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 [1]:
```
import pandas as pd
from pandas_profiling import ProfileReport
import matplotlib.pyplot as plt
%matplotlib inline
import phik
import seaborn as sns
import numpy as np
```

In [2]:
```
from pycaret.classification import setup, tune_model, compare_models, predict_moc
```

In [3]:
```python
df = pd.read_csv('prepped_churn_data.csv', index_col='customerID')
df
```

7590-VHVEG	1	0	0	2	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	3	42.30	1840.75
9237-HQITU	2	1	0	2	70.70	151.65
...	...	...	...	...	...	...
6840-RESVB	24	1	1	1	84.80	1990.50
2234-XADUH	72	1	1	0	103.20	7362.90

In [4]:
```python
automl = setup(df, target='Churn', numeric_features=['PhoneService','PaymentMetho
```

3	Label Encoded	0: 0, 1: 1
4	Original Data	(7032, 7)
5	Missing Values	False
6	Numeric Features	6
7	Categorical Features	0
8	Ordinal Features	False
9	High Cardinality Features	False
10	High Cardinality Method	None
11	Transformed Train Set	(4922, 6)
12	Transformed Test Set	(2110, 6)
13	Shuffle Train-Test	True
14	Stratify Train-Test	False
15	Fold Generator	StratifiedKFold

In [5]:
```python
automl[6]
```

Out[5]: 10

In [6]:
```
best_model = compare_models(sort='AUC')
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
gbc	Gradient Boosting Classifier	0.7944	0.8330	0.4985	0.6473	0.5625	0.4312	0.4379	0.1790
catboost	CatBoost Classifier	0.7952	0.8322	0.5092	0.6457	0.5691	0.4373	0.4428	1.5780
ada	Ada Boost Classifier	0.7940	0.8295	0.4970	0.6460	0.5612	0.4298	0.4364	0.0950
lightgbm	Light Gradient Boosting Machine	0.7956	0.8241	0.5245	0.6406	0.5759	0.4433	0.4476	0.0950
lr	Logistic Regression	0.7950	0.8239	0.5069	0.6470	0.5679	0.4362	0.4421	0.5590
qda	Quadratic Discriminant Analysis	0.7489	0.8174	0.7240	0.5200	0.6050	0.4282	0.4411	0.0100
lda	Linear Discriminant Analysis	0.7865	0.8162	0.4993	0.6239	0.5540	0.4160	0.4209	0.0090
xgboost	Extreme Gradient Boosting	0.7800	0.8122	0.5016	0.6048	0.5478	0.4042	0.4076	0.4030
rf	Random Forest Classifier	0.7818	0.8031	0.5084	0.6065	0.5526	0.4100	0.4131	0.2360
nb	Naive Bayes	0.7544	0.7851	0.6537	0.5309	0.5856	0.4138	0.4186	0.0090
et	Extra Trees Classifier	0.7617	0.7759	0.4924	0.5574	0.5222	0.3646	0.3661	0.1970
knn	K Neighbors Classifier	0.7631	0.7497	0.4557	0.5698	0.5056	0.3524	0.3568	0.0460
dt	Decision Tree Classifier	0.7330	0.6656	0.5115	0.4973	0.5038	0.3214	0.3218	0.0140
svm	SVM - Linear Kernel	0.6809	0.0000	0.6460	0.5191	0.5230	0.3189	0.3597	0.0200
ridge	Ridge Classifier	0.7930	0.0000	0.4656	0.6556	0.5439	0.4152	0.4256	0.0090

In [7]:
```
best_model
```

Out[7]:
```
GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='deviance', max_depth=3,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=100,
                           n_iter_no_change=None, presort='deprecated',
                           random_state=42, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)
```

In [8]:
```
tuned_best_model = tune_model(best_model)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8032	0.8406	0.5115	0.6700	0.5801	0.4546	0.4617
1	0.7972	0.8157	0.4580	0.6742	0.5455	0.4210	0.4340
2	0.7764	0.8290	0.4580	0.6061	0.5217	0.3795	0.3859
3	0.7967	0.8198	0.4580	0.6742	0.5455	0.4206	0.4337
4	0.8008	0.8327	0.5115	0.6634	0.5776	0.4501	0.4566
5	0.8130	0.8324	0.4885	0.7191	0.5818	0.4670	0.4815
6	0.7805	0.8269	0.3969	0.6420	0.4906	0.3604	0.3774
7	0.8130	0.8569	0.5191	0.7010	0.5965	0.4783	0.4874
8	0.8008	0.8424	0.4846	0.6702	0.5625	0.4378	0.4475
9	0.8049	0.8618	0.4846	0.6848	0.5676	0.4463	0.4574
Mean	0.7987	0.8358	0.4771	0.6705	0.5569	0.4316	0.4423
SD	0.0115	0.0141	0.0345	0.0292	0.0303	0.0355	0.0346

In [9]:
```
df.iloc[-2:-1]
```

Out[9]:

	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges	Chur
customerID							
8361-LTMKD	4	1	0	1	74.4	306.6	

In [10]:
```
predict_model(tuned_best_model, df.iloc[-2:-1])
```

Out[10]:

	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges	Chur
customerID							
8361-LTMKD	4	1	0	1	74.4	306.6	

In [11]:
```
save_model(tuned_best_model, 'gbc')
```

```
Transformation Pipeline and Model Succesfully Saved
```

Out[11]:
```
(Pipeline(memory=None,
          steps=[('dtypes',
                  DataTypes_Auto_infer(categorical_features=[],
                                       display_types=True, features_todrop=[],
                                       id_columns=[],
                                       ml_usecase='classification',
                                       numerical_features=['PhoneService',
                                                           'PaymentMethod',
                                                           'Contract'],
                                       target='Churn', time_features=[])),
                 ('imputer',
                  Simple_Imputer(categorical_strategy='not_available',
                                 fill_value_categorical=None...
                                            loss='deviance', max_depth=1,
                                            max_features=1.0,
                                            max_leaf_nodes=None,
                                            min_impurity_decrease=0.4,
                                            min_impurity_split=None,
                                            min_samples_leaf=2,
                                            min_samples_split=2,
                                            min_weight_fraction_leaf=0.0,
                                            n_estimators=270,
                                            n_iter_no_change=None,
                                            presort='deprecated',
                                            random_state=42, subsample=0.8,
                                            tol=0.0001, validation_fraction=0.
1,
                                            verbose=0, warm_start=False)]],
          verbose=False),
 'gbc.pkl')
```

In [12]:
```
import pickle

with open('gbc_model.pk', 'wb') as f:
    pickle.dump(best_model, f)
```

In [13]:
```
with open('gbc_model.pk', 'rb') as f:
    loaded_model = pickle.load(f)
```

In [14]:
```
new_data = df.iloc[-2:-1].copy()
new_data.drop('Churn', axis=1, inplace=True)
loaded_model.predict(new_data)
```

Out[14]:
```
array([1])
```

In [15]:
```
loaded_gbc = load_model('gbc')
```

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

In [16]:
```
predict_model(loaded_gbc, new_data)
```

Out[16]:

customerID	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges	Label
8361-LTMKD	4	1	0	1	74.4	306.6	1

In [17]:
```
from IPython.display import Code
Code('predict_churn.py')
```

Out[17]:
```
import pandas as pd
from pycaret.classification import predict_model, load_model
class PredictChurn:

    def load_data(filepath):
        """
        Loads churn 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 dataf
rame.
        """
        model = load_model('gbc')
        predictions = predict_model(model, data=df)
        predictions.rename({'Label': 'of Churn', 'Score': 'Probability'}, axis=
1, inplace=True)
        predictions['of Churn'].replace({1: 'Churn', 0: 'No Churn'},
                                        inplace=True)
        return predictions[{'of Churn', 'Probability'}]

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

In [18]:
```
%run predict_churn.py
print
```

```
Transformation Pipeline and Model Successfully Loaded
            of Churn  Probability
customerID
9305-CKSKC     Churn       0.5985
1452-KNGVK  No Churn       0.5580
6723-OKKJM  No Churn       0.9491
7832-POPKP  No Churn       0.7240
6348-TACGU  No Churn       0.7272
```

Out[18]:
```
<function print>
```

Summary

I began by loading the prepped data from week 2. I had to change the way I prepped the data because pycaret had too many n_features. I'm seeing how I can use sklearn param and predict proba features to enhance my ml algorithm. I didn't get a chance to test the more advanced autoML libraries like H20 and TPOT. I also didn't understand why the new_data was converting the contracts numerically so that only one-year contracts produced a 1. I would've liked to see the diffences in the predictions with month-to-month and two year contracts being quantified individually. Overall I optimized the best_model by sorting the data by "AUC" rather than Accuracy. I think the area under the curve better models the true predictions and minimizes false positives better when greater than the Accuracy. I tuned the hyperparameters using PyCaret and the accuracy improved to nearly 80%. This accuracy is much better than the randomforestclassifications from last week, though less intuitive when setting hyperparameters. I look forward to practicing with these libraries more in the future. I will play with GUI and python packaging as this will help me run the reports I am automating at work.