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]:

```
#use pycaret to find an ML algorithm that performs best on the data
##First, load the data
import pandas as pd
df = pd.read_csv('clean_churn_data.csv', index_col='customerID')#'Unnamed: 0')
df = df.drop(['Unnamed: 0', 'monthly_total_chg_ratio'], axis=1)
df
```

Out[1]:

customerID	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges	Churn
7590-VHVEG	1	0	12	1	29.85	29.85	0
5575-GNVDE	34	1	1	2	56.95	1889.50	0

customerID	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges	Churn
3668-QPYBK	2	1	12	2	53.85	108.15	1
7795-CFOCW	45	0	1	3	42.30	1840.75	0
9237-HQITU	2	1	12	1	70.70	151.65	1
...	...	...	...	...	...	...	...
6840-RESVB	24	1	1	2	84.80	1990.50	0
2234-XADUH	72	1	1	4	103.20	7362.90	0
4801-JZAZL	11	0	12	1	29.60	346.45	0
8361-LTMKD	4	1	12	2	74.40	306.60	1
3186-AJIEK	66	1	2	3	105.65	6844.50	0

7043 rows × 7 columns

In [2]:

```
from pycaret.classification import setup, compare_models, predict_model, save_model, load_model
automl = setup(df, target='Churn', numeric_features=['PhoneService', 'Contract', 'PaymentMethod'])
```

	Description	Value
0	session_id	2815
1	Target	Churn
2	Target Type	Binary
3	Label Encoded	None
4	Original Data	(7043, 7)
5	Missing Values	0
6	Numeric Features	6
7	Categorical Features	0
8	Ordinal Features	0
9	High Cardinality Features	0
10	High Cardinality Method	None
11	Transformed Train Set	(4930, 6)

	Description	Value
12	Transformed Test Set	(2113, 6)
13	Shuffle Train-Test	True
14	Stratify Train-Test	False
15	Fold Generator	StratifiedKFold
16	Fold Number	10
17	CPU Jobs	-1
18	Use GPU	0
19	Log Experiment	0
20	Experiment Name	clf-default-name
21	USI	6b3d
22	Imputation Type	simple
23	Iterative Imputation Iteration	None
24	Numeric Imputer	mean
25	Iterative Imputation Numeric Model	None
26	Categorical Imputer	constant
27	Iterative Imputation Categorical Model	None
28	Unknown Categoricals Handling	least_frequent
29	Normalize	0
30	Normalize Method	None
31	Transformation	0
32	Transformation Method	None
33	PCA	0
34	PCA Method	None
35	PCA Components	None
36	Ignore Low Variance	0
37	Combine Rare Levels	0
38	Rare Level Threshold	None
39	Numeric Binning	0

	Description	Value
40	Remove Outliers	0
41	Outliers Threshold	None
42	Remove Multicollinearity	0
43	Multicollinearity Threshold	None
44	Remove Perfect Collinearity	1
45	Clustering	0
46	Clustering Iteration	None
47	Polynomial Features	0
48	Polynomial Degree	None
49	Trignometry Features	0
50	Polynomial Threshold	None
51	Group Features	0
52	Feature Selection	0
53	Feature Selection Method	classic
54	Features Selection Threshold	None
55	Feature Interaction	0
56	Feature Ratio	0
57	Interaction Threshold	None
58	Fix Imbalance	0
59	Fix Imbalance Method	SMOTE

In [5]:

```
# index 0 seems to have the original data. let's check the documentation
# after checking the documentation, it's not clear what pycaret.classification.setup returns. it just says 'global variables'. If yo
automl[0]
```

Out[5]:

customerID	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges
1240-HCBOH	67.0	1.0	2.0	2.0	26.100000	1759.550049
2080-GKCWQ	2.0	1.0	12.0	1.0	74.949997	151.750000

	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges
customerID						
2654-VBVPB	1.0	1.0	12.0	3.0	19.900000	19.900000
4102-OQUPX	1.0	1.0	12.0	1.0	74.400002	74.400002
9074-KGVOX	50.0	0.0	12.0	4.0	39.450001	2021.349976
...	...	...	...	...	...	...
1755-RMCXH	2.0	1.0	12.0	2.0	20.299999	40.250000
5985-TBABQ	32.0	1.0	1.0	2.0	74.750000	2282.949951
0654-PQKDW	62.0	1.0	1.0	3.0	70.750000	4263.450195
1897-RCFUM	39.0	1.0	1.0	2.0	24.200001	914.599976
9244-ZVAPM	1.0	1.0	12.0	2.0	45.599998	45.599998

2113 rows × 6 columns

In [6]:

```
#now we campare models
##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, re
###I will choose the default accuracy, simply because I believe it's a good starting point
best_model = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
ada	Ada Boost Classifier	0.7976	0.8437	0.5266	0.6561	0.5832	0.4519	0.4573	0.0900
gbc	Gradient Boosting Classifier	0.7963	0.8426	0.5064	0.6600	0.5722	0.4419	0.4490	0.1810
lr	Logistic Regression	0.7941	0.8367	0.5372	0.6424	0.5844	0.4492	0.4528	4.5190
ridge	Ridge Classifier	0.7923	0.0000	0.4861	0.6565	0.5579	0.4262	0.4348	0.0080
lda	Linear Discriminant Analysis	0.7899	0.8300	0.5350	0.6321	0.5788	0.4402	0.4434	0.0140
lightgbm	Light Gradient Boosting Machine	0.7892	0.8275	0.5244	0.6323	0.5729	0.4347	0.4383	0.0480
svm	SVM - Linear Kernel	0.7728	0.0000	0.4320	0.6200	0.4995	0.3617	0.3756	0.0250
knn	K Neighbors Classifier	0.7694	0.7568	0.4635	0.5943	0.5195	0.3712	0.3768	0.0280
rf	Random Forest Classifier	0.7669	0.7995	0.4740	0.5848	0.5229	0.3711	0.3751	0.2080
et	Extra Trees Classifier	0.7568	0.7796	0.4763	0.5598	0.5138	0.3532	0.3558	0.1700
qda	Quadratic Discriminant Analysis	0.7556	0.8252	0.7378	0.5343	0.6195	0.4462	0.4592	0.0070

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
dummy	Dummy Classifier	0.7300	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0090
nb	Naive Bayes	0.7270	0.8062	0.7521	0.4966	0.5979	0.4042	0.4245	0.0090
dt	Decision Tree Classifier	0.7170	0.6504	0.4891	0.4764	0.4821	0.2877	0.2881	0.0130

In [7]:

```
best_model
```

Out[7]:

```
AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=1.0,
                   n_estimators=50, random_state=2815)
```

In [8]:

```
# Therefore, the best model available is the Ada Boost Classifier mode!
#save the model to disk
save_model(best_model, 'ABC_best')
```

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

Out[8]:

```
(Pipeline(memory=None,
          steps=[('dtypes',
                  DataTypes_Auto_infer(categorical_features=[],
                                       display_types=True, features_todrop=[],
                                       id_columns=[],
                                       ml_usecase='classification',
                                       numerical_features=['PhoneService',
                                                           'Contract',
                                                           'PaymentMethod'],
                                       target='Churn', time_features=[])),
                 ('imputer',
                  Simple_Imputer(categorical_strategy='not_available',
                                 fill_value_categorical=None...
                 ('dummy', Dummify(target='Churn')),
                 ('fix_perfect', Remove_100(target='Churn')),
                 ('clean_names', Clean_Colum_Names()),
                 ('feature_select', 'passthrough'), ('fix_multi', 'passthrough'),
                 ('dfs', 'passthrough'), ('pca', 'passthrough'),
                 ['trained_model',
                  AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,
                                     learning_rate=1.0, n_estimators=50,
                                     random_state=2815)]],
          verbose=False),
 'ABC_best.pkl')
```

In [9]:

```
#create a Python script/file/module with a function that takes a pandas dataframe as an input and returns the probability of churn j
##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
%run predict_churn.py
```

```
Please enter filename for predicting: new_churn_data.csv
Transformation Pipeline and Model Successfully Loaded
predictions:
customerID
9305-CKSKC    No churn
1452-KNGVK    No churn
6723-OKKJM    No churn
7832-POPKP    No churn
6348-TACGU    No churn
Name: Churn_prediction, dtype: object
```

In [ ]:

Summary

Write a short summary of the process and results here.

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

In this assignment:

- I used the pandas Python package to load in my clean_churn_data.csv from Week 3.
- Once the data was loaded, I dropped the unnamed column and the monthly_total_chg_ratio column as they are not present in the new_churn_data.csv data, causing an error.
- Using the pycaret Python package, I created a comparison of ML models based on the clean_churn_data, ensuring to set all columns as numeric since they have already been cleaned.
- Comparing the models accuracies, I chose to use the Ada Boost Classifier model because it was the most accurate.
- I saved this Ada Boost Classifier model to a pickle file.
- I then created a Python script based on the FTE example to prompt the user for a filename, and execute predictions on that file to predict churn vs. no-churn.
- These predictions are outputted to the users console.

Was fun to create some Python scripting!

Thank you, Jeremy

In [ ]: