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.

Loaded Data

In [179]:

```
import pandas as pd
```

In [180]:

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

Out[180]:

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

7032 rows × 9 columns

I deleted the below columns because python file and %run predictions was not working.

In [181]:

```
del df['totalcharges_monthlycharges_ratio']
```

In [182]:

```
del df['totalcharges_tenure_ratio']
```

In [183]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7032 entries, 7590-VHVEG to 3186-AJIEK
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   tenure          7032 non-null   int64  
 1   PhoneService    7032 non-null   int64  
 2   Contract        7032 non-null   int64  
 3   PaymentMethod   7032 non-null   int64  
 4   MonthlyCharges  7032 non-null   float64
 5   TotalCharges    7032 non-null   float64
 6   Churn           7032 non-null   int64  
dtypes: float64(2), int64(5)
memory usage: 439.5+ KB
```

AutoML with Pycaret

In [184]:

```
conda install -c conda-forge pycaret -y
```

```
Collecting package metadata (current_repodata.json): ...working... done
Note: you may need to restart the kernel to use updated packages.
Solving environment: ...working... done

# All requested packages already installed.
```

In [185]:

```
from pycaret.classification import setup, compare_models, predict_model, save_model, load_m
```

In [186]:

```
automl = setup(data = df, target = 'Churn', fold_shuffle=True, preprocess=False)
```

	Description	Value
0	session_id	5164
1	Target	Churn
2	Target Type	Binary
3	Label Encoded	0: 0, 1: 1
4	Original Data	(7032, 7)
5	Missing Values	False
6	Numeric Features	3
7	Categorical Features	3
8	Transformed Train Set	(4922, 6)
9	Transformed Test Set	(2110, 6)
10	Shuffle Train-Test	True
11	Stratify Train-Test	False
12	Fold Generator	StratifiedKFold
13	Fold Number	10
14	CPU Jobs	-1
15	Use GPU	False
16	Log Experiment	False
17	Experiment Name	clf-default-name
18	USI	895e
19	Fix Imbalance	False
20	Fix Imbalance Method	SMOTE

In [211]:

```
automl[6]
```

Out[211]:

```
-1
```

In [188]:

```
best_model = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
catboost	CatBoost Classifier	0.7938	0.8384	0.5176	0.6517	0.5766	0.4427	0.4481	2.0520
ada	Ada Boost Classifier	0.7928	0.8379	0.5221	0.6461	0.5770	0.4420	0.4466	0.3310
ridge	Ridge Classifier	0.7922	0.0000	0.4712	0.6666	0.5516	0.4216	0.4326	0.1080
gbc	Gradient Boosting Classifier	0.7907	0.8381	0.5138	0.6443	0.5713	0.4353	0.4404	0.2580
lr	Logistic Regression	0.7901	0.8346	0.5310	0.6362	0.5787	0.4405	0.4438	8.4090
lda	Linear Discriminant Analysis	0.7879	0.8224	0.5153	0.6350	0.5687	0.4301	0.4344	0.0260
rf	Random Forest Classifier	0.7777	0.8110	0.5026	0.6111	0.5509	0.4052	0.4090	0.4140
svm	SVM - Linear Kernel	0.7710	0.0000	0.4070	0.6409	0.4759	0.3449	0.3705	0.0350
et	Extra Trees Classifier	0.7696	0.7892	0.5078	0.5885	0.5446	0.3917	0.3939	0.3060
knn	K Neighbors Classifier	0.7639	0.7429	0.4405	0.5892	0.5038	0.3530	0.3597	0.0540
qda	Quadratic Discriminant Analysis	0.7493	0.8255	0.7449	0.5289	0.6180	0.4397	0.4544	0.2740
dt	Decision Tree Classifier	0.7401	0.6769	0.5281	0.5212	0.5241	0.3455	0.3459	0.0360
nb	Naive Bayes	0.7192	0.8082	0.7659	0.4906	0.5976	0.3980	0.4215	0.0200
xgboost	Extreme Gradient Boosting	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0130
lightgbm	Light Gradient Boosting Machine	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0160

In [189]:

```
best_model
```

Out[189]:

```
<catboost.core.CatBoostClassifier at 0x2015b5d3a00>
```

In [190]:

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

Out[190]:

```
(1, 7)
```

In [191]:

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

Out[191]:

	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges	C
customerID							
8361-LTMKD	4	1	0	2	74.4	306.6	

Saving and Loading our trained Model:

In [192]:

```
save_model(best_model, 'catboost')
```

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

Out[192]:

```
(Pipeline(memory=None,
          steps=[('dtypes',
                  DataTypes_Auto_infer(categorical_features=[],
                                       display_types=True, features_todrop=
[],
                                       id_columns=[],
                                       ml_usecase='classification',
                                       numerical_features=[], target='Chur
n',
                                       time_features=[])),
                 ['trained_model',
                  <catboost.core.CatBoostClassifier object at 0x000002015B5D
3A00>]],
          verbose=False),
 'catboost.pkl')
```

In [193]:

```
import pickle

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

In [194]:

```
with open('catboost_model.pk', 'rb') as f:
    loaded_model = pickle.load(f)
```

In [195]:

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

Out[195]:

```
array([1], dtype=int64)
```

In [196]:

```
loaded_ada = load_model('catboost')
```

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

In [197]:

```
predict_model(loaded_ada, new_data)
```

Out[197]:

	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges	L
customerID							
8361-LTMKD	4	1	0	2	74.4	306.6	

Making a Python Module to Make Predictions:

In [209]:

```
from IPython.display import Code
Code('predict_churn.py')
```

Out[209]:

```
import pandas as pd
from pycaret.classification import predict_model, load_model


def load_data(prepped_Churn_data):
    """
    Loads diabetes data into a DataFrame from a string filepath.
    """
    df = pd.read_csv(prepped_Churn_data, 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('ada')
    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 [210]:

```
%run predict_Churn.py
```

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

Summary:

Firstly, imported pandas and prepped_Churn_data file. Droping the totalcharges_monthlycharges_ratio and totalcharges_tenure_ratio colums from the dataset. Installed the pycaret library and imported the required packages like setup, compare_models, predict_models, save_model, load_model from pycaret.classification. We ran automl to find the best model, But in this case the best model here is Catboost Classifer (Catboost). Here we used pycaret predict model to make predictions. The score should be greater than or equal to 0.5 (My_Score = 0.560). Next saved and loaded my trained model as a pickle file. We use pickle to save our data to file. Transformation pipeline and model successfully loaded. We used VS Code to write functions for predict model, load model and print predictions. So the final true values are [0,0,0,1,0].