Benjamin Brufffey - 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

df = pd.read_csv('data/prepped_churn_data_bruffey.csv', index_col='cus
df
```

Out[1]:

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

7032 rows × 8 columns

Updating conda was required prior to being able to install pycaret correctly. Without this step, pycaret was not installing correctly based upon FTE instructions. This is likely due to variances in versioning as to when the assignment was created to now.

In [2]:
```
conda update -n base conda
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /opt/anaconda3

  added / updated specs:
    - conda


The following packages will be SUPERSEDED by a higher-priority channel:

  ca-certificates    conda-forge::ca-certificates-2022.5.1~ --> pkgs/mai
n::ca-certificates-2022.4.26-hecd8cb5_0
  certifi            conda-forge::certifi-2022.5.18.1-py39~ --> pkgs/mai
n::certifi-2022.5.18.1-py39hecd8cb5_0
  conda              conda-forge::conda-4.13.0-py39h6e9494~ --> pkgs/mai
n::conda-4.13.0-py39hecd8cb5_0
  openssl            conda-forge::openssl-1.1.1o-hfe4f2af_0 --> pkgs/mai
n::openssl-1.1.1o-hca72f7f_0


Preparing transaction: done
Verifying transaction: done
Executing transaction: done

Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
conda install -c conda-forge pycaret -y
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /opt/anaconda3

  added / updated specs:
    - pycaret


The following packages will be UPDATED:

  ca-certificates    pkgs/main::ca-certificates-2022.4.26-~ --> conda-for
ge::ca-certificates-2022.5.18.1-h033912b_0
  conda              pkgs/main::conda-4.13.0-py39hecd8cb5_0 --> conda-for
ge::conda-4.13.0-py39h6e9494a_1

The following packages will be SUPERSEDED by a higher-priority channel:

  certifi            pkgs/main::certifi-2022.5.18.1-py39he~ --> conda-for
ge::certifi-2022.5.18.1-py39h6e9494a_0
  openssl              pkgs/main::openssl-1.1.1o-hca72f7f_0 --> conda-for
ge::openssl-1.1.1o-hfe4f2af_0


Preparing transaction: done
Verifying transaction: done
Executing transaction: done

Note: you may need to restart the kernel to use updated packages.
```

AutoML with pycaret

In [4]:
```
from pycaret.classification import setup, compare_models, predict_mode
```

As there are categorical columns that will produce additional "feature" columns, the categorical columns need converted into numeric_features so as to maintain the 7 feature columns. This was causing issues down below in the prediction test against new_data.

In [5]:
```
#automl = setup(df, target='Churn')
# Running the below line to convert categorical columns into numeric
```

In [6]:
```
#This needs to be run in order to keep features to 7 in the model, ot
automl = setup(df, target='Churn', preprocess=False, numeric_features
```

	Description	Value
0	session_id	1446
1	Target	Churn
2	Target Type	Binary
3	Label Encoded	0: 0, 1: 1
4	Original Data	(7032, 8)
5	Missing Values	0
6	Numeric Features	7
7	Categorical Features	0
8	Transformed Train Set	(4922, 7)
9	Transformed Test Set	(2110, 7)
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	0
16	Log Experiment	0
17	Experiment Name	clf-default-name
18	USI	25a4
19	Fix Imbalance	0
20	Fix Imbalance Method	SMOTE

In [45]:
```
#This index seems to bounce around with what it actually correlates. 
automl[24]
```

Out[45]:

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

7032 rows × 8 columns

compare_models() was used to produce a list of models ranked by accuracy. A sort filter could be applied to change which category is used to rank by such as precision or AUC, etc.

In [8]:
```
best_model = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	0.7960	0.8317	0.5274	0.6442	0.5795	0.4467	0.4509	0.2400
ada	Ada Boost Classifier	0.7942	0.8324	0.4947	0.6508	0.5609	0.4301	0.4376	0.0210
catboost	CatBoost Classifier	0.7938	0.8348	0.5069	0.6463	0.5675	0.4348	0.4407	0.2680
ridge	Ridge Classifier	0.7926	0.0000	0.4635	0.6590	0.5439	0.4149	0.4258	0.0040
gbc	Gradient Boosting Classifier	0.7924	0.8375	0.5008	0.6432	0.5623	0.4291	0.4354	0.0590
lda	Linear Discriminant Analysis	0.7871	0.8183	0.4992	0.6275	0.5557	0.4182	0.4231	0.0040
lightgbm	Light Gradient Boosting Machine	0.7810	0.8261	0.5144	0.6062	0.5562	0.4122	0.4149	2.0820
xgboost	Extreme Gradient Boosting	0.7781	0.8146	0.5023	0.6017	0.5470	0.4018	0.4050	0.9420
rf	Random Forest Classifier	0.7714	0.8031	0.4741	0.5895	0.5247	0.3768	0.3812	0.0660
knn	K Neighbors Classifier	0.7710	0.7420	0.4635	0.5917	0.5188	0.3718	0.3771	0.0080
svm	SVM - Linear Kernel	0.7680	0.0000	0.4156	0.6021	0.4779	0.3394	0.3561	0.0070
et	Extra Trees Classifier	0.7564	0.7792	0.4727	0.5513	0.5080	0.3477	0.3500	0.0560
qda	Quadratic Discriminant Analysis	0.7466	0.8159	0.7024	0.5209	0.5972	0.4188	0.4295	0.0040
nb	Naive Bayes	0.7422	0.8112	0.7054	0.5131	0.5934	0.4117	0.4235	0.0040
dt	Decision Tree Classifier	0.7186	0.6470	0.4810	0.4742	0.4769	0.2847	0.2851	0.0050

In [9]:
```
best_model
```

Out[9]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tr
ue,
                   intercept_scaling=1, l1_ratio=None, max_iter=1000,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=1446, solver='lbfgs', tol=0.0001, verbose
=0,
                   warm_start=False)
```

In the case of this current run, logistic regression was the highest accuracy model. In other attemps not changing anything, I've seen GBC and Catboost also rank as the top which seems to align with the FTE mentioning randomness.

In [10]:
```
df.iloc[-2:-1].shape
```
Out[10]:
```
(1, 8)
```

In [11]:
```
df.iloc[-1].shape
```
Out[11]:
```
(8,)
```

In [12]:
```
predict_model(best_model, df.iloc[-2:-1])
```
Out[12]:

	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges	Churn
customerID							
8361-LTMKD	4	1	0	3	74.4	306.6	1

predict_model creates a new column, 'Score', with the probability of class 1. It also creates a 'Label' column with the predicted label, where it rounds up if score is >= 0.5 (greater than or equal to 0.5).

Saving and loading our model

Save the model as a pickle file, resulting file is `LR.pkl`

In [13]:
```
save_model(best_model, 'LR')
```

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

Out[13]:
```
(Pipeline(memory=None,
          steps=[('dtypes',
                  DataTypes_Auto_infer(categorical_features=[],
                                       display_types=True, features_todro
p=[],
                                       id_columns=[],
                                       ml_usecase='classification',
                                       numerical_features=['Contract',
                                                           'PhoneServic
e',
                                                           'PaymentMetho
d'],
                                       target='Churn', time_features=
[])),
                 ['trained_model',
                  LogisticRegression(C=1.0, class_weight=None, dual=Fals
e,
                                     fit_intercept=True, intercept_scalin
g=1,
                                     l1_ratio=None, max_iter=1000,
                                     multi_class='auto', n_jobs=None,
                                     penalty='l2', random_state=1446,
                                     solver='lbfgs', tol=0.0001, verbose=
0,
                                     warm_start=False)]],
          verbose=False),
 'LR.pkl')
```

In [14]:
```
import pickle

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

In [15]:
```
with open('LR_model.pk', 'rb') as f:
    loaded_model = pickle.load(f)
```

In [16]:
```
new_data = df.iloc[-2:-1].copy()
new_data.drop('Churn', axis=1, inplace=True)
loaded_model.predict(new_data)
```

Out[16]:
```
array([1])
```

Test loading of the LR model

In [22]:
```
loaded_lr = load_model('LR')
```

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

Use loaded LR model against the new data created above to produce a prediction of churn.

In [23]:
```
predict_model(loaded_lr, new_data)
```

Out[23]:

	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges	charg
customerID							
8361-LTMKD	4	1	0	3	74.4	306.6	

Using pip to install ipyhon

```
In [24]: pip install ipython
```

```
Requirement already satisfied: ipython in /opt/anaconda3/lib/python3.9/si
te-packages (8.3.0)
Requirement already satisfied: pickleshare in /opt/anaconda3/lib/python3.
9/site-packages (from ipython) (0.7.5)
Requirement already satisfied: setuptools>=18.5 in /opt/anaconda3/lib/pyt
hon3.9/site-packages (from ipython) (61.2.0)
Requirement already satisfied: stack-data in /opt/anaconda3/lib/python3.
9/site-packages (from ipython) (0.2.0)
Requirement already satisfied: traitlets>=5 in /opt/anaconda3/lib/python
3.9/site-packages (from ipython) (5.1.1)
Requirement already satisfied: pexpect>4.3 in /opt/anaconda3/lib/python3.
9/site-packages (from ipython) (4.8.0)
Requirement already satisfied: jedi>=0.16 in /opt/anaconda3/lib/python3.
9/site-packages (from ipython) (0.18.1)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.
0.0 in /opt/anaconda3/lib/python3.9/site-packages (from ipython) (3.0.20)
Requirement already satisfied: appnope in /opt/anaconda3/lib/python3.9/si
te-packages (from ipython) (0.1.2)
Requirement already satisfied: matplotlib-inline in /opt/anaconda3/lib/py
thon3.9/site-packages (from ipython) (0.1.2)
Requirement already satisfied: decorator in /opt/anaconda3/lib/python3.9/
site-packages (from ipython) (5.1.1)
Requirement already satisfied: backcall in /opt/anaconda3/lib/python3.9/s
ite-packages (from ipython) (0.2.0)
Requirement already satisfied: pygments>=2.4.0 in /opt/anaconda3/lib/pyth
on3.9/site-packages (from ipython) (2.11.2)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in /opt/anaconda3/lib/
python3.9/site-packages (from jedi>=0.16->ipython) (0.8.3)
Requirement already satisfied: ptyprocess>=0.5 in /opt/anaconda3/lib/pyth
on3.9/site-packages (from pexpect>4.3->ipython) (0.7.0)
Requirement already satisfied: wcwidth in /opt/anaconda3/lib/python3.9/si
te-packages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython)
(0.2.5)
Requirement already satisfied: pure-eval in /opt/anaconda3/lib/python3.9/
site-packages (from stack-data->ipython) (0.2.2)
Requirement already satisfied: executing in /opt/anaconda3/lib/python3.9/
site-packages (from stack-data->ipython) (0.8.3)
Requirement already satisfied: asttokens in /opt/anaconda3/lib/python3.9/
site-packages (from stack-data->ipython) (2.0.5)
Requirement already satisfied: six in /opt/anaconda3/lib/python3.9/site-p
ackages (from asttokens->stack-data->ipython) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

Display the code from the `predict_churn.py` file

In [25]:
```
from IPython.display import Code

Code('predict_churn.py')
```

Out[25]:
```
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 dat
aframe.
    """
    model = load_model('LR')
    predictions = predict_model(model, data=df)
    predictions.rename({'Label': 'Churn_prediction'}, axis=1, inplace=Tru
e)
    predictions['Churn_prediction'].replace({1: 'Churn', 0: 'No Churn'},
                                            inplace=True)
    return predictions['Churn_prediction']


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

Run the `predict_churn.py` file which is targeting the `new_churn_data.csv` file. This should produce a set of predictions for churn.

In [26]:
```
%run predict_churn.py
```

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

Summary

First the prepped data from week 2 was loaded into a padas dataframe. There was a slight modifificaiton to this data. The ratio of total to monthly charges column was dropped as it was not included in the new_churn_data.csv provided for the assignment which caused issues when attempting to run the predicions model against it. Additionally, the column for charge to tenure ratio was renamed to charge_per_tenure.

Next, conda needed to be updated. Without this step, the install of pycaret was failing. Once conda was updated, then pycharet was installed.

Then, `automl = setup(df, target='Churn')` could be run, however this causes issues where the model ends up with 11 features (due to the categorial columns), and it shows the three categorical columns in its output. To make all columns be numerical (and keep features to 7), `automl = setup(df, target='Churn', preprocess=False, numeric_features=['Contract','PhoneService','PaymentMethod'])` was run to set them to numeric.

`best_model = compare_models()` was then run to show the rankings of the models. Logistic Regression (LR) was found to be the highest for Accuracy, Kappa, and MCC and so the assignment proceeded with this as the best model. The default sort of Accuracy was used and so the primary metric that this decision was based up on was accuracy.

This model was saved to disk for use later in the python module/function.

A python file was created with a function that takes the pandas DF as an input and returns the probability for churn. This is based off of the logistic regression model that was selected for its highest accuracy.

The python module and function were used against the file `new_churn_data.csv`. The results from using the model with the function are [1, 0, 0, 0, 0], this is in comparison to the true values of [1, 0, 0, 1, 0], so there is one false negative that arose as a result from this model/prediction so the model is working relatively OK.

In [ ]:
```

```