PROJECT SPECIFICATION

Predict Customer Churn with Clean Code

Code Quality

CRITERIA	MEETS SPECIFICATIONS

The code is following the PEP 8 coding style.	All the code written for this project should follow the PEP 8 guidelines. Objects have meaningful names and syntax. Code is properly commented and organized. Imports are correctly ordered.
	
	Running the below can assist with formatting.
	
	`autopep8 --in-place --aggressive --aggressive script.py`
	
	Then students should aim for a score exceeding 7 when using `pylint`
	
	`pylint script.py`

The README file provides an overview of the project, the instructions to use the code	The file contains a summary of the purpose and description of the project. Someone should be able to run the code by reading the README.
All functions and files have document strings.	All functions have a document string that correctly identifies the inputs, outputs, and purpose of the function. All files have a document string that identifies the purpose of the file, the author, and the date the file was created.

Testing & Logging

CRITERIA	MEETS SPECIFICATIONS
Tests written for each function.	Each function in `churn_script_logging_and_tests.py` is complete with tests for the input function.
Log for info and errors.	Each function in `churn_script_logging_and_tests.py` is complete with logging for if the function successfully passes the tests or errors.
Logs stored in a `.log` file.	All log information should be stored in a `.log` file, so it can be viewed post the run of the script.
Easy to understand error and info messages.	The log messages should easily be understood and traceable that appear in the `.log` file.
Test and logging can be completed on the command line.	The README should inform a user how they would test and log the result of each function. Something similar to the below should produce the `.log` file with the result from running all tests. `ipython churn_script_logging_and_tests_solution.py`

Save Images & Models

CRITERIA	MEETS SPECIFICATIONS
Store EDA plots.	Store result plots including at least one: 1. Univariate, quantitative plot 2. Univariate, categorical plot 3. Bivariate plot
Store result plots.	Store result plots including: 1. ROC curves 2. Feature Importances

CRITERIA	MEETS SPECIFICATIONS
Store model objects that can easily be loaded and used in a production environment.	Store at least two models. Recommended using `joblib` and storing models with `.pkl` extension.

Problem Solving

CRITERIA	MEETS SPECIFICATIONS
Code completes the process for solving the data science process.	Code in `churn_library.py` completes the process for solving the data science process including: 1. EDA 2. Feature Engineering (including encoding of categorical variables) 3. Model Training 4. Prediction 5. Model Evaluation
Handle categorical columns.	Use one-hot encoding or mean of the response to fill in categorical columns. Currently, the notebook does this in an inefficient way that can be refactored by looping. Make this code more efficient using the same method as in the notebook or using one-hot encoding. Tip: Creating a list of categorical column names can help with looping through these items and create an easier way to extend this logic.

Suggestions to Make Your Project Stand Out!

- Re-organize each script to work as a class.
- Update functions to move constants to their own `constants.py` file, which can then be passed to the necessary functions, rather than being created as variables within functions.
- Work towards pylint score of 10/10.
- Add dependencies and libraries (or dockerfile) to README.md
- Add requirements.txt with needed dependencies and libraries for simple install.