Adopting an MLOps mindset

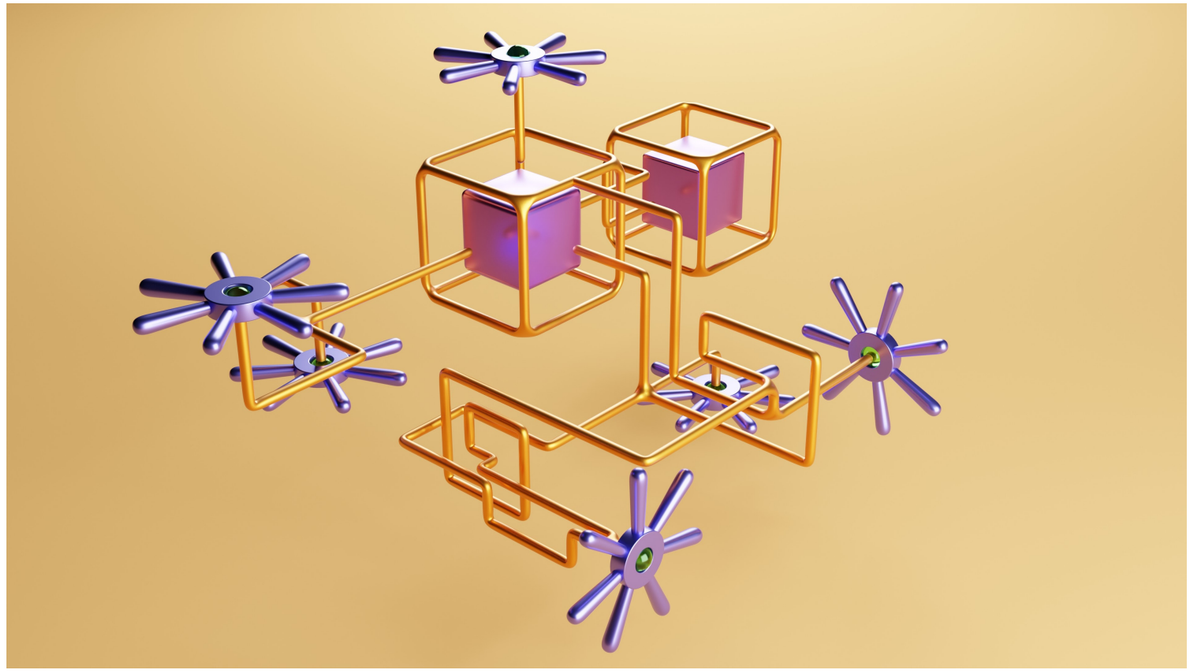
DEVELOPING MACHINE LEARNING MODELS FOR PRODUCTION

Sinan Ozdemir
Data Scientist, Entrepreneur, and Author

datacamp

MLOps

- The process of automating and streamlining the ML workflow from experimentation to production
- Ensures that ML experiments are properly tested and ready to be deployed and scaled

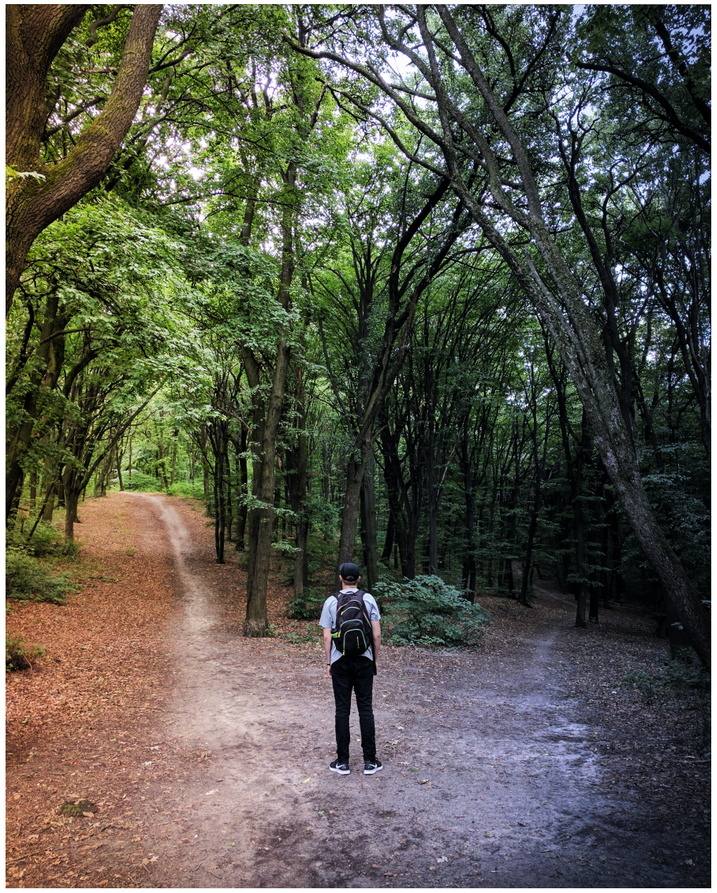


ML experiments

ML experiments involves testing models and determine which one is best

- MLOps includes model experimentation
- Evaluating models on different datasets
- Careful model selection essential
- Selection process may be time-consuming
- Ensures project success

From experiments to production



What Makes an ML Experiment Ready for Production?

- Tested and validated using appropriate metrics
- Proper documentation
- Performing monitoring is in place
- Production environment is secure and scalable

Why most ML experiments fail

There are many reasons why this might happen:

- Lack of clear goals and objectives
- Poor data quality
- Complex model architectures
- Insufficient training data
- Overfitting or underfitting

Technical debt

Code written in a hurry without proper testing or validation or missing/incomplete/out-of-date documenation

- Costly errors or bugs can occur if not addressed early on.
- Avoid technical debt by writing proper code and documentation from the start.

Let's practice!

DEVELOPING MACHINE LEARNING MODELS FOR PRODUCTION

Writing maintainable ML code

DEVELOPING MACHINE LEARNING MODELS FOR PRODUCTION

Sinan Ozdemir
Data Scientist, Entrepreneur, and Author

datacamp

Project structuring

- Organize project files into a logical structure
- Group related files together
  - Organize data sets and ML models in separate folders
- Ensure files are properly named and labeled

Sample project structure

Sample project directory with a README file, requirements file and three subfolders: data, models, notebooks

```
├── README.md
├── data
|   ├── raw
|   ├── processed
|   └── interim
├── models
|   ├── model1.py
|   ├── model2.py
|   └── model3.py
├── notebooks
|   ├── data_exploration.ipynb
|   ├── model_training.ipynb
|   └── model_evaluation.ipynb
├── requirements.txt
```

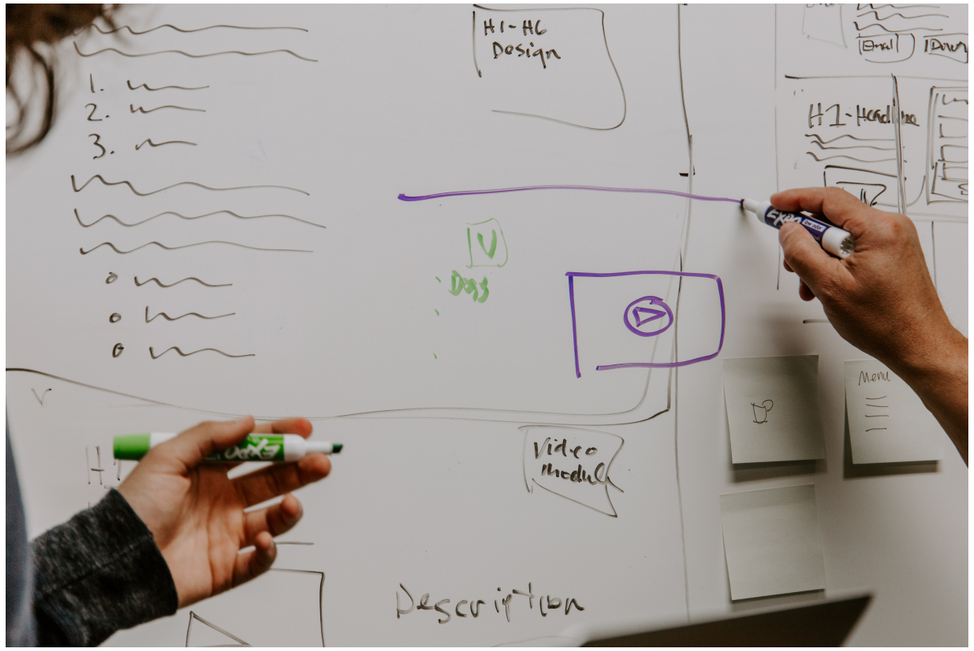
- README.md: Explains the purpose of the repository and how to use it.
- requirements.txt: Lists all dependencies
- data: contains data-related files, including raw data and processed data
- models: contains all model-related files, including scripts for creating models.
- notebooks: contains notebooks for data exploration, model training, and model evaluation.

Code versioning

- Use a version control system like git to keep track of changes to the code
- Allows for rollback of changes if necessary
- Can help identify the source of bugs and errors
- Allows for parallel work

Documentation

- Document code and project structure
- Explain the purpose of each file and function
- Describe how to use the code
- Include instructions on how to deploy the ML model



Adaptability of code

- Easier to understand, modify, and update
- Reduces the time and effort required to make changes to the codebase
- More easily adapt to data + code + requirements changes
- Less prone to bugs
- Easier to integrate new features or technologies as needed
- Essential for building ML applications that can evolve and adapt over time

Let's practice!

DEVELOPING MACHINE LEARNING MODELS FOR PRODUCTION

Writing effective ML documentation

DEVELOPING MACHINE LEARNING MODELS FOR PRODUCTION

Sinan Ozdemir
Data Scientist, Entrepreneur, and Author

datacamp

The components of excellent ML documentation

- Data sources
- Data schemas
- Labeling methods
- Model experimentation + selection
- Training environments
- Model pseudocode

Documenting data sources

Allows us to establish processes for evaluating the quality of our data.

This also offers other benefits:

- Keep track of where data comes from.
- Evaluate and iterate on the quality of data.

Data schemas

A structure that describes the organization of data.

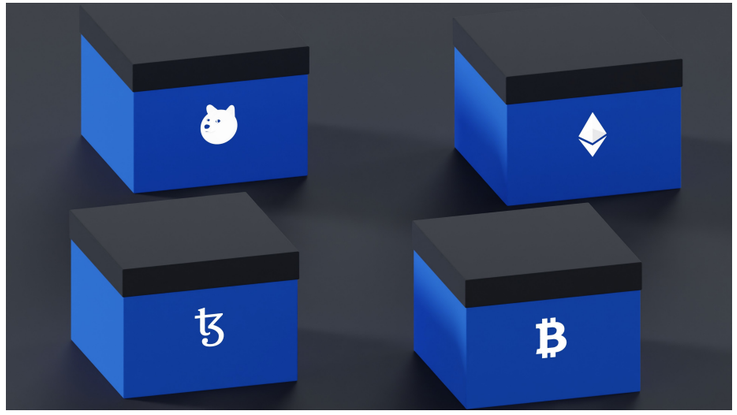
For a relational database schema:

Database key	Data type	Data order
`Person.name`	`string`	`nominal`
`Person.survey_score`	`integer`	`ordinal`

Labeling methods (for classification)

Documenting how we labeled our response variable enhances:

1. **Reproducibility** of the training pipeline.
2. **Model reliability** through label quality.
3. **Model performance** through label improvement.



Labeling methods can evolve over time.

Model pseudocode

A visual representation of the different steps involved in building your machine learning model.

This often includes:

- Feature engineering steps.
- Components of an ensemble pipeline.
- Example inputs and outputs of the model.

Model experimentation + selection

Documenting the process of experimentation and selection of the best model includes documenting:

- The model development process.
- The models considered.
- The metrics used.
- The hyperparameter combinations tried for each model.

Training environments

To document our training environment, we should include:

- Packages used with versions (eg. `scikit-learn==1.1.3`).
- Any random seeds used for non-deterministic training (eg. dimensionality reduction algorithms).

Why?

- Ability to reproduce the results of our machine learning models.
- Ensuring consistency between training and production deployments.

Let's practice!

DEVELOPING MACHINE LEARNING MODELS FOR PRODUCTION