Reply

Project: End to End Machine Learning Development for large-scale Model deployment (72 models in production)

Industry: Telecommunication
Role: Lead Machine Learning Engineer, Team Lead, and Client SPOC.
Team Structure: Team included 50 Engineers, Analysts and Data Scientists from different companies. I grew up and led the Reply team from 2 to 12 Machine Learning Engineers and Data Scientists working together with the rest of the team-members. I was also responsible for developing, architecting and leading smaller development teams for defined development stories.
Duration: 3,5 years - ongoing

Description: The project covered the end-to-end lifecycle of Machine Learning products. This includes developing an existing framework for Machine Learning, adding different features for and end-to-end Machine Learning life-cycle with MLOps paradigms, and further maintaining it, where I actively implemented and led smaller development teams. The outcome of the platform was decreasing the time-to-deploy from 72 days to 21 days. Platform includes different components:
Shared Machine Learning Library
A shared Machine Learning library which is implemented in Python with an ability to run locally in Pandas workflows, or in cluster mode utilizing Pyspark serves as a basis for common tasks including:

1. Standardizing a YAML configuration for all different stages, and implementing a parsing mechanism to ingest different YAML hierarchies into a configuration class.
2. Data loading and integration from different sources (Hive, Oracle, SQL-Server). This also includes a data versioning mechanism implemented binding YAML, Git, and data sources.
3. Model loading from HDFS, and DBFS.
4. Batch and Real-time inference utilizing Pyspark.
5. Model monitoring to detect model drifts, implemented for different model types (ex. clustering, classification and regression). The outcome is stored in an Oracle database; which is a base for a ticketing system for model performance issues, and a Tableau dashboard for model performance monitoring. Monitoring is done in multiple approaches, first for labels provided by domain experts through a labeling interface implemented in Flask.
6. Feature monitoring to detect data drifts, implemented with a similar logic to model monitoring, for different data and feature types.
7. Automated model retraining, this starts with splitting the dataset, considering the newly labeled data, through time, date, water-fall split-strategy and random split strategy. Next

step is retraining while performing Grid-Search to identify a model which improves the performance of the current one and deploy it.
8. Automated JIRA tickets creation for Model Quality issues resulting from monitoring or retraining.

Different Platforms built on-top of the shared Machine Learning library, and interfaced via YAML configurations:

1. Time-series platform which runs Machine Learning models which include a time-based components (~15 models running).
2. General platform for different Machine Learning use-cases including classification, regression, and clustering. This platform supports SKLearn, Tensorflow, and Pytorch models (~47 models running).
3. Tickets platform which runs Machine Learning models that support analyzing support tickets with respect to the different fields included and the free-text (~10 models running).

Additionally, my team was responsible for:

1. Building up the infrastructure for migration from Cloudera to Azure and Databricks
2. Building up the Databricks environment and maintaining it, including different updates and newly released products.
3. Building up different Data Science products to be deployed on the platforms (ex. churn prediction, next-best offer prediction, customer-service routing, revenue assurance..)
4. Operation of the platforms, infrastructure and Machine Learning models.

Technologies Used:

- **Programming Language:** Python
- **Data Processing:** Pandas, PySpark
- **Configuration Management:** YAML, with a custom parsing mechanism
- **Data Storage:** Hive, Oracle, SQL Server, HDFS, DBFS
- **Containerization and Orchestration:** Docker and Kubernetes
- **Web Framework**: Flask for labeling interface
- **MLOps**: MLFlow, model monitoring, feature monitoring, data versioning, model versioning, model retraining
- **Data Visualization:** Tableau
- **Machine Learning Libraries:** SKLearn, TensorFlow, PyTorch
- **Infrastructure and Cloud Services:** Cloudera, Azure, Databricks
- **Project Management:** JIRA, Confluence

Project: Building up a Machine Learning Workflow to Deploy Data Science Use-cases.

Industry: Insurance
Role: Team Lead and Client SPOC - Machine Learning Engineer
Team Structure: Leading a team of two Data Scientists and four Machine Learning Engineers working in an Agile mode together with the stakeholders on two weeks sprint basis.
Duration: 2 years

Description: The project's goal was to build and deploy multiple Data Science models efficiently on an OpenShift cluster. Throughout the project time, together with the team we managed to achieve different pillars:

Standardizing a Git Workflow based on Gitflow, to achieve an efficient CI/CD and Release Management Process
The process allows for a full release management process connected to JIRA, Confluence, and Git allowing for transparency with respect to the PO and increasing efficiency within the scrum team. It also implements different CI/CD steps upgrading code-quality using Pylint thresholds, implementing tests and standardizing coverage process, and standardizing review process. GitFlow was adapted for feature, develop, release, and main branches to regulate releases in dev, test, and prod.

Building and deploying different data science use-cases, chosen examples:

1. Sentiment Analysis for customer complaints, running on OpenShift.
2. Fraud detection for car-images sent by customers requesting insurance claims, running on OpenShift.
3. Detecting solar panels on roof-tops for buildings utilizing a Flask application, running on Google Kubernetes Engine, utilizing Google Maps API, and interfaced via Apigee API Gateway.

Development of an MLOps Platform for model deployment
The model workflow included a hand-over process from the Data Science team which allows for saving the model file along with a metadata file including features, output categories, model type, and the fields to be monitored. The workflow picks up the model, allows for batch/real-time inference, orchestration, and deployment over OpenShift either as an API or as a scheduled workflow. MLFlow based inference and experiment tracking capabilities were also introduced to control model development and versioning. Common monitoring abilities were developed as a Python library which is interfaced through decorators and utlilized to interface metrics to a Prometheus based end-point to visualize the model performance with Grafana and send alerts when there is a data or model drift based on the model type. This allowed for a decreased time to deploy for the models, a standardized MLOps approach, monitoring, alerting, and stable deployment together with the Release workflow.

Technologies Used:

- **Version Control System:** Git

- **Workflow Management:** GitFlow, JIRA, Confluence
- **Code Quality and Testing:** Pylint, Unit Testing Frameworks (pytest, unittest)
- **Cloud Computing and Orchestration:** OpenShift, Google Kubernetes Engine (GKE)
- **API Management:** Apigee API Gateway
- **Machine Learning Operations (MLOps)**: MLFlow, JFrog
- **Programming Language:** Python
- **Data Manipulation Libraries:** Pandas, NumPy
- **Machine Learning Libraries:** SKLearn, TensorFlow
- **Monitoring and Alerting:** Prometheus, Grafana

Project: Performing Demand Forecasting for large retail supermarket.

Industry: Retail
Role: Team Lead - Machine Learning Engineering
Team Structure: Leading a team of two Data Scientists presenting weekly to the stakeholders on an Agile workflow.
Duration: 8 months

Description: The project included two phases for applying demand forecasting for a large retail supermarket.

Phase (1)
The project started with an initial model for performing demand forecasting which was developed initially by the respective department. The model's aim is to forecast the demand on different items which was sold by the supermarket to optimize with respect to inventory management, reducing waste, and improving customer satisfaction regarding products' availability. I was leading a team of two Machine Learning Engineers with a mission to check the performance of the model and suggest further optimizations. We started with performing an exploratory data analysis to the dataset, identifying transformations done over the pipeline along with Feature Engineering, and identifying different mistakes the model was making and cross-checking them with the domain-owners. Afterwards, we started with stress-testing the Model Architecture according to the performance metrics, in accordance with the domain owners' KPIs (ex. over-estimation is better than under-estimation of products' quantities). Finally, we also calculated SHAP values for explainability of the model's performance. The output of that round was having a better trust in the model's prediction, as well as an improved performance both on a MSE basis, and with respect to corner cases provided by the domain owners.

Phase(2)
The next phase was to move the model to production. For that, we modularized the training code-base, implemented the necessary data pipelines in Python for performing the transformation, linted the full-code base to a 10/10 shown by Pylint, applied unit-tests bringing a higher test-coverage, and allowed for different parametrization on run-time bases. We have

also architected the AWS Architecture for deployment of the model inference both on scheduled, and real-time basis utilizing AWS SageMaker end-points, AWS Lambda, API Gateway, and CloudWatch events. The next step was to setup the model monitoring utilizing CloudWatch, and finally to have the alerting mechanism set-up using SNS topics.

Technologies Used:

- **Programming Language:** Python
- **Data Manipulation Libraries:** Pandas, NumPy
- **Machine Learning Libraries:** SKLearn
- **Data Visualization Tools:** Matplotlib, Seaborn
- **Model Interpretability:** SHAP (SHapley Additive exPlanations)
- **Code Quality and Testing:** Pylint, Pytest
- **Cloud Computing and Deployment:** AWS SageMaker, AWS Lambda, AWS API Gateway, AWS CloudWatch, AWS Managed AirFlow, AWS Step Functions.
- **Monitoring and Alerting:** AWS CloudWatch, AWS SNS, AWS S3

Project: Predictive Maintenance for Aircraft Spare Parts

Industry: Defense and Security
Role: Lead Data Scientist
Team Structure: Leading a team of two Data Scientists presenting frequently to the stakeholders on an Agile workflow.
Duration:

- PoC Phase: 2 days
- Full Project Duration: 4 months

Description: The project started with a two-days PoC where I was presented with a dataset of different error codes for planes occurring over different flights. My initial approach involved an in-depth exploratory data analysis, including viewing the distributions of different features including sensor data, identifying discrete values within categorical variables, and performing correlation analysis between different columns. Following step was applying multiple Machine Learning models, where the XGBoost showed the best performance among all. The result of the PoC was presented to the stakeholders, and accordingly, a 4 months project was started where I led a couple of team members to perform further data discovery and implement a Machine Learning Model predicting error classes from occurring errors on pervious flights.
Technologies Used:

- **Programming Language:** Python
- **Machine Learning Libraries:** SKLearn, MLFlow
- **Data Visualization Tools:** Plotly, Matplotlib
- **Model Interpretability:** SHAP (SHapley Additive exPlanations)

Project: Building up a scalable Microservice for Real-time GeoService Information.

Industry: Defense and Security
Role: Software Engineer
Team Structure: Developing the whole service based on the requirements while presenting weekly updates and the final outcome.
Duration: 1 month

Description: The project's goal was to develop and deploy a microservice which would be highly available for simulation software to query Geoservice information such as Temperature, and Precipitation for the closest point available to the given Longitude/Latitude at a certain date/time. The implementation was done via a KD-Tree for an O(n) worst case retrieval. The web application was Flask-based with a Web Server Application Gateway which was dockerized. It was architected as a Pyms Chasis Framework, had a Swagger API Spec interfaced via Connexion Endpoint mapping.

Technologies Used:

- **Programming Language:** Python
- **Web Framework:** Flask
- **Others:** Pyms, Swagger, Connexion
- **Data Structure for Spatial Information:** KD-Tree
- **Deployment:** Web Server Application Gateway, Docker
- **Testing:** Unit-tests (Pytest), Load-test (Locust)

Project: PoC for Analyzing Vehicle Test Drive Data via Natural Language Processing

Industry: Automotive
Role: Team Lead, and client SPOC
Team Structure: Leading a team of 5 engineers and Data Scientists, competing against 10 other teams from different companies and winning the first place.
Duration: 10 days

Description: The initial status was that we were presented with an Access DB including the test-drive data for different vehicles, including numeric variables and free-text comments. The challenge was to query the data efficiently to perform different analysis based on free-text queries. We developed a solution with an LLM-based Semantic Search model based on fine-tuning the LLM to automotive-based vocabulary, deploying the LLM to get embeddings of the query and the free-text fields, and performing Cosine similarity between the query and the free-text to get the top-n records matching the query utilizing Elastic Search. Next step was to visualize the outcome on Python Stream-lit application, dockerizing and deploying the different components. The approahc also included using MLFlow for experiment tracking.

Technologies Used:

- **Database:** Access DB
- **Natural Language Processing:** Large Language Model, Sentence-Bert.
- **Search Engine:** Elastic Search for indexing and querying free-text data
- **Similarity Measurement:** Cosine similarity for comparing query and text embeddings
- **Data Visualization**: Python Streamlit
- **Containerization:** Docker
- **Experiment Tracking:** MLFlow

Project: Quality control for food in real-time in a factory setting using Computer Vision

Industry: Food Manufacturing
Role: Project Manager, Machine Learning Engineer lead, and client SPOC
Team Structure: End-to-end development of the project from ideation phase to production. I am leading a team which consists of three developers and a team-lead.
Duration: 3 months - ongoing

Description: The initial status that we were provided with a problem of detecting quality issues with raw food running on a production line which was manually performed utilizing staff with high availability over the production line. We started by evaluating the production line in the factory for the cameras placement, taking multiple photos, creating a similar environment in the lab with conveyor belts and cameras for mimicking the production line status. Next, the team collected different images in different lighting conditions, speeds of the conveyor belt and introducing different error classes within the normal stream of food. The next step was utilizing Label studio to perform labeling on object level for the photos which were produced on mimicked production line. Afterwards, a YOLO model was trained to detect and classify different error classes (11 error classes). The outcome of the PoC met the acceptance criteria of the customer in terms of Precision, Recall, False Positives, and False Negatives. This started the actual project of transferring the outcome over the production line. For that, multiple cloud architectures including MLOps paradigms were proposed on AWS, Azure, and GCP along with the foreseen costs to choose the Cloud provider. Currently, the project is in the initial phase of development over the actual production line with the first food-type. The plan is to generalize the approach over 6 different food-types with different production lines over different factories.

Technologies Used:

- **Programming Language:** Python
- **Image Labeling:** Label Studio
- **Machine Learning Model:** YOLO (You Only Look Once) for object detection and classification
- **Cloud Computing:** AWS, Azure, GCP
- **Data Collection and Simulation Environment:** Custom lab setup with conveyor belts and cameras

- **Experiment and Model Management:** MLFlow for tracking experiments
- **Project Management:** JIRA, Confluence

Project: Machine Learning Workshop for the Data Science and Analytics Department

Industry: Defense and Security
Role: Lead Data Scientist
Team Structure: Together with a colleague of mine we led the workshop including 24 engineers and Data Scientists.
Date/Duration: 1 month

Description: The initial situation included a motivated team of 24 members who had access to data and resources with a good domain knowledge, but with less prioritization and technical knowledge about Machine Learning techniques. We delivered initially a three days' workshop where we started by refreshing and teaching the basics for Machine Learning, and presenting different examples of successful Machine Learning applications to spark the interest. Next, we divided the participants into teams of 4 and did an ideation workshop where we tried to spark their creativity to come up with use-cases which could be turned into Machine Learning use-cases. The ideation workshop resulted into ~90 ideas. Afterwards, we prioritized the use-cases according to different metrics including time-to-deploy, data availability, impact, and effort. This resulted in a top priority use-cases per team. Next step was to plan the PoC and success criteria for the teams. We spent a month of coaching with the different teams to help them implement the PoCs. The outcome of the workshop was a successful 5 out 6 use-cases, which made it to production afterwards, a backlog of prioritized use-cases, and an exemplary use-case per-team for reference to the next use-cases.

Technologies Used:
- **Programming Language:** Python
- **Data Processing:** Pandas, PySpark
- **Machine Learning Libraries:** SKLearn
- **Containerization:** Docker

Project: Prediction of Failures of Military Vehicles and Equipment.

Industry: Defense and Security
Role: Senior Machine Learning Engineer
Team Structure: Developing with a colleague of mine, and mentoring a junior colleague.
Date/Duration: 8 month

Description: The initial status was having different CSV files including the failures for military vehicles. We started by building and deploying the infrastructure and services for the database, data ingestion and data pipelines. Next was defining the data model for the available vehicles and equipment to design the three data layers. Afterwards, we developed the data pipelines for

handling the ingestion into different data layers in Python. The first RAW layer included the data imported from the CSV files. The pipelines for transforming the data into the second ODS layer was doing translation to English language, simplification and acronyms unrolling, data augmentation, and casting the data into the correct data types. We additionally, performed different analysis on the datasets, including mean time to failure (MTTF), mean time between failure (MTBF) and came up with an efficient algorithm to provide the prediction. Those different analysis tasks were stored in the final Analysis layer. Additionally, connectors for the simulation software were developed, as well as a library for the Data Scientists to be able to easily load different data sources. Finally, dashboards for visualizing the outcome from the different analysis, as well as, from the ODS were designed and developed in Python Streamlit.

Technologies Used:

- **Programming Language:** Python
- **Data Modeling:** yEd for ERD diagrams
- **Database:** PostgreSQL
- **Data Analysis:** MTTF (Mean Time to Failure), MTBF (Mean Time Between Failures)
- **Data Visualization:** Python Streamlit for Dashboards

Scout24

Project: Architecting and Developing an AI platform for end-to-end Data Science

Industry: E-commerce
Role: Machine Learning Engineer
Team Structure: Team of three Machine Learning Engineers and a part-time product owner.
Date/Duration: 1 year 3 months

Description: The initial status included multiple Data Science efforts and models developed and deployed utilizing different approaches. We started by developing the model development life-cycle to allow different teams to build and deploy Machine Learning models efficiently and in a scalable-manner.

- Self-service notebooks: A self-service notebooks solution which is started via a TeamCity build running an AWS Cloud Formation template which starts an AWS Sagemaker instance connected to An AWS EMR instance utilizing Apache Livy.
- Self-service real-time model serving: A real-time stack for model-serving utilizing AWS Lambda for inference and authorization, AWS API Gateway for API serving, AWS Route53 for DNS, and Cloudwatch for monitoring.
- Self-service deployment for Lambda layers: A Team-City build which is parameterized for the libraries which need to be installed for the deployment stacks. The libraries are provided as Lambda layers for the
- Model monitoring library: Python library for performing common model monitoring tasks for detecting drifts and uploading metrics to CloudWatch.

- Self-service command-line-interface: Command line for deploying, installing, and running any of the AI Platform services.

Additionally, we were responsible for conducting workshops for different teams to explain Machine Learning concepts, onboard the teams on the AI Platform, and discover/prioritize further use-cases which could be developed and deployed on the platform.

Technologies Used

- **Cloud Computing and Services:** AWS SageMaker, AWS EMR, AWS Lambda, AWS API Gateway, AWS Route53, AWS CloudWatch, AWS S3
- **Programming Language:** Python, Bash
- **Web Frameworks:** Flask
- **Infrastructure as Code (IaC)**: AWS CloudFormation
- **Continuous Integration:** TeamCity, Jenkins

Project: Deploying and Scaling a client-facing service for estimating the cars prices

Industry: E-commerce
Role: Machine Learning Engineer
Team Structure: Team of two Machine Learning Engineers
Date/Duration: 6 months

Description: The initial status was 781 models performing the task of predicting a car's listing price based, they were merged into a single model which was able to perform different predictions treating each model as a categorical variable with an improved performance on RMSE basis. The next step was deploying the model intro production, the stack implemented was a Flask, uWSGI, Nginx, with a swagger UI. The application was unit-tested via Pytest, and implementing a pact-python contract for integration tests with different modules. Infrastructure-wise, the microservice was deployed on AWS ECR/ECS implementing load-balancing. The application stack was written as IaC via AWS CloudFormation. Blue-green deployment was used to roll-out updates, with zero down-time. Application monitoring was done via CloudWatch metrics, connected to AWS SNS topics. The application was running 24/7 with high availability SLAs.

Technologies Used:

- **Programming Language:** Python
- **Web Development**: Flask, uWSGI, Nginx
- **API Documentation:** Swagger UI
- **Testing:** Pytest, Pact-Python
- **Containerization and Orchestration:** Docker, AWS ECR (Elastic Container Registry), AWS ECS (Elastic Container Service)
- **Load Balancing:** AWS Load Balancer
- **Infrastructure as Code:** AWS CloudFormation

- **Monitoring and Alerting:** AWS CloudWatch, AWS SNS (Simple Notification Service), Datadog
- **CI/CD:** Jenkins

Project: Migrating central Data Warehouse into a DataLake on AWS

Industry: E-commerce
Role: Data Engineer
Team Structure: Team of 10 Data Engineers, Lead Engineer, Engineering Manager and a Technical Product Owner. I started as a working student, then shortly got the responsibilities for a Data Engineer role.
Date/Duration: 8 months

Description: Since the company was migrating from on-premise servers to AWS, the on-premise Oracle Data-warehouse was planned to be migrated towards AWS. For that, an AWS S3 DataLake was designed to include raw, and processed partitioned Parquet/ORC files. Access management controls were governed through IAM. GDPR regulations were implemented to separate the PII data, and the obfuscated/non-PII data. Right to be forgotten was implemented via obfuscation of the personal identifier information fields, and connecting them to real fields via a look-up table; if the right to be forgotten was demanded, then the link between the real and obfuscated fields is deleted. Data pipelines were re-written in Pyspark and orchestrated via AWS Data Pipelines. Microstrategy was used as a BI tool, connected to Presto DB. Since Presto DB didn't support materialized views, data pipelines were implemented for materializing the views into static tables.

Technologies Used:
- **Cloud Storage:** AWS S3, DataLake
- **File Formats:** Parquet, ORC
- **Access Management:** AWS IAM (Identity and Access Management)
- **Data Processing and Orchestration:** Python, PySpark, AWS Data Pipelines
- **Compliance and Data Management:** GDPR Regulations, Data Obfuscation Techniques
- **Business Intelligence Tool:** MicroStrategy
- **Query Engine:** Presto DB
- **CI/CD:** Jenkins, TeamCity

Project: Data Platform and Data Pipelines Development on AWS

Industry: E-commerce
Role: Data Engineer
Team Structure: Team of 10 Data Engineers, Lead Engineer, Engineering Manager and a Technical Product Owner. I started as a working student, then shortly got the responsibilities for a Data Engineer role.
Date/Duration: 1 year

Description: The responsibility of the team was developing a Data Platform including multiple services like a Personal Analytics cluster based on AWS EMR, interfaced via AWS CloudFormation and parameterized using TeamCity builds. Another custom service is a wrapper around AWS Data Pipeline to allow for parameterizing Data Pipelines easily and efficiently to perform generic ETLs from one S3 bucket to another. The wrapper was written in Python, transformations in Pyspark, interfaced via CloudFormation and parameterized using Bash and TeamCity. Additionally, among the responsibilities were developing Pyspark Data Pipelines for extracting data from different sources, including S3 buckets, Oracle, and APIs like Google Analytics.

Technologies Used:

- **Cloud Computing:** AWS EMR (Elastic MapReduce)
- **Infrastructure as Code:** AWS CloudFormation
- **CI/CD:** TeamCity
- **Scripting Language:** Python, Bash
- **Data Processing:** PySpark
- **Data Sources:** AWS S3, Oracle, Google Analytics APIs
- **Data Management:** Parameterized ETLs for Data Transformation and Loading