DS Automation Assignment

Using our prepared churn data from week 2:

- use TPOT 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.
  - REMEMBER: TPOT only finds the optimized processing pipeline and model. It doesn't create the model.
    - You can use `tpot.export('my_model_name.py')` (assuming you called your TPOT object tpot) and it will save a Python template with an example of the optimized pipeline.
    - Use the template code saved from the `export()` function in your program.
- create a Python script/file/module using code from the exported template above that
  - create 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.

Find the best pipeline with TPOT

In [19]:
```
import pandas as pd

# import the usual packages
from tpot import TPOTClassifier
from sklearn.model_selection import train_test_split

# Any results you write to the current directory are saved as output.
import timeit

import warnings
warnings.filterwarnings("ignore")
```

Load data and inspect data

In [20]:
```
df = pd.read_csv('../../w2/clean_churn_data.csv',index_col='customerID')
df.head()
```

Out[20]:

customerID	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges	Ch
7590-VHVEG	1	0	0	0	29.85	29.85	
5575-GNVDE	34	1	1	1	56.95	1889.50	
3668-QPYBK	2	1	0	1	53.85	108.15	
7795-CFOCW	45	0	1	2	42.30	1840.75	
9237-HQITU	2	1	0	0	70.70	151.65	

Split target and features data

In [39]:
```
features = df.drop('Churn', axis=1)
targets = df['Churn']

x_train, x_test, y_train, y_test = train_test_split(features, targets, stratify=target
features
```

Out[39]:

customerID	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges
7590-VHVEG	1	0	0	0	29.85	29.85
5575-GNVDE	34	1	1	1	56.95	1889.50
3668-QPYBK	2	1	0	1	53.85	108.15
7795-CFOCW	45	0	1	2	42.30	1840.75
9237-HQITU	2	1	0	0	70.70	151.65
...	...	...	...	...	...	...
6840-RESVB	24	1	1	1	84.80	1990.50
2234-XADUH	72	1	1	3	103.20	7362.90
4801-JZAZL	11	0	0	0	29.60	346.45
8361-LTMKD	4	1	0	1	74.40	306.60
3186-AJIEK	66	1	2	2	105.65	6844.50

7043 rows × 6 columns

Utilize TPOTClassifier

In [46]:
```
%%time
tpot = TPOTClassifier(generations=5, population_size=50, verbosity=2, n_jobs=-1, rando
tpot.fit(x_train, y_train)
print(tpot.score(x_test, y_test))
```

```
Generation 1 - Current best internal CV score: 0.7993157091253118

Generation 2 - Current best internal CV score: 0.7993157091253118

Generation 3 - Current best internal CV score: 0.800640929159141

Generation 4 - Current best internal CV score: 0.800640929159141

Generation 5 - Current best internal CV score: 0.800640929159141

Best pipeline: XGBClassifier(input_matrix, learning_rate=0.01, max_depth=6, min_child
_weight=14, n_estimators=100, n_jobs=1, subsample=0.15000000000000002, verbosity=0)
0.7938671209540034
CPU times: total: 1min 39s
Wall time: 10min 52s
```

Best pipeline is XGBClassifier based on the recommendation of TPOTClassifier. We can export this to be a python file for later use.

In [13]:
```
tpot.export("tpot_template.py")
```

Predict new data with the recommended pipeline

In [15]:
```
from xgboost import XGBClassifier
```

In [50]:
```
# exported_pipeline = XGBClassifier(learning_rate=0.01, max_depth=6, min_child_weight=
# exported_pipeline = XGBClassifier(learning_rate=0.01, max_depth=6, min_child_weight=
exported_pipeline = XGBClassifier(learning_rate=0.01, max_depth=6, min_child_weight=14
```

In [53]:
```
df2 = pd.read_csv('../new_churn_data.csv',index_col='customerID')
new_features = df2.drop('charge_per_tenure', axis=1)

exported_pipeline.fit(x_train, y_train)
exported_pipeline.predict(new_features)
```

Out[53]:
```
array([1, 0, 0, 0, 0])
```

```
We used the recommended pipeline to predict the new_churn_data.csv and get 1, 0, 0, 0,
0 as results.
```

Summary

This week, we started from the clean churn data file prepared in week 2. We use TPOTClassifier to fit the train data, and it is recommended by TPOT to use XGBClassifier, which has average CV score on the training set 0.800640929159141.

To be able to create our own pipeline, we then exported the model to a Python file, which enables us to just run the file and prints the prediction for the new_churn_data.csv file. We correctly predicted 4 out of 5 results.

Currently, the file path is hard-coded, but in the future, it can be expanded to accept file path to accept where training data and predicting data are stored.