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.

In [128]:
```
## import modules and data frame
import pandas as pd
from pycaret.classification import setup, compare_models, predict_model, save_model, l
from IPython.display import Code
import pickle

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

Out[128]:

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
...	...	...	...	...	...	...	...
6840-RESVB	24	Yes	One year	Mailed check	84.80	1990.50	No
2234-XADUH	72	Yes	One year	Credit card (automatic)	103.20	7362.90	No
4801-JZAZL	11	No	Month-to-month	Electronic check	29.60	346.45	No
8361-LTMKD	4	Yes	Month-to-month	Mailed check	74.40	306.60	Yes
3186-AJIEK	66	Yes	Two year	Bank transfer (automatic)	105.65	6844.50	No

7043 rows × 7 columns

In [129]:
```
## submit data set to auto ML
automl = setup(df, target='Churn')
```

	Description	Value
0	session_id	188
1	Target	Churn
2	Target Type	Binary
3	Label Encoded	No: 0, Yes: 1
4	Original Data	(7043, 7)
5	Missing Values	True
6	Numeric Features	3
7	Categorical Features	3
8	Ordinal Features	False
9	High Cardinality Features	False
10	High Cardinality Method	None
11	Transformed Train Set	(4930, 11)

In [130]:
```
## create best model
best_model = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
ada	Ada Boost Classifier	0.7923	0.8379	0.5103	0.6357	0.5657	0.4315	0.4362	0.0230
gbc	Gradient Boosting Classifier	0.7919	0.8382	0.5057	0.6365	0.5635	0.4292	0.4342	0.0510
ridge	Ridge Classifier	0.7917	0.0000	0.4744	0.6488	0.5468	0.4160	0.4253	0.0030
lr	Logistic Regression	0.7901	0.8326	0.5179	0.6264	0.5661	0.4295	0.4334	0.0100
lda	Linear Discriminant Analysis	0.7862	0.8256	0.5271	0.6151	0.5666	0.4260	0.4290	0.0040
lightgbm	Light Gradient Boosting Machine	0.7777	0.8259	0.5004	0.5980	0.5444	0.3990	0.4020	0.0800
rf	Random Forest Classifier	0.7702	0.8067	0.4912	0.5810	0.5320	0.3811	0.3837	0.0630
knn	K Neighbors Classifier	0.7645	0.7393	0.4377	0.5744	0.4964	0.3465	0.3522	0.0150
et	Extra Trees Classifier	0.7643	0.7845	0.4950	0.5655	0.5270	0.3712	0.3732	0.0580
svm	SVM - Linear Kernel	0.7475	0.0000	0.4553	0.5682	0.4787	0.3237	0.3437	0.0060
dt	Decision Tree Classifier	0.7256	0.6572	0.4912	0.4854	0.4873	0.3003	0.3009	0.0090
nb	Naive Bayes	0.6848	0.8107	0.8472	0.4506	0.5882	0.3696	0.4183	0.0030
qda	Quadratic Discriminant Analysis	0.5132	0.5030	0.4809	0.2319	0.2932	0.0041	0.0071	0.0040

In [131]:
```
## Save best model to disk
save_model(best_model, 'gbc')
```

...

In [132]:
```
## save best model as LDA_model.pk
with open('gbc_model.pk', 'wb') as f:
    pickle.dump(best_model, f)
```

In [133]:
```
## open best model as "loaded_model"
with open('gbc_model.pk', 'rb') as f:
    loaded_model = pickle.load(f)
```

In [ ]:
```
```

In [134]:
```
loaded_lda = load_model('gbc')
```

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

In [135]:
```
predict_model(loaded_lda, new_data)
```

Out[135]:

customerID	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges	charge_per_t
8361-LTMKD	4	1	0.083	1	74.4	306.6	

In [140]:
```
Code('predict_churn.py')
```

Out[140]:
```
import pandas as pd
from pycaret.classification import predict_model, load_model

def load_data(filepath):
    """
    Loads 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.
    """
    model = load_model('gbc')
    predictions = predict_model(model, data=df)
    predictions.rename({'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_unmodified.csv')
    predictions = make_predictions(df)
    print('predictions:')
    print(predictions)
```

In [146]:
```
%run predict_churn.py
```

```
Please enter the filepath for analysis
C:\Users\adamg\Documents\MSDS 600\Week 5\new_churn_data_unmodified.csv
Transformation Pipeline and Model Successfully Loaded
predictions:
customerID
9305-CKSKC    Yes
1452-KNGVK     No
6723-OKKJM     No
7832-POPKP    Yes
6348-TACGU     No
Name: Churn_prediction, dtype: object
```

Summary

The original preparation and development of a prediction model for churn took an extensive amount of manual coding and action to prepare. The process can be improved upon by implementing autoML tools that will do much of this work on their own. To find the best possible model and hyperparameters for the churn data set, the data was run through Pycaret's autoML tool. This tool was instructed to find the best learning model to predict for churn.

Pycaret determined an Ada Boost Classifier was the most accurate model based on a predictive accuracy score of 79%. This model is not significantly outperformed by the gradient boost classifier which has the highest AUC. Since the AUC scores are so close together (delta of 0.0003%) there is not a significant advantage to using one over the other. The Ada Boost Classier has been selected for our model based on it having the highest accuracy. The Ada Boost Classifier was then saved to a pickle file to allow it to be called by a python script later in the code.

The data we used for the Ada Boost model is the original unmodified data set provided for churn. The program encountered errors when attempting to predict for churn when using my custom prepared data set from week 2. The model expected to see columns in the test file that contained custom calculations different from the ones I had added to the prepared data. To correct the issue, the autoML process was rerun top to bottom using the original unmodified data in both places. This ultimately produced a model that could accurately predict the values for the test set.

Essential preparation steps like converting categorical fields to numerical ones were automatically performed on the original data by the Pycaret autoML process. This reduced the overall prep time required to get the data into a ML usable format to several seconds. The program only needed confirmation that it had correctly identified which columns were categorical and numerical to begin preprocessing the data.

The autoML process was successfully implemented here on the churn data set to produce a useable model in significantly less time. The manual model I tried to fine tweak over the last 4 weeks was also predicting churn with 79% accuracy. This proves that you can use autoML to generate models that are just as good in a fraction of the time. The new autoML generated Ada Boost model can be implemented to predict with prefect accuracy the 5 target churn answers in the new_churn_data test set above.