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.

Load Data

In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv('prepped_churn_data.csv', index_col='customerID')
#df = pd.read_csv('new_churn_data.csv', index_col='customerID')
df
```

Out[2]:

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

7032 rows × 8 columns

AutoML with pycaret

In [3]:
```
from pycaret.classification import *
```

In [4]:
```
automl = setup(data = df, target = 'Churn',  fold_shuffle=True)
```

	Description	Value
0	session_id	8860
1	Target	Churn
2	Target Type	Binary
3	Label Encoded	0: 0, 1: 1
4	Original Data	(7032, 8)
5	Missing Values	False
6	Numeric Features	4
7	Categorical Features	3
8	Ordinal Features	False
9	High Cardinality Features	False
10	High Cardinality Method	None
11	Transformed Train Set	(4922, 12)
12	Transformed Test Set	(2110, 12)
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	False
19	Log Experiment	False
20	Experiment Name	clf-default-name
21	USI	a8f7
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	False
30	Normalize Method	None
31	Transformation	False
32	Transformation Method	None

	Description	Value
33	PCA	False
34	PCA Method	None
35	PCA Components	None
36	Ignore Low Variance	False
37	Combine Rare Levels	False
38	Rare Level Threshold	None
39	Numeric Binning	False
40	Remove Outliers	False
41	Outliers Threshold	None
42	Remove Multicollinearity	False
43	Multicollinearity Threshold	None
44	Clustering	False
45	Clustering Iteration	None
46	Polynomial Features	False
47	Polynomial Degree	None
48	Trignometry Features	False
49	Polynomial Threshold	None
50	Group Features	False
51	Feature Selection	False
52	Features Selection Threshold	None
53	Feature Interaction	False
54	Feature Ratio	False
55	Interaction Threshold	None
56	Fix Imbalance	False
57	Fix Imbalance Method	SMOTE

In [5]:
```
automl[6]
```
Out[5]:
```
-1
```

In [6]:
```
best_model = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lda	Linear Discriminant Analysis	0.7901	0.8325	0.5045	0.6445	0.5650	0.4295	0.4357	0.0030
lr	Logistic Regression	0.7891	0.8380	0.4940	0.6454	0.5589	0.4235	0.4306	0.2280
ridge	Ridge Classifier	0.7885	0.0000	0.4571	0.6584	0.5386	0.4072	0.4192	0.0030
gbc	Gradient Boosting Classifier	0.7859	0.8345	0.4910	0.6341	0.5529	0.4152	0.4214	0.0410
ada	Ada Boost Classifier	0.7816	0.8311	0.4789	0.6258	0.5420	0.4021	0.4086	0.0160
catboost	CatBoost Classifier	0.7814	0.8297	0.4940	0.6203	0.5496	0.4078	0.4126	0.2710
lightgbm	Light Gradient Boosting Machine	0.7735	0.8197	0.4947	0.5985	0.5413	0.3928	0.3962	0.0100
xgboost	Extreme Gradient Boosting	0.7733	0.8083	0.4857	0.6000	0.5366	0.3887	0.3927	0.0530
rf	Random Forest Classifier	0.7643	0.7853	0.4759	0.5784	0.5216	0.3674	0.3708	0.0350
knn	K Neighbors Classifier	0.7574	0.7336	0.4308	0.5664	0.4889	0.3340	0.3395	0.1250
et	Extra Trees Classifier	0.7475	0.7581	0.4677	0.5407	0.5005	0.3329	0.3351	0.0350
dt	Decision Tree Classifier	0.7170	0.6512	0.4925	0.4786	0.4849	0.2901	0.2904	0.0030
svm	SVM - Linear Kernel	0.7109	0.0000	0.4293	0.5646	0.4199	0.2574	0.2996	0.0040
nb	Naive Bayes	0.7064	0.8159	0.8075	0.4749	0.5979	0.3904	0.4248	0.0940
qda	Quadratic Discriminant Analysis	0.5645	0.5271	0.4459	0.2066	0.2616	0.0385	0.0525	0.0030

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

Out[7]:
```
LinearDiscriminantAnalysis(n_components=None, priors=None, shrinkage=None,
                           solver='svd', store_covariance=False, tol=0.0001)
```

In [8]:
```
df.iloc[-2:-1].shape
```

Out[8]:
```
(1, 8)
```

In [9]:
```
predict_model(best_model, df.iloc[-2:-1])
```

Out[9]:

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

Saving and Loading Model

In [10]:
```
save_model(best_model, 'LDA')
```

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

Out[10]:
```
(Pipeline(memory=None,
          steps=[('dtypes',
                  DataTypes_Auto_infer(categorical_features=[],
                                       display_types=True, features_todrop=[],
                                       id_columns=[],
                                       ml_usecase='classification',
                                       numerical_features=[], target='Churn',
                                       time_features=[])),
                 ('imputer',
                  Simple_Imputer(categorical_strategy='not_available',
                                 fill_value_categorical=None,
                                 fill_value_numerical=None,
                                 numeric_strate...
                 ('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',
                  LinearDiscriminantAnalysis(n_components=None, priors=None,
                                             shrinkage=None, solver='svd',
                                             store_covariance=False,
                                             tol=0.0001)]],
          verbose=False),
 'LDA.pkl')
```

In [11]:
```
import pickle

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

In [12]:
```
with open('LDA_model.pk', 'rb') as f:
    loaded_model = pickle.load(f)
```

In [13]:
```
new_data = df.iloc[-2:-1].copy()
#new_data.drop('MonthlyCharges_tenure_ratio', axis=1, inplace=True)
new_data['dummy1']=0
new_data['dummy2']=0
new_data['dummy3']=0
new_data['dummy4']=0

new_data
```

Out[13]:

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

In [14]:
```
loaded_model.predict(new_data)
```

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

In [15]:
```
loaded_lda = load_model('LDA')
```

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

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

Out[16]:

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

Python Predictions Module

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

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 datafr
ame.
    """
    model = load_model('LDA')
    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.csv')
    predictions = make_predictions(df)
    print('predictions:')
    print(predictions)
```

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

```
Transformation Pipeline and Model Successfully Loaded
predictions:
customerID
9305-CKSKC       Churn
1452-KNGVK       Churn
6723-OKKJM    No churn
7832-POPKP       Churn
6348-TACGU       Churn
Name: Churn_prediction, dtype: object
```

Summary

Using the pycaret module, we setup the autoML by setting the data to the preppared churn data and target the Churn catergory of the data set. By using the compare model function, the liner discriminant analysis model provides the best accuracy for the data set. Then a predict model function was used on the last row to make a 2D array. The save model function is used to save the trained model in a pickle file for later use. A test was carried out on the save file by opening the saved pickle with the pickle load function. Using the loaded data, the predict model function is used to make predictions on the new data using the preppared data set. The predictions shows that out of the 5 customers listed, 4 of the customers will churn.