Erik Lundin

2021-07-15

The Lightweight IBM Cloud Garage Method for Data Science

Architectural Decisions Document Template

Project: Predicting Heart Attacks using Machine Learning

For context, this document explains Architectural Decision made to implement a machine learning application that helps users of IoT devices to predict whether they have a higher change of having a heart attack or not. I have chosen this project because it is an interesting topic, and one that could help save lives.

# Architectural Components Overview



IBM Data and Analytics Reference Architecture. Source: IBM Corporation

## Data Source

### Technology Choice

Please describe what technology you have defined here. Please justify below, why. In case this component is not needed justify below.

The source of the data for my use case varies. Since the data is used for predicting probability of heart attacks in patients, the data source could be various IOT Devices, such as measuring devices, mobile phones and computers. This is mainly during inference though.

The training data source comes from a dataset found on Kaggle. Since this dataset is small, it does not need to reside on a cluster of nodes powered by technology stacks like Apache Hadoop + Spark. In other words, LocalStorage is used for training data storage, which is enough since the dataset is so small.

### Justification

As stated above, I am dealing with a smaller dataset so LocalStorage as a data source is enough

## Enterprise Data

### Technology Choice

The enterprise data will likely come from a large amount of IOT Devices that are connected to health care – blood pressure measuring devices, doctor’s computers, but also potentially patients mobile phones and sensors/devices that patients use at home to monitor their health. To handle this enterprise data, I have chosen to use a server that is reachable over HTTP so that IOT devices can send feature data to the server, which can then perform inference on the model and return a prediction to the device which requested it. This device, whether it is a computer or a phone or a smart watch, can then display to the used if he or she should visit a doctor to get a heart check since they run a higher chance of having a heart attack.

The data that will be sent to the HTTP server should then be stored in a SQL database, since the data is suitably modelled in a relational database and the relational structure will not change over time. Specifically, it should be stored in a PostgreSQL database, since they are powerful and can handle the large amounts of data that will potentially be sent from users

This SQL Database can then easily be connected to the HTTP Server that runs the model

### Justification

Using a relational database fits the context since data will only be numerical values, so ObjectStorage is not needed. Furthermore, a SQL Server suits the use case of sending prediction requests over HTTP well: Everytime a Doctor/Patient using an IoT device sends their data, it can be used to make a prediction by the model, and the server can also store the data in the SQL database for later use. Since the database schema is well-structured, SQL will also be benefitted by the use of secondary indices and provide high search speeds

## Streaming analytics

### Technology Choice

Since the architecture doesn’t require devices to stream data, streaming analytics are not needed

### Justification

Please justify your technology choices here.

## Data Integration

### Technology Choice

The data was integrated into the model by using python and several data science and machine learning libraries:

* Pandas
* Numpy
* Scikit-learn

### Justification

First of all, pandas provides a really powerful way to process large datasets that are in a row-column format, by using it, processing data, looking for missing values, merging datasets and exploring data becomes easy and intuitive

Numpy is a fast and powerful numerical library for python which provided good tools for feature extraction, transforming the data (PCA, scaling etc). It is therefore really useful in the context of data integration

Scikit-learn is really useful since it provides ways to transform data (e.g. normalizing). Since it is of paramount importance that the scaling operations are the same on the data that is used for real life inference as the data used for training, scikit-learn is useful since it allows the user to save the scalers for later use. Therefore, we can

* Fit a scaler to the dataset used for training
* Scale the data
* Save the scaler
* When the model is used on real life data for inference, the saved scaler can be used to scale the data to be suitable for the model
* Inference can then be performed properly

## Data Repository

### Technology Choice

As stated, the data repository was stored using LocalStorage

### Justification

Since the dataset was only 300 entries long, LocalStorage was sufficient

## Discovery and Exploration

### Technology Choice

For discovery and exploration, python along with Scikit-learn, pandas, numpy has been used

Methodologies have included counting NaN values, looking up what different features mean, correlation plots and Principal Component Analysis

### Justification

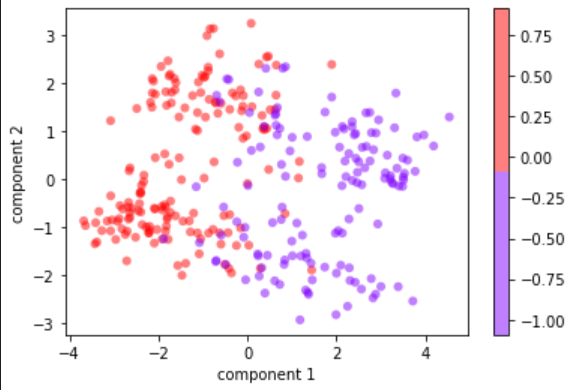
**Data Quality Assessment:**

Pandas provides ways to check the data for NaNs and Null values, which was useful for assessing data quality

Numpy provides functions for performing Principal Component Analysis and calculating the correlation, which was useful to determine the quality of the features and how the correlated with the label

It is extremely important to understand the dataset before using it in the model, for instance, two features were categorical values which are hard to interpret by models if they are left as-is. By understanding that these features are categorical, we can use methods to engineer the features in a way that makes them usable by our models

Checking for NaN values is very important since they can completely break our model if they are included in the training dataset.

PCA is a powerful way to visualize high dimensional data and can also be used to transform the data before training. In my case, the data is very high, dimensional, and PCA provides a way to visualize the data despite this fact: 

Correlation plots can tell us how much the features are correlated to the label. It can tell us if a feature will only bring noise to the model instead of helping it, which means that we by then removing that feature can increase model performance.

## Actionable Insights

### Technology Choice

Please describe what technology you have defined here. Please justify below, why. In case this component is not needed justify below.

The technology used for this part, including engineering the data, has been numpy, scikit-learn and pandas. The methods used for feature engineering have been:

* Normalization
* One-hot encoding
* Adding a bias term

### Justification

Without normalizing the data, some features may overpower the model, (compare feature ‘Age’=64 to a binary feature=1) . My normalizing the data we circumvent this issue and make sure that all features are valued equally by the model when they are fed into it (not accounting for the parameters)

Since some features are categorical, it is absolutely necessary to process these features in a way that makes them usable by the model. In my case, what I mean by one-hot encoding is

{categorical features with values between 0 and 3} -> binary values in a 4-row vector. For example

[3, 2, 0] -> [[0, 0, 0, 1] [0, 0, 1, 0] [1, 0, 0 ,0]]

This can significantly improve model performance. In my case, it increased accuracy by about 2%, which could mean a difference between life and death when predicting heart attacks.

Adding a bias term is also absolutely crucial, since without it all predictions will make models decision boundaries in hyperspace able to form a hyperplane that does not intercept the origin.

## Applications / Data Products

### Technology Choice

For the application, I have/will use the following technologies:

Flask

SQL database

For the data product, I have used:

Pytorch

Scikit Learn

Specifically, the model I have chosen to use is a Support Vector Machine model with a second degree polynomial kernel.

### Justification

Flask and SQL are a powerful combination due to their ease of use, and the fact that they are both running in python. This makes integration between model and server seamless

For the data product, AKA model, the support vector machine proved to be the best performing one. I have justified this by running 10-fold cross-validation on the dataset with all the different models that I have tested, namely

* Logistic Regression
* SVM:s with 3 different kernels
* Decision trees with 2 decision criterions
* 2 different neural networks implemented in PyTorch

I have used test accuracy as my main performance metric since it is extremely important that the model used performs as well as possible (it is about life and death)

Out of all the models, the polynomial SVM performed best in terms of accuracy. This justifies the fact that I decided to use that model. Other than that, it is also a very fast classifier compared to e.g. a neural network, and it does not require that much memory to run since the only parameters are the support vectors. This means that the model could theoretically be deployed onto smaller IoT devices and be used to perform inference locally, which is very useful for users without an internet connection.

## Security, Information Governance and Systems Management

### Technology Choice

The technologies that are related to Security, Information Governance and Systems Management are Flask and SQL database

### Justification

To make sure that the application is secure, users should authenticate themselves using HTTP authentication. This authentication can be linked to a persons SSN to make sure that data sent to the server is legitimate, and it makes sense if we want to store each individual patients data.

To prevent hacker attacks, the SQL server should have the flask server as an intermediary endpoint. This ensures that SQL injections attacks are avoided.

When it comes to systems management, the server could with benefits be running on a cluster that supports SQL databases, for example Apache spark could be used since it would allow the data to be distributed as a RDD on several nodes. It would also work well with the SVM since an ML pipeline could then be set up to streamline the flow of data to the model and the outputs of the model to the user.