10 Future-Ready MLOps Project Ideas



**Anish Mahapatra**

Contents

[List of Figures 3](#_Toc82481195)

[Introduction 4](#_Toc82481196)

[Designing an MLOps Solution 5](#_Toc82481197)

[1 Perfect project Structure – Cookiecutter & readme.so 6](#_Toc82481198)

[Project Idea 8](#_Toc82481199)

[2 Speed Exploratory Data Analysis to minutes – Pandas Profiling, SweetViz 8](#_Toc82481200)

[Project Idea 10](#_Toc82481201)

[3 Track Data Science Projects with CI, CD, CT, CM – Data Version Control (DVC) 10](#_Toc82481202)

[Project Idea 11](#_Toc82481203)

[4 Explainable AI / XAI – SHAP, LIME, SHAPASH 11](#_Toc82481204)

[Project Idea 11](#_Toc82481205)

[5 Deploy projects in minutes – Docker, FastAPI 12](#_Toc82481206)

[Project idea 12](#_Toc82481207)

[6 End to End Machine Learning – mlflow 13](#_Toc82481208)

[Project Idea 13](#_Toc82481209)

[7 Model Registry – MLStore, FeatureStore 14](#_Toc82481210)

[8 Big Data using Python, instead of PySpark – DASK 14](#_Toc82481211)

[9 Build a Chatbot and deploy it (open-source) 14](#_Toc82481212)

[10 FaaS framework implementation – Apache OpenWhisk, OpenFaas 14](#_Toc82481213)

[Conclusion 14](#_Toc82481214)

# List of Figures

[Fig 1: The complete MLOps process 4](#_Toc82480909)

[Fig 2: The MLOps Stack Template (Source) 5](#_Toc82480910)

[Fig 3: Cookiecutter Folder Structure (Source) 7](#_Toc82480911)

[Fig 4: GUI for Markdown Editor (readme.so) 8](#_Toc82480912)

[Fig 5: Pandas Profiling (Source) 9](#_Toc82480913)

[Fig 6: SweetViz Features (Source) 10](#_Toc82480914)

[Fig 7: SHAPASH monitor for Interpretable Machine Learning (Source) 12](#_Toc82480915)

# Introduction

Data Science is losing its title of the “Sexiest Job of the 21st Century”.

*87% of Data Science Projects never make it to production -* [*VentureBeat*](https://venturebeat.com/2019/07/19/why-do-87-of-data-science-projects-never-make-it-into-production/)

Jobs over the next decade are going to leverage the fundamentals of Data Science and build on top of that for production. Data Science has flourished over the decade on the promise that companies will make money using analytics. Data Science can make money only when AI/ML projects are in production. **MLOps roles will replace Data Science jobs** in the next decade. MLOps seems to be the new buzzword floating around in the Data Science world, and it is critical to understand what it really means.

MLOps aims to provide an end-to-end machine learning development process to design, build and manage reproducible, testable and evolvable ML-powered software. The cloud strategy of organizations plays a key role when it comes to production as custom solutions have to be built depending on the needs of the company. The complexity associated with MLOps is one of the main reasons we have not seen a one-size-fits-all approach. In this article, we will go through a few end-to-end projects that will explain how MLOps can be leveraged to make a Data Science project production-ready.

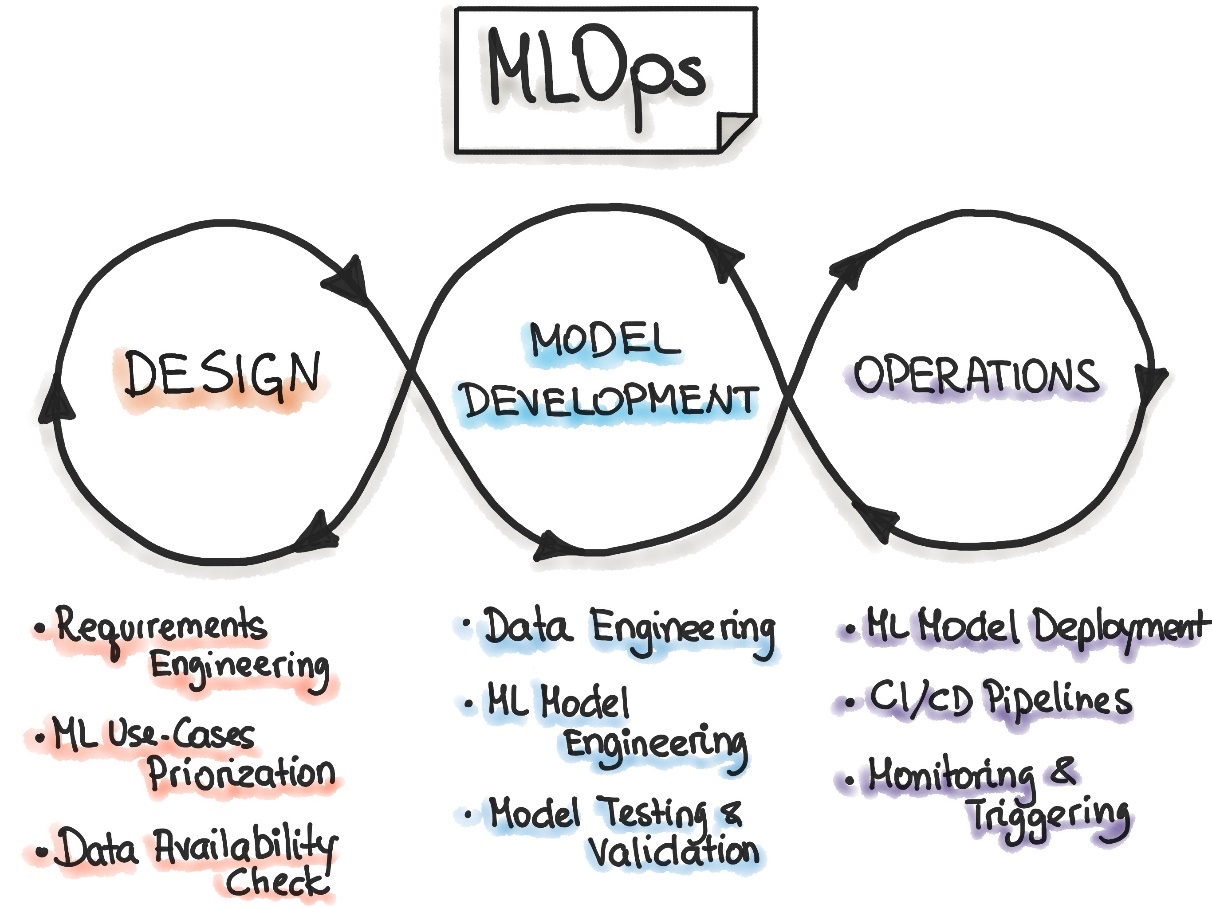


Fig 1: The complete MLOps process

# Designing an MLOps Solution

At the moment, the MLOps landscape is frequently evolving. The number of tools available for a single task is many – this leads to multiple options, which is always a good thing, but it also means that we have to decide what will work best in our use case. A good way to make projects more tangible is to focus on the stack. This can be complicated in different settings, and we have to make MLOps real for it to work. Where DevOps has a CI/CD approach to software solutions, MLOps has a Continuous Integration (CI), Continuous Delivery (CD), Continuous Training (CT), Continuous Monitoring (CM) approach.

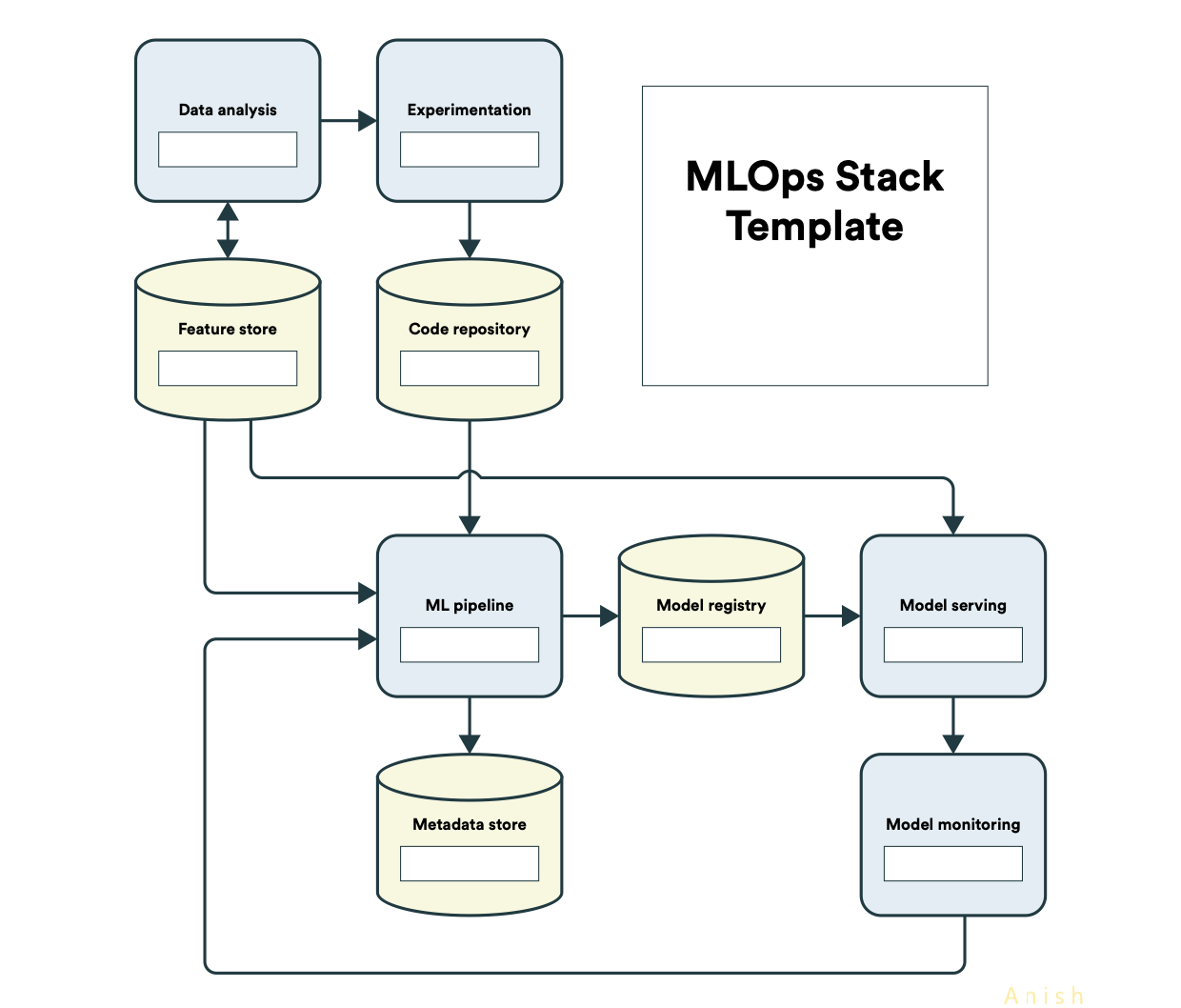


Fig 2: The MLOps Stack Template ([Source](https://ml-ops.org/content/state-of-mlops))

The MLOps Stack template is a good start on how we can think about MLOps solution design when implementing at scale for various organizations. The components are as follows:

* **Data Analysis**: The analysis component of the MLOps flow can be done with the help of various tools
* **Experimentation**: Output-focused experimentation along with domain knowledge can help select the relevant toolset
* **Feature Store**: Feature stores are used to store variations on the feature set leveraged for machine learning models that multiple teams can access
* **Code Repository**: The data and code repository has to be selected such that it fits into the MLOps stack being used, especially if it is on the cloud
* **ML Pipeline**: Similar to data pipeline, ML pipelines help carry the state of the machine learning project from data to ML output
* **Metadata Store**: Metadata for larger and evolving datasets can be housed in metadata stores
* **Model Registry**: Logging models are done in the model registry; this setup helps reflect back on multiple iterations
* **Model Serving**: Model serving is critical to production; it is the interface of machine learning with the real world
* **Model Monitoring**: Monitoring the model parameters in real-world post-production is critical to adjust various components to ensure that the feedback loop is met and the expected output is delivered in production consistently

Now that we have got a clear understanding of the components required to design for the solution, we are on track to look at project ideas for MLOps. One nuance of the current state of MLOps is that it is in the very nascent stages of development – all major cloud platforms and various open-source applications are all trying to solve for production-ready machine learning. So, in the project ideas below, we will look at the project ideas where we can implement components of MLOps to make data science projects more MLOps ready.

# 1 Perfect project Structure – Cookiecutter & readme.so

From the MLOps process flow, if there is one thing you can incorporate in your projects right now, it is improving your Data Science Structure. A great package called [cookiecutter](https://drivendata.github.io/cookiecutter-data-science/#cookiecutter-data-science) helps build the structure of real-world machine learning implementations. The goal of cookiecutter is to make it easier to start, structure and share analysis.

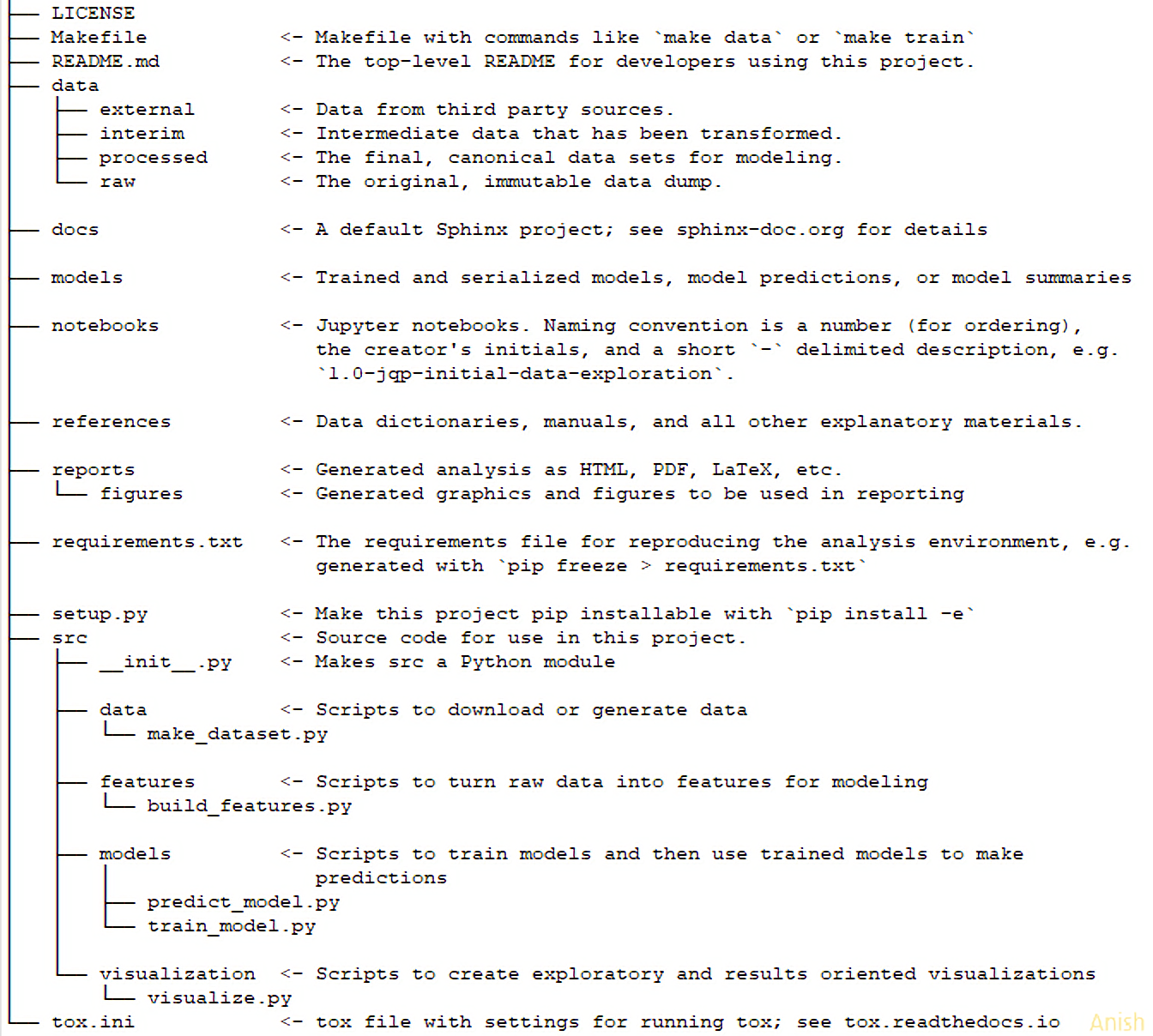


Fig 3: Cookiecutter Folder Structure ([Source](https://drivendata.github.io/cookiecutter-data-science/#cookiecutter-data-science))

Another major component where Data Scientists lack is in providing information and documentation in their projects. So, when returning to your own project after a period of time, a lot of information is missed out. Markdown files, particularly readme.md files can be complicated to create, especially if you want to do them well. So, use an editor from readme.so to help you in real-time.

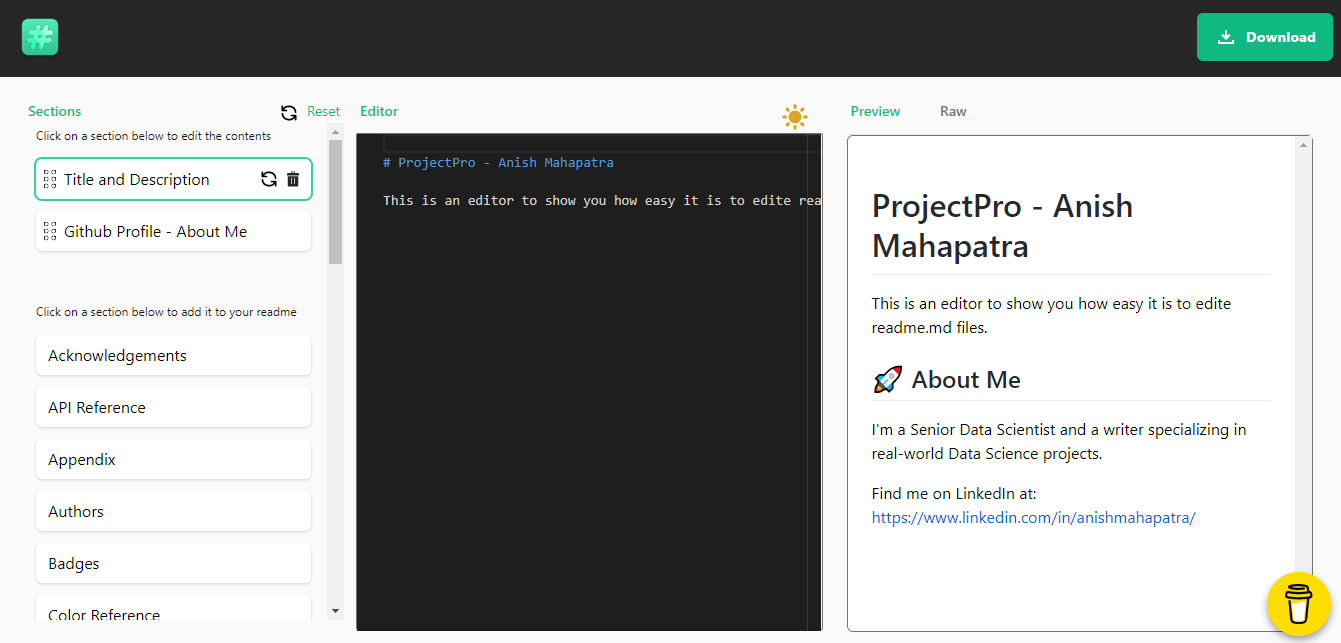


Fig 4: GUI for Markdown Editor ([readme.so](https://readme.so/editor))

This is bound to impress interviewers and demarcate you as an expert in most interviews, as a lot of teams face issues when it comes to the standardization of machine learning in production. Adopting these in your personal GitHub projects will definitely make you stand out as an expert in MLOps.

## Project Idea

Take any project you have done and implement it with a high-quality readme.md file leveraging [readme.so](https://readme.so/editor) and data science project structure using [cookiecutter](https://github.com/drivendata/cookiecutter-data-science). Let’s say you are a beginner and want to get started. Head over to the [Kaggle Titanic EDA to ML Challenge](https://www.kaggle.com/dejavu23/titanic-eda-to-ml-beginner) and focus on building the project out using cookiecutter and high-quality documentation as expected from a production machine learning perspective. Please remember, in production, the actual process of Data Science does not change. Rather the way we approach solution design evolves.

# 2 Speed Exploratory Data Analysis to minutes – Pandas Profiling, SweetViz

Exploratory Data Analysis (EDA) is something beginners in Data Science fear. Especially when multiple teams are working towards demystifying the data to ensure that a complete understanding of the data is done, there is bound to be some gaps between the teams and the business. Another gap that is prevalent in the real world is the presentation of EDA to business stakeholders. The process of retrieving images/analysis from Jupyter Notebooks to a presentation can be a tedious process prone to errors. There are ways to effectively speed up the process of EDA.

The packages that can be used to generate reports that can be downloaded as Html files to showcase end to end exploratory analysis are as follows:

* [Pandas Profiling](https://github.com/pandas-profiling/pandas-profiling): Helps with a quick analysis of the data with inclusions such as type inference, unique values, missing values, quantile statistics, descriptive statistics, histogram, correlations, text analysis etc. Just one line of code will enable you to quickly perform the initial EDA in a visually appealing and shareable format. This is one of the best packages you can try to impress quickly.

pandasreport = ProfileReport(df, title=”New Report Sample”)

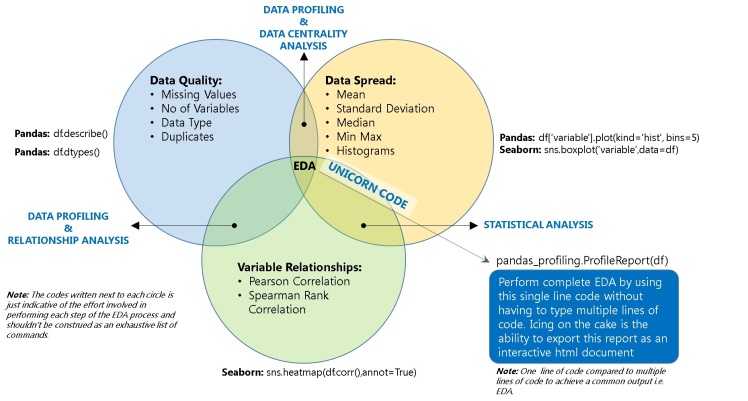


Fig 5: Pandas Profiling ([Source](https://dzone.com/articles/pandas-one-line-magical-code-for-eda-pandas-profil))

In the Project idea section, we will talk about a good dataset you can use to try Pandas Profiling out.

* [SweetViz](https://pypi.org/project/sweetviz/): SweetViz is a fantastic package to perform exploratory data analysis designed by a graphic designer. These packages work inline on Jupyter Notebook as well as online editors such as Google Colab. SweetViz offers an in-depth EDA that goes a step further from Pandas profiling by offering target analysis, feature analysis, comparison and correlation analysis.  
    
  import sweetviz as sv  
    
  sweetVizReport = sv.analyze (sampleDataFrame)  
  sweetVizReport.show\_html()

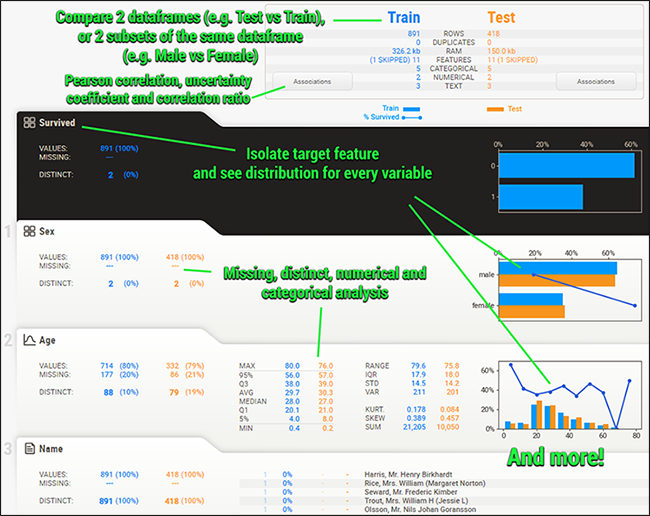


Fig 6: SweetViz Features ([Source](https://pypi.org/project/sweetviz/))

## Project Idea

A good dataset to understand visual bias is the [Zomato Restaurants dataset](https://www.kaggle.com/shrutimehta/zomato-restaurants-data). It is a decent dataset to query with multiple nuances that can be analyzed. Focus on performing a preliminary analysis of the data using Python, followed by leveraging pandas profiling and sweetviz. The source code for inspiration can be found [here](https://github.com/anishmahapatra/Zomato-Data-Visualization). It can either be performed on Jupyter Notebook or on Google Colab (inline as well as separate files that can be generated).

# 3 Track Data Science Projects with CI, CD, CT, CM – Data Version Control (DVC)

Data Science Projects in production need multiple components to work seamlessly to generate reliable results. [DVC](https://dvc.org/) is built to make models shareable and reproducible. Where GitHub only focuses on the code, consider DVC as the big brother for Data Science. It is designed to handle large files, data sets, machine learning models, metrics as well as code. The three main components that differentiate the offering of DVC are as follows:

* **ML project version control**: Will work with Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform, SFTP, HDFS and HTTP
* **ML experiment management**: Git branches along with data and metrics tracking to enable Continuous Training (CT) and Continuous Monitoring (CM), this enables a cleaner structure
* **Deployment & Collaboration**: Leverage the classic push/pull along with data version control to collaborate and even orchestrate as a team.

This helps save and reproduce experiments, version control models and data and aids in establishing a workflow for deployment & collaboration.

## Project Idea

Build out a machine learning project and perform version control with DVC. Build a dependency graph to operationalize pipelines and showcase how DVC can be leverage to reproduce AI/ML production models at scale. The main components that can be worked upon in this project are Data and Model Version Control, Data and Model access, Data pipelines (DAGs), Metrics, Parameters and Plots, and finally, experiments. Leverage the [Kaggle Telco Customer Churn Dataset](https://www.kaggle.com/blastchar/telco-customer-churn) to perform EDA, data cleaning and modelling and invoke the same using DVC. You can take code inspiration from this repository [here](https://github.com/anishmahapatra/Classification-Telecom-Customer-Churn).

# 4 Explainable AI / XAI – SHAP, LIME, SHAPASH

An end-to-end project where the output of the model and the reasoning behind each of the decisions of the model can be really helpful in the real world to explain to the business stakeholders. Where Feature Selection tackles the issue of the selection of important columns for modelling, Explainable AI puts forth to focus on what are the micro-decisions that lead to the final macro decision. This can be done in a model-agnostic manner using Shapely Additive Explanations (SHAP) and Local-Interpretable model-agnostic explanations (LIME). Consider SHAPASH as the combination of the best of SHAP and LIME with the addition of a full-fledged web interface.

## Project Idea

Consider the [Kaggle House Prices Dataset](https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data) to build an interpretable machine learning model on top. Build a supervised model along with the stated pre-requisites and launch the web app to be able to visualize the data. Build a Light GBM regressor model and visualize what an interpret what the important decisions that go into making a prediction. The sample source code can be found [here](https://github.com/MAIF/shapash/blob/master/tutorial/tutorial01-Shapash-Overview-Launch-WebApp.ipynb) to build on top of.

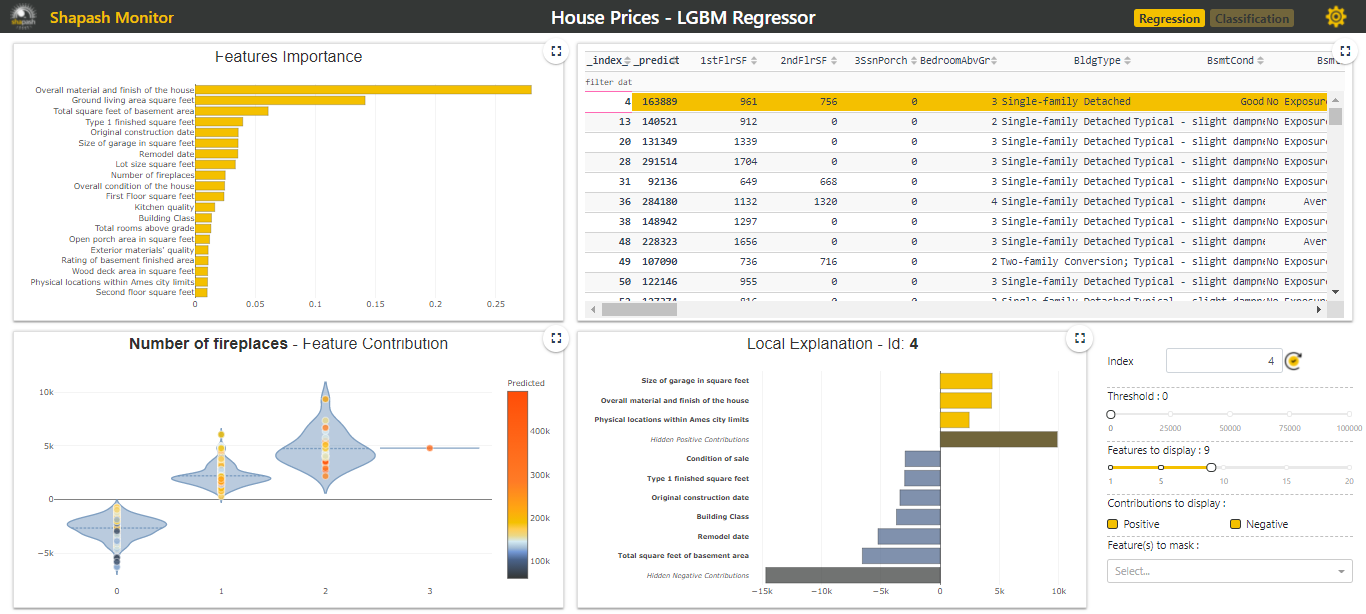


Fig 7: SHAPASH monitor for Interpretable Machine Learning ([Source](https://shapash-demo.ossbymaif.fr/))

# 5 Deploy projects in minutes – Docker, FastAPI

Once the base machine learning model is developed, the ultimate question is now, what do we do? It is one of the most common questions Googled by the lucky few Data Scientists that are able to reach the point where they get to deploy their model. A great way to be ready for this experience is to do a project where you leverage Docker or FastAPI to package and interface your machine learning for production use.

* **Docker**: Think of your machine learning project and the number of localized dependencies it has on the Operating System, such as folder structure, OS, requirements, package version control etc. Docker simplifies the deployment with the help of OS-level virtualization to deliver software in packages called containers.
* **FastAPI**: Software framework to develop web applications in Python. Essentially, FastAPI is a modern, high-performance web framework to build APIs.

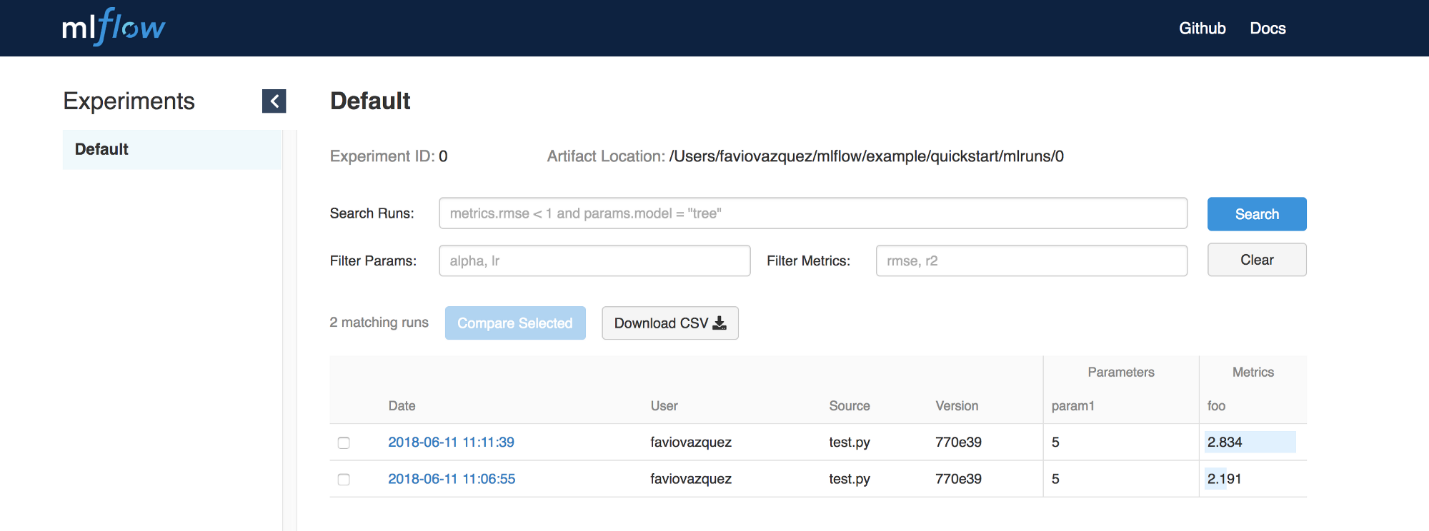
## Project idea

I admit this would be an intermediate to an advanced implementation as it includes an end-to-end implementation. However, if you are able to implement and deploy this project, you will get through multiple interview rounds as a front-runner. For this implementation, we will be using the [Kaggle Boston House Prices](https://www.kaggle.com/vikrishnan/boston-house-prices) dataset on which we will build a random forest regressor. The expected flow of tasks are as follows:

* Package the model and build an API to communicate with the model
* Design a user interface
* Set up a virtual environment and use FastAPI
* Packaging machine learning model with Docker
* Use dependency injection to be testing-ready
* Test the API
* Package with Docker and Docker Compose
* Use GitHub Actions to Automate Testing

For your reference, the source code for this ambitious but very doable project can be found [here](https://github.com/cosmic-cortex/fastAPI-ML-quickstart).

# 6 End to End Machine Learning – mlflow

Truly one of the widely adopted methods to implement MLOps in the real world, Mlflow is a platform to manage the end-to-end machine learning lifecycle.   
  


Mlflow can be implemented on multiple cloud platforms and consists of the following components:

* **MLflow Tracking**: Used to log parameters, code and results in ML experiments. These logged values can be compared using the interactive UI
* **MLflow Projects**: Used to work reliably reproduce runs using conda and Docker
* **MLflow Models**: Model packaging format and tools to help deploy the same model to batch and real-time settings on multiple platforms such as Docker, Apache Spark, Azure ML and AWS SageMaker
* **MLflow Model Registry**: Centralized model store to collaboratively manage the full lifecycle of MLflow models

## Project Idea

For this project idea, we will leverage the [UCI Wine Quality Dataset](https://archive.ics.uci.edu/ml/datasets/wine+quality) and work with MLflow to register the nuances of the model. This can be visualized on the MLflow visual interface. This is an intermediate implementation that will help strengthen your foundation of MLOps. The source code for this example can be found [here](https://github.com/databricks/mlflow/tree/master/examples/sklearn_elasticnet_wine).

# 7 Building Production ML Pipelines - Model Registry, Feature Store (Feast, ButterFlow)

Data Scientists are duplicating work because there is no centralized feature store. If a team were to use features that another parallel team has made, it is a complex and messy project that ends up with the original team simply building out new features on their own. Large companies are building out their own Feature Store to tackle customized issues that they are solving. The big players in the game – Amazon, Google and Databricks have their own feature store. The interesting thing to note here are the other players that also have a feature store – Uber (Michelangelo Palette), Netflix (Runaway), Pinterest (Galaxy), Apple (Overton), GoJek/Google (Feast), where further information can be found on [featurestore.org](https://www.featurestore.org/)

The two popular feature stores where we can build projects to learn more are:

* [Feast](https://github.com/feast-dev/feast): Built by GoJek and Google, Feast helps operationalize analytic models for model training and online inference.
* [ButterFree](https://github.com/quintoandar/butterfree): A tool to build feature stores, to help transform raw data infor feature stores. It is used to build ETL pipelines for Feature Stores using Apache Spark

## Project Ideas

* **Butterfree**: If you are comfortable with PySpark, then you could build on top of the fundamentals of feature store using Butterfree [here](https://github.com/quintoandar/butterfree/blob/master/examples/simple_feature_set/simple_feature_set.ipynb), where there is code for extract, transform, load followed by building pipelines using the feature store.
* **Feast**:Another great project idea with Feast (Python) to build on top of is **Driver Ranking** and **Fraud Detection** in Google Cloud Plaform in a production ready manner. Building this on Jupyter Notebook can be a little complicated, so I would advise to build your project on Google Colab and commit it to your GitHub post understanding and building on top of the code. The notebooks can be found [here](https://colab.research.google.com/github/feast-dev/feast-driver-ranking-tutorial/blob/master/notebooks/Driver_Ranking_Tutorial.ipynb) and [here](https://colab.research.google.com/github/feast-dev/feast-fraud-tutorial/blob/master/notebooks/Fraud_Detection_Tutorial.ipynb) respectively.

# 8 Big Data using Python, instead of PySpark – DASK

…

# 9 Build a Chatbot and deploy it (open-source)

…

# 10 FaaS framework implementation – Apache OpenWhisk, OpenFaas

…

# Conclusion