McBride_Week_5_Assignment

November 25, 2023

1 DS Automation Assignment

Using our prepared churn data from week 2: - use pycaret to find an ML algorithm that performs best on the data - Choose a metric you think is best to use for finding the best model; by default, it is accuracy but it could be AUC, precision, recall, etc. The week 3 FTE has some information on these different metrics. - save the model to disk - create a Python script/file/module with a function that takes a pandas dataframe as an input and returns the probability of churn for each row in the dataframe - your Python file/function should print out the predictions for new data (new_churn_data.csv) - the true values for the new data are [1, 0, 0, 1, 0] if you're interested - test your Python module and function with the new data, new_churn_data.csv - write a short summary of the process and results at the end of this notebook - upload this Jupyter Notebook and Python file to a Github repository, and turn in a link to the repository in the week 5 assignment dropbox

Optional challenges: - return the probability of churn for each new prediction, and the percentile where that prediction is in the distribution of probability predictions from the training dataset (e.g. a high probability of churn like 0.78 might be at the 90th percentile) - use other autoML packages, such as TPOT, H2O, MLBox, etc, and compare performance and features with pycaret - create a class in your Python module to hold the functions that you created - accept user input to specify a file using a tool such as Python's `input()` function, the `click` package for command-line arguments, or a GUI - Use the unmodified churn data (new_unmodified_churn_data.csv) in your Python script. This will require adding the same preprocessing steps from week 2 since this data is like the original unmodified dataset from week 1.

1.1 Load Data

[36]:
```python
import pandas as pd

df = pd.read_csv('cleaned_churn_data.csv', index_col='customerID')
df
```

[36]:
```
            tenure  PhoneService  Contract  PaymentMethod  MonthlyCharges  \
customerID
7590-VHVEG       1             0         0              0           29.85
5575-GNVDE      34             1         1              1           56.95
3668-QPYBK       2             1         0              1           53.85
7795-CFOCW      45             0         1              2           42.30
9237-HQITU       2             1         0              0           70.70
```

```
…               …              …      …            …            …
6840-RESVB      24             1      1            1        84.80
2234-XADUH      72             1      1            3       103.20
4801-JZAZL      11             0      0            0        29.60
8361-LTMKD       4             1      0            1        74.40
3186-AJIEK      66             1      2            2       105.65

            TotalCharges  Churn  Total_Charge_Validation  Potential_Discount
customerID
7590-VHVEG         29.85      0                    29.85                0.00
5575-GNVDE       1889.50      0                  1936.30               46.80
3668-QPYBK        108.15      1                   107.70               -0.45
7795-CFOCW       1840.75      0                  1903.50               62.75
9237-HQITU        151.65      1                   141.40              -10.25
…                    …      …                        …                  …
6840-RESVB       1990.50      0                  2035.20               44.70
2234-XADUH       7362.90      0                  7430.40               67.50
4801-JZAZL        346.45      0                   325.60              -20.85
8361-LTMKD        306.60      1                   297.60               -9.00
3186-AJIEK       6844.50      0                  6972.90              128.40

[7032 rows x 9 columns]
```

[37]:

```python
# Modifying dataset to fit later parameters on the model
df.drop('Total_Charge_Validation', axis=1, inplace=True)
df.drop('Potential_Discount', axis=1, inplace=True)
df
```

[37]:

```
            tenure  PhoneService  Contract  PaymentMethod  MonthlyCharges  \
customerID
7590-VHVEG       1             0         0              0           29.85
5575-GNVDE      34             1         1              1           56.95
3668-QPYBK       2             1         0              1           53.85
7795-CFOCW      45             0         1              2           42.30
9237-HQITU       2             1         0              0           70.70
…               …              …      …            …            …
6840-RESVB      24             1         1              1           84.80
2234-XADUH      72             1         1              3          103.20
4801-JZAZL      11             0         0              0           29.60
8361-LTMKD       4             1         0              1           74.40
3186-AJIEK      66             1         2              2          105.65

            TotalCharges  Churn
customerID
7590-VHVEG         29.85      0
5575-GNVDE       1889.50      0
3668-QPYBK        108.15      1
```

```
7795-CFOCW        1840.75      0
9237-HQITU         151.65      1
…                    …     …
6840-RESVB        1990.50      0
2234-XADUH        7362.90      0
4801-JZAZL         346.45      0
8361-LTMKD         306.60      1
3186-AJIEK        6844.50      0

[7032 rows x 7 columns]
```

1.2 Initialize Pycaret

[38]:
```
from pycaret.classification import ClassificationExperiment
```

[39]:
```
automl = ClassificationExperiment()
```

[40]:
```
automl.setup(df, target='Churn')
```

```
<pandas.io.formats.style.Styler at 0x148170f90>
```

[40]:
```
<pycaret.classification.oop.ClassificationExperiment at 0x1499b6b90>
```

[41]:
```
automl
```

[41]:
```
<pycaret.classification.oop.ClassificationExperiment at 0x1499b6b90>
```

1.3 Find the best model

[42]:
```
#Using pycaret to identify which model performs best on the data

best_model = automl.compare_models()
```

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

<pandas.io.formats.style.Styler at 0x145ff7110>

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

The comparison here shows that if I deemed Accuracy the best metric to deterine fit then Logistic Regression would be the best model to use, yielding a 79% accuracy rate.

[43]:
```
# Choose a metric you think is best to use for finding the best model; by␣
↪default, it is accuracy but it could be AUC, precision, recall, etc.

best_model = automl.compare_models(sort='Precision')
```

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

<pandas.io.formats.style.Styler at 0x146291310>
```

<IPython.core.display.HTML object>

I sorted the list by Precision to find that a Ridge Classifer is the best model if precision was the primary focus.

[44]:
```
# Clearing sort and setting back to default
best_model = automl.compare_models()
```

<IPython.core.display.HTML object>

<pandas.io.formats.style.Styler at 0x14b85b990>

<IPython.core.display.HTML object>

[45]:
```
# Identify the best model for the data set
best_model
```

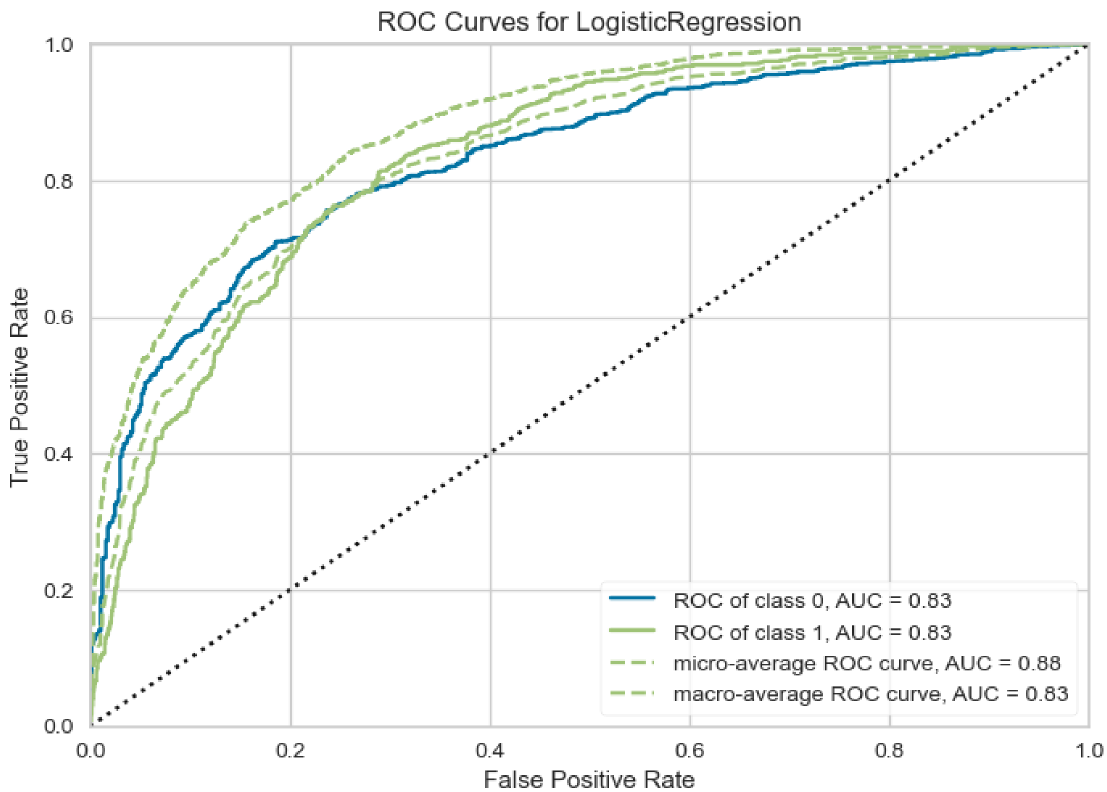
[45]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=1000,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=3546, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

[46]:
```
# Showcase various plots for the best model
automl.evaluate_model(best_model)
```

```
interactive(children=(ToggleButtons(description='Plot Type:', icons=('',),␣
↪options=(('Pipeline Plot', 'pipelin…
```

[47]:
```
automl.plot_model(best_model)
```

<IPython.core.display.HTML object>



1.4 Test it

[48]:
```
automl.predict_model(best_model, df.iloc[-2:-1])
```

```
<pandas.io.formats.style.Styler at 0x14c878390>
```

[48]:
```
            tenure  PhoneService  Contract  PaymentMethod  MonthlyCharges  \
customerID
8361-LTMKD       4             1         0              1       74.400002

            TotalCharges  Churn  prediction_label  prediction_score
customerID
8361-LTMKD    306.600006      1                 1            0.5742
```

1.5 Save the model to disk

[49]:
```
automl.save_model(best_model, 'pycaret model')
```

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

[49]:
```
(Pipeline(memory=Memory(location=None),
          steps=[('numerical_imputer',
                  TransformerWrapper(exclude=None,
```

```
                                      include=['tenure', 'PhoneService',
                                               'Contract', 'PaymentMethod',
                                               'MonthlyCharges', 'TotalCharges'],
transformer=SimpleImputer(add_indicator=False,
                                                                  copy=True,
                                                                  fill_value=None,
keep_empty_features=False,
missing_values=nan,
                                                                  strategy='mean',
verbose='deprecated'))),
                 ('…
keep_empty_features=False,
missing_values=nan,
strategy='most_frequent',
verbose='deprecated'))),
                 ('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=3546,
                                     solver='lbfgs', tol=0.0001, verbose=0,
                                     warm_start=False))],
          verbose=False),
 'pycaret model.pkl')
```

1.6 Test Load

[52]:
```python
new_pycaret = ClassificationExperiment()
loaded_model = new_pycaret.load_model('pycaret model')
```

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

[53]:
```python
new_data = df.iloc[-2:-1]
```

[54]:
```python
new_pycaret.predict_model(loaded_model, new_data)
```

[54]:
```
            tenure  PhoneService  Contract  PaymentMethod  MonthlyCharges  \
customerID
8361-LTMKD       4             1         0              1       74.400002

            TotalCharges  Churn  prediction_label  prediction_score
customerID
8361-LTMKD    306.600006      1                 1            0.5742
```

2 Churn Probability Python Script

2.1 Load data for evaluation

[63]:
```
from IPython.display import Code

Code('predict_churn.py')
```

[63]:
```
#!/usr/bin/env python
# coding: utf-8

# Python Script

# In[ ]:


import pandas as pd
from pycaret.classification import ClassificationExperiment

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


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


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

Make predications on the entire dataset

[64]:
```
%run predict_churn.py
```

```
Transformation Pipeline and Model Successfully Loaded
predictions:
```

```
            tenure  PhoneService  Contract  PaymentMethod  MonthlyCharges  \
customerID
9305-CKSKC      22             1         0              2       97.400002
1452-KNGVK       8             0         1              1       77.300003
6723-OKKJM      28             1         0              0       28.250000
7832-POPKP      62             1         0              2      101.699997
6348-TACGU      10             0         0              1       51.150002

            TotalCharges  charge_per_tenure  prediction_label  \
customerID
9305-CKSKC    811.700012          36.895454                 0
1452-KNGVK   1701.949951         212.743744                 1
6723-OKKJM    250.899994           8.960714                 0
7832-POPKP   3106.560059          50.105808                 0
6348-TACGU   3440.969971         344.096985                 1

            prediction_score
customerID
9305-CKSKC            0.5568
1452-KNGVK            0.5225
6723-OKKJM            0.8729
7832-POPKP            0.8525
6348-TACGU            0.7091
```

3 Summary

I begin by installing pycaret within my environment (set up a virtual environment as well) and loaded my cleaned dataset from an earlier week. I set up the autoML and ran it to determine the best model, Logistic Regression. I selected a new metric for evaluation and then set the comparison back to its default before identifying the best model. Before saving my pycaret model, I made sure to test it to ensure it worked properly. Once saved, I ran a test load to double check my work. Lastly, I created a script to read a dataset (new_churn_data) and make a prediction on the entire file.