369.450195
29      17             1       64.699997   1093.099976
30      71             1       96.349998   6766.950195
31       2             1       95.500000    181.649994

    TotalCharges_to_MonthlyCharges_ratio  \
20                              1.000000
21                             10.214646
22                              1.000000
23                             58.515862
```

24	49.837250
25	27.678120
26	47.802216
27	1.000000
28	70.575623
29	16.894899
30	70.233002
31	1.902094

	PaymentMethod_Bank transfer (automatic) \
20	0
21	1
22	0
23	0
24	0
25	1
26	0
27	0
28	0
29	0
30	0
31	0

	PaymentMethod_Credit card (automatic)	PaymentMethod_Electronic check \
20	0	0
21	0	1
22	0	1
23	1	1
24	1	1
25	0	1
26	0	0
27	0	0
28	1	1
29	0	1
30	1	1
31	1	1

	PaymentMethod_Mailed check	Contract_Month-to-month	Contract_One year \
20	0	0	0
21	0	1	1
22	1	0	0
23	0	1	0
24	0	0	0
25	0	0	0
26	0	0	0
27	0	0	0
28	0	1	0

```
29                                 1                            0                       0
30                                 0                            1                       0
31                                 0                            0                       0

     Contract_Two year  Churn  prediction_label  prediction_score
20                   0      1                 1            0.6476
21                   0      0                 0            0.9219
22                   0      1                 0            0.7724
23                   1      0                 0            0.9869
24                   0      0                 0            0.9184
25                   0      0                 0            0.8040
26                   0      1                 0            0.5762
27                   0      1                 1            0.5910
28                   1      0                 0            0.9732
29                   0      1                 0            0.6854
30                   1      0                 0            0.9654
31                   0      0                 1            0.6554
```

Interpreting a few rows,

Row 20 - A customer with 1 month tenure, using PhoneService, has a MonthlyCharge of 39.65. The model predicts Churn (prediction_label = 1) with a probability score of 0.6476.

Row 21 - A customer with 12 months tenure, using PhoneService, MonthlyCharge of 19.80, and TotalCharges of 202.25. The model predicts No Churn (prediction_label = 0) with a high probability score of 0.9219.

Row 22 - A customer with 1 month tenure, using PhoneService, MonthlyCharge of 20.15. The model predicts No Churn (prediction_label = 0) with a probability score of 0.7724.

These interpretations demonstrate how the model predicts churn based on the input features and provides a probability score, which is useful for understanding the model's confidence in its predictions.

0.4 Save best model

[26]:
```
save_model(best_model, 'LR')
```

```
Transformation Pipeline and Model Successfully Saved
```

[26]:
```
(Pipeline(memory=Memory(location=None),
          steps=[('numerical_imputer',
                  TransformerWrapper(exclude=None,
                                     include=['tenure', 'PhoneService',
                                              'MonthlyCharges', 'TotalCharges',
 'TotalCharges_to_MonthlyCharges_ratio',
                                              'PaymentMethod_Bank transfer '
                                              '(automatic)',
                                              'PaymentMethod_Credit card '
                                              '(automatic)',
```

```
                                                'PaymentMethod_Electronic check',
                                                'PaymentMethod_Mailed check',
                                                'Contr…
                     TransformerWrapper(exclude=None, include=None,
transformer=CleanColumnNames(match='[\\]\\[\\,\\{\\}\\"\\:]+'))),
                    ('trained_model',
                     LogisticRegression(C=1.0, class_weight=None, dual=False,
                                        fit_intercept=True, intercept_scaling=1,
                                        l1_ratio=None, max_iter=1000,
                                        multi_class='auto', n_jobs=None,
                                        penalty='l2', random_state=8517,
                                        solver='lbfgs', tol=0.0001, verbose=0,
                                        warm_start=False))],
          verbose=False),
 'LR.pkl')
```

[27]:
```python
with open('LR_model.pk', 'wb') as f:
    pickle.dump(best_model, f)
```

[28]:
```python
with open('LR_model.pk', 'rb') as f:
    loaded_model = pickle.load(f)
```

0.5 new_data from the selected rows

[29]:
```python
new_data = selected_rows.drop('Churn', axis=1).copy()
new_data.to_csv('new_churn_data.csv', index=False)
```

0.6 predict churn for the new data

[30]:
```python
loaded_model.predict(new_data)
```

[30]:
```
array([1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1], dtype=int8)
```

[32]:
```python
loaded_lr = load_model('LR')
predict_model(loaded_lr, new_data)
```

```
Transformation Pipeline and Model Successfully Loaded
```

```
<IPython.core.display.HTML object>
```

[32]:
```
    tenure  PhoneService  MonthlyCharges  TotalCharges  \
20       1             0       39.650002     39.650002
21      12             1       19.799999    202.250000
22       1             1       20.150000     20.150000
23      58             1       59.900002   3505.100098
24      49             1       59.599998   2970.300049
25      30             1       55.299999   1530.599976
26      47             1       99.349998   4749.149902
```

```
27       1             0       30.200001     30.200001
28      72             1       90.250000   6369.450195
29      17             1       64.699997   1093.099976
30      71             1       96.349998   6766.950195
31       2             1       95.500000    181.649994

    TotalCharges_to_MonthlyCharges_ratio  \
20                              1.000000
21                             10.214646
22                              1.000000
23                             58.515862
24                             49.837250
25                             27.678120
26                             47.802216
27                              1.000000
28                             70.575623
29                             16.894899
30                             70.233002
31                              1.902094

    PaymentMethod_Bank transfer (automatic)  \
20                                        0
21                                        1
22                                        0
23                                        0
24                                        0
25                                        1
26                                        0
27                                        0
28                                        0
29                                        0
30                                        0
31                                        0

    PaymentMethod_Credit card (automatic)  PaymentMethod_Electronic check  \
20                                      0                               0
21                                      0                               1
22                                      0                               1
23                                      1                               1
24                                      1                               1
25                                      0                               1
26                                      0                               0
27                                      0                               0
28                                      1                               1
29                                      0                               1
30                                      1                               1
31                                      1                               1
```

```
    PaymentMethod_Mailed check  Contract_Month-to-month  Contract_One year  \
20                          0                        0                  0
21                          0                        1                  1
22                          1                        0                  0
23                          0                        1                  0
24                          0                        0                  0
25                          0                        0                  0
26                          0                        0                  0
27                          0                        0                  0
28                          0                        1                  0
29                          1                        0                  0
30                          0                        1                  0
31                          0                        0                  0

    Contract_Two year  prediction_label  prediction_score
20                  0                 1            0.6476
21                  0                 0            0.9219
22                  0                 0            0.7724
23                  1                 0            0.9869
24                  0                 0            0.9184
25                  0                 0            0.8040
26                  0                 0            0.5762
27                  0                 1            0.5910
28                  1                 0            0.9732
29                  0                 0            0.6854
30                  1                 0            0.9654
31                  0                 1            0.6554
```

0.7 Display predict_churn.py script

[43]:

```
code_display = Code('predict_churn.py')
code_display
```

[43]:

```
import pandas as pd
from pycaret.classification import predict_model, load_model
import os

print("Current Working Directory:", os.getcwd())

df = pd.read_csv('new_churn_data.csv')

model = load_model('LR')

# Make predictions
predictions = predict_model(model, df)

# Rename the prediction label and replace values
```

```
predictions.rename({'prediction_label': 'Churn_prediction'}, axis=1,␣
 ↪inplace=True)
predictions['Churn_prediction'].replace({1: 'Churn', 0: 'No Churn'},␣
 ↪inplace=True)

print(predictions['Churn_prediction'])
```

0.8 Execute script

[45]:
```
%run predict_churn.py
```

```
Transformation Pipeline and Model Successfully Loaded

<IPython.core.display.HTML object>

0        Churn
1     No Churn
2     No Churn
3     No Churn
4     No Churn
5     No Churn
6     No Churn
7        Churn
8     No Churn
9     No Churn
10    No Churn
11       Churn
Name: Churn_prediction, dtype: object
```

The output indicates the predictions for each row in the dataset after loading the model. The `Churn_prediction` column contains the predicted labels, where `Churn` represents instances predicted as churn and `No Churn` represents instances predicted as not churn.

0.9 Summary

The goal is to perform churn prediction using PyCaret. The process begins by reading churn data from a CSV file into a pandas DataFrame. An auto ML environment is then set up using PyCaret's setup function, specifying Churn as the target variable. A variety of classification models are compared using the compare_models function, and the best-performing model which is LogisticRegression, is selected.

The 12 (from 20 to 32) rows of the dataset are extracted, and the best model is used to predict the target variable for these selected rows. The model is saved with the name LR using both PyCaret's save_model function and pickle serialization. The saved model is then loaded back into memory using pickle deserialization.

A new dataset (new_data) is created by copying the selected rows and dropping the Churn column. The loaded model is employed to predict the target variable for this new dataset, and the results are printed. Additionally, the LR model is loaded again using PyCaret's load_model function, and predictions are made for the new dataset.

We conclude by displaying the code for creating a Python module named `predict_churn.py` using IPython's Code display. The %run predict_churn.py command is executed to run the script, making churn predictions using the saved model.