Week5_Assignment

February 20, 2024

[1]:
```
import pandas as pd
from pycaret.classification import setup, compare_models, predict_model,␣
↪save_model, load_model
import pickle
from IPython.display import Code
```

0.1 Load data

[3]:
```
df = pd.read_csv("prepped_churn_data.csv")
df
```

[3]:
```
      tenure  MonthlyCharges  TotalCharges  Churn  MonthlyCharges_log  \
0          1           29.85         29.85      0            3.396185
1         34           56.95       1889.50      0            4.042174
2          2           53.85        108.15      1            3.986202
3         45           42.30       1840.75      0            3.744787
4          2           70.70        151.65      1            4.258446
…        …             …           …      …                   …
7027      24           84.80       1990.50      0            4.440296
7028      72          103.20       7362.90      0            4.636669
7029      11           29.60        346.45      0            3.387774
7030       4           74.40        306.60      1            4.309456
7031      66          105.65       6844.50      0            4.660132

      TotalCharges_Tenure_Ratio  MonthlyCharges_to_TotalCharges_Ratio  \
0                     29.850000                              1.000000
1                     55.573529                              0.030140
2                     54.075000                              0.497920
3                     40.905556                              0.022980
4                     75.825000                              0.466205
…                           …                                    …
7027                  82.937500                              0.042602
7028                 102.262500                              0.014016
7029                  31.495455                              0.085438
7030                  76.650000                              0.242661
7031                 103.704545                              0.015436

      Bank transfer (automatic)  Credit card (automatic)  Electronic check  \
```

```
0                            0                        0                 0
1                            0                        0                 1
2                            0                        0                 1
3                            1                        0                 1
4                            0                        0                 0
…                        …                      …               …
7027                         0                        0                 1
7028                         0                        1                 1
7029                         0                        0                 0
7030                         0                        0                 1
7031                         1                        0                 1

      Mailed check  Month-to-month  One year  Two year
0                0               0         0         0
1                1               1         1         0
2                1               0         0         0
3                0               1         1         0
4                0               0         0         0
…              …             …       …       …
7027             1               1         1         0
7028             0               1         1         0
7029             0               0         0         0
7030             1               0         0         0
7031             0               1         0         1

[7032 rows x 14 columns]
```

0.2 initialize auto ML environment

[4]:
```
automl_setup = setup(df, target='Churn')
```

```
<pandas.io.formats.style.Styler at 0x7f263f095710>
```

The output summarizes the setup information for the PyCaret auto ML environment.

Session id: 7041 - A unique identifier for the PyCaret session.

Target: Churn - The target variable for the classification task is Churn.

Target type: Binary - The target variable is binary, indicating a binary classification task (Churn or no Churn).

Original data shape: (7032, 14) - The original dataset has 7032 rows and 14 columns.

Transformed data shape: (7032, 14) - The transformed dataset after preprocessing remains the same size as the original dataset.

Transformed train set shape: (4922, 14) - The training set after preprocessing contains 4922 samples.

Transformed test set shape: (2110, 14) - The test set after preprocessing contains 2110 samples.

Numeric features: 13 - There are 13 numeric features in the dataset.

Preprocess: True - The data has been preprocessed.

Imputation type: simple - Simple imputation method has been used for handling missing values.

Numeric imputation: mean - Mean imputation has been applied to numeric features.

Categorical imputation: mode - Mode imputation has been applied to categorical features.

Fold Generator: StratifiedKFold - Stratified K-Fold cross-validation is used during model training.

Number: 10 - 10 folds are used in cross-validation.

CPU Jobs: -1 - The number of CPU jobs is set to -1, allowing PyCaret to utilize all available CPUs.

Use GPU: False - GPU acceleration is not utilized for model training.

Log Experiment: False - Logging of the experiment is turned off.

Experiment Name: clf-default-name - The default name for the classification experiment is 'clf-default-name'.

USI: cc2a - A unique identifier for the experiment setup.

[5]:
```
automl_type = type(automl_setup)
automl_type
```

[5]:
```
pycaret.classification.oop.ClassificationExperiment
```

0.3 Compare and select best model

[6]:
```
best_model = compare_models()
```

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

```
<pandas.io.formats.style.Styler at 0x7f263ea74050>
```

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

This output summarizes the performance metrics of various machine learning models trained on the prepped churn dataset, including accuracy, area under the curve (AUC), recall, precision, F1 score, Kappa, Matthews correlation coefficient (MCC), and training time in seconds.

The best performing model based on accuracy:

The Ridge Classifier achieved the highest accuracy of 79.42% followed closely by Logistic Regression accuracy of 79.38%. LDA is another high performer with an accuracy of 79.36%.

Interpreting the results

Accuracy: Indicates the proportion of correctly classified instances out of the total instances.

AUC: Represents the area under the receiver operating characteristic (ROC) curve, which measures the model's ability to distinguish between classes.

Recall: Denotes the proportion of actual positive cases that were correctly identified by the model.

Precision: Indicates the proportion of positive identifications that were actually correct.

F1 Score: Harmonic mean of precision and recall, providing a balance between the two metrics.

Kappa: Measures the agreement between predicted and actual classifications, considering the possibility of the agreement occurring by chance.

MCC (Matthews Correlation Coefficient): Another measure of the quality of binary classifications, considering both false positives and false negatives.

Training Time (TT): Indicates the time taken by each model to train on the dataset.

[7]:
```
best_model_info = best_model
best_model_info
```

[7]:
```
RidgeClassifier(alpha=1.0, class_weight=None, copy_X=True, fit_intercept=True,
                max_iter=None, positive=False, random_state=7041, solver='auto',
                tol=0.0001)
```

0.4 Select specific rows

[8]:
```
selected_rows = df.iloc[400:415]
selected_rows
```

[8]:
```
     tenure  MonthlyCharges  TotalCharges  Churn  MonthlyCharges_log  \
400      32           19.75        624.15      0            2.983153
401      11           20.05        237.70      0            2.998229
402      69           99.45       7007.60      1            4.599655
403      68           55.90       3848.80      0            4.023564
404      20           19.70        419.40      0            2.980619
405      72           19.80       1468.75      0            2.985682
406      60           95.40       5812.00      0            4.558079
407      32           93.95       2861.45      0            4.542763
408       1           19.90         19.90      1            2.990720
409       1           19.60         19.60      1            2.975530
410       3           81.35        233.70      1            4.398761
411      46           24.45       1066.15      0            3.196630
412      29           74.95       2149.05      0            4.316821
413      51           87.35       4473.00      0            4.469923
414      48           70.65       3545.05      0            4.257738

     TotalCharges_Tenure_Ratio  MonthlyCharges_to_TotalCharges_Ratio  \
400                  19.504687                              0.031643
401                  21.609091                              0.084350
402                 101.559420                              0.014192
403                  56.600000                              0.014524
404                  20.970000                              0.046972
405                  20.399306                              0.013481
406                  96.866667                              0.016414
407                  89.420312                              0.032833
408                  19.900000                              1.000000
409                  19.600000                              1.000000
```

```
410              77.900000                                  0.348096
411              23.177174                                  0.022933
412              74.105172                                  0.034876
413              87.705882                                  0.019528
414              73.855208                                  0.019929

     Bank transfer (automatic)  Credit card (automatic)  Electronic check  \
400                          1                        0                 1
401                          0                        1                 1
402                          0                        1                 1
403                          1                        0                 1
404                          0                        0                 1
405                          0                        1                 1
406                          1                        0                 1
407                          0                        0                 1
408                          0                        0                 1
409                          0                        0                 1
410                          0                        0                 0
411                          1                        0                 1
412                          0                        0                 0
413                          0                        0                 0
414                          1                        0                 1

     Mailed check  Month-to-month  One year  Two year
400             0               1         1         0
401             0               1         1         0
402             0               0         0         0
403             0               1         1         0
404             1               1         0         1
405             0               1         0         1
406             0               1         1         0
407             1               0         0         0
408             1               0         0         0
409             1               0         0         0
410             0               0         0         0
411             0               1         1         0
412             0               0         0         0
413             0               0         0         0
414             0               0         0         0
```

0.5 Utilize best model to predict churn

[9]:
```
predict_model(best_model, selected_rows)
```

```
<pandas.io.formats.style.Styler at 0x7f263e72c250>
```

[9]:

```
     tenure  MonthlyCharges  TotalCharges  MonthlyCharges_log  \
400      32       19.750000    624.150024            2.983154
401      11       20.049999    237.699997            2.998229
402      69       99.449997   7007.600098            4.599655
403      68       55.900002   3848.800049            4.023564
404      20       19.700001    419.399994            2.980619
405      72       19.799999   1468.750000            2.985682
406      60       95.400002   5812.000000            4.558079
407      32       93.949997   2861.449951            4.542763
408       1       19.900000     19.900000            2.990720
409       1       19.600000     19.600000            2.975530
410       3       81.349998    233.699997            4.398761
411      46       24.450001   1066.150024            3.196630
412      29       74.949997   2149.050049            4.316821
413      51       87.349998   4473.000000            4.469923
414      48       70.650002   3545.050049            4.257738

     TotalCharges_Tenure_Ratio  MonthlyCharges_to_TotalCharges_Ratio  \
400                  19.504688                              0.031643
401                  21.609091                              0.084350
402                 101.559418                              0.014192
403                  56.599998                              0.014524
404                  20.969999                              0.046972
405                  20.399305                              0.013481
406                  96.866669                              0.016414
407                  89.420311                              0.032833
408                  19.900000                              1.000000
409                  19.600000                              1.000000
410                  77.900002                              0.348096
411                  23.177174                              0.022933
412                  74.105171                              0.034876
413                  87.705879                              0.019528
414                  73.855209                              0.019929

     Bank transfer (automatic)  Credit card (automatic)  Electronic check  \
400                          1                        0                 1
401                          0                        1                 1
402                          0                        1                 1
403                          1                        0                 1
404                          0                        0                 1
405                          0                        1                 1
406                          1                        0                 1
407                          0                        0                 1
408                          0                        0                 1
409                          0                        0                 1
410                          0                        0                 0
411                          1                        0                 1
```

412	0	0	0
413	0	0	0
414	1	0	1

	Mailed check	Month-to-month	One year	Two year	Churn	prediction_label
400	0	1	1	0	0	0
401	0	1	1	0	0	0
402	0	0	0	0	1	0
403	0	1	1	0	0	0
404	1	1	0	1	0	0
405	0	1	0	1	0	0
406	0	1	1	0	0	0
407	1	0	0	0	0	0
408	1	0	0	0	1	0
409	1	0	0	0	1	0
410	0	0	0	0	1	1
411	0	1	1	0	0	0
412	0	0	0	0	0	0
413	0	0	0	0	0	0
414	0	0	0	0	0	0

Metrics

Model: Ridge Classifier Accuracy: 80% AUC: 62.5% Recall: 25% Precision: 100% F1 Score: 40% Kappa: 32.84% MCC: 44.32%

While the model exhibits high precision, suggesting it correctly identifies churn when it occurs, its recall is quite low, indicating it misses many actual churn instances.

0.6 Incorrect predictions

[10]:

```
predicted_rows = predict_model(best_model, selected_rows)
incorrect_predictions = (predicted_rows['Churn'] !=␣
 ↪predicted_rows['prediction_label']).sum()

print("Incorrect Predictions:", incorrect_predictions)
```

```
<pandas.io.formats.style.Styler at 0x7f263fec20d0>
```

```
Incorrect Predictions: 3
```

Out of the total predictions made, the model was incorrect in predicting the churn status of 3 customers.

0.7 Save model

[11]:

```
save_model(best_model, 'ridge')
```

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

```
[11]: (Pipeline(memory=Memory(location=None),
                steps=[('numerical_imputer',
                        TransformerWrapper(exclude=None,
                                           include=['tenure', 'MonthlyCharges',
                                                    'TotalCharges',
                                                    'MonthlyCharges_log',
                                                    'TotalCharges_Tenure_Ratio',
      'MonthlyCharges_to_TotalCharges_Ratio',
                                                    'Bank transfer (automatic)',
                                                    'Credit card (automatic)',
                                                    'Electronic check', 'Mailed
      check',
                                                    'Month-to-month', 'One year',
                                                    'Two y…
      strategy='most_frequent',
      verbose='deprecated'))),
                       ('clean_column_names',
                        TransformerWrapper(exclude=None, include=None,
      transformer=CleanColumnNames(match='[\\]\\[\\,\\{\\}\\"\\:]+'))),
                       ('trained_model',
                        RidgeClassifier(alpha=1.0, class_weight=None, copy_X=True,
                                        fit_intercept=True, max_iter=None,
                                        positive=False, random_state=7041,
                                        solver='auto', tol=0.0001))],
                verbose=False),
       'ridge.pkl')
```

0.8 Use pickle to save and load the model (serialization and deserialization)

```
[14]: with open('ridge_model.pk', 'wb') as f:
          pickle.dump(best_model, f)
```

```
[16]: with open('ridge_model.pk', 'rb') as f:
          loaded_model = pickle.load(f)
```

0.9 Create new data

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

0.10 Make predictions for churn on the loaded data

```
[18]: loaded_model.predict(new_data)
```

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

[19]:
```
loaded_ridge = load_model('ridge')
predict_model(loaded_ridge, new_data)
```

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

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

[19]:
```
     tenure  MonthlyCharges  TotalCharges  MonthlyCharges_log  \
400      32       19.750000    624.150024            2.983154
401      11       20.049999    237.699997            2.998229
402      69       99.449997   7007.600098            4.599655
403      68       55.900002   3848.800049            4.023564
404      20       19.700001    419.399994            2.980619
405      72       19.799999   1468.750000            2.985682
406      60       95.400002   5812.000000            4.558079
407      32       93.949997   2861.449951            4.542763
408       1       19.900000     19.900000            2.990720
409       1       19.600000     19.600000            2.975530
410       3       81.349998    233.699997            4.398761
411      46       24.450001   1066.150024            3.196630
412      29       74.949997   2149.050049            4.316821
413      51       87.349998   4473.000000            4.469923
414      48       70.650002   3545.050049            4.257738

     TotalCharges_Tenure_Ratio  MonthlyCharges_to_TotalCharges_Ratio  \
400                  19.504688                              0.031643
401                  21.609091                              0.084350
402                 101.559418                              0.014192
403                  56.599998                              0.014524
404                  20.969999                              0.046972
405                  20.399305                              0.013481
406                  96.866669                              0.016414
407                  89.420311                              0.032833
408                  19.900000                              1.000000
409                  19.600000                              1.000000
410                  77.900002                              0.348096
411                  23.177174                              0.022933
412                  74.105171                              0.034876
413                  87.705879                              0.019528
414                  73.855209                              0.019929

     Bank transfer (automatic)  Credit card (automatic)  Electronic check  \
400                          1                        0                 1
401                          0                        1                 1
402                          0                        1                 1
403                          1                        0                 1
404                          0                        0                 1
405                          0                        1                 1
```

```
406                 1                  0              1
407                 0                  0              1
408                 0                  0              1
409                 0                  0              1
410                 0                  0              0
411                 1                  0              1
412                 0                  0              0
413                 0                  0              0
414                 1                  0              1

     Mailed check  Month-to-month  One year  Two year  prediction_label
400             0               1         1         0                 0
401             0               1         1         0                 0
402             0               0         0         0                 0
403             0               1         1         0                 0
404             1               1         0         1                 0
405             0               1         0         1                 0
406             0               1         1         0                 0
407             1               0         0         0                 0
408             1               0         0         0                 0
409             1               0         0         0                 0
410             0               0         0         0                 1
411             0               1         1         0                 0
412             0               0         0         0                 0
413             0               0         0         0                 0
414             0               0         0         0                 0
```

0.11 Python Module to predict churn

[20]:

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

[20]:

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

def predict_churn():
    df = pd.read_csv('new_churn_data.csv')
    model = load_model('ridge')
    predictions = predict_model(model, df)
    predictions.rename({'prediction_label': 'Churn_prediction'}, axis=1,␣
 ↪inplace=True)
    predictions['Churn_prediction'].replace({1: 'Churn', 0: 'No Churn'},␣
 ↪inplace=True)
    return predictions['Churn_prediction']

# Call the function and print the predictions
print(predict_churn())
```

[21]:
```
%run predict_churn.py
```

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

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

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

The output indicates the churn predictions for each customer in the dataset. Each entry in the output corresponds to a customer, and it shows whether the model predicts that the customer will churn or not churn.

Necessary libraries and functions were imported for this process, which involves building a churn prediction model using PyCaret, a Python library for automating machine learning workflows.

We successfully achieved the following,

Loaded and prepared the churn data.

Set up an auto ML environment and compared classification models.

Selected the best-performing model which was Ridge Classifiet

Predicted the churn status for 15 specific rows of data using the selected model.

Saved the best-performing model to a file using PyCaret's save_model function.

Serialized and deserialized the model using pickle.

Predicted the churn status for new data using both the loaded model and PyCaret's load_model function