Model Deployment with Python

Learning Outcomes

By the end of this topic, you will have achieved the following learning outcomes:

- I can define model deployment in data science.
- I can explain the importance of the model deployment process.
- I can understand the steps taken when deploying a model.
- I can prepare code in preparation for model deployment.
- I can create a web service to deploy a model.

“Predicting the future isn’t magic, it’s artificial intelligence.” ~Dave Waters

Overview

This session will aim to demystify what it means to deploy a model, what factors to consider when deploying models, the tools and frameworks to utilize.

Reading

What is Model Deployment?

Model deployment is the integration of a machine learning model into a production environment with the goal of making practical business decisions based on data.

This is the last step in the data science life cycle / CRISP-DM lifecycle and requires coordination between data scientists, IT teams, software developers, and business professionals to ensure the model works reliably in the organization’s production environment.

The Importance of the Model Deployment Process

It's important to seamlessly deploy models into production so that a business or organisation can start to use them to make critical decisions could lead to operational effectiveness or profitability / better financial health.

Some of the needs that might arise while performing model deployment include:

- **Data and Feature Preparation:**
    - Ensuring that the data collected for input is appropriate for the model. Some data inputs are might be unstable and perhaps changing over time, thus there arises a need for tracking feature engineering and selection changes.
- **Detecting Model Errors:**
    - Ensuring that model errors as a result of deploying the wrong version of a model, forgetting a feature, and training on an outdated dataset are mitigated.
- **Separation of Expertise:**
    - As the deployment of machine learning systems requires cooperation between multiple teams, there's a need for methodologies which would help avoid failures and general inefficiencies from the teams working with each other.
- **Entanglement:**
    - There's a need for designing systems that allow for the tracking of feature engineering and selection changes while a model is in production.

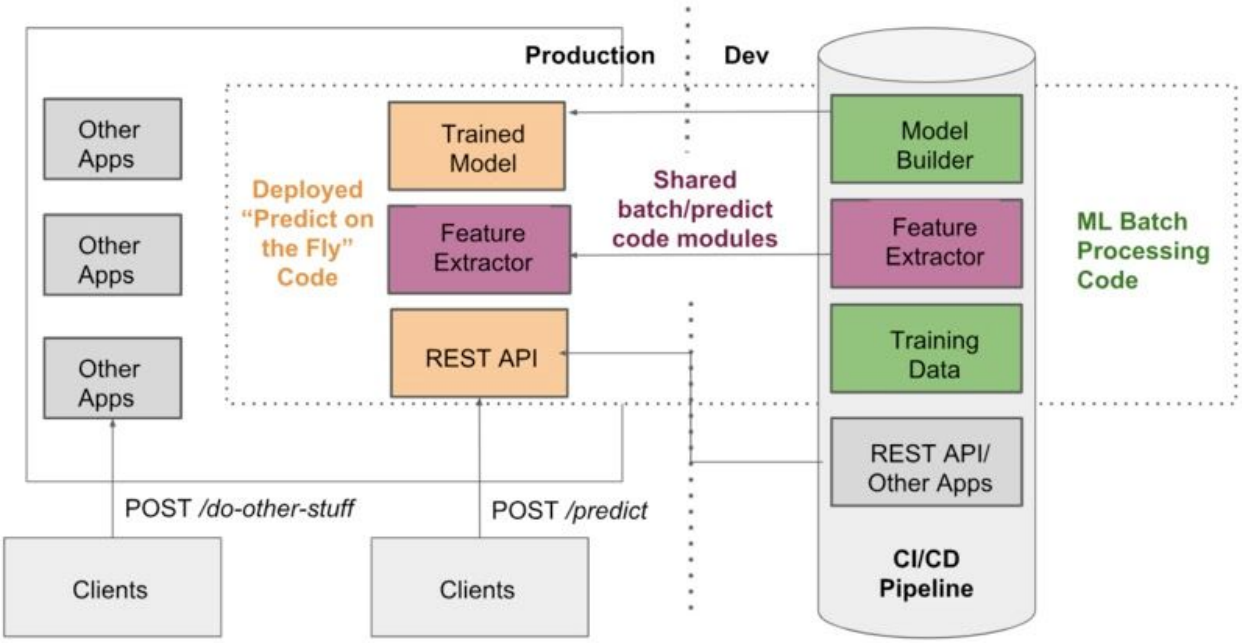
Machine Learning Deployment System Architecture

Upon the creation of our best machine learning model, we then need to prepare our model for deployment. Below are some of the common steps that we can take to achieve this goal:

1. **Step 1:** We start by ensuring that the code which will be taken into production takes into account of the outlined key principles outlined in the next section below.
2. **Step 2:** We then package our model in a file known as a pickle file.
3. **Step 3:** We then build our server infrastructure (where now the model resides) that allows us to be able to deploy our model solution for use by external applications. This can be done via a RESTful API or a web service. A web service is an application that runs on a server and allows a client (such as a browser) to remotely write/retrieve data to/from the server. A RESTful API - is a method of web communication that allows data to be exchanged between applications via a representation of the state of the requested resource (i.e. JSON Format).

4. **Step 4:** New input users are then sent to the server, and where preprocessing steps happen and the passed to the model which returns the prediction.
5. **Step 5:** The prediction is then sent back to the external requesting application via a Restful API or back to the web service where it's displayed.
6. **Step 6:** Lastly, we create systems that will allow us to monitor and maintain our model in production.

Below is an image that shows the structure of a typical model in deployment.



Key Principles For Designing a Machine Learning System

The following principles should be taken into consideration when designing a machine learning deployment system.

- **Reproducibility**: Model outputs/ results need to be reproducible. This ensures that the results are correct, transparent and gives us confidence in understanding exactly what was done. Thus metadata such as config, dependencies, geography, timezones needs to be well documented.
- **Plan for extensibility**: If models will be updated on a regular basis then data processing, feature engineering, modelling and optimisation techniques need to be designed from the beginning with consideration for extensibility.
- **Modularity**: For efficiency and simplicity purposes, preprocessing and feature engineering code from the research environment need to be reusable such that code duplication doesn't happen.

- **Testing**: The testing of various parts of your code needs to happen to ensure that the models work as expected. Testing in machine learning applications differs from software engineering testing.
- **Pipelines**: There is a need for the creation of reproducible pipelines for the model. In some cases, this can be a regulatory requirement which would involve controls and versioning on files, databases, S3 buckets and other data sources used during your model training. Aspects of the data science process encompassed in the pipeline include gathering data sources, performing data pre-processing, variable selection and model building.

Tools for Performing Machine Learning Deployment

Containers

- Containers are a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. Yes, you can think of a container as a portable/shareable environment that contains all the software you need to deploy your model.
- Reproducing containerized systems is much easier because the container images/copies ensure operating system and runtime dependencies stay fixed. The ability to consistently and quickly generate precise environments is a huge advantage for reproducibility during testing and training. Docker and Kubernetes are container technologies that can be used.

Continuous Integration and Deployment (CI/CD)

- In a team, continuous integration (CI) is a development practice where code is submitted into a shared repository frequently, preferably several times a day with each integration being verified by an automated build and automated tests.
- This allows for the quick detection of errors in the code thus allows for the team to move fast while keeping high-quality standards that can be checked automatically.

Testing

- As mentioned testing / automated testing allows for the quick detection of errors in the code i.e. whether new code causes an unexpected outcome or behaviour in another part of the code/ model solution.
- In addition to the traditional testing strategies used in software engineering such as unit/integration/acceptance tests, machine learning deployment requires for the use of differential, benchmark and load/stress tests.
    - **Differential tests:** In this type of tests, a comparison is made between predictions given by a new model vs predictions given by the previous model. These tests can be vital for detecting otherwise healthy-seeming models, for example where an outdated dataset has been used in training, or a feature has been accidentally removed from the feature selection code.

- **Benchmark tests**: These tests compare the time taken to either train or serve predictions from your model from one version to the next. They prevent you from introducing inefficient code additions to your ML applications.
- **Load/Stress tests:** These tests help assess the need for CPU/Memory needs of a machine learning application.

Model Deployment Strategies

While performing model deployment strategies, a deployment plan needs to be formulated and would include summarizing the deployment strategy by performing the following activities:

1. Developing and evaluate alternative plans for deployment.
2. Deciding on each distinct knowledge or information result.
3. Determining how knowledge or information will be propagated to users.
4. Deciding how the use of the result will be monitored and its benefits measured (where applicable).
5. Establishing how the model result will be deployed within the organization's systems.
6. Determining how its use will be monitored and its benefits measured (where applicable).
7. Identifying possible problems during deployment (pitfalls to be avoided)

Monitoring and Maintenance

Monitoring and maintenance are important issues if model results become part of the day-to-day business and its environment. Careful preparation of a maintenance strategy helps to avoid unnecessarily long periods of incorrect usage of model results. In order to monitor the deployment of the data mining result(s), a detailed plan for monitoring and maintenance needs to be taken into account.

This would involve the following:

1. Checking for dynamic aspects (i.e., what things could change in the environment?)
2. Deciding how accuracy will be monitored (metric score to be used)
3. Determine when the data mining result or model should not be used any more. Identify criteria (validity, the threshold of accuracy, new data, change in the application domain, etc.), and what should happen if the model or result could no longer be used. (update model, set up a new data mining project, etc.).
4. Will the business objectives of the use of the model change over time? Fully document the initial problem the model was attempting to solve.
5. Develop a monitoring and maintenance plan.

References

You can also use the following references for your further reading:

1. You can also use the following references for your further reading:
2. Machine Learning Model Deployment [Link]
3. Deploy a Machine Learning Model [Link]
4. How to Deploy Machine Learning Models [Link]
5. 150 successful Machine Learning models: 6 lessons learned at Booking.com [Link] and Paper [Link]