MSDS600 Week 5 Assignment Starter - Rafael Fernandes

Getting Ready

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

In [4]:

```
import pandas as pd
from pycaret.classification import setup, compare_models, predict_model, save_model
from IPython.display import Code
```

In [5]:

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

Out[5]:

	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharg
customerID						
7590-VHVEG	1	No	Month-to-month	Electronic check	29.85	29.
5575-GNVDE	34	Yes	One year	Mailed check	56.95	1889.
3668-QPYBK	2	Yes	Month-to-month	Mailed check	53.85	108.
7795-CFOCW	45	No	One year	Bank transfer (automatic)	42.30	1840.
9237-HQITU	2	Yes	Month-to-month	Electronic check	70.70	151.
9305-CDSKC	8	Yes	Month-to-month	Electronic check	99.65	820.
1452-KIOVK	22	Yes	Month-to-month	Credit card (automatic)	89.10	1949.
6713-OKOMC	10	No	Month-to-month	Mailed check	29.75	301.
7892-POOKP	28	Yes	Month-to-month	Electronic check	104.80	3046.
6388-TABGU	62	Yes	One year	Bank transfer (automatic)	56.15	3487.

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

In [7]:

```
!jupyter kernelspec list
```

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

```
0.00s - Debugger warning: It seems that frozen modules are being used, which may
0.01s - 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 disabl
e this validation.
```

In [8]:

```
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
```

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.

```
In [10]: automl = ClassificationExperiment()
```

```
In [11]: automl = setup(data=df, target='Churn')
```

	Description	Value
0	Session id	5151
1	Target	Churn
2	Target type	Binary
3	Target mapping	No: 0, Yes: 1
4	Original data shape	(7043, 7)
5	Transformed data shape	(7043, 12)
6	Transformed train set shape	(4930, 12)
7	Transformed test set shape	(2113, 12)
8	Numeric features	3
9	Categorical features	3
10	Rows with missing values	0.2%
11	Preprocess	True
12	Imputation type	simple
13	Numeric imputation	mean
14	Categorical imputation	mode
15	Maximum one-hot encoding	25
16	Encoding method	None
17	Fold Generator	StratifiedKFold
18	Fold Number	10
19	CPU Jobs	-1
20	Use GPU	False
21	Log Experiment	False
22	Experiment Name	clf-default-name
23	USI	2002

In [12]:

```
best_model = compare_models(sort='recall')
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	0.7903	0.8320	0.7903	0.7806	0.7828	0.4279	0.4326	1.1690
gbc	Gradient Boosting Classifier	0.7880	0.8341	0.7880	0.7760	0.7767	0.4068	0.4159	0.1930
lightgbm	Light Gradient Boosting Machine	0.7880	0.8233	0.7880	0.7779	0.7801	0.4201	0.4251	0.1760
ridge	Ridge Classifier	0.7878	0.8221	0.7878	0.7741	0.7746	0.3986	0.4094	0.0420
ada	Ada Boost Classifier	0.7868	0.8336	0.7868	0.7759	0.7777	0.4122	0.4186	0.0970
lda	Linear Discriminant Analysis	0.7856	0.8221	0.7856	0.7761	0.7787	0.4184	0.4220	0.0460
svm	SVM - Linear Kernel	0.7708	0.7426	0.7708	0.7591	0.7527	0.3431	0.3613	0.0490
rf	Random Forest Classifier	0.7602	0.7941	0.7602	0.7494	0.7528	0.3508	0.3539	0.1810
knn	K Neighbors Classifier	0.7560	0.7468	0.7560	0.7413	0.7455	0.3268	0.3316	0.0740
et	Extra Trees Classifier	0.7491	0.7633	0.7491	0.7383	0.7423	0.3244	0.3265	0.2000
dummy	Dummy Classifier	0.7347	0.5000	0.7347	0.5398	0.6223	0.0000	0.0000	0.0580
dt	Decision Tree Classifier	0.7260	0.6530	0.7260	0.7264	0.7260	0.2976	0.2979	0.0480
nb	Naive Bayes	0.6801	0.8065	0.6801	0.7881	0.6987	0.3580	0.4043	0.0530
qda	Quadratic Discriminant Analysis	0.6408	0.6674	0.6408	0.6833	0.6164	0.1475	0.1640	0.0410

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

```
In [13]: automl
```

```
Out[13]: <pycaret.classification.oop.ClassificationExperiment at 0x29f53b73210>
```

```
In [14]: best_model
```

Out[14]:

```
LogisticRegression
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
                   intercept_scaling=1, l1_ratio=None, max_iter=1000,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=5151, 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, but AUC had the highest number with 0.8320. Now I'm going to evaluate the model plotting the best model.

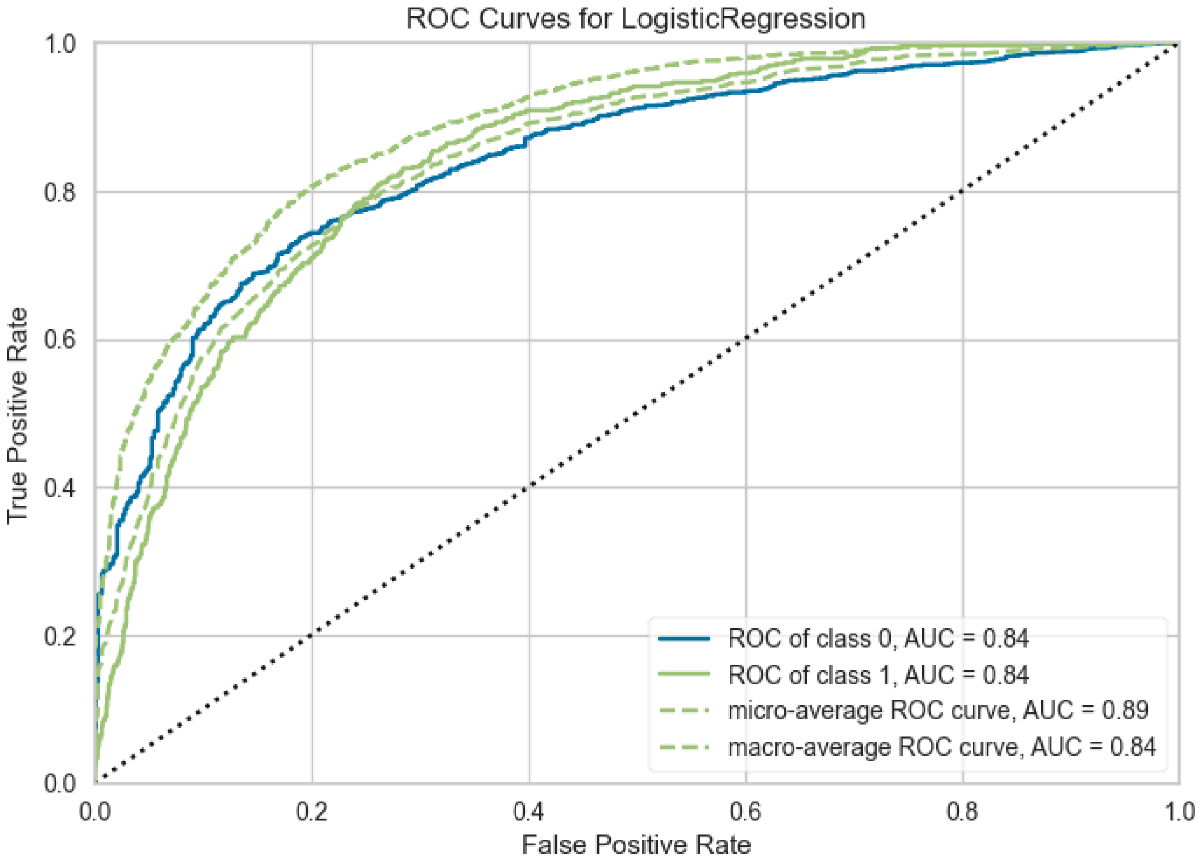
In [16]:

```
automl.evaluate_model(best_model)
```

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

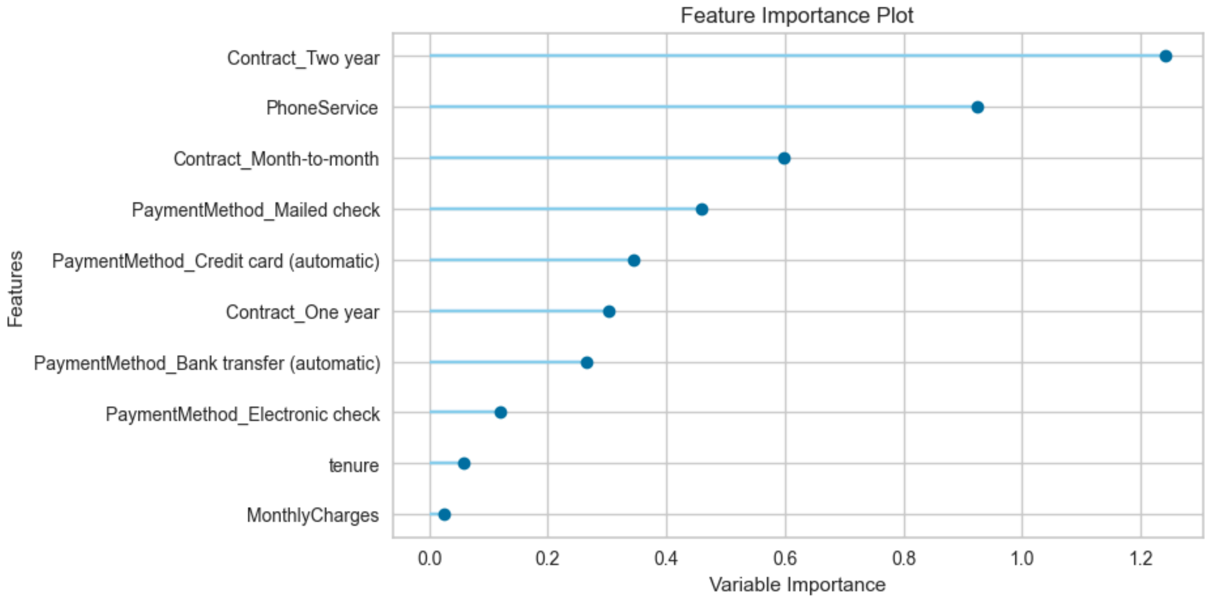
In [17]:

```
automl.plot_model(best_model)
```



In [18]:

```
automl.plot_model(best_model, plot = 'feature')
```



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

```
In [20]: new_churn_data = df.iloc[-3:-2]
```

```
In [21]: predictions = predict_model(best_model, data=new_churn_data)
         predictions
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Logistic Regression	1.0000	0	1.0000	1.0000	1.0000	nan	0.0000

Out[21]:

	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharg
customerID						
4801-JZAZL	11	No	Month-to-month	Electronic check	29.6	346.4500

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.

```
In [23]: automl.save_model(best_model, 'pyca_data_model')
```

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

Out[23]:

```
(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=Fals
e,
                                                               copy=True,
                                                               fill_value=None,
                                                               keep_empty_features
=False,...
                                                               handle_missing='ret
urn_nan',
                                                               handle_unknown='val
ue',
                                                               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=5151,
                                     solver='lbfgs', tol=0.0001, verbose=0,
                                     warm_start=False))],
          verbose=False),
 'pyca_data_model.pkl')
```

In [24]:

```python
pyca_model = ClassificationExperiment()
tested_model = pyca_model.load_model('pyca_data_model')
```

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

In [25]:

```python
new_pyca = ClassificationExperiment()
loaded_model = new_pyca.load_model('pyca_data_model')
```

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

In [26]:

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

Out[26]:

	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharg
customerID						
4801-JZAZL	11	No	Month-to-month	Electronic check	29.6	346.4500

Creating a Python module for predictions

```
In [28]: Code('predict_churn.py')
```

```
Out[28]:
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.

```
In [30]: %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>
```

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

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 Logistic Regression had the best 5 results from 7 comparisons.