Week5_Mounika_Lakureddy

February 18, 2024

[21]:
```
# ! pip install pycaret
```

[22]:
```
import pandas as pd
```

0.1 Load data

[60]:
```
df = pd.read_csv("prepped_churn_data.csv")
df.head(15)
```

[60]:
```
    tenure  PhoneService  MonthlyCharges  TotalCharges  Churn  \
0        1             0           29.85         29.85      0
1       34             1           56.95       1889.50      0
2        2             1           53.85        108.15      1
3       45             0           42.30       1840.75      0
4        2             1           70.70        151.65      1
5        8             1           99.65        820.50      1
6       22             1           89.10       1949.40      0
7       10             0           29.75        301.90      0
8       28             1          104.80       3046.05      1
9       62             1           56.15       3487.95      0
10      13             1           49.95        587.45      0
11      16             1           18.95        326.80      0
12      58             1          100.35       5681.10      0
13      49             1          103.70       5036.30      1
14      25             1          105.50       2686.05      0

    MonthlyCharges_to_TotalCharges_Ratio  Bank transfer (automatic)  \
0                               1.000000                          0
1                               0.030140                          0
2                               0.497920                          0
3                               0.022980                          1
4                               0.466205                          0
5                               0.121450                          0
6                               0.045706                          0
7                               0.098543                          0
8                               0.034405                          0
9                               0.016098                          1
10                              0.085029                          0
```

```
11                        0.057987                         0
12                        0.017664                         0
13                        0.020591                         1
14                        0.039277                         0

    Credit card (automatic)  Electronic check  Mailed check  Month-to-month  \
0                         0                 0             0               0
1                         0                 1             1               1
2                         0                 1             1               0
3                         0                 1             0               1
4                         0                 0             0               0
5                         0                 0             0               0
6                         1                 1             0               0
7                         0                 1             1               0
8                         0                 0             0               0
9                         0                 1             0               1
10                        0                 1             1               0
11                        1                 1             0               1
12                        1                 1             0               1
13                        0                 1             0               0
14                        0                 0             0               0

    One year  Two year
0          0         0
1          1         0
2          0         0
3          1         0
4          0         0
5          0         0
6          0         0
7          0         0
8          0         0
9          1         0
10         0         0
11         0         1
12         1         0
13         0         0
14         0         0
```

0.2 Import Pycaret functions and classes

[24]:
```python
from pycaret.classification import setup, compare_models, predict_model,␣
↪save_model, load_model
```

0.3 setup

[25]:
```
automl = setup(df, target='Churn')
```

```
<pandas.io.formats.style.Styler at 0x7f4b1c2618d0>
```

The output from the setup function in PyCaret, is used to set up the environment for machine learning. It provides information about the configuration and the preprocessing steps applied to the dataset.

Session id: A unique identifier for the current PyCaret session which is 581.

Target: The target variable for the machine learning task. In this case, it is Churn, indicating thatwe are working on a binary classification problem where the goal is to predict whether a customer will churn or not.

Target type: Specifies the nature of the target variable. Binary indicates that it is a binary classification task.

Original data shape: The shape of the original dataset before any preprocessing. In this case, it was (7032, 13), meaning there are 7032 rows and 13 columns in the original dataset.

Transformed data shape: The shape of the dataset after preprocessing. It remains the same in this case, indicating that no feature engineering or dimensionality reduction was performed.

Transformed train set shape: The shape of the training set after preprocessing. In this case, it is (4922, 13), meaning that 4922 samples are used for training.

Transformed test set shape: The shape of the test set after preprocessing is (2110, 13), showing that 2110 samples are used for testing.

Numeric features: The number of numeric features in the dataset, which are 12 numeric features.

Preprocess: Indicates whether preprocessing was performed. `True` suggests that preprocessing steps, such as imputation and scaling, were applied.

Imputation type: Specifies the type of imputation used for missing values. `Simple` means basic imputation techniques were applied.

Numeric imputation: The strategy used for imputing missing values in numeric features. `Mean` means that the mean value was used.

Categorical imputation: The strategy used for imputing missing values in categorical features. `Mode` indicates that the mode (most frequent value) was used.

Fold Generator: The cross-validation strategy used. `StratifiedKFold` indicates that stratified k-fold cross-validation was employed.

Fold Number: The number of folds used in cross-validation - 10

CPU Jobs: The number of CPU cores used during parallel processing. `-1` usually means to use all available cores.

Experiment Name: The name assigned to the current machine learning experiment. In this case, it is `clf-default-name.`

USI: The User System Identifier, a unique identifier for the current user's system.

[26]:
```
type(automl)
```

[26]:
```
pycaret.classification.oop.ClassificationExperiment
```

0.4 compare models

[27]:
```
best_model = compare_models()
```

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

<IPython.core.display.HTML object>

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

A machine learning model selection process was performed using PyCaret. The summary provides information about various classification models and their performance metrics based on a 10-fold cross-validation.

Below is the best model according to the metrics

```
Best Model: Logistic Regression (lr)
    Accuracy: 0.7997
    AUC (Area Under the Curve): 0.8419
    Recall: 0.5252
    Precision: 0.6549
    F1 Score: 0.5824
    Kappa: 0.4528
    MCC (Matthews Correlation Coefficient): 0.4579
    Training Time: 2.0710 seconds
```

Other models were also evaluated, and their respective performance metrics are presented in the table. The evaluation metrics include Accuracy, AUC, Recall, Precision, F1 Score, Kappa, MCC, and Training Time.

[28]:
```
best_model
```

[28]:
```
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=663, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

here we print out the best_model hyperparameters'. It was determined to be Logistic Regression

0.5 select 2nd-to-last row from the DF

[29]:
```
df.iloc[-2:-1]
```

[29]:
```
      tenure  PhoneService  MonthlyCharges  TotalCharges  Churn  \
7030       4             1            74.4         306.6      1
```

```
      MonthlyCharges_to_TotalCharges_Ratio  Bank transfer (automatic)  \
7030                              0.242661                          0

      Credit card (automatic)  Electronic check  Mailed check  Month-to-month  \
7030                        0                 1             1               0

      One year  Two year
7030         0         0
```

[30]:
```
predict_model(best_model, df.iloc[-2:-1])
```

```
<pandas.io.formats.style.Styler at 0x7f4b1c32ad50>
```

[30]:
```
      tenure  PhoneService  MonthlyCharges  TotalCharges  \
7030       4             1       74.400002    306.600006

      MonthlyCharges_to_TotalCharges_Ratio  Bank transfer (automatic)  \
7030                              0.242661                          0

      Credit card (automatic)  Electronic check  Mailed check  Month-to-month  \
7030                        0                 1             1               0

      One year  Two year  Churn  prediction_label  prediction_score
7030         0         0      1                 0            0.5686
```

Here we call and predict the Churn label and assign a prediction score 0 (indicates the predicted class is 'No Churn') with a prediction score of 0.5686

Interpretation

The actual Churn value for this instance is 1, indicating that Churn occurred. However, the model predicted a Churn label of 0 ('No Churn') with a prediction score of 0.5686.

Possible issues The model may not be well-calibrated or might need further tuning. There could be an issue with the evaluation setup or the way the model was trained.

0.6 save model to disk

[31]:
```
save_model(best_model, 'LR')
```

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

[31]:
```
(Pipeline(memory=Memory(location=None),
          steps=[('numerical_imputer',
                  TransformerWrapper(exclude=None,
                                     include=['tenure', 'PhoneService',
                                              'MonthlyCharges', 'TotalCharges',
'MonthlyCharges_to_TotalCharges_Ratio',
                                              'Bank transfer (automatic)',
                                              'Credit card (automatic)',
```

```
                                                        'Electronic check', 'Mailed
check',
                                                        'Month-to-month', 'One year',
                                                        'Two year'],
                                             transformer=SimpleImputer(ad…
                      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=663,
                                         solver='lbfgs', tol=0.0001, verbose=0,
                                         warm_start=False))],
            verbose=False),
 'LR.pkl')
```

[32]:

```
import pickle
```

0.7 save and load model

[33]:

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

[34]:

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

[49]:

```
new_data = pd.concat([df.iloc[-1:].copy()] * 10, ignore_index=True)
new_data.drop('Churn', axis=1, inplace=True)

new_data
```

[49]:

```
   tenure  PhoneService  MonthlyCharges  TotalCharges  \
0      66             1          105.65        6844.5
1      66             1          105.65        6844.5
2      66             1          105.65        6844.5
3      66             1          105.65        6844.5
4      66             1          105.65        6844.5
5      66             1          105.65        6844.5
6      66             1          105.65        6844.5
7      66             1          105.65        6844.5
8      66             1          105.65        6844.5
9      66             1          105.65        6844.5

   MonthlyCharges_to_TotalCharges_Ratio  Bank transfer (automatic)  \
0                              0.015436                          1
```

```
1                                 0.015436                                1
2                                 0.015436                                1
3                                 0.015436                                1
4                                 0.015436                                1
5                                 0.015436                                1
6                                 0.015436                                1
7                                 0.015436                                1
8                                 0.015436                                1
9                                 0.015436                                1

   Credit card (automatic)  Electronic check  Mailed check  Month-to-month  \
0                        0                 1             0               1
1                        0                 1             0               1
2                        0                 1             0               1
3                        0                 1             0               1
4                        0                 1             0               1
5                        0                 1             0               1
6                        0                 1             0               1
7                        0                 1             0               1
8                        0                 1             0               1
9                        0                 1             0               1

   One year  Two year
0         0         1
1         0         1
2         0         1
3         0         1
4         0         1
5         0         1
6         0         1
7         0         1
8         0         1
9         0         1
```

We create new_data by copying the second-to-last row of the DataFrame and dropping the'Churn column

0.8 Predict the target variable for the new_data

[50]:
```
loaded_model.predict(new_data)
```

[50]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int8)
```

[52]:
```
loaded_lr = load_model('LR')
```

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

[53]:
```
predict_model(loaded_lr, new_data)
```

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

[53]:

```
   tenure  PhoneService  MonthlyCharges  TotalCharges  \
0      66             1      105.650002        6844.5
1      66             1      105.650002        6844.5
2      66             1      105.650002        6844.5
3      66             1      105.650002        6844.5
4      66             1      105.650002        6844.5
5      66             1      105.650002        6844.5
6      66             1      105.650002        6844.5
7      66             1      105.650002        6844.5
8      66             1      105.650002        6844.5
9      66             1      105.650002        6844.5

   MonthlyCharges_to_TotalCharges_Ratio  Bank transfer (automatic)  \
0                              0.015436                          1
1                              0.015436                          1
2                              0.015436                          1
3                              0.015436                          1
4                              0.015436                          1
5                              0.015436                          1
6                              0.015436                          1
7                              0.015436                          1
8                              0.015436                          1
9                              0.015436                          1

   Credit card (automatic)  Electronic check  Mailed check  Month-to-month  \
0                        0                 1             0               1
1                        0                 1             0               1
2                        0                 1             0               1
3                        0                 1             0               1
4                        0                 1             0               1
5                        0                 1             0               1
6                        0                 1             0               1
7                        0                 1             0               1
8                        0                 1             0               1
9                        0                 1             0               1

   One year  Two year  prediction_label  prediction_score
0         0         1                 0             0.925
1         0         1                 0             0.925
2         0         1                 0             0.925
3         0         1                 0             0.925
4         0         1                 0             0.925
5         0         1                 0             0.925
6         0         1                 0             0.925
7         0         1                 0             0.925
```

```
8          0          1                  0              0.925
9          0          1                  0              0.925
```

[55]:
```
# Save the DataFrame to a CSV file
new_data.to_csv('new_churn_data.csv', index=False)
```

0.9 Using Python Module to make predictions

[58]:
```
from IPython.display import Code
Code('predict_churn.py')
```

[58]:
```
import pandas as pd
from pycaret.classification import predict_model, load_model

def load_data(filepath):
    """
    Loads churn data into a DataFrame from a string filepath.
    """
    df = pd.read_csv(filepath)
    return df


def make_predictions(df):
    """
    Uses the pycaret best model to make predictions on data in the df dataframe.
    """
    model = load_model('LR')
    predictions = predict_model(model, data=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']


if __name__ == "__main__":
    df = load_data('new_churn_data.csv')
    predictions = make_predictions(df)
    print('predictions:')
    print(predictions)
```

[59]:
```
%run predict_churn.py
```

```
Transformation Pipeline and Model Successfully Loaded

<IPython.core.display.HTML object>

predictions:
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
Name: Churn_prediction, dtype: object
```

The Python module is successfully loading the transformation pipeline and model, and it's making predictions on the new data. The predictions are currently all No Churn.

0.10 Summary

We began by importing the necessary libraries, including pandas for data manipulation and PyCaret for automated machine learning tasks. Then, we loaded the prepped churn data from a CSV file into a pandas DataFrame. Using PyCaret's setup function, we initialized the auto ML environment, specifying the target variable as `Churn`. After setting up the environment, we compared different classification models to select the best-performing one which was found to be Logistic Regression.

Once the best model was identified, it was saved both as a file named `LR` and using the pickle serialization method for Python objects. Subsequently, we loaded the saved model using pickle deserialization and used it to make predictions on a new dataset by copying 10 rows of the DataFrame and dropping the Churn column.

Additionally, we demonstrated how to load the saved model using PyCaret's load_model function and make predictions on the same new dataset. Finally, we created a Python module named `predict_churn.py` using IPython's Code display feature and ran the script using the %run magic command, effectively summarizing the entire process of loading the model and making predictions on new data encapsulated within a reusable Python module.