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 [34]:
```
import pandas as pd

df = pd.read_csv('C:/Users/Asmita Bamma/Downloads/prepped_churn_data.csv', ir
df
```

Out[34]:

customerID	Unnamed: 0	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges
7590-VHVEG	0	1	0	0	0	29.85
5575-GNVDE	1	34	1	1	1	56.95
3668-QPYBK	2	2	1	0	1	53.85
7795-CFOCW	3	45	0	1	2	42.30
9237-HQITU	4	2	1	0	0	70.70
...	...	...	...	...	...	...
6840						

In [33]:
```
df_copy = df.copy()
```

In [37]:
```
del df['Unnamed: 0']
```

In [38]:
```
del df['tenure_TotalCharges_ratio']
```

In [39]:
```
del df['tenure_MonthlyCharges_ratio']
```

In [40]:
```
from pycaret.classification import setup, compare_models, predict_model, save
```

In [41]:
```
automl = setup(data = df, target = 'Churn',  fold_shuffle=True, preprocess=Fa
```

	Description	Value
0	session_id	2
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	2845
19	Fix Imbalance	False
20	Fix Imbalance Method	SMOTE

In [42]:
```
best_model = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	0.7938	0.8379	0.5088	0.6377	0.5651	0.4324	0.4376	0.3820
ridge	Ridge Classifier	0.7938	0.0000	0.4465	0.6620	0.5327	0.4069	0.4202	0.0070
gbc	Gradient Boosting Classifier	0.7920	0.8370	0.4896	0.6392	0.5541	0.4215	0.4282	0.0880
lda	Linear Discriminant Analysis	0.7915	0.8261	0.5004	0.6333	0.5582	0.4245	0.4300	0.0080
ada	Ada Boost Classifier	0.7899	0.8369	0.5042	0.6292	0.5584	0.4230	0.4283	0.0950
catboost	CatBoost Classifier	0.7899	0.8344	0.4788	0.6353	0.5455	0.4126	0.4198	0.8850
rf	Random Forest Classifier	0.7694	0.8021	0.4758	0.5767	0.5209	0.3710	0.3743	0.1010
knn	K Neighbors Classifier	0.7623	0.7449	0.4404	0.5623	0.4931	0.3412	0.3459	0.0130
et	Extra Trees Classifier	0.7546	0.7797	0.4765	0.5410	0.5059	0.3437	0.3454	0.0860
qda	Quadratic Discriminant Analysis	0.7493	0.8266	0.7428	0.5182	0.6102	0.4341	0.4496	0.0080
svm	SVM - Linear Kernel	0.7329	0.0000	0.4726	0.5636	0.4754	0.3136	0.3387	0.0110
dt	Decision Tree Classifier	0.7214	0.6484	0.4788	0.4726	0.4746	0.2854	0.2860	0.0090
nb	Naive Bayes	0.7192	0.8123	0.7713	0.4810	0.5921	0.3954	0.4214	0.0070
xgboost	Extreme Gradient Boosting	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0060
lightgbm	Light Gradient Boosting Machine	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0060

In [43]:
```
best_model
```

Out[43]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
                   intercept_scaling=1, l1_ratio=None, max_iter=1000,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=2, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [44]:
```
df.iloc[-2:-1].shape
```

Out[44]: (1, 7)

In [45]:
```
predict_model(best_model, df.iloc[-2:-1])
```

Out[45]:

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

Saving and loading our model

In [46]:
```
save_model(best_model, 'LDA')
```

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

Out[46]:
```
(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',
                  LogisticRegression(C=1.0, class_weight=None, dual=False,
                                     fit_intercept=True, intercept_scaling=
1,
                                     l1_ratio=None, max_iter=1000,
                                     multi_class='auto', n_jobs=None,
                                     penalty='l2', random_state=2,
                                     solver='lbfgs', tol=0.0001, verbose=0,
                                     warm_start=False)]],
          verbose=False),
 'LDA.pkl')
```

In [47]:
```
import pickle

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

In [48]:
```
with open('LDA_model.pk', 'rb') as f:
    loaded_model = pickle.load(f)
```

In [49]:
```
new_data = df.iloc[-2:-1].copy()
new_data.drop('Churn', axis=1, inplace=True)
loaded_model.predict(new_data)
```

Out[49]:
```
array([1])
```

In [50]:
```
loaded_model.predict(new_data)
```

Out[50]:
```
array([1])
```

In [51]:
```
loaded_lda = load_model('LDA')
```

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

In [54]:
```
predict_model(loaded_lda, new_data)
```

Out[54]:

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

Making a Python module to make predictions

In [53]:
```
from IPython.display import Code

Code('predict_churn.py')
```

```
taFrame.
    """

    model = load_model('LDA')
    predictions = predict_model(model, data=df)
    predictions.rename({'Label': 'Churn_prediction'}, axis=1, inplace=Tr
ue)
    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 [55]:
```
%run predict_churn.py
```

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

Summary

First, prepped churn data was imported and then we also imported functions like setup, compare_models, predict_model, save_model, load_model from pycaret.classification. After running setup function, we got list of features with values which seem fine. By default, it converted categorical columns into numeric. Afterthat, we ran automl to find the best model. Linear Discriminant Analysis (LDA) model came out to be the best model. Then we used pycaret's predict_model to make predictions. The score was greater than 0.5. Then, we saved and loaded our model as pickle file. Python file and %run predictions was not working for me so I had to delete some of the features. And finally got the results [0,1,0,0,1].