

[< Return to Classroom](#)

Predict Customer Churn with Clean Code



REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Hello Udacity Student ,

Let me start by congratulating you on the work done . Your submission meets all of our specifications in one shot. You could implement your learnings to identify credit card customers that are most likely to churn. Your project includes a Python Package in which your code follows the coding (PEP8) and engineering best practices for implementing software (modular, documented, and tested). Your package also has the flexibility of being run interactively or from the command-line interface (CLI). The Machine learning part of the project is executed flawlessly. I hope the project could give you practice using your skills for testing, logging, and best coding practices from this lesson. Through it, you were also introduced to a problem data scientists across companies face all the time. Continue the great work and good luck with future projects. 

Extra materials

The following documentation on the topic could help you for further learning. Please have a look at them in your free time.

- [Predict Customer Churn in Python](#)
- [Hands-on: Predict Customer Churn](#)
- [Predict Customer Churn Using Python & Machine Learning](#)

#StaySafe! 🙏

Code Quality

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
```

Awesome, your code is well optimized with an intuitive and easy-to-follow structure that follows PEP8 style guidelines 👍

- Running `pylint` on `churn_library.py` and `churn_script_logging_and_tests.py` produce scores over the threshold of `7`. Nice job with this section. 🙌

Suggestions

- [PEP-8 Tutorial: Code Standards in Python](#)
- [Clean Code Syntax for Python: Introduction to PEP 8 Style Guide: PEP 8 Naming Conventions](#)

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.

Nice effort, your project contains a README that describes your project. You have provided the project summary and instructions on how to run your project in your README.

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.

Excellent work! All functions have a document string that correctly identifies the inputs, outputs, and purpose of the function. Also, all your files have a document string that identifies the purpose of the file, the author, and the date the file was created. 🙌🙌

Suggestions

- [PEP 8: Style Guide for Python Code](#)
- [How to Write Beautiful Python Code With PEP 8](#)
- [PEP 257 – Docstring Conventions](#)

Testing & Logging

Each function in `churn_script_logging_and_tests.py` is complete with tests for the input function.

Good job, the logging script has been well completed and tests all functions. I see you made use of assert statements to test if the functions work properly. 👍

Each function in `churn_script_logging_and_tests.py` is complete with logging for if the function successfully passes the tests or errors.

Indeed, each function found in the `churn_script_logging_and_tests.py` is complete and tests all of its functions' functionality. The `.log` file shows the success for each of the functions passing as expected. 🍀

Pro Tips

[Logging](#) is a very useful tool in a programmer's toolbox. It can help you develop a better understanding of the flow of a program and discover scenarios that you might not even have thought of while developing. You can have more information on this from this [link](#).

All log information should be stored in a `.log` file, so it can be viewed post the run of the script.

The log file is populated with all the results. The log file is saved and viewable after the script is executed. Whether it fails or succeeds this file is created and populated with information about each test. 🎉

The log messages should easily be understood and traceable that appear in the `.log` file.

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
```

The README informs a user how they would test and log the result of each function.

Save Images & Models

Store result plots including at least one:

1. Univariate, quantitative plot
2. Univariate, categorical plot
3. Bivariate plot

Store result plots including:

1. ROC curves
2. Feature Importances

Store at least two models. Recommended using `joblib` and storing models with `.pkl` extension.

Problem Solving

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

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.

 [DOWNLOAD PROJECT](#)

RETURN TO PATH

Rate this review

START