MSDS600_Week_5_Assignment_starter_Rafael_Fernandes

June 11, 2024

1 MSDS600 Week 5 Assignment Starter - Rafael Fernandes

1.1 Getting Ready

Loading the necessary libraries, dataset, and filters and checking if the data has any error.

[5]:
```
import pandas as pd
from pycaret.classification import setup, compare_models, predict_model,␣
 ↪save_model, load_model, ClassificationExperiment
from IPython.display import Code
```

[6]:
```
df = pd.read_csv('churn_data.csv', index_col='customerID')
df.head(10)
```

[6]:

customerID	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges	Churn
7590-VHVEG	1	No	Month-to-month	Electronic check	29.85	29.85	No
5575-GNVDE	34	Yes	One year	Mailed check	56.95	1889.50	No
3668-QPYBK	2	Yes	Month-to-month	Mailed check	53.85	108.15	Yes
7795-CFOCW	45	No	One year	Bank transfer (automatic)	42.30	1840.75	No
9237-HQITU	2	Yes	Month-to-month	Electronic check	70.70	151.65	Yes
9305-CDSKC	8	Yes	Month-to-month	Electronic check	99.65	820.50	Yes
1452-KIOVK	22	Yes	Month-to-month	Credit card (automatic)	89.10	1949.40	No
6713-OKOMC	10	No	Month-to-month	Mailed check	29.75	301.90	No
7892-POOKP	28	Yes	Month-to-month	Electronic check	104.80	3046.05	Yes
6388-TABGU	62	Yes	One year	Bank transfer (automatic)			

```
6388-TABGU           56.15       3487.95     No
```

To use pycaret I created a virtual environment and I called it 'pyca'.

[8]:
```
!jupyter kernelspec list
```

```
Available kernels:
  pyca       C:\Users\rafaf\AppData\Roaming\jupyter\kernels\pyca
  python3    C:\Users\rafaf\AppData\Roaming\jupyter\kernels\python3
```

```
0.01s - Debugger warning: It seems that frozen modules are being used, which may
0.00s - make the debugger miss breakpoints. Please pass -Xfrozen_modules=off
0.00s - to python to disable frozen modules.
0.00s - Note: Debugging will proceed. Set PYDEVD_DISABLE_FILE_VALIDATION=1 to
disable this validation.
```

[9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7043 entries, 7590-VHVEG to 3186-AJIEK
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   tenure          7043 non-null   int64
 1   PhoneService    7043 non-null   object
 2   Contract        7043 non-null   object
 3   PaymentMethod   7043 non-null   object
 4   MonthlyCharges  7043 non-null   float64
 5   TotalCharges    7032 non-null   float64
 6   Churn           7043 non-null   object
dtypes: float64(2), int64(1), object(4)
memory usage: 440.2+ KB
```

1.2 Automation

In this part I start the process for auto ML, setting it up, comparing the models and I'm sorting 'recall' as first model.

[11]:
```
automl = ClassificationExperiment()
```

[12]:
```
automl = setup(data=df, target='Churn')
```

```
<pandas.io.formats.style.Styler at 0x2234b5cbe50>
```

[13]:
```
best_model = compare_models(sort='recall')
```

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

```
<pandas.io.formats.style.Styler at 0x2234e186f10>
```

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

[14]:
```
automl
```

[14]:
```
<pycaret.classification.oop.ClassificationExperiment at 0x2234bef2410>
```

[15]:
```
best_model
```

[15]:
```
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=358, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

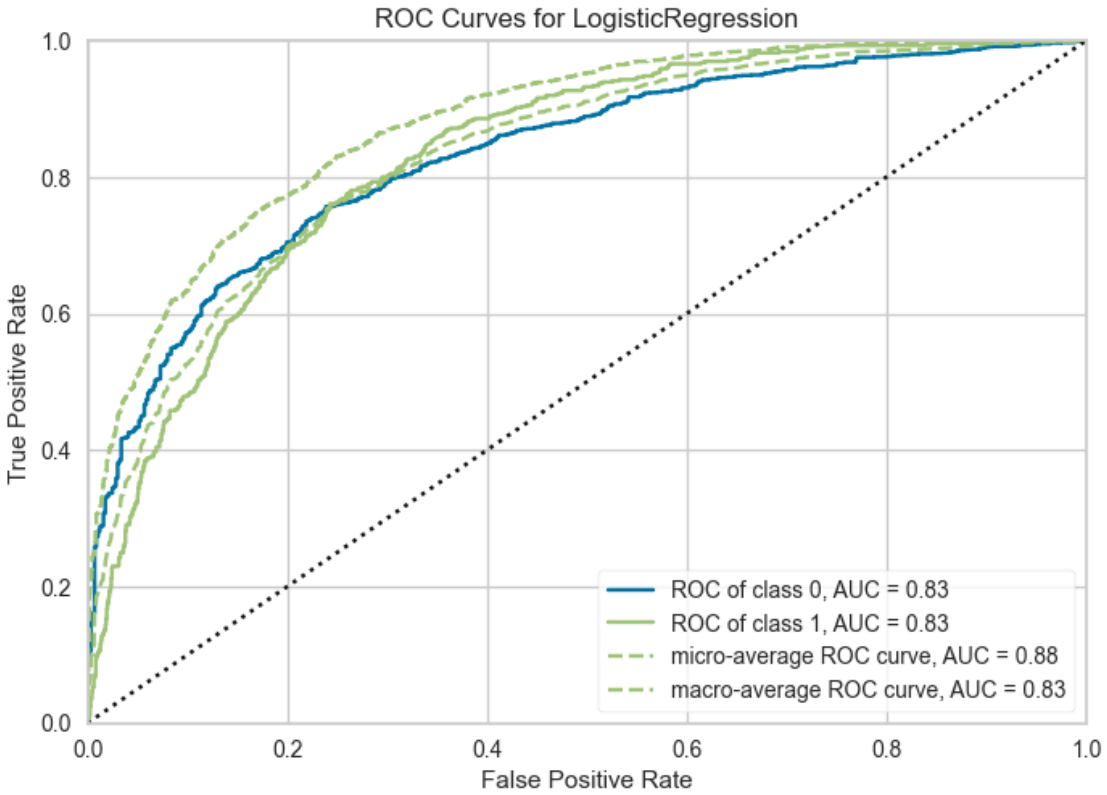
Above we can see that 'Logistic Regression' was the best model for the 'Recall' and it showed that 'Accuracy' model with the same result. Now I'm going to evaluate the model plotting the best model.

[17]:
```
automl.evaluate_model(best_model)
```

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

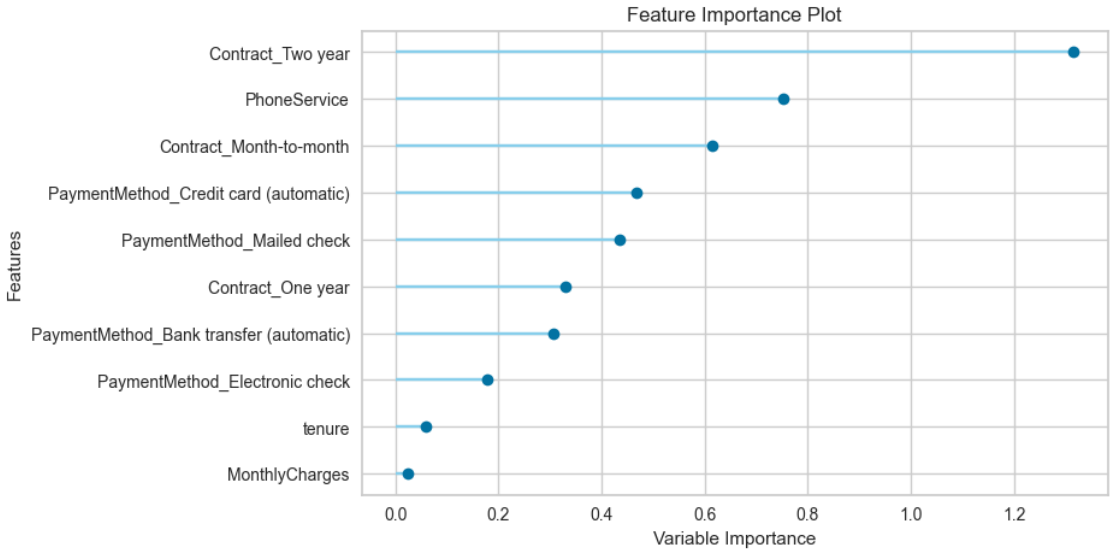
[18]:
```
automl.plot_model(best_model)
```

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



[19]:
```
automl.plot_model(best_model, plot = 'feature')
```

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



I'm going to create a new churn data to predict the best model.

[21]:
```
new_churn_data = df.iloc[-3:-2]
```

[22]:
```
predictions = predict_model(best_model, data=new_churn_data)
predictions
```

```
<pandas.io.formats.style.Styler at 0x2234e005810>
```

[22]:
```
            tenure PhoneService        Contract     PaymentMethod  \
customerID
4801-JZAZL      11           No  Month-to-month  Electronic check

            MonthlyCharges  TotalCharges Churn prediction_label  \
customerID
4801-JZAZL            29.6    346.450012    No               No

            prediction_score
customerID
4801-JZAZL            0.5371
```

1.3 Saving, testing and loading the model

In this part, I'm going to save the model in a pickle file, then I'm going to test, load, and predict the file.

[24]:
```
automl.save_model(best_model, 'pyca_data_model')
```

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

[24]:
```
(Pipeline(memory=Memory(location=None),
          steps=[('label_encoding',
                  TransformerWrapperWithInverse(exclude=None, include=None,
                                                transformer=LabelEncoder())),
                 ('numerical_imputer',
                  TransformerWrapper(exclude=None,
                                     include=['tenure', 'MonthlyCharges',
                                              'TotalCharges'],
transformer=SimpleImputer(add_indicator=False,
                                                               copy=True,
                                                               fill_value=None,
keep_empty_features=False,…
handle_missing='return_nan',
handle_unknown='value',
                                                               return_df=True,
use_cat_names=True,
                                                               verbose=0))),
                 ('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=358,
                                     solver='lbfgs', tol=0.0001, verbose=0,
                                     warm_start=False))],
          verbose=False),
 'pyca_data_model.pkl')
```

[25]:
```
pyca_model = ClassificationExperiment()
tested_model = pyca_model.load_model('pyca_data_model')
```

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

[26]:
```
new_pyca = ClassificationExperiment()
loaded_model = new_pyca.load_model('pyca_data_model')
```

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

[27]:
```
new_pyca.predict_model(loaded_model, df.iloc[-3:-2])
```

[27]:
```
            tenure PhoneService        Contract     PaymentMethod  \
customerID
4801-JZAZL      11           No  Month-to-month  Electronic check

            MonthlyCharges  TotalCharges Churn prediction_label  \
```

```
customerID
4801-JZAZL            29.6    346.450012    No               No

            prediction_score
customerID
4801-JZAZL            0.5371
```

1.4 Creating a Python module for predictions

[29]:
```
Code('predict_churn.py')
```

[29]:
```
import pandas as pd
from pycaret.classification import ClassificationExperiment

def load_data(filepath):
    "Load the churn_data.csv data into a DataFrame."

    df = pd.read_csv('churn_data.csv', index_col='customerID')
    return df


def make_predictions(df):
    "Use the best model (LogisticRegression) pycaret to make predictions"

    classifier = ClassificationExperiment()
    model = classifier.load_model('pyca_data_model')
    predictions = classifier.predict_model(model, data=df)
    predictions.rename({'Label': 'Churn'}, axis=1, inplace=True)
    predictions['Churn'].replace({1: 'Churn', 0: 'No churn'},
                                                 inplace=True)
    return predictions['Churn']


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

Lastly I'm running the file to test it and see the predictions.

[31]:
```
%run predict_churn.py
```

```
Transformation Pipeline and Model Successfully Loaded
predictions:
customerID
7590-VHVEG     No
5575-GNVDE     No
3668-QPYBK    Yes
```

```
7795-CFOCW     No
9237-HQITU    Yes
              …
6840-RESVB     No
2234-XADUH     No
4801-JZAZL     No
8361-LTMKD    Yes
3186-AJIEK     No
Name: Churn, Length: 7043, dtype: category
Categories (2, object): ['No', 'Yes']
```

```
<Figure size 800x550 with 0 Axes>
```

1.5 References

The following links are references used as resources to complete and improve this project.

A step-by-step guide to install PyCaret in Python

A Complete Guide to PyCaret!!!

Analysis and model explainability functions in PyCaret

joblib 1.4.2

FTE_Week_3 MSDS600 W3 FTE advanced section

1.6 Summary

I used the pycaret auto ML package to predict if customers are going to churn. I set 'recall' as the metric used for finding the best model and it showed 'Logistic Regression' as the best one, however, 'Accuracy' was the same, and both for all the models had the same result. I trained the model, I plotted the best model and the best model with 'feature'. After I estimated the predictions for the new DF, I saved the model to the disk as a pickle file, tested the functions with the new data, and printed the predictions.