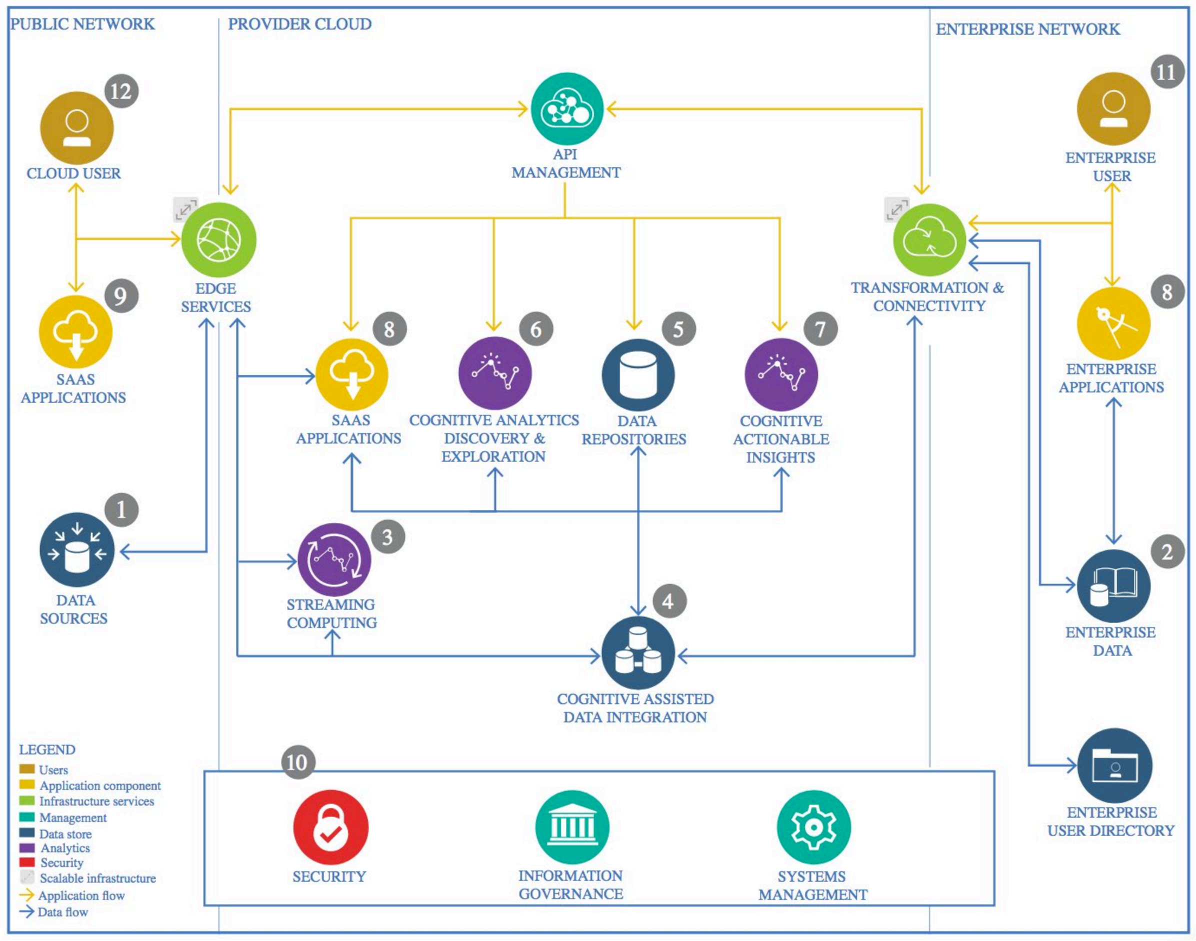
The Lightweight IBM Cloud Garage Method for Data Science

Architectural Decisions Document Template

1 Architectural Components Overview



IBM Data and Analytics Reference Architecture. Source: IBM Corporation

1.1 Data Source

1.1.1 Technology Choice

The data source used was the Daily Global Top 200 Rankings from Spotify for the years 2017 to 2021. It was downloaded from the following Kaggle link: https://www.kaggle.com/datasets/c0lydxmas/spotify-top-200-daily-global-2017-2021.

Each date's data was stored in a separate JSON file and thus needed to be looped through, normalized, and then combined to form each year's complete dataset. No enterprise or streaming analytic data was used in this analysis – only the JSON files obtained from the above Kaggle link.

1.1.2 Justification
I chose this dataset because of my interest and background in music. The goal with this dataset was to determine whether my models could effectively distinguish between Top 100 and Bottom 100 songs given the features present in the datasets. Then, the Feature Importances of that classification would be used to inform business insights, recommendations, and future improvements.

1.2 Data Integration

1.2.1 Technology Choice
For data integration, as referenced above, a custom Python ETL script was written to loop through all the JSON files for a particular calendar year, normalize the JSON within each file, cleanse the normalized data, and finally concatenate the data together to form a structured CSV dataset for each year. This ETL script was run on my local machine.

1.2.2 Justification
The choice was made to build a custom ETL script because of practicality and my background knowledge of the subject matter. Building and running a custom script allowed me to extract the specific/key nested information needed from each file. It also avoided uploading 1000+ individual JSON files onto the IBM Cloud just to run the IBM Watson ETL tool. In the end, I was happy with my decision as it extracted the data that I was looking for and packaged it neatly into 5 CSV files for subsequent steps.

1.3 Feature Engineering and Processing

1.3.1 Technology Choice
For Feature Engineering, a number of different steps were taken. After removing records with Key Feature Information missing (ex: Artist Name, Genre, etc.), missing values were imputed as "NABlankValue" to prevent model training from failing due to the presence of null values. Genre information was extracted, sorted, and then replaced in order to allow the model to properly compare genre information. Flags for "# of genres" and "multigenre" were created as additions to the 12 individual genre columns. Finally, songs that were released outside of the dataset year were removed in order to avoid penalizing Top 100 songs with notable longevity. I.e. being miscategorized as primarily Bottom 100 due to staying on the charts for "too long".

1.3.2 Justification
Of all the feature engineering steps mentioned above, the biggest impact was made by genre extraction, sorting, and replacement. Prior to doing so, the Random Forest models were performing in the mid 70s. After doing so, the Random Forest models began performing closer to the mid 80s. This was a very important step because the previous alphabetical sorting was not allowing the models to properly make use of those features.

1.4 Data Repository

1.4.1 Technology Choice

The choice was made to house the finalized CSV files on the IBM Cloud via Cloud Object Storage.

1.4.2 Justification

This was done in order to be able to easily access consume them in the Apache Spark environment set up on the IBM Cloud. The 5 CSV files were no more than 15MB in size so storage capacity and cost were also not a concern at the free tier.

1.5 Discovery, Exploration, and Data Quality Assessment

1.5.1 Technology Choice

For Discovery, Exploration, and Data Quality Assessment, the SweetViz Python library was leveraged. SweetViz generates descriptive statistics and visuals for datasets and subsets given the labels being trained on.

1.5.2 Justification

SweetViz is a very powerful tool that generates all the Discovery, Exploration, and Data Quality Assessment elements needed using just a fraction of code. For this analysis, via SweetViz, I looked at the following components for each feature of the 5 datasets: Correlation Matrices (Heatmaps), Distribution Bar Graphs (Histograms), Percentile Splits, IQR, Mean, Median, Mode, Standard Deviation, Skew, Kurtosis, Distribution Shape, Missing Value Counts, and Frequent Values. Together, these components allowed me to understand what the distribution looked like for each feature, and whether there were any major flags to be aware of. The added benefit of being able to run subset analyses with SweetViz was that additional insights could be gleaned once the model was trained (as seen in the PDF Report and Presentation).

1.6 Actionable Insights

1.6.1 Technology Choice

Three algorithms were chosen as part of this analysis: Support Vector Machine (SVM) via Spark, Neural Network Perceptron via Keras, and Random Forest via Spark. PySpark and Keras were chosen as the frameworks, and IBM Cloud was chosen as the environment to house the overall project. “Area Under the ROC Curve” was chosen as the “Model Performance Indicator”, and “Feature Importance” was chosen as the main driver for final Actionable Insights and Recommendations.

1.6.2 Justification

PySpark and Keras were chosen as the frameworks in order to put into practice the learnings from the IBM lectures, and IBM Cloud was also chosen in order to put into practice the learnings from the IBM lectures as well. “AUROC” was chosen as the “Model Performance Indicator” because it looks at a model's ability to discriminate between two classes while also taking into consideration any imbalance in the dataset itself. The datasets that were used for this project (2017,2018,2019,2020,2021 Daily Global Top 200 Data) were not perfectly balanced when it comes to label distribution – thus “AUROC” made sense.

SVM was chosen to provide a baseline using strictly numerical features within the dataset. RF was chosen as the final algorithm because there are a number of key high-cardinality, non-numerical features within the dataset. These types of features work well with tree-based algorithms, but not as well with non-tree-based algorithms due to the need for indexing and one-hot-encoding. When used on high-cardinality features, OHE is incredibly inefficient and can actually even cause performance degradation of a model.

The tree-based algorithm implementations within Spark utilize metadata to remove the need for OHE after StringIndexing. This improves efficiency and performance, and is part of the reason why RandomForest was a great choice as the final algorithm for this project. The other reason RandomForest was a great choice is that the ensemble nature of the RandomForest algorithm also helps improve performance over simpler tree-based algorithms like DecisionTrees.

Finally, a Neural Network Perceptron was also implemented in order to practice the Deep Learning teachings from the Advanced Data Science Specialization course material.

1.7 Applications / Data Products

1.7.1 Technology Choice

The Data Product decided upon for this project was an exported PDF Report of the Jupyter Notebooks and an exported PDF of the Final Slide Deck Presentation.

1.7.2 Justification

As mentioned in the Model Deployment Guidelines, "The key to everything is that the business insights that result from the model are made available to stakeholders. This can happen in various ways. At the simplest level a PDF report is generated… and handed over to business stakeholders".

The Jupyter Notebook PDF report and the Final Slide Deck PDF are perfect ways to detail every step of the project while also highlighting the final business insights to be considered for end users.

1.8 Security, Information Governance and Systems Management

1.8.1 Technology Choice

Cloud Object Storage was used to handle all data access within the IBM Cloud. Before uploading my code to my GitHub, all access tokens and IDs were redacted in order to maintain privacy and security. I have chosen to share my code on GitHub for others to learn from as well.

1.8.2 Justification

Because this dataset is publicly available on Kaggle, there is not really a privacy concern with regards to the data. The redaction of tokens and IDs helps maintain the privacy and security of my IBM account, however.