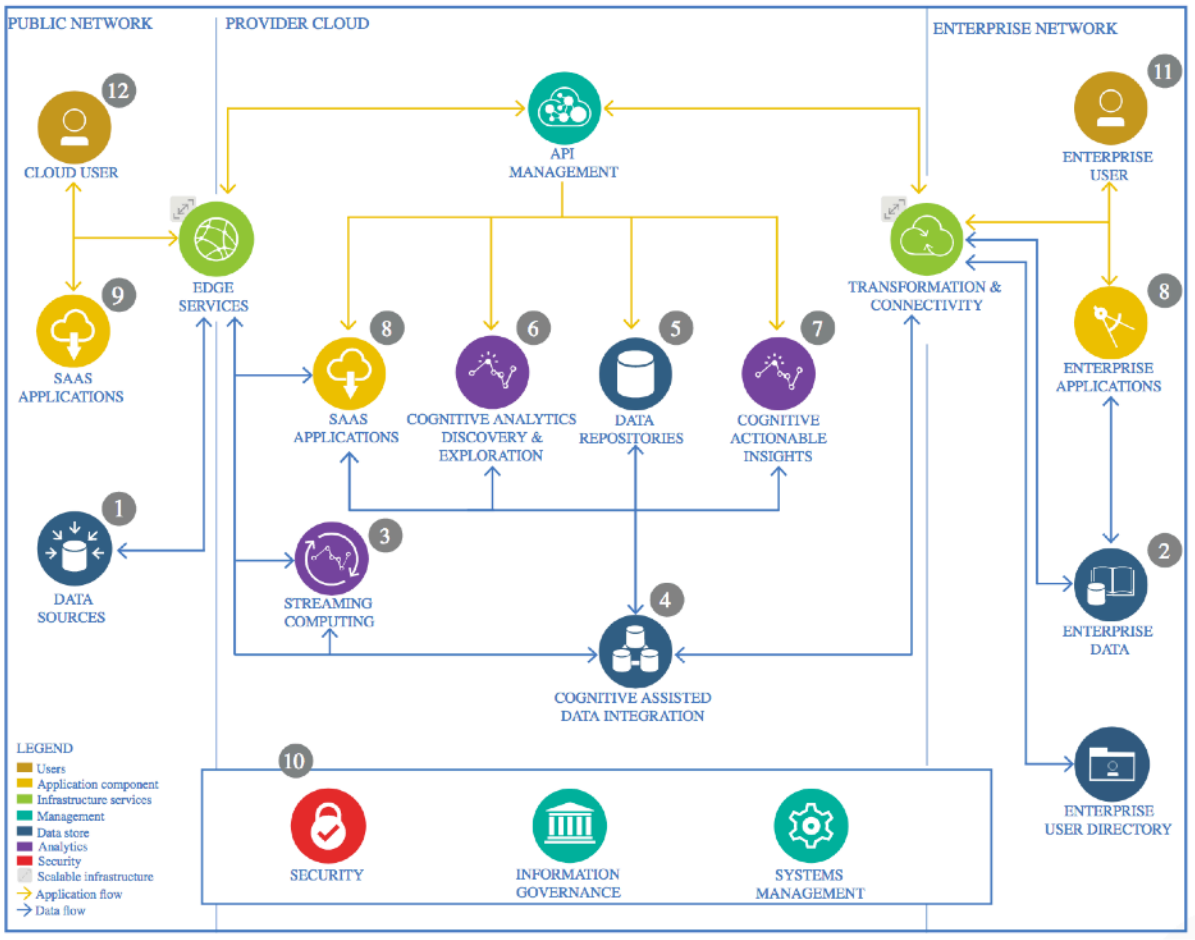
EXOPLANET HUNT — BINARY CLASSIFICATION OF STAR SYSTEMS

The Lightweight IBM Cloud Garage Method for Data Science

Architectural Decisions Document Template

https://developer.ibm.com/articles/data-science-architectural-decisions-guidelines/

1. Architectural Components Overview



IBM Data and Analytics Reference Architecture. Source: IBM Corporation

1.1. Data Source

1.1.1. Technology Choice

The data for this project comes from kaggle.com and was originally proposed by NASA. The dataset is called Exoplanet hunting and consists of light intensity

measurements from 5657 star systems and corresponding binary labels: 1 — the star does not have any exoplanets in its system, 2 — it does. Thus, the aim of the project is to be able to distinct between those two with a sufficient degree of certainty based on the light intensity data. Every star contains 3197 time measurements taken with an average interval of ≈ 80 minutes.
Dataset URL: https://www.kaggle.com/keplersmachines/kepler-labelled-time-series-data

1.1.2. Justification
In my opinion, this is an interesting exercice: measurements for each star are light intensities in different moments of time. Which means, that direct approach will most likely not work as information is stored in the signal in a sort of hidden way, however, there are many opportunities to incorporate feature extraction or data augmentation techniques.

1.2. Enterprise Data

1.2.1. Technology Choice
It was decided to store the data at the cloud, using IBM Watson Studio and its Cloud Object Storage.

1.2.2. Justification
Initial data is represented by 2 csv files which take $\approx 300MB$ and may be downloaded directly from Kaggle, using its API or from NASA site.
By storing it at the cloud, one may perform any kind of processing or modeling and profit from a variety of IBM services at the same time.
It also allows one to access some nice parallelization features in case they are needed.
Finally, it is especially practical and even essential for research teams where each member may work from a different place or even continent.

1.3. Streaming analytics

1.3.1. Technology Choice
The chosen approach is batch processing. It is considered to be state-of-the-art for the moment not without a reason. It is reliable, efficient and fits perfectly the studied case.

1.3.2. Justification
In fact, there is even no option to include real-time data streaming in this case. The reason for that is that the data comes originally from NASA Kepler space telescope. It is transmitted down to Earth, where NASA applies particular de-noising algorithms to remove some unnecessary elements, produced by telescope activity.

So, this data comes a long way before becoming available, which means one can only wait for a new Kepler mission to acquire a new batch.

1.4. Data Integration

1.4.1. Technology Choice

The data comes pretty clean in its initial form already. There are no missing values and values of all the columns are coherent. However, there are way less data on star systems witho planets than on those without exoplanets, so one needs to squeeze the maximum of the available instances. This is why, various processing techniques were used as a part of feature creation.

1.4.2. Justification

It was decided to work with 4 kinds of data.

1) Row-Normalized data. This may be considered as “clean”, as normalization is a necessary step to feed the data to the ML algorithms. It is row normalized, as each row represents a new star, so it makes no sense to mix light intensities of different stars, considering they may be of an entirely different order.
2) Detrended data. This is a common signal processing technique. Firstly, one computes a smoothed version of the signal; it was chosen to do that using gaussian smoothing, but there are numerous possibilities. Then the smoothed signal is removed from the original one and normalized. In such a way one obtains a kind of high-pass filtered data. Finally, as we are looking for the moments of time where the star is eclipsed by one of its planets, we are interested in drops in light intensity; spikes present no interest. So they are clipped.
3) Frequency data. Instead of working in time domain, it was decided to compare the efficiency of passing into frequency domain. To do that one may perform FFT or Laplace Transform for example. The first one was chosen for its computational efficiency. However, initial signal contains too many frequencies, so this operation is conducted on the detrended data.
4) Wavelet-transformed data. This one is especially interesting to test, as it somehow combines both time and frequency domains: after applying CWT for each row one obtains a scale-time matrix of coefficients. Although this approach is more time-consuming, it may have generated some additional insight. In the ETL notebook, one shows how easily two scalograms of each type of stars may be distinguished between each other, so if a network succeeds in learning these differences, one may expect a very powerful result. Despite the fact that these data was produced locally and tested briefly, it has proven to be very impractical in usage: it can not be converted into .csv files, which demands different approach, and the training is significantly longer.

1.5. Data Repository

1.5.1. Technology Choice

After ETL is done, all the kinds of processed data are stored as ".csv" files in IBM COS. For safety and universal possibility of access all the pieces of data also cloned to Google Drive and to local repository.

1.5.2. Justification

As such problem would most likely occur in the domain of scientific research, it seems to me, that storing data on cloud is most practical and available solution.

The dataset is not gigantic, as it was mentioned before, so there is no need to introduce spark data frames for the current goal. Nonetheless, it was tried and proved to not be worth at such scale due to low computational gain and not so practical row oriented data format. Probably the limitation of free plan are the reason for that, as well.

Approximately $5.5GB$ of space is needed to store the original data and every processed variation, including both train and test samples (with wavelet augmented data taking $4GB$).

Pandas is strict requirement to be able to successfully manipulate the data in the given project.

In case of growth of the dataset and new data being discovered, COS is also a very nice choice as it supports simple addition of a new ".csv" file.

1.6. Discovery and Exploration

1.6.1. Technology Choice

In this section one performed a variety of different plots using Python's Seaborn and Matplotlib libraries to obtain an idea of how the data belonging to two classes are different.

1.6.2. Justification

Taking into account the nature of the data, it does not make much sense to perform statistical analysis of the whole dataset. One may of course look at the mean or standard deviation across all the stars for, let's say, moment of time $t = 10$, but what possible information might it result in ?

On the other side, it is perfectly fine to compare these various statistical measures computed over each row, between multiple stars. So, a brief analysis of such type is performed.

Moreover, data visualization produces a much more valuable insight. It allows one to conclude, for instance, that:

- Stars with exoplanets data is likely periodic;

- There is usually a considerable difference between scales and variance of data of two kinds;
- Smoothing or filtering leads one to an assumption that data from planetless systems contains more of high and mid frequencies;
- Scalograms emphasize the differences between classes to even greater extent.

1.7. Actionable Insights

1.7.1. Technology Choice

To resolve this problem 4 different models were trained and tested on every variation of produced features. Those models are:

- ➡ Multilayer Perceptron (3 dense hidden layers) — *baseline model*
- ➡ Convolutional Network (3 convolutional layers and 1 flattening Dense)
- ➡ LSTM (2 LSTM layers and 1 flattening Dense)
- ➡ Support Vector Classifier (basic setting from sklearn)

First 3 ones are implemented using Keras API of Tensorflow. However, custom generators are used instead of those offered by tensorflow. They include the possibility to boost the minority class which is very helpful with such a huge difference in occurrences of two classes in the present dataset. Generators are common in Python and with a must-have knowledge of Numpy it is simple to create or maintain one.

Convolutional network has shown the best results in most of the cases and so it was chosen as a final model. In a separate notebook grid search was applied to find best model and training meta-parameters.

To compare models and evaluate their performance one needs to define a set of measures that will serve this purpose.

Having in mind the objective of a problem - binary classification - we suggest next indicators could be used as evaluating measurements of the models:

1. Loss - Binary CrossEntropy
2. Accuracy
3. Sensitivity & Specificity
4. Precision & Recall
5. F1-score

1.7.2. Justification

As for models choice — those are basically most common models that one should test when working with time series kind of data.

As for the framework — both Keras and sklearn are universally accepted as data science standards and offer an incredibly wide range of possibilities. Both support parallel training in case it's needed. Tensorflow and Keras may also be accelerated with GPU execution.

All of those models may be serialized and saved on the disk.

1.8. Applications / Data Products

1.8.1. Technology Choice

Models are fine, but their value rises when they can be consumed by the ordinary user. In the current use case, there is really no 'ordinary' user, but still model is way more practical to use when it is deployed and the data product is created. So it is done, using Watson Machine Learning Service.

After the deployment, model was tested and has produced the expected result. The next step would be to push it to the application.

1.8.2. Justification

WMS is an obvious choice since one already uses IBM Cloud's infrastructure in all the elements of the project. The deployment is simple and fast and clear tutorials from IBM themselves are available online.

1.9. Security, Information Governance and Systems Management

1.9.1. Technology Choice

This dataset is open for anyone to use and does not contain any secret information, thus, does not need any special security. The whole projected was conducted by myself, the only person who has access to the current data product. All in all, security breach is possible and there is, most likely, no person who might be interested in it.

1.9.2. Justification

This dataset is open for anyone to use and does not contain any secret information, thus, does not need any special security.