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 [6]:

```
Requirement already satisfied: scikit-learn==0.23.2 in c:\users\ediso
n\anaconda3\lib\site-packages (0.23.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\ediso
n\anaconda3\lib\site-packages (from scikit-learn==0.23.2) (2.1.0)
Requirement already satisfied: numpy>=1.13.3 in c:\users\edison\anaco
nda3\lib\site-packages (from scikit-learn==0.23.2) (1.20.1)
Requirement already satisfied: joblib>=0.11 in c:\users\edison\anacon
da3\lib\site-packages (from scikit-learn==0.23.2) (1.0.1)
Requirement already satisfied: scipy>=0.19.1 in c:\users\edison\anaco
nda3\lib\site-packages (from scikit-learn==0.23.2) (1.6.2)
```

In [11]:

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

# All requested packages already installed.
```

In [12]:

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

# All requested packages already installed.
```

In [13]:

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

# All requested packages already installed.
```

In [17]:

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

## Package Plan ##

  environment location: C:\Users\Edison\anaconda3

  added / updated specs:
    - scikit-plot
```

In [18]:

```python
import pandas as pd

df = pd.read_csv('data/prepped_churn_data.csv', index_col='customerID'
```

Out[18]:

customerID	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges
5375	1	0	0	2	29.85	29.85
3962	34	1	1	3	56.95	1889.50
2564	2	1	0	3	53.85	108.15
5535	45	0	1	0	42.30	1840.75
6511	2	1	0	2	70.70	151.65
...	...	...	...	...	...	...
4853	24	1	1	3	84.80	1990.50
1525	72	1	1	1	103.20	7362.90
3367	11	0	0	2	29.60	346.45
5934	4	1	0	3	74.40	306.60
2226	66	1	2	0	105.65	6844.50

7043 rows × 7 columns

In [9]:

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

# All requested packages already installed.
```

In [19]:

In [20]:

Description	Value

	Description	Value
0	session_id	6884
1	Target	Churn
2	Target Type	Binary
3	Label Encoded	0: 0, 1: 1
4	Original Data	(7043, 7)
5	Missing Values	False
6	Numeric Features	3
7	Categorical Features	3
8	Ordinal Features	False
9	High Cardinality Features	False
10	High Cardinality Method	None
11	Transformed Train Set	(4930, 11)
12	Transformed Test Set	(2113, 11)
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	20ac
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
33	PCA	False
34	PCA Method	None

	Description	Value
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

In [21]:

In [26]:

Out[26]:

	tenure	MonthlyCharges	TotalCharges	PhoneService_0	Contract_0	Contract_1
customerID						
4303	23.0	57.200001	1423.349976	0.0	0.0	1.0
6278	68.0	84.400002	5746.750000	0.0	0.0	0.0
4152	32.0	19.799999	607.700012	0.0	0.0	1.0
3650	2.0	75.800003	160.750000	0.0	1.0	0.0
6533	3.0	24.600000	86.349998	0.0	1.0	0.0
...	...	...	...	...	...	...
1382	1.0	55.700001	55.700001	0.0	1.0	0.0
4226	12.0	19.350000	219.350006	0.0	0.0	1.0
4600	8.0	51.299999	411.600006	0.0	1.0	0.0

	tenure	MonthlyCharges	TotalCharges	PhoneService_0	Contract_0	Contract_1
customerID						
1405	5.0	49.400002	232.550003	0.0	1.0	0.0

In [34]:

Out[34]:

	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges
customerID						
5375	1	0	0	2	29.85	29.85
3962	34	1	1	3	56.95	1889.50
2564	2	1	0	3	53.85	108.15
5535	45	0	1	0	42.30	1840.75
6511	2	1	0	2	70.70	151.65
...	...	...	...	...	...	...
4853	24	1	1	3	84.80	1990.50
1525	72	1	1	1	103.20	7362.90
3367	11	0	0	2	29.60	346.45
5934	4	1	0	3	74.40	306.60
2226	66	1	2	0	105.65	6844.50

7043 rows × 7 columns

In [35]:

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	0.7968	0.8385	0.5113	0.6377	0.5663	0.4359	0.4412	0.6420
lda	Linear Discriminant Analysis	0.7955	0.8291	0.5269	0.6292	0.5722	0.4395	0.4433	0.0180
catboost	CatBoost Classifier	0.7951	0.8398	0.5113	0.6313	0.5638	0.4322	0.4370	1.7170
gbc	Gradient Boosting Classifier	0.7949	0.8440	0.5058	0.6339	0.5614	0.4300	0.4354	0.2560
ridge	Ridge Classifier	0.7927	0.0000	0.4467	0.6493	0.5272	0.4007	0.4133	0.0160
ada	Ada Boost Classifier	0.7925	0.8398	0.4848	0.6332	0.5483	0.4168	0.4236	0.1130
lightgbm	Light Gradient Boosting Machine	0.7907	0.8338	0.5261	0.6146	0.5659	0.4293	0.4322	0.2770

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
xgboost	Extreme Gradient Boosting	0.7878	0.8259	0.5122	0.6106	0.5558	0.4182	0.4217	0.3070
rf	Random Forest Classifier	0.7789	0.8079	0.4973	0.5918	0.5386	0.3951	0.3987	0.2960
svm	SVM - Linear Kernel	0.7718	0.0000	0.4256	0.5966	0.4843	0.3464	0.3607	0.0300
knn	K Neighbors Classifier	0.7682	0.7458	0.4568	0.5690	0.5053	0.3566	0.3611	0.0520
et	Extra Trees Classifier	0.7649	0.7825	0.4965	0.5556	0.5230	0.3680	0.3699	0.2670
dt	Decision Tree Classifier	0.7436	0.6713	0.5137	0.5084	0.5100	0.3367	0.3374	0.0240

In [36]:

Out[36]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercep
t=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=1000,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=6884, solver='lbfgs', tol=0.0001, ver
bose=0,
                   warm_start=False)
```

In [37]:

Out[37]:

```
(1, 7)
```

In [38]:

Out[38]:

	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges
customerID						
5934	4	1	0	3	74.4	306.6

In [40]:

Out[40]:

	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges
customerID						
2521	55	1	1	1	60.00	3316.10
4893	1	1	0	2	75.75	75.75
6875	38	1	0	1	69.50	2625.25
437	67	1	0	1	102.95	6886.25
5995	19	1	0	0	78.70	1495.10
5504	12	0	1	2	60.65	743.30

	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges
customerID						
1758	72	1	2	0	21.15	1419.40
4853	24	1	1	3	84.80	1990.50
1525	72	1	1	1	103.20	7362.90

In [41]:

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

Out[41]:

```
(Pipeline(memory=None,
          steps=[('dtypes',
                  DataTypes_Auto_infer(categorical_features=[],
                                       display_types=True, features_t
odrop=[],
                                       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...
                 ('feature_select', 'passthrough'), ('fix_multi', 'pa
ssthrough'),
                 ('dfs', 'passthrough'), ('pca', 'passthrough'),
                 ['trained_model',
                  LogisticRegression(C=1.0, class_weight=None, dual=F
alse,
                                     fit_intercept=True, intercept_sc
aling=1,
                                     l1_ratio=None, max_iter=1000,
                                     multi_class='auto', n_jobs=None,
                                     penalty='l2', random_state=6884,
                                     solver='lbfgs', tol=0.0001, verb
ose=0,
                                     warm_start=False)]],
          verbose=False),
 'LR.pkl')
```

In [42]:

```python
import pickle

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

In [43]:

```python
with open('LR_model.pk', 'rb') as f:
```

In [54]:

In [57]:

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

In [58]:

Out[58]:

	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges
customerID						
5934	4	1	0	3	74.4	306.6

In [68]:

```python
from IPython.display import Code
```

Out[68]:

```python
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
dataframe.
    """
    model = load_model('LR')
    predictions = predict_model(model, data=df)
    predictions.rename({'Label': 'Churn prediction'}, axis=1, inplace
```

In [69]:

```
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       Churn
Name: Churn_prediction, dtype: object
```

In [ ]:

Summary

I previously had errors due to what seemed to be package version incompatibilities with pycaret. Using "!pip install -U scikit-learn==0.23.2" with a frehs installation of Anaconda on a different computer solved that problem, and required me to install previously installed packages such as scikit-plot.

After using "compare_models" to identify best models for different metrics, can see that Logistic Regression is best for Accuracy, while Gradient Boosting Classifier is best for AUC, Naive Bayes is best for Recall and F1, and Ridge Classifier is best for Precision. Looking at Week 3 summary, AUC (so the Gradient Boosting Classifier) is probably the best fit model, since the dataset is somewhat skewed, and we're trying to have the model predict a binary outcome ('yes Churn vs no Churn'). Eithe way, the Logistic Regression and Gradient Boosting Classifier models have fairly close values for both Accuracy and AUC.

I then tested and saved the traind model, and prepared it for Python with pickle. Then created the Python script and ran it -- loading the new_churn_data.csv and and testing the application of the LR model. Last I saved the updated Jupiter notebook and Python script to GitHub.

In [ ]: