Week5_Assignment

February 19, 2024

[1]:
```python
import pandas as pd
```

0.1 Load dataset

[2]:
```python
df = pd.read_csv("prepared_churn_data.csv")
df.head(15)
```

[2]:
```
    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 Setup autoML Environment

[3]:
```
from pycaret.classification import setup, compare_models, predict_model,␣
↪save_model, load_model
```

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

```
<pandas.io.formats.style.Styler at 0x7f2d8857c850>
```

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 1589.

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 - 73dc

[5]:
```
type(automl)
```

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

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

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

```
<pandas.io.formats.style.Styler at 0x7f2d83f15b90>
```

<IPython.core.display.HTML object>

We performed Machine learning model selection process 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: Gradient Boosting Classifier (gbc)
    Accuracy: 0.7917
    AUC (Area Under the Curve): 0.8347
    Recall: 0.4793
    Precision: 0.6549
    F1 Score: 0.5490
    Kappa: 0.4178
    MCC (Matthews Correlation Coefficient): 0.4263
```

Other models were also evaluated, and their respective performance metrics are presented in the table.

[7]:
```
best_model
```

[7]:
```
GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='log_loss', max_depth=3,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_samples_leaf=1,
                           min_samples_split=2, min_weight_fraction_leaf=0.0,
                           n_estimators=100, n_iter_no_change=None,
                           random_state=1589, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)
```

0.3 Select specific rows

[8]:
```
df.iloc[90:100]
```

[8]:
```
    tenure  PhoneService  MonthlyCharges  TotalCharges  Churn  \
90      30             1           82.05       2570.20      0
91       1             1           74.70         74.70      0
92      66             1           84.00       5714.25      0
93      65             1          111.05       7107.00      0
94      72             1          100.90       7459.05      0
95      12             1           78.95        927.35      1
96      71             1           66.85       4748.70      0
97       5             1           21.05        113.85      1
98      52             1           21.00       1107.20      0
99      25             1           98.50       2514.50      1

    MonthlyCharges_to_TotalCharges_Ratio  Bank transfer (automatic)  \
```

```
90                   0.031924                          1
91                   1.000000                          0
92                   0.014700                          0
93                   0.015625                          0
94                   0.013527                          1
95                   0.085135                          0
96                   0.014078                          0
97                   0.184892                          0
98                   0.018967                          1
99                   0.039173                          0

    Credit card (automatic)  Electronic check  Mailed check  Month-to-month  \
90                        0                 1             0               0
91                        0                 0             0               0
92                        0                 1             1               1
93                        1                 1             0               0
94                        0                 1             0               1
95                        0                 0             0               0
96                        1                 1             0               1
97                        0                 1             1               0
98                        0                 1             0               1
99                        0                 0             0               0

    One year  Two year
90         0         0
91         0         0
92         0         1
93         0         0
94         0         1
95         0         0
96         1         0
97         0         0
98         0         1
99         0         0
```

[9]:
```
predict_model(best_model, df.iloc[90:100])
```

```
<pandas.io.formats.style.Styler at 0x7f2d8851a110>
```

[9]:
```
    tenure  PhoneService  MonthlyCharges  TotalCharges  \
90      30             1       82.050003   2570.199951
91       1             1       74.699997     74.699997
92      66             1       84.000000   5714.250000
93      65             1      111.050003   7107.000000
94      72             1      100.900002   7459.049805
95      12             1       78.949997    927.349976
96      71             1       66.849998   4748.700195
97       5             1       21.049999    113.849998
```

```
98      52             1       21.000000   1107.199951
99      25             1       98.500000   2514.500000

    MonthlyCharges_to_TotalCharges_Ratio  Bank transfer (automatic)  \
90                              0.031924                          1
91                              1.000000                          0
92                              0.014700                          0
93                              0.015625                          0
94                              0.013527                          1
95                              0.085135                          0
96                              0.014078                          0
97                              0.184892                          0
98                              0.018967                          1
99                              0.039173                          0

    Credit card (automatic)  Electronic check  Mailed check  Month-to-month  \
90                        0                 1             0               0
91                        0                 0             0               0
92                        0                 1             1               1
93                        1                 1             0               0
94                        0                 1             0               1
95                        0                 0             0               0
96                        1                 1             0               1
97                        0                 1             1               0
98                        0                 1             0               1
99                        0                 0             0               0

    One year  Two year  Churn  prediction_label  prediction_score
90         0         0      0                 0            0.7120
91         0         0      0                 1            0.8048
92         0         1      0                 0            0.9664
93         0         0      0                 0            0.7174
94         0         1      0                 0            0.9753
95         0         0      1                 1            0.5302
96         1         0      0                 0            0.9741
97         0         0      1                 0            0.8441
98         0         1      0                 0            0.9647
99         0         0      1                 1            0.7361
```

observations regarding the prediction of the churn label

Model Performance: The model achieved an accuracy of 80%, which indicates that 80% of the predictions made by the model were correct.

AUC Score: The AUC score is 0.7619, which suggests that the model has some ability to distinguish between churned and non-churned customers. AUC values closer to 1 indicate better discrimination ability.

Recall and Precision: The recall, precision, and F1 score are all approximately 0.67, which means that the model correctly identifies around 67% of the churned customers (recall), and when it predicts a customer will churn, it is correct around 67% of the time (precision). While these values are not extremely high, they indicate a moderate performance in identifying churned customers.

Kappa and MCC: The Kappa statistic and MCC are both 0.5238, which suggests moderate agreement between the actual and predicted churn labels. These metrics take into account the possibility of the agreement occurring by chance.

0.4 save model to disk

[10]:
```
save_model(best_model, 'GBC')
```

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

[10]:
```
(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…
                                             criterion='friedman_mse',
init=None,
                                             learning_rate=0.1, loss='log_loss',
                                             max_depth=3, max_features=None,
                                             max_leaf_nodes=None,
                                             min_impurity_decrease=0.0,
                                             min_samples_leaf=1,
                                             min_samples_split=2,
                                             min_weight_fraction_leaf=0.0,
                                             n_estimators=100,
                                             n_iter_no_change=None,
                                             random_state=1589, subsample=1.0,
                                             tol=0.0001,
validation_fraction=0.1,
                                             verbose=0, warm_start=False))],
          verbose=False),
 'GBC.pkl')
```

[11]:
```
import pickle
```

0.5 Save and load model

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

[15]:
```
with open('GBC_model.pk', 'rb') as f:
    loaded_model = pickle.load(f)
```

[23]:
```
rows = df.iloc[90:100]
new_data = rows.copy()
new_data.drop('Churn', axis=1, inplace=True)
new_data.to_csv('new_churn_data.csv', index=False)


new_data
```

[23]:
```
    tenure  PhoneService  MonthlyCharges  TotalCharges  \
90      30             1           82.05       2570.20
91       1             1           74.70         74.70
92      66             1           84.00       5714.25
93      65             1          111.05       7107.00
94      72             1          100.90       7459.05
95      12             1           78.95        927.35
96      71             1           66.85       4748.70
97       5             1           21.05        113.85
98      52             1           21.00       1107.20
99      25             1           98.50       2514.50

    MonthlyCharges_to_TotalCharges_Ratio  Bank transfer (automatic)  \
90                              0.031924                          1
91                              1.000000                          0
92                              0.014700                          0
93                              0.015625                          0
94                              0.013527                          1
95                              0.085135                          0
96                              0.014078                          0
97                              0.184892                          0
98                              0.018967                          1
99                              0.039173                          0

    Credit card (automatic)  Electronic check  Mailed check  Month-to-month  \
90                        0                 1             0               0
91                        0                 0             0               0
92                        0                 1             1               1
93                        1                 1             0               0
94                        0                 1             0               1
95                        0                 0             0               0
96                        1                 1             0               1
```

```
97                       0                  1             1               0
98                       0                  1             0               1
99                       0                  0             0               0

    One year  Two year
90         0         0
91         0         0
92         0         1
93         0         0
94         0         1
95         0         0
96         1         0
97         0         0
98         0         1
99         0         0
```

0.6 predict churn for the loaded data

[25]:
```python
loaded_model.predict(new_data)
```

[25]:
```
array([0, 1, 0, 0, 0, 1, 0, 0, 0, 1], dtype=int8)
```

[26]:
```python
loaded_gbc = load_model('GBC')
```

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

[27]:
```python
predict_model(loaded_gbc, new_data)
```

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

[27]:
```
    tenure  PhoneService  MonthlyCharges  TotalCharges  \
90      30             1       82.050003   2570.199951
91       1             1       74.699997     74.699997
92      66             1       84.000000   5714.250000
93      65             1      111.050003   7107.000000
94      72             1      100.900002   7459.049805
95      12             1       78.949997    927.349976
96      71             1       66.849998   4748.700195
97       5             1       21.049999    113.849998
98      52             1       21.000000   1107.199951
99      25             1       98.500000   2514.500000

    MonthlyCharges_to_TotalCharges_Ratio  Bank transfer (automatic)  \
90                              0.031924                          1
91                              1.000000                          0
92                              0.014700                          0
93                              0.015625                          0
94                              0.013527                          1
```

```
95                           0.085135                          0
96                           0.014078                          0
97                           0.184892                          0
98                           0.018967                          1
99                           0.039173                          0

    Credit card (automatic)  Electronic check  Mailed check  Month-to-month  \
90                        0                 1             0               0
91                        0                 0             0               0
92                        0                 1             1               1
93                        1                 1             0               0
94                        0                 1             0               1
95                        0                 0             0               0
96                        1                 1             0               1
97                        0                 1             1               0
98                        0                 1             0               1
99                        0                 0             0               0

    One year  Two year  prediction_label  prediction_score
90         0         0                 0            0.7120
91         0         0                 1            0.8048
92         0         1                 0            0.9664
93         0         0                 0            0.7174
94         0         1                 0            0.9753
95         0         0                 1            0.5302
96         1         0                 0            0.9741
97         0         0                 0            0.8441
98         0         1                 0            0.9647
99         0         0                 1            0.7361
```

0.7 Python module to make predictions

[28]:
```python
from IPython.display import Code
Code('predict_churn.py')
```

[28]:
```python
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('GBC')
    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)
```

[29]:
```
%run predict_churn.py
```

```
Transformation Pipeline and Model Successfully Loaded

<IPython.core.display.HTML object>

predictions:
0    No Churn
1       Churn
2    No Churn
3    No Churn
4    No Churn
5       Churn
6    No Churn
7    No Churn
8    No Churn
9       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 7 No Churn and 3 Churn

0.8 Summary

Started off by importing essential libraries, leveraging pandas for data manipulation and PyCaret for automating machine learning tasks. Next, we fetched the prepared churn data from a CSV file and loaded it into a pandas DataFrame. Using PyCaret's setup function, we initialized the automated ML environment, specifying the target variable as Churn.

Once the environment was set up, we conducted a comparative analysis of various classification models to pinpoint the top performer, which turned out to be Gradient Boosting Classifier. After identifying the best model, we saved it both as a file named `GBC` and using Python's pickle serialization method for objects.

Following this, we loaded the saved model using pickle deserialization and applied it to predict outcomes on a new dataset. This involved copying 10 rows of the DataFrame and omitting the Churn column.

In addition, we demonstrated how to load the saved model using PyCaret's load_model function and make predictions on the same new dataset. Finally, we encapsulated the entire process, from loading the model to making predictions on new data, within a reusable Python module named predict_churn.py. This module was created using IPython's Code display feature and executed using the %run magic command, thereby summarizing the workflow effectively for future use.