Architecture for machine learning apps

Django Flavored

About me

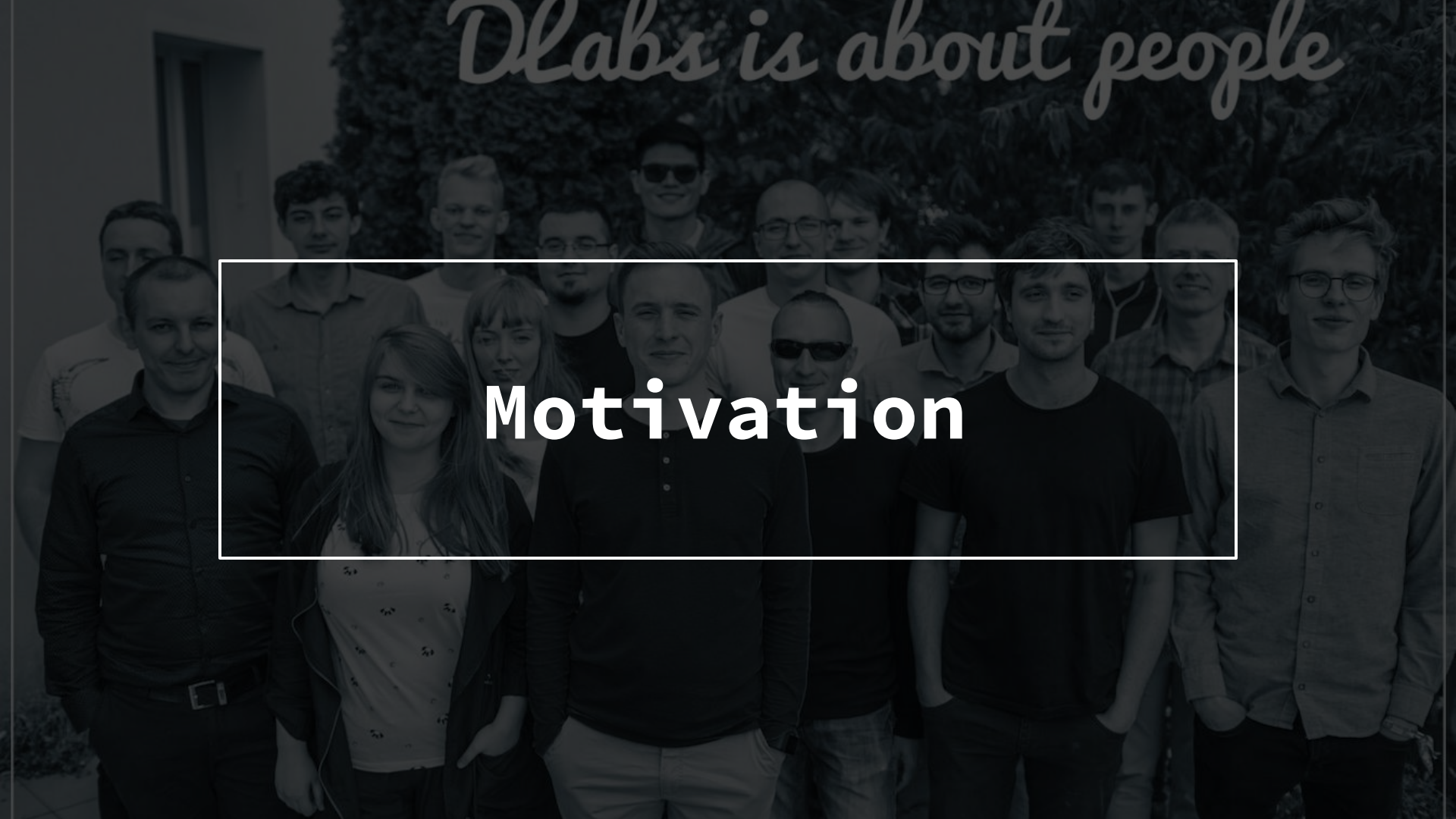
I’m a software engineer.

Lately I’ve been working for National Geographic / Globant Contractor

Coffee addicted

Batman’s fan

You can find me at @jorlugaqui on twitter



Motivation

Global view (Diagram)

Topics

- Defining the model
- Placing the model into the app
- Running the model into the app
- Engineering aspects

Defining the model

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

ppn = Perceptron(max_iter=40, eta0=0.1, random_state=0)
ppn.fit(X_train_std, y_train)
```

Iris Data set

Placing the model into the app

```
class IrisModel(object):

    _instance = None

    def __init__(self):
        if self._instance is not None:
            raise ValueError('The model was already loaded')

    @classmethod
    def get_instance(cls):
        if cls._instance is None:
            try:
                model_path = os.path.join(settings.BASE_DIR, 'data', IrisModelConfig.MODEL_PKL)
                with open(model_path, 'rb') as model:
                    cls._instance = pickle.load(model)
            except IOError as e:
                logger.exception('Serialized model was not found')
            except pickle.UnpicklingError as e:
                logger.exception('Error while loading the model')

        return cls._instance
```

Load the model (SINGLETON)

Running the model into the app

```
api:
  image: jorlugaqui/pycon:latest
  command: ./scripts/run_api.sh
  depends_on:
    - general-worker
    - db
  env_file: .env

general-worker:
  image: jorlugaqui/pycon:latest
  command: ./scripts/run_worker.sh
  depends_on:
    - rabbit
    - db
  env_file: .env

rabbit:
  image: rabbitmq:3.7.1-management-alpine
  environment:
    RABBITMQ_DEFAULT_USER: ${PYCON_RABBIT_USER}
    RABBITMQ_DEFAULT_PASS: ${PYCON_RABBIT_PASS}
  env_file: .env
```

Use celery for running the model

Others engineering aspects

```
import pickle
import os

pickle.dump(ppn, open(os.path.join('iris_predictor.pkl'), 'wb'), protocol=4)
```

Serializing the trained model

- Container for every component
- Make team members life easier: automate as much as possible
- Test the models
- Cache the results
- Model as Service / Model in an Independent container

Miscellaneous

- Code style
- Keeping unused code
- Global variables
- Debugger being no used
- Long Pull request
- Naive implementations
- Lack of tests

Issues

Conclusions

- Don’t get lazy with testing, Include as many related model tests in the CI workflow as possible.
- Don’t run the model in the same thread / request
- Trust the data team in model correctness, but don’t rely on them about software engineering stuff

Thank you!