Week_5_assignment_starter

March 2, 2024

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]:
```
#Import all the required modules

import pandas as pd
from pycaret.classification import *
import pickle
```

[2]:
```
# Import the churn dataset

df = pd.read_csv('churn_data.csv')
df
```

[2]:
```
   customerID  tenure PhoneService        Contract  \
0  7590-VHVEG       1           No  Month-to-month
1  5575-GNVDE      34          Yes        One year
```

```
2     3668-QPYBK       2           Yes  Month-to-month
3     7795-CFOCW      45            No        One year
4     9237-HQITU       2           Yes  Month-to-month
…           …     …          …             …
7038  6840-RESVB      24           Yes        One year
7039  2234-XADUH      72           Yes        One year
7040  4801-JZAZL      11            No  Month-to-month
7041  8361-LTMKD       4           Yes  Month-to-month
7042  3186-AJIEK      66           Yes        Two year

                  PaymentMethod  MonthlyCharges  TotalCharges Churn
0              Electronic check           29.85         29.85    No
1                  Mailed check           56.95       1889.50    No
2                  Mailed check           53.85        108.15   Yes
3     Bank transfer (automatic)           42.30       1840.75    No
4              Electronic check           70.70        151.65   Yes
…                           …             …           …   …
7038               Mailed check           84.80       1990.50    No
7039    Credit card (automatic)          103.20       7362.90    No
7040           Electronic check           29.60        346.45    No
7041               Mailed check           74.40        306.60   Yes
7042  Bank transfer (automatic)          105.65       6844.50    No

[7043 rows x 8 columns]
```

[3]:
```python
# drop the records with null values

df.dropna(inplace=True)
df
```

[3]:
```
      customerID  tenure PhoneService        Contract  \
0     7590-VHVEG       1            No  Month-to-month
1     5575-GNVDE      34           Yes        One year
2     3668-QPYBK       2           Yes  Month-to-month
3     7795-CFOCW      45            No        One year
4     9237-HQITU       2           Yes  Month-to-month
…           …     …          …             …
7038  6840-RESVB      24           Yes        One year
7039  2234-XADUH      72           Yes        One year
7040  4801-JZAZL      11            No  Month-to-month
7041  8361-LTMKD       4           Yes  Month-to-month
7042  3186-AJIEK      66           Yes        Two year

                  PaymentMethod  MonthlyCharges  TotalCharges Churn
0              Electronic check           29.85         29.85    No
1                  Mailed check           56.95       1889.50    No
2                  Mailed check           53.85        108.15   Yes
```

```
3     Bank transfer (automatic)          42.30       1840.75    No
4              Electronic check          70.70        151.65   Yes
…                           …              …            …    …
7038                Mailed check          84.80       1990.50    No
7039     Credit card (automatic)         103.20       7362.90    No
7040           Electronic check          29.60        346.45    No
7041                Mailed check          74.40        306.60   Yes
7042  Bank transfer (automatic)         105.65       6844.50    No

[7032 rows x 8 columns]
```

[4]:
```python
# drop the customerID and PhoneService columns
df.drop(columns=['customerID', 'PhoneService'], inplace = True)
df
```

[4]:
```
      tenure        Contract              PaymentMethod  MonthlyCharges  \
0          1  Month-to-month           Electronic check           29.85
1         34        One year               Mailed check           56.95
2          2  Month-to-month               Mailed check           53.85
3         45        One year  Bank transfer (automatic)           42.30
4          2  Month-to-month           Electronic check           70.70
…        …             …                          …             …
7038      24        One year               Mailed check           84.80
7039      72        One year    Credit card (automatic)          103.20
7040      11  Month-to-month           Electronic check           29.60
7041       4  Month-to-month               Mailed check           74.40
7042      66        Two year  Bank transfer (automatic)          105.65

      TotalCharges Churn
0            29.85    No
1          1889.50    No
2           108.15   Yes
3          1840.75    No
4           151.65   Yes
…              …   …
7038       1990.50    No
7039       7362.90    No
7040        346.45    No
7041        306.60   Yes
7042       6844.50    No

[7032 rows x 6 columns]
```

[5]:
```python
df
```

[5]:
```
      tenure        Contract              PaymentMethod  MonthlyCharges  \
0          1  Month-to-month           Electronic check           29.85
```

```
1         34        One year              Mailed check           56.95
2          2  Month-to-month              Mailed check           53.85
3         45        One year  Bank transfer (automatic)          42.30
4          2  Month-to-month          Electronic check           70.70
…        …             …                         …              …
7038      24        One year              Mailed check           84.80
7039      72        One year    Credit card (automatic)         103.20
7040      11  Month-to-month          Electronic check           29.60
7041       4  Month-to-month              Mailed check           74.40
7042      66        Two year  Bank transfer (automatic)         105.65

      TotalCharges Churn
0            29.85    No
1          1889.50    No
2           108.15   Yes
3          1840.75    No
4           151.65   Yes
…              …   …
7038       1990.50    No
7039       7362.90    No
7040        346.45    No
7041        306.60   Yes
7042       6844.50    No

[7032 rows x 6 columns]
```

[6]:

```
# Set up the PyCaret environment
clf_setup = setup(data=df, target='Churn',session_id = 2209)
```

```
<pandas.io.formats.style.Styler at 0x22f85a1d460>
```

[7]:

```
# Compare models and choose the best one
best_model = compare_models()
```

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

```
<pandas.io.formats.style.Styler at 0x22f85c35ee0>
```

```
Processing:   0%|          | 0/61 [00:00<?, ?it/s]
```

The ada boost classifier model was the best model with an accuracy of 0.7850 and recall of 0.7850. It was the second best when ranked on F1 and Kappa metrics too. This makes it undoubtedly the best model for the churn percentage prediction.

[8]:

```
# save the model as ML_model.pickle
with open('ML_model.pickle', 'wb') as ml_file:
    pickle.dump(best_model, ml_file)
```

[9]:
```
# Load in the new_churn_data.csv file that contains the data to be predicted.
new_churn_data = pd.read_csv('new_churn_data.csv')

# Remove the 'customerID' and 'PhoneService' columns
new_churn_data.drop(columns=['customerID','PhoneService'], inplace = True)
new_churn_data
```

[9]:
```
   tenure        Contract            PaymentMethod  MonthlyCharges  \
0      22  Month-to-month         Electronic check           97.40
1       8        One year             Mailed check           77.30
2      28  Month-to-month  Credit card (automatic)           28.25
3      62  Month-to-month         Electronic check          101.70
4      10        Two year  Credit card (automatic)           51.15

   TotalCharges
0        811.70
1       1701.95
2        250.90
3       3106.56
4       3440.97
```

[10]:
```
# Import the created probability_of_churn function  data from the function.py␣
 ↪file
from function import probability_of_churn

# Use the function to get the prediction
probability_of_churn(new_churn_data)
```

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

[10]:
```
[1, 0, 0, 1, 0]
```

2 Summary

I used pycaret to find the best ML model to predict the probability of churn. I saved the model and used the saved model to predict new data's probability of churn.

I began by importing all the modules I'll use in this project; pandas, pycaret and pikle to save the ML model. I read in the churn_data.csv that contained the data that will be used for training and testing the models. I removed all the records with null values, and removed the customerID and PhoneService columns since they have very little correlation with the churn probability. I then set up the PyCaret environment specifying the data, target which is the churn column and the session_id.

I compared the models using the compare_models() method and choose the best one. The ada boost classifier model was the best model with an accuracy of 0.7850 and recall of 0.7850. It was the second best when ranked on F1 and Kappa metrics too. This makes it undoubtedly the best model for the churn percentage prediction. I proceeded to use the pickle.dump() method to savethe model as a pickle file.

I created a new python file function.py with a python function probability_of_churn() that takes in a dataframe and uses the saved model to make predictions for the probability of churn. I imported the function and used it to make predictions of the records in new_churn_data.csv. The predictions made were accurate.

```
[ ]:
```