Mastering A/B Testing: A Practical Guide for Production
Project Overview

Overview
A/B testing, or split testing, is a statistical method used in marketing and product development to compare two versions of a web page, advertisement, email, or other marketing material to determine which performs better.

In A/B testing, two versions of the same marketing material are created: version A (the control group) and version B (the treatment group). The two versions are then randomly shown to different groups of users, and their responses are measured and compared.

In A/B testing, the treatment group is the group that receives the modified version of the marketing material being tested (version B). In contrast, the control group is the group that receives the original or existing version of the marketing material (version A).

The purpose of the control group is to establish a baseline or benchmark against which the performance of the treatment group can be measured. By measuring the performance of both groups, analysts can determine whether the changes made in the treatment group had a statistically significant impact on the measured response metrics.

In a production environment, conducting A/B testing can be challenging as it requires careful planning and implementation to avoid negatively impacting the user experience. This project aims to develop a hands-on approach to conducting A/B testing in a production environment.

The project will focus on implementing the A/B testing process in a hypothetical e-commerce platform. The platform wants to increase the conversion rate of its product pages by experimenting with different variations.

The project is demonstrated in two parts:

- In the first part, we will conduct A/B testing using Jupyter Notebook and understand the statistical tests involved.

- In the second part, we will focus on what the code base for a larger company might look like if they were running multiple A/B tests. We will use a fake events log generated with the fake web events library to simulate data processing, computing t-tests, and permutation tests, and analyzing significance. We will also examine the production code setup, including data generation and processing, visualize the result graphics, and explore how to integrate A/B testing into this setup.

Aim

This project aims to analyze logged events through an A/B test and understand the challenges of building such a solution at a production level.

Data Description

The data used in this A/B testing project was generated by the "fake web events" library. This package is designed to generate semi-random web events that can be used for prototyping purposes.

Tech Stack

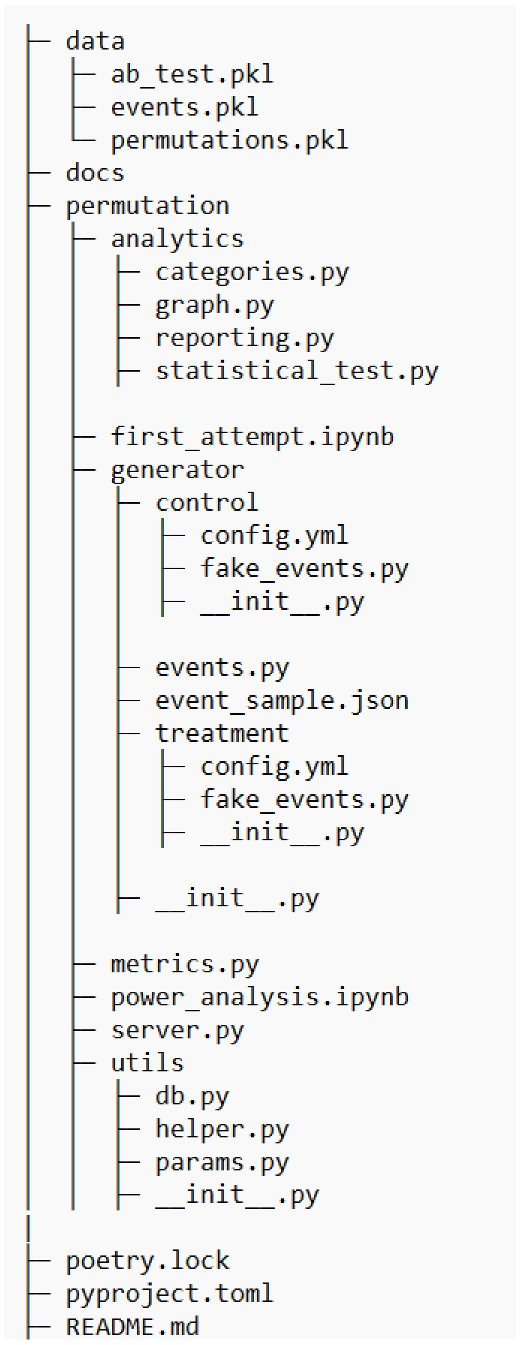
The A/B testing project uses several technologies to build and run the code.

- ➔ Language: Python (version 3.10.4)
- ➔ Libraries: pandas, requests, jupyter, notebook, duckdb, fake-web-events, statsmodels, matplotlib, tqdm, customtkinter, pandera, black, ipykernel.

Approach

- Statistical Analysis with Jupyter Notebook
    - Use Jupyter Notebook to perform A/B testing on a single metric using T-Tests and Permutation Tests.
    - Generate random events log data using the fake-web-events library.
    - Perform data processing and analysis using Pandas, NumPy, Statsmodels, and Matplotlib libraries.
    - Analyze the results and determine the statistical significance of the experiment.
- Production Code Setup
    - Running multiple A/B tests using a production-level code base
    - Use the fake-web-events library to generate more complex and realistic events log data.
    - Use Pandas and DuckDB libraries for data processing and querying.
    - Compute T-Tests and Permutation Tests to analyze the significance of the experiment.
    - Visualize the results.

Modular Code Overview:



```
├─ data
│  ├─ ab_test.pkl
│  ├─ events.pkl
│  └─ permutations.pkl
├─ docs
├─ permutation
│  ├─ analytics
│  │  ├─ categories.py
│  │  ├─ graph.py
│  │  ├─ reporting.py
│  │  ├─ statistical_test.py
│  │
│  ├─ first_attempt.ipynb
│  ├─ generator
│  │  ├─ control
│  │  │  ├─ config.yml
│  │  │  ├─ fake_events.py
│  │  │  ├─ __init__.py
│  │  │
│  │  ├─ events.py
│  │  ├─ event_sample.json
│  │  ├─ treatment
│  │  │  ├─ config.yml
│  │  │  ├─ fake_events.py
│  │  │  ├─ __init__.py
│  │  │
│  │  ├─ __init__.py
│  │
│  ├─ metrics.py
│  ├─ power_analysis.ipynb
│  ├─ server.py
│  ├─ utils
│  │  ├─ db.py
│  │  ├─ helper.py
│  │  ├─ params.py
│  │  ├─ __init__.py
│
├─ poetry.lock
├─ pyproject.toml
├─ README.md
```

Once you unzip the modular_code.zip file, you can find the following folders.

1. data
2. docs

3. permutation
4. poetry.lock
5. pyproject.toml
6. README.md

1. The data directory contains various data files, such as ab_test.pkl, events.pkl, and permutations.pkl, which stores data related to the A/B testing and event tracking.

2. **The docs directory contains documentation files or resources related to the project which are highly recommended to go through before moving on with the project.**

3. The permutation directory is the main directory for the A/B testing and permutation analysis. This contains the main server.py, and the example notebooks as explained in the tutorial videos.

4. The poetry.lock and pyproject.toml files are configuration files for the Poetry dependency management tool, specifying project dependencies and their versions.

5. **The README.md file contains project documentation, and all the execution instructions.**

Project Takeaways

1. Understanding of the statistical concepts behind A/B testing.
2. Hands-on experience in conducting A/B tests using Python and relevant libraries.
3. Insight into the challenges of building a solution for A/B testing.
4. Understanding the process of generating fake web events data using external libraries like fake-web-events.
5. Familiarity with statistical concepts such as t-tests, degrees of freedom, and p-values.
6. Learning to perform statistical analysis on A/B test results using t-tests, permutation tests, and statistical power graphs.
7. Understanding the difference between A/B and A/A testing.

8. Learning how to perform 10,000 A/A tests using a loop of permutation and temporary data storage.
9. Analyzing various categories and filters to understand the significance of A/B testing in measuring business impact.