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.

Importing all the required packages and installing them

```
!conda create -n pycaret_env1 python=3.10 -y
!conda activate pycaret_env1
!pip install pycaret
```

```
Channels:
 - defaults
Platform: win-64
```

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

## Package Plan ##

  environment location: C:\Users\Chandra\anaconda3\envs\pycaret_env

  added / updated specs:
    - python=3.10.14


The following NEW packages will be INSTALLED:

  bzip2              pkgs/main/win-64::bzip2-1.0.8-h2bbff1b_6
  ca-certificates    pkgs/main/win-64::ca-certificates-2024.7.2-
haa95532_0
  libffi             pkgs/main/win-64::libffi-3.4.4-hd77b12b_1
  openssl            pkgs/main/win-64::openssl-3.0.15-h827c3e9_0
  pip                pkgs/main/win-64::pip-24.2-py310haa95532_0
  python             pkgs/main/win-64::python-3.10.14-he1021f5_1
  setuptools         pkgs/main/win-64::setuptools-75.1.0-
py310haa95532_0
  sqlite             pkgs/main/win-64::sqlite-3.45.3-h2bbff1b_0
  tk                 pkgs/main/win-64::tk-8.6.14-h0416ee5_0
  tzdata             pkgs/main/noarch::tzdata-2024a-h04d1e81_0
  vc                 pkgs/main/win-64::vc-14.40-h2eaa2aa_1
  vs2015_runtime     pkgs/main/win-64::vs2015_runtime-14.40.33807-
h98bb1dd_1
  wheel              pkgs/main/win-64::wheel-0.44.0-py310haa95532_0
  xz                 pkgs/main/win-64::xz-5.4.6-h8cc25b3_1
  zlib               pkgs/main/win-64::zlib-1.2.13-h8cc25b3_1



Downloading and Extracting Packages: ...working... done
Preparing transaction: ...working... done
Verifying transaction: ...working... done
Executing transaction: ...working... done
#
# To activate this environment, use
#
#     $ conda activate pycaret_env
#
# To deactivate an active environment, use
#
#     $ conda deactivate

Collecting pycaret
  Using cached pycaret-3.3.2-py3-none-any.whl.metadata (17 kB)
Requirement already satisfied: ipython>=5.5.0 in c:\users\chandra\
```

```
appdata\roaming\python\python312\site-packages (from pycaret) (8.26.0)
Requirement already satisfied: ipywidgets>=7.6.5 in c:\users\chandra\
anaconda3\lib\site-packages (from pycaret) (7.8.1)
Requirement already satisfied: tqdm>=4.62.0 in c:\users\chandra\
anaconda3\lib\site-packages (from pycaret) (4.66.4)
Requirement already satisfied: numpy<1.27,>=1.21 in c:\users\chandra\
anaconda3\lib\site-packages (from pycaret) (1.26.4)
Requirement already satisfied: pandas<2.2.0 in c:\users\chandra\
anaconda3\lib\site-packages (from pycaret) (2.1.4)
Requirement already satisfied: jinja2>=3 in c:\users\chandra\
anaconda3\lib\site-packages (from pycaret) (3.1.4)
Requirement already satisfied: scipy<=1.11.4,>=1.6.1 in c:\users\
chandra\anaconda3\lib\site-packages (from pycaret) (1.11.4)
Requirement already satisfied: joblib<1.4,>=1.2.0 in c:\users\chandra\
anaconda3\lib\site-packages (from pycaret) (1.3.2)
Requirement already satisfied: scikit-learn>1.4.0 in c:\users\chandra\
anaconda3\lib\site-packages (from pycaret) (1.4.2)
Requirement already satisfied: pyod>=1.1.3 in c:\users\chandra\
anaconda3\lib\site-packages (from pycaret) (2.0.2)
Requirement already satisfied: imbalanced-learn>=0.12.0 in c:\users\
chandra\anaconda3\lib\site-packages (from pycaret) (0.12.3)
Collecting category-encoders>=2.4.0 (from pycaret)
  Using cached category_encoders-2.6.3-py2.py3-none-any.whl.metadata
(8.0 kB)
Collecting lightgbm>=3.0.0 (from pycaret)
  Using cached lightgbm-4.5.0-py3-none-win_amd64.whl.metadata (17 kB)
Requirement already satisfied: numba>=0.55.0 in c:\users\chandra\
anaconda3\lib\site-packages (from pycaret) (0.60.0)
Requirement already satisfied: requests>=2.27.1 in c:\users\chandra\
anaconda3\lib\site-packages (from pycaret) (2.32.2)
Requirement already satisfied: psutil>=5.9.0 in c:\users\chandra\
appdata\roaming\python\python312\site-packages (from pycaret) (6.0.0)
Requirement already satisfied: markupsafe>=2.0.1 in c:\users\chandra\
anaconda3\lib\site-packages (from pycaret) (2.1.3)
Requirement already satisfied: importlib-metadata>=4.12.0 in c:\users\
chandra\anaconda3\lib\site-packages (from pycaret) (7.0.1)
Requirement already satisfied: nbformat>=4.2.0 in c:\users\chandra\
anaconda3\lib\site-packages (from pycaret) (5.9.2)
Requirement already satisfied: cloudpickle in c:\users\chandra\
anaconda3\lib\site-packages (from pycaret) (2.2.1)
Requirement already satisfied: deprecation>=2.1.0 in c:\users\chandra\
anaconda3\lib\site-packages (from pycaret) (2.1.0)
Requirement already satisfied: xxhash in c:\users\chandra\anaconda3\
lib\site-packages (from pycaret) (3.5.0)
Collecting matplotlib<3.8.0 (from pycaret)
  Using cached matplotlib-3.7.5-cp312-cp312-win_amd64.whl.metadata
(5.8 kB)
Requirement already satisfied: scikit-plot>=0.3.7 in c:\users\chandra\
anaconda3\lib\site-packages (from pycaret) (0.3.7)
```

```
Collecting yellowbrick>=1.4 (from pycaret)
  Using cached yellowbrick-1.5-py3-none-any.whl.metadata (7.7 kB)
Requirement already satisfied: plotly>=5.14.0 in c:\users\chandra\
anaconda3\lib\site-packages (from pycaret) (5.22.0)
Requirement already satisfied: kaleido>=0.2.1 in c:\users\chandra\
anaconda3\lib\site-packages (from pycaret) (0.2.1)
Requirement already satisfied: schemdraw==0.15 in c:\users\chandra\
anaconda3\lib\site-packages (from pycaret) (0.15)
Collecting plotly-resampler>=0.8.3.1 (from pycaret)
  Using cached plotly_resampler-0.10.0-py3-none-any.whl.metadata (13
kB)
Requirement already satisfied: statsmodels>=0.12.1 in c:\users\
chandra\anaconda3\lib\site-packages (from pycaret) (0.14.2)
Collecting sktime==0.26.0 (from pycaret)
  Using cached sktime-0.26.0-py3-none-any.whl.metadata (29 kB)
Collecting tbats>=1.1.3 (from pycaret)
  Using cached tbats-1.1.3-py3-none-any.whl.metadata (3.8 kB)
Collecting pmdarima>=2.0.4 (from pycaret)
  Using cached pmdarima-2.0.4-cp312-cp312-win_amd64.whl.metadata (8.0
kB)
Requirement already satisfied: packaging in c:\users\chandra\appdata\
roaming\python\python312\site-packages (from sktime==0.26.0->pycaret)
(24.1)
Requirement already satisfied: scikit-base<0.8.0 in c:\users\chandra\
anaconda3\lib\site-packages (from sktime==0.26.0->pycaret) (0.7.8)
Requirement already satisfied: patsy>=0.5.1 in c:\users\chandra\
anaconda3\lib\site-packages (from category-encoders>=2.4.0->pycaret)
(0.5.6)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\
chandra\anaconda3\lib\site-packages (from imbalanced-learn>=0.12.0-
>pycaret) (3.5.0)
Requirement already satisfied: zipp>=0.5 in c:\users\chandra\
anaconda3\lib\site-packages (from importlib-metadata>=4.12.0->pycaret)
(3.17.0)
Requirement already satisfied: decorator in c:\users\chandra\appdata\
roaming\python\python312\site-packages (from ipython>=5.5.0->pycaret)
(5.1.1)
Requirement already satisfied: jedi>=0.16 in c:\users\chandra\appdata\
roaming\python\python312\site-packages (from ipython>=5.5.0->pycaret)
(0.19.1)
Requirement already satisfied: matplotlib-inline in c:\users\chandra\
appdata\roaming\python\python312\site-packages (from ipython>=5.5.0-
>pycaret) (0.1.7)
Requirement already satisfied: prompt-toolkit<3.1.0,>=3.0.41 in c:\
users\chandra\appdata\roaming\python\python312\site-packages (from
ipython>=5.5.0->pycaret) (3.0.47)
Requirement already satisfied: pygments>=2.4.0 in c:\users\chandra\
appdata\roaming\python\python312\site-packages (from ipython>=5.5.0-
>pycaret) (2.18.0)
```

```
Requirement already satisfied: stack-data in c:\users\chandra\appdata\
roaming\python\python312\site-packages (from ipython>=5.5.0->pycaret)
(0.6.3)
Requirement already satisfied: traitlets>=5.13.0 in c:\users\chandra\
appdata\roaming\python\python312\site-packages (from ipython>=5.5.0-
>pycaret) (5.14.3)
Requirement already satisfied: colorama in c:\users\chandra\appdata\
roaming\python\python312\site-packages (from ipython>=5.5.0->pycaret)
(0.4.6)
Requirement already satisfied: comm>=0.1.3 in c:\users\chandra\
appdata\roaming\python\python312\site-packages (from
ipywidgets>=7.6.5->pycaret) (0.2.2)
Requirement already satisfied: ipython-genutils~=0.2.0 in c:\users\
chandra\anaconda3\lib\site-packages (from ipywidgets>=7.6.5->pycaret)
(0.2.0)
Requirement already satisfied: widgetsnbextension~=3.6.6 in c:\users\
chandra\anaconda3\lib\site-packages (from ipywidgets>=7.6.5->pycaret)
(3.6.6)
Requirement already satisfied: jupyterlab-widgets<3,>=1.0.0 in c:\
users\chandra\anaconda3\lib\site-packages (from ipywidgets>=7.6.5-
>pycaret) (1.0.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\chandra\
anaconda3\lib\site-packages (from matplotlib<3.8.0->pycaret) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\chandra\
anaconda3\lib\site-packages (from matplotlib<3.8.0->pycaret) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\chandra\
anaconda3\lib\site-packages (from matplotlib<3.8.0->pycaret) (4.51.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\chandra\
anaconda3\lib\site-packages (from matplotlib<3.8.0->pycaret) (1.4.4)
Requirement already satisfied: pillow>=6.2.0 in c:\users\chandra\
anaconda3\lib\site-packages (from matplotlib<3.8.0->pycaret) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\chandra\
anaconda3\lib\site-packages (from matplotlib<3.8.0->pycaret) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\
chandra\appdata\roaming\python\python312\site-packages (from
matplotlib<3.8.0->pycaret) (2.9.0.post0)
Requirement already satisfied: fastjsonschema in c:\users\chandra\
anaconda3\lib\site-packages (from nbformat>=4.2.0->pycaret) (2.16.2)
Requirement already satisfied: jsonschema>=2.6 in c:\users\chandra\
anaconda3\lib\site-packages (from nbformat>=4.2.0->pycaret) (4.19.2)
Requirement already satisfied: jupyter-core in c:\users\chandra\
appdata\roaming\python\python312\site-packages (from nbformat>=4.2.0-
>pycaret) (5.7.2)
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in c:\users\
chandra\anaconda3\lib\site-packages (from numba>=0.55.0->pycaret)
(0.43.0)
Requirement already satisfied: pytz>=2020.1 in c:\users\chandra\
anaconda3\lib\site-packages (from pandas<2.2.0->pycaret) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\chandra\
```

```
anaconda3\lib\site-packages (from pandas<2.2.0->pycaret) (2023.3)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\chandra\
anaconda3\lib\site-packages (from plotly>=5.14.0->pycaret) (8.2.2)
Collecting dash>=2.9.0 (from plotly-resampler>=0.8.3.1->pycaret)
  Using cached dash-2.18.1-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: orjson<4.0.0,>=3.8.0 in c:\users\
chandra\anaconda3\lib\site-packages (from plotly-resampler>=0.8.3.1-
>pycaret) (3.10.7)
Requirement already satisfied: tsdownsample>=0.1.3 in c:\users\
chandra\anaconda3\lib\site-packages (from plotly-resampler>=0.8.3.1-
>pycaret) (0.1.3)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in c:\
users\chandra\anaconda3\lib\site-packages (from pmdarima>=2.0.4-
>pycaret) (3.0.11)
Requirement already satisfied: urllib3 in c:\users\chandra\anaconda3\
lib\site-packages (from pmdarima>=2.0.4->pycaret) (2.2.2)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in c:\
users\chandra\anaconda3\lib\site-packages (from pmdarima>=2.0.4-
>pycaret) (69.5.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\
chandra\anaconda3\lib\site-packages (from requests>=2.27.1->pycaret)
(2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\chandra\
anaconda3\lib\site-packages (from requests>=2.27.1->pycaret) (3.7)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\chandra\
anaconda3\lib\site-packages (from requests>=2.27.1->pycaret)
(2024.8.30)
Requirement already satisfied: Flask<3.1,>=1.0.4 in c:\users\chandra\
anaconda3\lib\site-packages (from dash>=2.9.0->plotly-
resampler>=0.8.3.1->pycaret) (3.0.3)
Requirement already satisfied: Werkzeug<3.1 in c:\users\chandra\
anaconda3\lib\site-packages (from dash>=2.9.0->plotly-
resampler>=0.8.3.1->pycaret) (3.0.3)
Requirement already satisfied: dash-html-components==2.0.0 in c:\
users\chandra\anaconda3\lib\site-packages (from dash>=2.9.0->plotly-
resampler>=0.8.3.1->pycaret) (2.0.0)
Requirement already satisfied: dash-core-components==2.0.0 in c:\
users\chandra\anaconda3\lib\site-packages (from dash>=2.9.0->plotly-
resampler>=0.8.3.1->pycaret) (2.0.0)
Requirement already satisfied: dash-table==5.0.0 in c:\users\chandra\
anaconda3\lib\site-packages (from dash>=2.9.0->plotly-
resampler>=0.8.3.1->pycaret) (5.0.0)
Requirement already satisfied: typing-extensions>=4.1.1 in c:\users\
chandra\anaconda3\lib\site-packages (from dash>=2.9.0->plotly-
resampler>=0.8.3.1->pycaret) (4.11.0)
Requirement already satisfied: retrying in c:\users\chandra\anaconda3\
lib\site-packages (from dash>=2.9.0->plotly-resampler>=0.8.3.1-
>pycaret) (1.3.4)
Requirement already satisfied: nest-asyncio in c:\users\chandra\
```

```
appdata\roaming\python\python312\site-packages (from dash>=2.9.0-
>plotly-resampler>=0.8.3.1->pycaret) (1.6.0)
Requirement already satisfied: parso<0.9.0,>=0.8.3 in c:\users\
chandra\appdata\roaming\python\python312\site-packages (from
jedi>=0.16->ipython>=5.5.0->pycaret) (0.8.4)
Requirement already satisfied: attrs>=22.2.0 in c:\users\chandra\
anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat>=4.2.0-
>pycaret) (23.1.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
c:\users\chandra\anaconda3\lib\site-packages (from jsonschema>=2.6-
>nbformat>=4.2.0->pycaret) (2023.7.1)
Requirement already satisfied: referencing>=0.28.4 in c:\users\
chandra\anaconda3\lib\site-packages (from jsonschema>=2.6-
>nbformat>=4.2.0->pycaret) (0.30.2)
Requirement already satisfied: rpds-py>=0.7.1 in c:\users\chandra\
anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat>=4.2.0-
>pycaret) (0.10.6)
Requirement already satisfied: six in c:\users\chandra\appdata\
roaming\python\python312\site-packages (from patsy>=0.5.1->category-
encoders>=2.4.0->pycaret) (1.16.0)
Requirement already satisfied: wcwidth in c:\users\chandra\appdata\
roaming\python\python312\site-packages (from prompt-
toolkit<3.1.0,>=3.0.41->ipython>=5.5.0->pycaret) (0.2.13)
Requirement already satisfied: notebook>=4.4.1 in c:\users\chandra\
anaconda3\lib\site-packages (from widgetsnbextension~=3.6.6-
>ipywidgets>=7.6.5->pycaret) (7.0.8)
Requirement already satisfied: platformdirs>=2.5 in c:\users\chandra\
anaconda3\lib\site-packages (from jupyter-core->nbformat>=4.2.0-
>pycaret) (3.10.0)
Requirement already satisfied: pywin32>=300 in c:\users\chandra\
appdata\roaming\python\python312\site-packages (from jupyter-core-
>nbformat>=4.2.0->pycaret) (306)
Requirement already satisfied: executing>=1.2.0 in c:\users\chandra\
appdata\roaming\python\python312\site-packages (from stack-data-
>ipython>=5.5.0->pycaret) (2.0.1)
Requirement already satisfied: asttokens>=2.1.0 in c:\users\chandra\
appdata\roaming\python\python312\site-packages (from stack-data-
>ipython>=5.5.0->pycaret) (2.4.1)
Requirement already satisfied: pure-eval in c:\users\chandra\appdata\
roaming\python\python312\site-packages (from stack-data-
>ipython>=5.5.0->pycaret) (0.2.3)
Requirement already satisfied: itsdangerous>=2.1.2 in c:\users\
chandra\anaconda3\lib\site-packages (from Flask<3.1,>=1.0.4-
>dash>=2.9.0->plotly-resampler>=0.8.3.1->pycaret) (2.2.0)
Requirement already satisfied: click>=8.1.3 in c:\users\chandra\
anaconda3\lib\site-packages (from Flask<3.1,>=1.0.4->dash>=2.9.0-
>plotly-resampler>=0.8.3.1->pycaret) (8.1.7)
Requirement already satisfied: blinker>=1.6.2 in c:\users\chandra\
anaconda3\lib\site-packages (from Flask<3.1,>=1.0.4->dash>=2.9.0-
```

```
>plotly-resampler>=0.8.3.1->pycaret) (1.6.2)
Requirement already satisfied: jupyter-server<3,>=2.4.0 in c:\users\
chandra\anaconda3\lib\site-packages (from notebook>=4.4.1-
>widgetsnbextension~=3.6.6->ipywidgets>=7.6.5->pycaret) (2.14.1)
Requirement already satisfied: jupyterlab-server<3,>=2.22.1 in c:\
users\chandra\anaconda3\lib\site-packages (from notebook>=4.4.1-
>widgetsnbextension~=3.6.6->ipywidgets>=7.6.5->pycaret) (2.25.1)
Requirement already satisfied: jupyterlab<4.1,>=4.0.2 in c:\users\
chandra\anaconda3\lib\site-packages (from notebook>=4.4.1-
>widgetsnbextension~=3.6.6->ipywidgets>=7.6.5->pycaret) (4.0.11)
Requirement already satisfied: notebook-shim<0.3,>=0.2 in c:\users\
chandra\anaconda3\lib\site-packages (from notebook>=4.4.1-
>widgetsnbextension~=3.6.6->ipywidgets>=7.6.5->pycaret) (0.2.3)
Requirement already satisfied: tornado>=6.2.0 in c:\users\chandra\
appdata\roaming\python\python312\site-packages (from notebook>=4.4.1-
>widgetsnbextension~=3.6.6->ipywidgets>=7.6.5->pycaret) (6.4.1)
Requirement already satisfied: anyio>=3.1.0 in c:\users\chandra\
anaconda3\lib\site-packages (from jupyter-server<3,>=2.4.0-
>notebook>=4.4.1->widgetsnbextension~=3.6.6->ipywidgets>=7.6.5-
>pycaret) (4.2.0)
Requirement already satisfied: argon2-cffi>=21.1 in c:\users\chandra\
anaconda3\lib\site-packages (from jupyter-server<3,>=2.4.0-
>notebook>=4.4.1->widgetsnbextension~=3.6.6->ipywidgets>=7.6.5-
>pycaret) (21.3.0)
Requirement already satisfied: jupyter-client>=7.4.4 in c:\users\
chandra\appdata\roaming\python\python312\site-packages (from jupyter-
server<3,>=2.4.0->notebook>=4.4.1->widgetsnbextension~=3.6.6-
>ipywidgets>=7.6.5->pycaret) (8.6.2)
Requirement already satisfied: jupyter-events>=0.9.0 in c:\users\
chandra\anaconda3\lib\site-packages (from jupyter-server<3,>=2.4.0-
>notebook>=4.4.1->widgetsnbextension~=3.6.6->ipywidgets>=7.6.5-
>pycaret) (0.10.0)
Requirement already satisfied: jupyter-server-terminals>=0.4.4 in c:\
users\chandra\anaconda3\lib\site-packages (from jupyter-
server<3,>=2.4.0->notebook>=4.4.1->widgetsnbextension~=3.6.6-
>ipywidgets>=7.6.5->pycaret) (0.4.4)
Requirement already satisfied: nbconvert>=6.4.4 in c:\users\chandra\
anaconda3\lib\site-packages (from jupyter-server<3,>=2.4.0-
>notebook>=4.4.1->widgetsnbextension~=3.6.6->ipywidgets>=7.6.5-
>pycaret) (7.10.0)
Requirement already satisfied: overrides>=5.0 in c:\users\chandra\
anaconda3\lib\site-packages (from jupyter-server<3,>=2.4.0-
>notebook>=4.4.1->widgetsnbextension~=3.6.6->ipywidgets>=7.6.5-
>pycaret) (7.4.0)
Requirement already satisfied: prometheus-client>=0.9 in c:\users\
chandra\anaconda3\lib\site-packages (from jupyter-server<3,>=2.4.0-
>notebook>=4.4.1->widgetsnbextension~=3.6.6->ipywidgets>=7.6.5-
>pycaret) (0.14.1)
Requirement already satisfied: pywinpty>=2.0.1 in c:\users\chandra\
```

```
anaconda3\lib\site-packages (from jupyter-server<3,>=2.4.0-
>notebook>=4.4.1->widgetsnbextension~=3.6.6->ipywidgets>=7.6.5-
>pycaret) (2.0.10)
Requirement already satisfied: pyzmq>=24 in c:\users\chandra\appdata\
roaming\python\python312\site-packages (from jupyter-server<3,>=2.4.0-
>notebook>=4.4.1->widgetsnbextension~=3.6.6->ipywidgets>=7.6.5-
>pycaret) (26.2.0)
Requirement already satisfied: send2trash>=1.8.2 in c:\users\chandra\
anaconda3\lib\site-packages (from jupyter-server<3,>=2.4.0-
>notebook>=4.4.1->widgetsnbextension~=3.6.6->ipywidgets>=7.6.5-
>pycaret) (1.8.2)
Requirement already satisfied: terminado>=0.8.3 in c:\users\chandra\
anaconda3\lib\site-packages (from jupyter-server<3,>=2.4.0-
>notebook>=4.4.1->widgetsnbextension~=3.6.6->ipywidgets>=7.6.5-
>pycaret) (0.17.1)
Requirement already satisfied: websocket-client>=1.7 in c:\users\
chandra\anaconda3\lib\site-packages (from jupyter-server<3,>=2.4.0-
>notebook>=4.4.1->widgetsnbextension~=3.6.6->ipywidgets>=7.6.5-
>pycaret) (1.8.0)
Requirement already satisfied: async-lru>=1.0.0 in c:\users\chandra\
anaconda3\lib\site-packages (from jupyterlab<4.1,>=4.0.2-
>notebook>=4.4.1->widgetsnbextension~=3.6.6->ipywidgets>=7.6.5-
>pycaret) (2.0.4)
Requirement already satisfied: ipykernel in c:\users\chandra\appdata\
roaming\python\python312\site-packages (from jupyterlab<4.1,>=4.0.2-
>notebook>=4.4.1->widgetsnbextension~=3.6.6->ipywidgets>=7.6.5-
>pycaret) (6.29.5)
Requirement already satisfied: jupyter-lsp>=2.0.0 in c:\users\chandra\
anaconda3\lib\site-packages (from jupyterlab<4.1,>=4.0.2-
>notebook>=4.4.1->widgetsnbextension~=3.6.6->ipywidgets>=7.6.5-
>pycaret) (2.2.0)
Requirement already satisfied: babel>=2.10 in c:\users\chandra\
anaconda3\lib\site-packages (from jupyterlab-server<3,>=2.22.1-
>notebook>=4.4.1->widgetsnbextension~=3.6.6->ipywidgets>=7.6.5-
>pycaret) (2.11.0)
Requirement already satisfied: json5>=0.9.0 in c:\users\chandra\
anaconda3\lib\site-packages (from jupyterlab-server<3,>=2.22.1-
>notebook>=4.4.1->widgetsnbextension~=3.6.6->ipywidgets>=7.6.5-
>pycaret) (0.9.6)
Requirement already satisfied: sniffio>=1.1 in c:\users\chandra\
anaconda3\lib\site-packages (from anyio>=3.1.0->jupyter-
server<3,>=2.4.0->notebook>=4.4.1->widgetsnbextension~=3.6.6-
>ipywidgets>=7.6.5->pycaret) (1.3.0)
Requirement already satisfied: argon2-cffi-bindings in c:\users\
chandra\anaconda3\lib\site-packages (from argon2-cffi>=21.1->jupyter-
server<3,>=2.4.0->notebook>=4.4.1->widgetsnbextension~=3.6.6-
>ipywidgets>=7.6.5->pycaret) (21.2.0)
Requirement already satisfied: python-json-logger>=2.0.4 in c:\users\
chandra\anaconda3\lib\site-packages (from jupyter-events>=0.9.0-
```

```
>jupyter-server<3,>=2.4.0->notebook>=4.4.1->widgetsnbextension~=3.6.6-
>ipywidgets>=7.6.5->pycaret) (2.0.7)
Requirement already satisfied: pyyaml>=5.3 in c:\users\chandra\
anaconda3\lib\site-packages (from jupyter-events>=0.9.0->jupyter-
server<3,>=2.4.0->notebook>=4.4.1->widgetsnbextension~=3.6.6-
>ipywidgets>=7.6.5->pycaret) (6.0.1)
Requirement already satisfied: rfc3339-validator in c:\users\chandra\
anaconda3\lib\site-packages (from jupyter-events>=0.9.0->jupyter-
server<3,>=2.4.0->notebook>=4.4.1->widgetsnbextension~=3.6.6-
>ipywidgets>=7.6.5->pycaret) (0.1.4)
Requirement already satisfied: rfc3986-validator>=0.1.1 in c:\users\
chandra\anaconda3\lib\site-packages (from jupyter-events>=0.9.0-
>jupyter-server<3,>=2.4.0->notebook>=4.4.1->widgetsnbextension~=3.6.6-
>ipywidgets>=7.6.5->pycaret) (0.1.1)
Requirement already satisfied: beautifulsoup4 in c:\users\chandra\
anaconda3\lib\site-packages (from nbconvert>=6.4.4->jupyter-
server<3,>=2.4.0->notebook>=4.4.1->widgetsnbextension~=3.6.6-
>ipywidgets>=7.6.5->pycaret) (4.12.3)
Requirement already satisfied: bleach!=5.0.0 in c:\users\chandra\
anaconda3\lib\site-packages (from nbconvert>=6.4.4->jupyter-
server<3,>=2.4.0->notebook>=4.4.1->widgetsnbextension~=3.6.6-
>ipywidgets>=7.6.5->pycaret) (4.1.0)
Requirement already satisfied: defusedxml in c:\users\chandra\
anaconda3\lib\site-packages (from nbconvert>=6.4.4->jupyter-
server<3,>=2.4.0->notebook>=4.4.1->widgetsnbextension~=3.6.6-
>ipywidgets>=7.6.5->pycaret) (0.7.1)
Requirement already satisfied: jupyterlab-pygments in c:\users\
chandra\anaconda3\lib\site-packages (from nbconvert>=6.4.4->jupyter-
server<3,>=2.4.0->notebook>=4.4.1->widgetsnbextension~=3.6.6-
>ipywidgets>=7.6.5->pycaret) (0.1.2)
Requirement already satisfied: mistune<4,>=2.0.3 in c:\users\chandra\
anaconda3\lib\site-packages (from nbconvert>=6.4.4->jupyter-
server<3,>=2.4.0->notebook>=4.4.1->widgetsnbextension~=3.6.6-
>ipywidgets>=7.6.5->pycaret) (2.0.4)
Requirement already satisfied: nbclient>=0.5.0 in c:\users\chandra\
anaconda3\lib\site-packages (from nbconvert>=6.4.4->jupyter-
server<3,>=2.4.0->notebook>=4.4.1->widgetsnbextension~=3.6.6-
>ipywidgets>=7.6.5->pycaret) (0.8.0)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\
chandra\anaconda3\lib\site-packages (from nbconvert>=6.4.4->jupyter-
server<3,>=2.4.0->notebook>=4.4.1->widgetsnbextension~=3.6.6-
>ipywidgets>=7.6.5->pycaret) (1.5.0)
Requirement already satisfied: tinycss2 in c:\users\chandra\anaconda3\
lib\site-packages (from nbconvert>=6.4.4->jupyter-server<3,>=2.4.0-
>notebook>=4.4.1->widgetsnbextension~=3.6.6->ipywidgets>=7.6.5-
>pycaret) (1.2.1)
Requirement already satisfied: debugpy>=1.6.5 in c:\users\chandra\
appdata\roaming\python\python312\site-packages (from ipykernel-
>jupyterlab<4.1,>=4.0.2->notebook>=4.4.1->widgetsnbextension~=3.6.6-
```

```
>ipywidgets>=7.6.5->pycaret) (1.8.5)
Requirement already satisfied: webencodings in c:\users\chandra\
anaconda3\lib\site-packages (from bleach!=5.0.0->nbconvert>=6.4.4-
>jupyter-server<3,>=2.4.0->notebook>=4.4.1->widgetsnbextension~=3.6.6-
>ipywidgets>=7.6.5->pycaret) (0.5.1)
Requirement already satisfied: fqdn in c:\users\chandra\anaconda3\lib\
site-packages (from jsonschema[format-nongpl]>=4.18.0->jupyter-
events>=0.9.0->jupyter-server<3,>=2.4.0->notebook>=4.4.1-
>widgetsnbextension~=3.6.6->ipywidgets>=7.6.5->pycaret) (1.5.1)
Collecting isoduration (from jsonschema[format-nongpl]>=4.18.0-
>jupyter-events>=0.9.0->jupyter-server<3,>=2.4.0->notebook>=4.4.1-
>widgetsnbextension~=3.6.6->ipywidgets>=7.6.5->pycaret)
  Using cached isoduration-20.11.0-py3-none-any.whl.metadata (5.7 kB)
Requirement already satisfied: jsonpointer>1.13 in c:\users\chandra\
anaconda3\lib\site-packages (from jsonschema[format-nongpl]>=4.18.0-
>jupyter-events>=0.9.0->jupyter-server<3,>=2.4.0->notebook>=4.4.1-
>widgetsnbextension~=3.6.6->ipywidgets>=7.6.5->pycaret) (2.1)
Requirement already satisfied: uri-template in c:\users\chandra\
anaconda3\lib\site-packages (from jsonschema[format-nongpl]>=4.18.0-
>jupyter-events>=0.9.0->jupyter-server<3,>=2.4.0->notebook>=4.4.1-
>widgetsnbextension~=3.6.6->ipywidgets>=7.6.5->pycaret) (1.3.0)
Requirement already satisfied: webcolors>=1.11 in c:\users\chandra\
anaconda3\lib\site-packages (from jsonschema[format-nongpl]>=4.18.0-
>jupyter-events>=0.9.0->jupyter-server<3,>=2.4.0->notebook>=4.4.1-
>widgetsnbextension~=3.6.6->ipywidgets>=7.6.5->pycaret) (24.8.0)
Requirement already satisfied: cffi>=1.0.1 in c:\users\chandra\
anaconda3\lib\site-packages (from argon2-cffi-bindings->argon2-
cffi>=21.1->jupyter-server<3,>=2.4.0->notebook>=4.4.1-
>widgetsnbextension~=3.6.6->ipywidgets>=7.6.5->pycaret) (1.16.0)
Requirement already satisfied: soupsieve>1.2 in c:\users\chandra\
anaconda3\lib\site-packages (from beautifulsoup4->nbconvert>=6.4.4-
>jupyter-server<3,>=2.4.0->notebook>=4.4.1->widgetsnbextension~=3.6.6-
>ipywidgets>=7.6.5->pycaret) (2.5)
Requirement already satisfied: pycparser in c:\users\chandra\
anaconda3\lib\site-packages (from cffi>=1.0.1->argon2-cffi-bindings-
>argon2-cffi>=21.1->jupyter-server<3,>=2.4.0->notebook>=4.4.1-
>widgetsnbextension~=3.6.6->ipywidgets>=7.6.5->pycaret) (2.21)
Requirement already satisfied: arrow>=0.15.0 in c:\users\chandra\
anaconda3\lib\site-packages (from isoduration->jsonschema[format-
nongpl]>=4.18.0->jupyter-events>=0.9.0->jupyter-server<3,>=2.4.0-
>notebook>=4.4.1->widgetsnbextension~=3.6.6->ipywidgets>=7.6.5-
>pycaret) (1.2.3)
Using cached pycaret-3.3.2-py3-none-any.whl (486 kB)
Using cached sktime-0.26.0-py3-none-any.whl (21.8 MB)
Using cached category_encoders-2.6.3-py2.py3-none-any.whl (81 kB)
Using cached lightgbm-4.5.0-py3-none-win_amd64.whl (1.4 MB)
Using cached matplotlib-3.7.5-cp312-cp312-win_amd64.whl (7.5 MB)
Using cached plotly_resampler-0.10.0-py3-none-any.whl (80 kB)
Using cached pmdarima-2.0.4-cp312-cp312-win_amd64.whl (625 kB)
Using cached tbats-1.1.3-py3-none-any.whl (44 kB)
```

```
Using cached yellowbrick-1.5-py3-none-any.whl (282 kB)
Using cached dash-2.18.1-py3-none-any.whl (7.5 MB)
Using cached isoduration-20.11.0-py3-none-any.whl (11 kB)
Installing collected packages: matplotlib, lightgbm, yellowbrick,
sktime, isoduration, dash, pmdarima, plotly-resampler, category-
encoders, tbats, pycaret

WARNING: Ignoring invalid distribution ~atplotlib (C:\Users\Chandra\
anaconda3\Lib\site-packages)
WARNING: Ignoring invalid distribution ~atplotlib (C:\Users\Chandra\
anaconda3\Lib\site-packages)
ERROR: Could not install packages due to an OSError: [WinError 32] The
process cannot access the file because it is being used by another
process: 'C:\\Users\\Chandra\\anaconda3\\Lib\\site-packages\\
matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSans.ttf'
Consider using the `--user` option or check the permissions.
```

Data science automation

This week is all about looking at automation tehcniques for data science and with Python. We can automate a lot of things with Python: collecting data, processing it, cleaning it, and many other parts of the data science pipeline. Here, we will show how to:

- use the pycaret autoML Python package to find an optimized ML model for our diabetes dataset
- create a Python script to ingest new data and make predictions on it

Often, next steps in fully operationalizing an ML pipeline like this are to use a cloud service to scale and serve our ML algorithm. We can use things like AWS lambda, GCP, AWS, or Azure ML depolyment with tools such as docker and kubernetes.

Data Preparation

It includes pulling the dataset from the system and we are going to load our same prepared data from week 2 where everything has been converted to numbers.

```
import pandas as pd
from pycaret.classification import *

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

```
            tenure  PhoneService  Contract  PaymentMethod
MonthlyCharges  \
customerID
```

```
7590-VHVEG     1.0              0         0               3
29.85
5575-GNVDE    34.0              1         1               2
56.95
3668-QPYBK     2.0              1         0               2
53.85
7795-CFOCW    45.0              0         1               1
42.30
9237-HQITU     2.0              1         0               3
70.70
...            ...            ...       ...             ...
...
6840-RESVB    24.0              1         1               2
84.80
2234-XADUH    72.0              1         1               0
103.20
4801-JZAZL    11.0              0         0               3
29.60
8361-LTMKD     4.0              1         0               2
74.40
3186-AJIEK    66.0              1         2               1
105.65

            TotalCharges  Churn  charge_per_tenure
customerID
7590-VHVEG         29.85      0          29.850000
5575-GNVDE       1889.50      0          55.573529
3668-QPYBK        108.15      1          54.075000
7795-CFOCW       1840.75      0          40.905556
9237-HQITU        151.65      1          75.825000
...                  ...    ...               ...
6840-RESVB       1990.50      0          82.937500
2234-XADUH       7362.90      0         102.262500
4801-JZAZL        346.45      0          31.495455
8361-LTMKD        306.60      1          76.650000
3186-AJIEK       6844.50      0         103.704545

[7043 rows x 8 columns]

df.head()

            tenure  PhoneService  Contract  PaymentMethod
MonthlyCharges  \
customerID

7590-VHVEG     1.0              0         0               3
29.85
5575-GNVDE    34.0              1         1               2
56.95
```

```
3668-QPYBK     2.0              1         0               2
53.85
7795-CFOCW    45.0              0         1               1
42.30
9237-HQITU     2.0              1         0               3
70.70

            TotalCharges  Churn  charge_per_tenure
customerID
7590-VHVEG         29.85      0          29.850000
5575-GNVDE       1889.50      0          55.573529
3668-QPYBK        108.15      1          54.075000
7795-CFOCW       1840.75      0          40.905556
9237-HQITU        151.65      1          75.825000
```

```
df.tail()
```

```
            tenure  PhoneService  Contract  PaymentMethod
MonthlyCharges  \
customerID

6840-RESVB    24.0              1         1               2
84.80
2234-XADUH    72.0              1         1               0
103.20
4801-JZAZL    11.0              0         0               3
29.60
8361-LTMKD     4.0              1         0               2
74.40
3186-AJIEK    66.0              1         2               1
105.65

            TotalCharges  Churn  charge_per_tenure
customerID
6840-RESVB       1990.50      0          82.937500
2234-XADUH       7362.90      0         102.262500
4801-JZAZL        346.45      0          31.495455
8361-LTMKD        306.60      1          76.650000
3186-AJIEK       6844.50      0         103.704545
```

```
from pycaret.classification import *

df.columns
```

```
Index(['tenure', 'PhoneService', 'Contract', 'PaymentMethod',
'MonthlyCharges',
       'TotalCharges', 'Churn', 'charge_per_tenure'],
      dtype='object')
```

```
?setup
```

```
Signature:
setup(
    data: Union[dict, list, tuple, numpy.ndarray,
scipy.sparse._matrix.spmatrix, pandas.core.frame.DataFrame, NoneType]
= None,
    data_func: Optional[Callable[[], Union[dict, list, tuple,
numpy.ndarray, scipy.sparse._matrix.spmatrix,
pandas.core.frame.DataFrame]]] = None,
    target: Union[int, str, list, tuple, numpy.ndarray,
pandas.core.series.Series] = -1,
    index: Union[bool, int, str, list, tuple, numpy.ndarray,
pandas.core.series.Series] = True,
    train_size: float = 0.7,
    test_data: Union[dict, list, tuple, numpy.ndarray,
scipy.sparse._matrix.spmatrix, pandas.core.frame.DataFrame, NoneType]
= None,
    ordinal_features: Optional[Dict[str, list]] = None,
    numeric_features: Optional[List[str]] = None,
    categorical_features: Optional[List[str]] = None,
    date_features: Optional[List[str]] = None,
    text_features: Optional[List[str]] = None,
    ignore_features: Optional[List[str]] = None,
    keep_features: Optional[List[str]] = None,
    preprocess: bool = True,
    create_date_columns: List[str] = ['day', 'month', 'year'],
    imputation_type: Optional[str] = 'simple',
    numeric_imputation: Union[int, float, str] = 'mean',
    categorical_imputation: str = 'mode',
    iterative_imputation_iters: int = 5,
    numeric_iterative_imputer: Union[str, Any] = 'lightgbm',
    categorical_iterative_imputer: Union[str, Any] = 'lightgbm',
    text_features_method: str = 'tf-idf',
    max_encoding_ohe: int = 25,
    encoding_method: Optional[Any] = None,
    rare_to_value: Optional[float] = None,
    rare_value: str = 'rare',
    polynomial_features: bool = False,
    polynomial_degree: int = 2,
    low_variance_threshold: Optional[float] = None,
    group_features: Optional[dict] = None,
    drop_groups: bool = False,
    remove_multicollinearity: bool = False,
    multicollinearity_threshold: float = 0.9,
    bin_numeric_features: Optional[List[str]] = None,
    remove_outliers: bool = False,
    outliers_method: str = 'iforest',
    outliers_threshold: float = 0.05,
    fix_imbalance: bool = False,
    fix_imbalance_method: Union[str, Any] = 'SMOTE',
    transformation: bool = False,
```

```
    transformation_method: str = 'yeo-johnson',
    normalize: bool = False,
    normalize_method: str = 'zscore',
    pca: bool = False,
    pca_method: str = 'linear',
    pca_components: Union[int, float, str, NoneType] = None,
    feature_selection: bool = False,
    feature_selection_method: str = 'classic',
    feature_selection_estimator: Union[str, Any] = 'lightgbm',
    n_features_to_select: Union[int, float] = 0.2,
    custom_pipeline: Optional[Any] = None,
    custom_pipeline_position: int = -1,
    data_split_shuffle: bool = True,
    data_split_stratify: Union[bool, List[str]] = True,
    fold_strategy: Union[str, Any] = 'stratifiedkfold',
    fold: int = 10,
    fold_shuffle: bool = False,
    fold_groups: Union[str, pandas.core.frame.DataFrame, NoneType] =
None,
    n_jobs: Optional[int] = -1,
    use_gpu: bool = False,
    html: bool = True,
    session_id: Optional[int] = None,
    system_log: Union[bool, str, logging.Logger] = True,
    log_experiment: Union[bool, str,
pycaret.loggers.base_logger.BaseLogger, List[Union[str,
pycaret.loggers.base_logger.BaseLogger]]] = False,
    experiment_name: Optional[str] = None,
    experiment_custom_tags: Optional[Dict[str, Any]] = None,
    log_plots: Union[bool, list] = False,
    log_profile: bool = False,
    log_data: bool = False,
    verbose: bool = True,
    memory: Union[bool, str, joblib.memory.Memory] = True,
    profile: bool = False,
    profile_kwargs: Optional[Dict[str, Any]] = None,
)
Docstring:
This function initializes the training environment and creates the
transformation
pipeline. Setup function must be called before executing any other
function. It takes
two mandatory parameters: ``data`` and ``target``. All the other
parameters are
optional.

Example
-------
>>> from pycaret.datasets import get_data
>>> juice = get_data('juice')
```

```
>>> from pycaret.classification import *
>>> exp_name = setup(data = juice,  target = 'Purchase')


data: dataframe-like = None
    Data set with shape (n_samples, n_features), where n_samples is
the
    number of samples and n_features is the number of features. If
data
    is not a pandas dataframe, it's converted to one using default
column
    names.


data_func: Callable[[], DATAFRAME_LIKE] = None
    The function that generate ``data`` (the dataframe-like input).
This
    is useful when the dataset is large, and you need parallel
operations
    such as ``compare_models``. It can avoid broadcasting large
dataset
    from driver to workers. Notice one and only one of ``data`` and
    ``data_func`` must be set.


target: int, str or sequence, default = -1
    If int or str, respectivcely index or name of the target column in
data.
    The default value selects the last column in the dataset. If
sequence,
    it should have shape (n_samples,). The target can be either binary
or
    multiclass.


index: bool, int, str or sequence, default = True
    Handle indices in the `data` dataframe.
        - If False: Reset to RangeIndex.
        - If True: Keep the provided index.
        - If int: Position of the column to use as index.
        - If str: Name of the column to use as index.
        - If sequence: Array with shape=(n_samples,) to use as index.


train_size: float, default = 0.7
    Proportion of the dataset to be used for training and validation.
Should be
    between 0.0 and 1.0.


test_data: dataframe-like or None, default = None
```

```
    If not None, test_data is used as a hold-out set and `train_size` parameter
    is ignored. The columns of data and test_data must match.


ordinal_features: dict, default = None
    Categorical features to be encoded ordinally. For example, a categorical
    feature with 'low', 'medium', 'high' values where low < medium < high can
    be passed as ordinal_features = {'column_name' : ['low', 'medium', 'high']}.


numeric_features: list of str, default = None
    If the inferred data types are not correct, the numeric_features param can
    be used to define the data types. It takes a list of strings with column
    names that are numeric.


categorical_features: list of str, default = None
    If the inferred data types are not correct, the categorical_features param
    can be used to define the data types. It takes a list of strings with column
    names that are categorical.


date_features: list of str, default = None
    If the inferred data types are not correct, the date_features param can be
    used to overwrite the data types. It takes a list of strings with column
    names that are DateTime.


text_features: list of str, default = None
    Column names that contain a text corpus. If None, no text features are
    selected.


ignore_features: list of str, default = None
    ignore_features param can be used to ignore features during preprocessing
    and model training. It takes a list of strings with column names that are
    to be ignored.
```

```
keep_features: list of str, default = None
    keep_features param can be used to always keep specific features
during
    preprocessing, i.e. these features are never dropped by any kind
of
    feature selection. It takes a list of strings with column names
that are
    to be kept.


preprocess: bool, default = True
    When set to False, no transformations are applied except for
train_test_split
    and custom transformations passed in ``custom_pipeline`` param.
Data must be
    ready for modeling (no missing values, no dates, categorical data
encoding),
    when preprocess is set to False.


create_date_columns: list of str, default = ["day", "month", "year"]
    Columns to create from the date features. Note that created
features
    with zero variance (e.g. the feature hour in a column that only
contains
    dates) are ignored. Allowed values are datetime attributes from
    `pandas.Series.dt`. The datetime format of the feature is inferred
    automatically from the first non NaN value.


imputation_type: str or None, default = 'simple'
    The type of imputation to use. Can be either 'simple' or
'iterative'.
    If None, no imputation of missing values is performed.


numeric_imputation: int, float or str, default = 'mean'
    Imputing strategy for numerical columns. Ignored when
``imputation_type=
    iterative``. Choose from:
        - "drop": Drop rows containing missing values.
        - "mean": Impute with mean of column.
        - "median": Impute with median of column.
        - "mode": Impute with most frequent value.
        - "knn": Impute using a K-Nearest Neighbors approach.
        - int or float: Impute with provided numerical value.


categorical_imputation: str, default = 'mode'
    Imputing strategy for categorical columns. Ignored when
```

```
``imputation_type=
    iterative``. Choose from:
        - "drop": Drop rows containing missing values.
        - "mode": Impute with most frequent value.
        - str: Impute with provided string.


iterative_imputation_iters: int, default = 5
    Number of iterations. Ignored when ``imputation_type=simple``.


numeric_iterative_imputer: str or sklearn estimator, default =
'lightgbm'
    Regressor for iterative imputation of missing values in numeric
features.
    If None, it uses LGBClassifier. Ignored when
``imputation_type=simple``.


categorical_iterative_imputer: str or sklearn estimator, default =
'lightgbm'
    Regressor for iterative imputation of missing values in
categorical features.
    If None, it uses LGBClassifier. Ignored when
``imputation_type=simple``.


text_features_method: str, default = "tf-idf"
    Method with which to embed the text features in the dataset.
Choose
    between "bow" (Bag of Words - CountVectorizer) or "tf-idf"
(TfidfVectorizer).
    Be aware that the sparse matrix output of the transformer is
converted
    internally to its full array. This can cause memory issues for
large
    text embeddings.


max_encoding_ohe: int, default = 25
    Categorical columns with `max_encoding_ohe` or less unique values
are
    encoded using OneHotEncoding. If more, the `encoding_method`
estimator
    is used. Note that columns with exactly two classes are always
encoded
    ordinally. Set to below 0 to always use OneHotEncoding.


encoding_method: category-encoders estimator, default = None
```

```
    A `category-encoders` estimator to encode the categorical columns
    with more than `max_encoding_ohe` unique values. If None,
    `category_encoders.target_encoder.TargetEncoder` is used.


rare_to_value: float or None, default=None
    Minimum fraction of category occurrences in a categorical column.
    If a category is less frequent than `rare_to_value * len(X)`, it
is
    replaced with the string in `rare_value`. Use this parameter to
group
    rare categories before encoding the column. If None, ignores this
step.


rare_value: str, default="rare"
    Value with which to replace rare categories. Ignored when
    ``rare_to_value`` is None.


polynomial_features: bool, default = False
    When set to True, new features are derived using existing numeric
features.


polynomial_degree: int, default = 2
    Degree of polynomial features. For example, if an input sample is
two dimensional
    and of the form [a, b], the polynomial features with degree = 2
are:
    [1, a, b, a^2, ab, b^2]. Ignored when ``polynomial_features`` is
not True.


low_variance_threshold: float or None, default = None
    Remove features with a training-set variance lower than the
provided
    threshold. If 0, keep all features with non-zero variance, i.e.
remove
    the features that have the same value in all samples. If None,
skip
    this transformation step.


group_features: dict or None, default = None
    When the dataset contains features with related characteristics,
    add new fetaures with the following statistical properties of that
    group: min, max, mean, std, median and mode. The parameter takes a
    dict with the group name as key and a list of feature names
    belonging to that group as value.
```

```
drop_groups: bool, default=False
    Whether to drop the original features in the group. Ignored when
    ``group_features`` is None.


remove_multicollinearity: bool, default = False
    When set to True, features with the inter-correlations higher than
    the defined threshold are removed. For each group, it removes all
    except the feature with the highest correlation to `y`.


multicollinearity_threshold: float, default = 0.9
    Minimum absolute Pearson correlation to identify correlated
    features. The default value removes equal columns. Ignored when
    ``remove_multicollinearity`` is not True.


bin_numeric_features: list of str, default = None
    To convert numeric features into categorical, bin_numeric_features
parameter can
    be used. It takes a list of strings with column names to be
discretized. It does
    so by using 'sturges' rule to determine the number of clusters and
then apply
    KMeans algorithm. Original values of the feature are then replaced
by the
    cluster label.


remove_outliers: bool, default = False
    When set to True, outliers from the training data are removed
using an
    Isolation Forest.


outliers_method: str, default = "iforest"
    Method with which to remove outliers. Ignored when
`remove_outliers=False`.
    Possible values are:
        - 'iforest': Uses sklearn's IsolationForest.
        - 'ee': Uses sklearn's EllipticEnvelope.
        - 'lof': Uses sklearn's LocalOutlierFactor.


outliers_threshold: float, default = 0.05
    The percentage of outliers to be removed from the dataset. Ignored
    when ``remove_outliers=False``.
```

```
fix_imbalance: bool, default = False
    When training dataset has unequal distribution of target class it
can be balanced
    using this parameter. When set to True, SMOTE (Synthetic Minority
Over-sampling
    Technique) is applied by default to create synthetic datapoints
for minority class.


fix_imbalance_method: str or imblearn estimator, default = "SMOTE"
    Estimator with which to perform class balancing. Choose from the
name
    of an `imblearn` estimator, or a custom instance of such. Ignored
when
    `fix_imbalance=False`.


transformation: bool, default = False
    When set to True, it applies the power transform to make data more
Gaussian-like.
    Type of transformation is defined by the ``transformation_method``
parameter.


transformation_method: str, default = 'yeo-johnson'
    Defines the method for transformation. By default, the
transformation method is
    set to 'yeo-johnson'. The other available option for
transformation is 'quantile'.
    Ignored when ``transformation`` is not True.


normalize: bool, default = False
    When set to True, it transforms the features by scaling them to a
given
    range. Type of scaling is defined by the ``normalize_method``
parameter.


normalize_method: str, default = 'zscore'
    Defines the method for scaling. By default, normalize method is
set to 'zscore'
    The standard zscore is calculated as z = (x - u) / s. Ignored when
``normalize``
    is not True. The other options are:

    - minmax: scales and translates each feature individually such
that it is in
      the range of 0 - 1.
    - maxabs: scales and translates each feature individually such
that the
```

```
      maximal absolute value of each feature will be 1.0. It does not
      shift/center the data, and thus does not destroy any sparsity.
    - robust: scales and translates each feature according to the
Interquartile
      range. When the dataset contains outliers, robust scaler often
gives
      better results.


pca: bool, default = False
    When set to True, dimensionality reduction is applied to project
the data into
    a lower dimensional space using the method defined in
``pca_method`` parameter.


pca_method: str, default = 'linear'
    Method with which to apply PCA. Possible values are:
        - 'linear': Uses Singular Value  Decomposition.
        - 'kernel': Dimensionality reduction through the use of RBF
kernel.
        - 'incremental': Similar to 'linear', but more efficient for
large datasets.


pca_components: int, float, str or None, default = None
    Number of components to keep. This parameter is ignored when
`pca=False`.
        - If None: All components are kept.
        - If int: Absolute number of components.
        - If float: Such an amount that the variance that needs to be
explained
                    is greater than the percentage specified by
`n_components`.
                    Value should lie between 0 and 1 (ony for
pca_method='linear').
        - If "mle": Minka’s MLE is used to guess the dimension (ony
for pca_method='linear').


feature_selection: bool, default = False
    When set to True, a subset of features is selected based on a
feature
    importance score determined by ``feature_selection_estimator``.


feature_selection_method: str, default = 'classic'
    Algorithm for feature selection. Choose from:
        - 'univariate': Uses sklearn's SelectKBest.
        - 'classic': Uses sklearn's SelectFromModel.
```

```
        - 'sequential': Uses sklearn's SequentialFeatureSelector.


feature_selection_estimator: str or sklearn estimator, default =
'lightgbm'
    Classifier used to determine the feature importances. The
    estimator should have a `feature_importances_` or `coef_`
    attribute after fitting. If None, it uses LGBClassifier. This
    parameter is ignored when `feature_selection_method=univariate`.


n_features_to_select: int or float, default = 0.2
    The maximum number of features to select with feature_selection.
If <1,
    it's the fraction of starting features. Note that this parameter
doesn't
    take features in ``ignore_features`` or ``keep_features`` into
account
    when counting.


custom_pipeline: list of (str, transformer), dict or Pipeline, default
= None
    Addidiotnal custom transformers. If passed, they are applied to
the
    pipeline last, after all the build-in transformers.


custom_pipeline_position: int, default = -1
    Position of the custom pipeline in the overal preprocessing
pipeline.
    The default value adds the custom pipeline last.


data_split_shuffle: bool, default = True
    When set to False, prevents shuffling of rows during
'train_test_split'.


data_split_stratify: bool or list, default = True
    Controls stratification during 'train_test_split'. When set to
True, will
    stratify by target column. To stratify on any other columns, pass
a list of
    column names. Ignored when ``data_split_shuffle`` is False.


fold_strategy: str or sklearn CV generator object, default =
'stratifiedkfold'
    Choice of cross validation strategy. Possible values are:
```

```
    * 'kfold'
    * 'stratifiedkfold'
    * 'groupkfold'
    * 'timeseries'
    * a custom CV generator object compatible with scikit-learn.

    For ``groupkfold``, column name must be passed in ``fold_groups``
parameter.
    Example: ``setup(fold_strategy="groupkfold",
fold_groups="COLUMN_NAME")``

fold: int, default = 10
    Number of folds to be used in cross validation. Must be at least
2. This is
    a global setting that can be over-written at function level by
using ``fold``
    parameter. Ignored when ``fold_strategy`` is a custom object.


fold_shuffle: bool, default = False
    Controls the shuffle parameter of CV. Only applicable when
``fold_strategy``
    is 'kfold' or 'stratifiedkfold'. Ignored when ``fold_strategy`` is
a custom
    object.


fold_groups: str or array-like, with shape (n_samples,), default =
None
    Optional group labels when 'GroupKFold' is used for the cross
validation.
    It takes an array with shape (n_samples, ) where n_samples is the
number
    of rows in the training dataset. When string is passed, it is
interpreted
    as the column name in the dataset containing group labels.


n_jobs: int, default = -1
    The number of jobs to run in parallel (for functions that supports
parallel
    processing) -1 means using all processors. To run all functions on
single
    processor set n_jobs to None.


use_gpu: bool or str, default = False
    When set to True, it will use GPU for training with algorithms
that support it,
    and fall back to CPU if they are unavailable. When set to 'force',
```

```
it will only
    use GPU-enabled algorithms and raise exceptions when they are
unavailable. When
    False, all algorithms are trained using CPU only.

    GPU enabled algorithms:

    - Extreme Gradient Boosting, requires no further installation

    - CatBoost Classifier, requires no further installation
      (GPU is only enabled when data > 50,000 rows)

    - Light Gradient Boosting Machine, requires GPU installation
      https://lightgbm.readthedocs.io/en/latest/GPU-Tutorial.html

    - Logistic Regression, Ridge Classifier, Random Forest, K
Neighbors Classifier,
      Support Vector Machine, requires cuML >= 0.15
      https://github.com/rapidsai/cuml


html: bool, default = True
    When set to False, prevents runtime display of monitor. This must
be set to False
    when the environment does not support IPython. For example,
command line terminal,
    Databricks Notebook, Spyder and other similar IDEs.


session_id: int, default = None
    Controls the randomness of experiment. It is equivalent to
'random_state' in
    scikit-learn. When None, a pseudo random number is generated. This
can be used
    for later reproducibility of the entire experiment.


log_experiment: bool or str or BaseLogger or list of str or
BaseLogger, default = False
    A (list of) PyCaret ``BaseLogger`` or str (one of 'mlflow',
'wandb', 'comet_ml')
    corresponding to a logger to determine which experiment loggers to
use.
    Setting to True will use just MLFlow.


system_log: bool or str or logging.Logger, default = True
    Whether to save the system logging file (as logs.log). If the
input
    is a string, use that as the path to the logging file. If the
```

```
input
    already is a logger object, use that one instead.


experiment_name: str, default = None
    Name of the experiment for logging. Ignored when
``log_experiment`` is False.


experiment_custom_tags: dict, default = None
    Dictionary of tag_name: String -> value: (String, but will be
string-ified
    if not) passed to the mlflow.set_tags to add new custom tags for
the experiment.


log_plots: bool or list, default = False
    When set to True, certain plots are logged automatically in the
``MLFlow`` server.
    To change the type of plots to be logged, pass a list containing
plot IDs. Refer
    to documentation of ``plot_model``. Ignored when
``log_experiment`` is False.


log_profile: bool, default = False
    When set to True, data profile is logged on the ``MLflow`` server
as a html file.
    Ignored when ``log_experiment`` is False.


log_data: bool, default = False
    When set to True, dataset is logged on the ``MLflow`` server as a
csv file.
    Ignored when ``log_experiment`` is False.


verbose: bool, default = True
    When set to False, Information grid is not printed.


memory: str, bool or Memory, default=True
    Used to cache the fitted transformers of the pipeline.
        If False: No caching is performed.
        If True: A default temp directory is used.
        If str: Path to the caching directory.


profile: bool, default = False
    When set to True, an interactive EDA report is displayed.
```

```
profile_kwargs: dict, default = {} (empty dict)
    Dictionary of arguments passed to the ProfileReport method used
    to create the EDA report. Ignored if ``profile`` is False.


Returns:
    ClassificationExperiment object.
File:      c:\users\chandra\anaconda3\envs\pycaret_env1\lib\site-
packages\pycaret\classification\functional.py
Type:      function
```

Use pycaret to find an ML algorithm that performs best on the data

```
# Set up the environment in PyCaret
automl = setup(data=df, target='Churn')
```

```
<pandas.io.formats.style.Styler at 0x22a0699bbb0>
```

INTERPRETATION:

Here, the preprocess is true it includes outliers treatment ,missing value treatment and feature engineering

```
best_model = compare_models()
```

```
<IPython.core.display.HTML object>
```

```
<pandas.io.formats.style.Styler at 0x22a0698ec80>
```

```
<IPython.core.display.HTML object>
```

```
best_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=531, solver='lbfgs', tol=0.0001,
verbose=0,
                   warm_start=False)
```

INTERPRETATION:

1. These are the best parameters for the model after hyperparameter tuning.

Evaluation metric you think is best to use for finding the best model

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

```
            tenure  PhoneService  Contract  PaymentMethod
MonthlyCharges  \
customerID

8361-LTMKD     4.0             1         0              2
74.4

            TotalCharges  Churn  charge_per_tenure
customerID
8361-LTMKD         306.6      1              76.65
```

We are selecting the last row, but using the indexing `[-2:-1]` to make it a 2D array instead of 1D (which throws an error). Try running `df.iloc[-1].shape` and `df.iloc[-2:-1].shape` to see how they differ.

However, this only works if we set `preprocess=False` in our setup function. Otherwise the order of features may be different

A more robust way (in case we are using preprocessing with autoML) is to use pycaret's predict_model function:

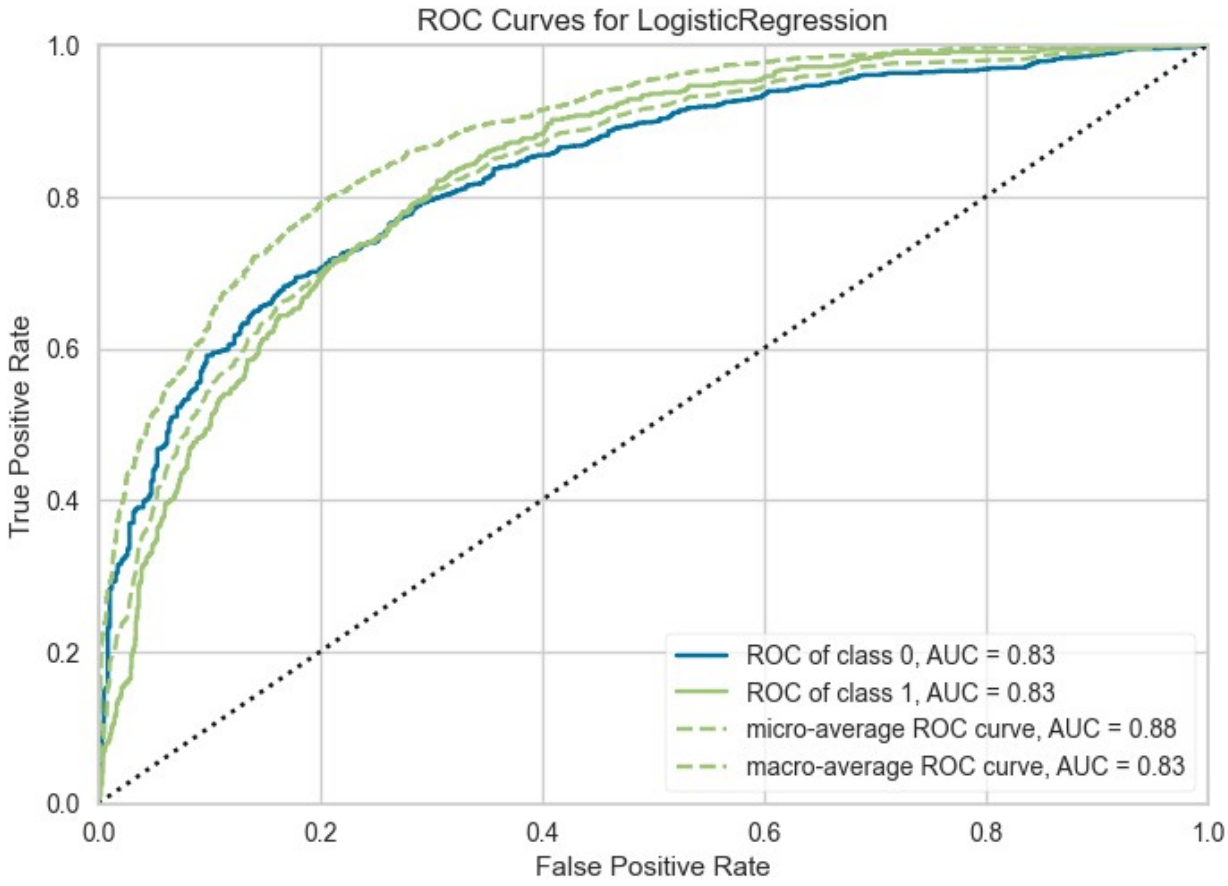
```
predict_model(best_model, df.iloc[-2:-1])
```

```
<pandas.io.formats.style.Styler at 0x22a0698c520>
```

```
            tenure  PhoneService  Contract  PaymentMethod
MonthlyCharges  \
customerID

8361-LTMKD     4.0             1         0              2
74.400002

            TotalCharges  charge_per_tenure  Churn
prediction_label  \
customerID

8361-LTMKD    306.600006          76.650002      1                   1


            prediction_score
customerID
8361-LTMKD            0.5662
```

```
plot_model(best_model, plot='auc')
```

```
<IPython.core.display.HTML object>
```

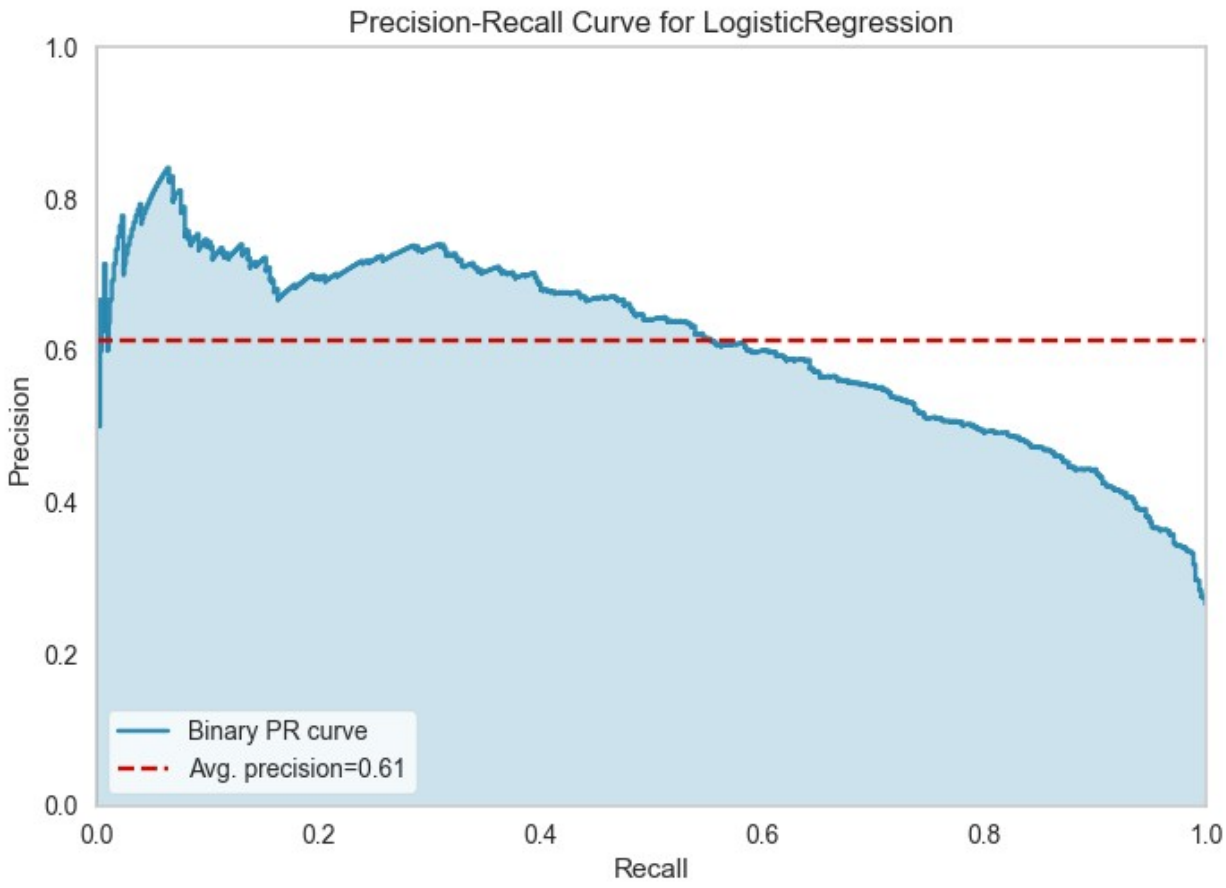


INTERPRETATIONS:

1. In this plot, The roc of class 0 and roc of class 1 both are positivily increasing against false positive rate and towards true positive rate which proves the efficiency of prediction.
2. the AUC values of class 0 and 1 are above 0.8 which indicates better discrimination performance of the model.

```
plot_model(best_model, plot='pr')
```

```
<IPython.core.display.HTML object>
```



INTERPRETATION:

1. The Precision-Recall (PR) curve is another evaluation metric used in binary classification tasks, particularly when dealing with imbalanced datasets.
2. The PR curve is a graphical representation of the trade-off between precision and recall for different threshold values used to classify instances as positive or negative.
3. the curve has downward trend i.e; with increase in Recall the Precision is decreasing rapidly.

SAVING AND LOADING MODEL

Next, we want to save our trained model so we can use it in a Python file

```
save_model(best_model, 'lr')

Transformation Pipeline and Model Successfully Saved

(Pipeline(memory=Memory(location=None),
          steps=[('numerical_imputer',
                  TransformerWrapper(exclude=None,
                                     include=['tenure',
```

```
'PhoneService',
                                                 'Contract',
'PaymentMethod',
                                                 'MonthlyCharges',
'TotalCharges',
                                                 'charge_per_tenure'],

transformer=SimpleImputer(add_indicator=False,

copy=True,

fill_value=None,

keep_empty_features=False,

missing_values=nan,

strategy='mean'))),
                 ('c...

fill_value=None,

keep_empty_features=False,

missing_values=nan,

strategy='most_frequent'))),
                 ('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=531,
                                     solver='lbfgs', tol=0.0001,
verbose=0,
                                     warm_start=False))],
          verbose=False),
 'lr.pkl')
```

Using `pickle` for Model Persistence

To save and load trained machine learning models in Python, we utilize the `pickle` module. This allows us to serialize the model object into a file, enabling easy storage and future reuse without needing to retrain the model. This approach supports efficient deployment and sharing of models across different environments.

```
import pickle
with open('lr.pkl', 'wb') as f:
    pickle.dump(best_model, f)

with open('lr.pkl', 'rb') as f:
    loaded_model = pickle.load(f)

loaded_lda = load_model('lr')
```

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

```
new_data=df.iloc[-2:-1]

predict_model(loaded_lda, new_data)
```

```
<pandas.io.formats.style.Styler at 0x22a7c750250>
```

```
            tenure  PhoneService  Contract  PaymentMethod
MonthlyCharges  \
customerID

8361-LTMKD     4.0             1         0              2
74.400002

            TotalCharges  charge_per_tenure  Churn
prediction_label  \
customerID

8361-LTMKD    306.600006          76.650002      1                   1


            prediction_score
customerID
8361-LTMKD            0.5662
```

Making a Python module to make predictions

```
from IPython.display import Code

Code(r"predict_churn_data.py")

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

model = load_model('lr')


def load_data(filepath):
    """
    Loading churn data into a DataFrame from a string filepath.
```

```
    """
    df = pd.read_csv(filepath, index_col='customerID')
    return df


def make_predictions(df, threshold=0.75):
    """
    Using the pycaret best model to make predictions on data in the df
dataframe.
    Rounds up to 1 if greater than or equal to the threshold.
    """
    predictions = predict_model(model, data=df)
    predictions['Churn_prediction'] = (predictions['prediction_score']
>= threshold)
    predictions['Churn_prediction'].replace({True: '0', False: '1'},
inplace=True)
    drop_cols = predictions.columns.tolist()
    drop_cols.remove('Churn_prediction')
    return predictions.drop(drop_cols, axis=1)


def calculate_accuracy(predicted_labels, true_labels):
    """
    Calculate the accuracy of predictions.
    """
    correct_predictions = [0 if pred == true else 1 for pred, true in
zip(predicted_labels, true_labels)]
    accuracy = sum(correct_predictions) / len(correct_predictions)
    return accuracy


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

    # True values for comparison
    true_values = [1, 0, 0, 1, 0]

    # Calculating accuracy
    predicted_labels = predictions['Churn_prediction'].tolist()
    accuracy = calculate_accuracy(predicted_labels, true_values)

    print(f"Prediction Accuracy: {accuracy:.2f}")
```

```
%run predict_churn_data.py
```

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

```
<IPython.core.display.HTML object>
```

```
predictions:
           Churn_prediction
customerID
9305-CKSKC                 1
1452-KNGVK                 0
6723-OKKJM                 0
7832-POPKP                 1
6348-TACGU                 0
Prediction Accuracy: 1.00
```

Summary

1. First, the data used to train and test the ML model is a telecommunication churn dataset with only numeric features.
2. The next is I have imported the Pycaret package which contains the 'automl' function which by default does the preprocessing, splitting into training-testing data, then loading every model with train-test data, also it performs hyperparameter tuning and gives us the best ML model with the best accuracy and required best parameters.
3. Using the 'automl' function information, we can load, predict data, and save the model using the respective functions defined in the 'pycaret' package.
4. To test the efficiency of the prediction I have created a Python module and interlinked it with the current directory or notebook using the 'code()' and 'run' functions from the 'IPython.display' package and for the data I have used new churn dataset with known values for prediction.
5. After predicting and checking efficiency I have saved and uploaded all the files used and created into the GITHUB.

Optional and Advanced Section

Churn Prediction and Analysis

```
# Import necessary libraries
import pandas as pd
import numpy as np
from pycaret.classification import *
from scipy.stats import percentileofscore

# Load the trained model
model = load_model('lr')  # Ensure your model named 'lr' exists

# Static file name for the input data
DATA_FILE_PATH = 'new_churn_data_unmodified.csv'
```

```
# Define a function to load data
def load_data(filepath):
    """
    Load churn data into a DataFrame from a string filepath.
    """
    df = pd.read_csv(filepath, index_col='customerID')
    return df

# Define a function to calculate percentiles
def calculate_percentile(prediction_score, prob_dist):
    """
    Calculate the percentile of the prediction score within the
distribution.
    """
    return percentileofscore(prob_dist, prediction_score)

# Preprocessing function to handle missing values and encode
categorical features
def preprocess_data(df):
    """
    Preprocess the DataFrame by handling missing values and encoding
categorical features.
    """
    print("Available columns in the DataFrame:", df.columns.tolist())

    # Convert 'TotalCharges' to numeric, handling errors
    df['TotalCharges'] = pd.to_numeric(df['TotalCharges'],
errors='coerce')

    # Fill missing values for numeric types
    df.fillna(df.mean(numeric_only=True), inplace=True)  # Mean
imputation for numeric columns

    # Create a simulated 'Churn' column for demonstration purposes.
    # Example Rule: If MonthlyCharges > 80, simulate churn as 1 (Yes),
or else 0 (No)
    df['Churn'] = (df['MonthlyCharges'] > 80).astype(int)  # Here '1'
represents churn and '0' represents no churn

    # Identify categorical columns
    categorical_cols = ['PhoneService', 'Contract', 'PaymentMethod']

    # One-hot encode categorical columns
    df = pd.get_dummies(df, columns=categorical_cols, drop_first=True)

    return df

# Define a class to encapsulate the prediction functions
class ChurnPredictor:
    def __init__(self):
```

```
        self.model = None

    def train_model(self, df):
        """
        Train the machine learning model using PyCaret.
        """
        print("Training the model...")
        setup(data=df, target='Churn', session_id=123)

        # Compare different models and select the best one based on
AUC
        self.model = compare_models(sort='AUC')

    def preprocess_and_predict(self, filepath):
        # Load and preprocess data
        df = load_data(filepath)
        df = preprocess_data(df)

        # Train the model
        self.train_model(df)

        # Make predictions
        if self.model:
            predictions = predict_model(self.model, data=df)

            # Calculate percentiles for the prediction score
            prob_distribution = predictions['prediction_score']
            predictions['Percentile'] =
predictions['prediction_score'].apply(
                lambda x: calculate_percentile(x, prob_distribution)
            )

            # Calculate prediction accuracy
            correct_predictions = (predictions['Label'] ==
df['Churn']).sum()  # Count correct predictions
            total_predictions = len(predictions)
            accuracy = correct_predictions / total_predictions  #
Calculate accuracy

            print(f"Prediction Accuracy: {accuracy:.2%}")  # Print
accuracy as percentage

            return predictions[['prediction_score', 'Percentile',
'Label']]  # Include the output columns

# Main execution block
if __name__ == "__main__":
    churn_predictor = ChurnPredictor()

    # Use the static file path for the dataset
```

```
    predictions =
churn_predictor.preprocess_and_predict(DATA_FILE_PATH)

    # Display predictions
    if predictions is not None:
        print("Predictions and Percentiles:")
        print(predictions)
```

```
Transformation Pipeline and Model Successfully Loaded
Available columns in the DataFrame: ['tenure', 'PhoneService',
'Contract', 'PaymentMethod', 'MonthlyCharges', 'TotalCharges']
Training the model...

<pandas.io.formats.style.Styler at 0x22a0c9316f0>

<IPython.core.display.HTML object>

<pandas.io.formats.style.Styler at 0x22a0c1d10c0>

<IPython.core.display.HTML object>
```