Analysis of Software Artifacts

CMU SE 17654

Data and Model Quality

J. Gennari / C. Kästner / S. Echeverria / M. Velez Due: Dates shown on syllabus

Overview

For this assignment you will dabble with machine learning. Pressured by competitors like Google Nest you want to make your smart home system actually smart. You have many ideas and are looking into hiring some data analysts that help you with the modeling for specific tasks, but as a early risk reduction step you want to demonstrate feasibility and establish that you have the capabilities for automating and testing the learning process.

As a test project you plan to build a model that predicts at what time the user of the smarthome system will turn on their lights in the morning and turn them off at night. If the models work out well, you want to integrate them in your system as smart alarm clock or energy savings and nudging[1] features. You will build a prototype model and automate and test the learning steps as well as build the infrastructure to serve the model and monitor model quality in production.

Learning goals:

- Apply the basic and iterative machine learning steps to create models from data
- Integrate a ML model into a software product
- Design and apply quality measures for model quality, both on a static dataset and in production
- Design and apply automation and quality assurance steps on the entire machine learning pipeline to gain confidence in the quality and reliability of the system

Tasks

Unfortunately, you still do not have a full running system and user data yet and your simulators are too simple to produce meaningful data. Instead you have convinced a number of your employees and friends to install a small data logger in their home, which logs every minute whether any lights are on. This has given you observations from 6 people in the Pittsburgh area over a few month that can be downloaded at:
https://drive.google.com/open?id=1Tm2pliySPYiThg2JRPIJ49tRl8prYc1X

[1] https://en.wikipedia.org/wiki/Nudge_theory

You will use this or part of this data set to build a prediction model, will build a REST API to allow other parts of your system to query predictions from your model, and integrate the model in your system, for example showing the predicted times for activating and deactivating the lights on the user's dashboard. As usual the focus of this assignment is on quality assurance though, so you will develop and implement a strategy to assess model and data quality, to assess the quality of the modeling infrastructure, and to monitor model quality in production.

Part 1: Modeling Pipeline and Static Dataset

- **Step 1: Basic Modeling.** Build models that predict when your smart-home users turn on the lights in the morning and when they turn them off at night. You will likely want separate models for morning and nights and per user, but you can explore other solutions. As part of the process you will need to extract features from the provided raw data and perform some data cleaning. Keep it simple for now.
- **Step 2: Serving the Model.** Build a REST API service that other parts of the system can use to query model predictions, for example, to get the predicted time when a specific user would turn on the light in the morning on a specific date (and possibly more context information). Document your API and deploy it as a Docker container in your infrastructure, ensuring that you can serve updated models with minimal downtime. Integrate this with your A/B testing infrastructure from the previous assignment, such that you can serve different models to different users.
- **Step 3: Evaluate Model Quality on a Static Dataset.** Design a strategy to evaluate the quality of your models on the given datasets and to compare the quality of two versions of a models. This will involve many design decisions, for example, how to split the data into learning and evaluation set, what quality measures to use, or whether to report quality per user or for all users.
- **Step 4: Automate the ML Pipeline.** Automate all previous steps, such that you can automatically build models from datasets, evaluate their quality, and release them through the REST API. As part of the process you should version data and models, such that you can reproduce models and roll back changes. Also, you should track quality measures in Jenkins (ideally plot model quality measures over revisions or variants).
- **Step 5: Quality Assurance for Your ML Pipleline.** Devise and implement a strategy to assure the quality of your ML pipeline (distinct from the model quality assessed in Step 2). As a minimum you likely want to target for reasonable test coverage of all steps in your pipeline that you developed yourself (feature extraction, learning, serving, ...) so that you are confident that it also works (or correctly identifies problems) with other training data in the future.[2]
- **Step 6: Improve your models.** Finally, with the confidence of a safety net from automation and tests, improve your model. This will likely involve some experimentation

[2] For example, does the feature extraction mechanism correctly handle cases where invalid dates are provided or other schema violations occur? How does it handle redundant, conflicting, or missing data? What happens when the model cannot be produced? The provided dataset contains various issues, but others could easily happen in future datasets.

and feature engineering (new features, different operationalization of features). For this assignment potentially interesting features might include: day of the week, when other people get up, holidays and other important days, location information and corresponding sunset and sunrise times, weather, and typical sleep duration.[3] We will assign up to 5% bonus points if you use at least one external data source (such as sunset times or weather) as a feature. Make sure that the resulting modeling pipeline undergoes the same quality assurance steps planned in Step 5.

Part 2: Monitoring in Production

- **Step 7: Integrate the Predictions.** Integrate the predictions into your platform. For example, show the predicted times for the next day or week on a user's dashboard. (You can assign users to specific users from the provided training set.)
- **Step 8: Monitor Model Quality in Production:** Build the infrastructure to monitor model quality in production (e.g., based on the data you collect in the mysql database). This will involve building an analytics service that can report model quality over time. You should be able to detect models that degrade in prediction accuracy over time.

Note that the current simulation infrastructure does not produce much meaningful "live" data. For testing purposes, you may want to reserve some of the provided data to use as "live" data to populate your database. For example, a good monitoring system would detect that model quality suddenly drops if you start replaying data with 1h delay or replaying data from a different user.

Given the companies current lack of deployed systems or meaningful simulators, you do *NOT* need to perform the following tasks: Learn and serve new models from live data for all users; conduct actual A/B tests or canary tests; check data quality of live data (e.g., schema conformance or expected distributions).

Hints:

- Keep the machine learning and feature engineering part simple. We do not require or expect a deep understanding of machine learning technology and it is sufficient to use learning tools that provide learners for decision trees, regression models or similar basic techniques as black boxes. It is sufficient to demonstrate that you can improve your original models when adding additional features, but it is not important to optimize model performance. We will not grade the accuracy of your models but your quality assurance efforts and infrastructure.
- We will demonstrate Weka in recitation, but feel free to use other technologies. We do not have a requirement on specific technologies or languages for any parts of this assignment. Feel free to explore existing open-source infrastructure, but in many cases it might be easier to develop simple solutions yourself.

[3] See hints below.

- We require that you use at least one external feature not derivable from the provided dataset (e.g., sunrise time), but leave it up to you how you acquire and integrate that data.
- If you haven't done so already, you likely want to split a single Jenkins task into more specific tasks, such that you can update the model without updating the rest of the infrastructure and vice versa. You might also consider revising the structure of your Git repositories (e.g., use a single structured repository vs many specific repositories).
- We encourage you to continue to use the Docker infrastructure from the previous assignment and create additional containers for the API, the monitoring infrastructure, and the learning components.

Deliverables

Push all code to your github repositories' main branch and tag the final revision with "G3_done".

Create a report as a single PDF that covers the following with a clear structure (graphics and screenshots do not count against the page limits):

- **Model description** (<1 page of text): Describe how you model the problem. What is the predicted outcome, what are the features used for prediction, how are those features collected, and what learning technique is used? Briefly justify your design decisions.
- **Model quality measure** (<1 page of text): Describe how you assess the quality of your model on the static dataset. What metric(s) do you use, how do you split learning and evaluation set, and how to you report or aggregate measures for morning vs evening and different users? Briefly justify your design decisions.
- **Model quality results** (<1 page of text): Report your model accuracy according to the previously described measures and show at least two results for different versions of the model. Include a screenshot of your Jenkins integration and provide instructions (e.g., links, accounts) how we can access your jenkins configuration.
- **ML Pipeline** (<1 page of text): Describe all components of the pipeline you develop and engineering decisions you have made (feature collection, learning, model serving, versioning, A/B test integration, etc). Briefly justify your design decisions. Include a pointer to the documentation of your REST API and explain where to find artifacts that are not committed to GitHub.
- **Quality assurance of the ML pipeline** (<1 page of text): Describe your quality assurance plan for the pipeline (e.g., reviewing policies, testing goals, performance goals, automation goals) and provide appropriate evidence of the achieved quality.
- **System integration and monitoring** (<1 page of text): Briefly describe your system integration and monitoring infrastructure (include screenshots where appropriate). Include how you measure model quality based on production data. Additionally, provide evidence that the monitoring infrastructure can detect degradation of the model predictions.

Create a 10 min presentation (PDF) that covers at a minimum: (a) an overview of your pipeline and key decisions you made, (b) your quality assurance decisions and outcomes, and (c) lessons learned.

Upload report and presentation as a zip file to Canvas before the deadline.