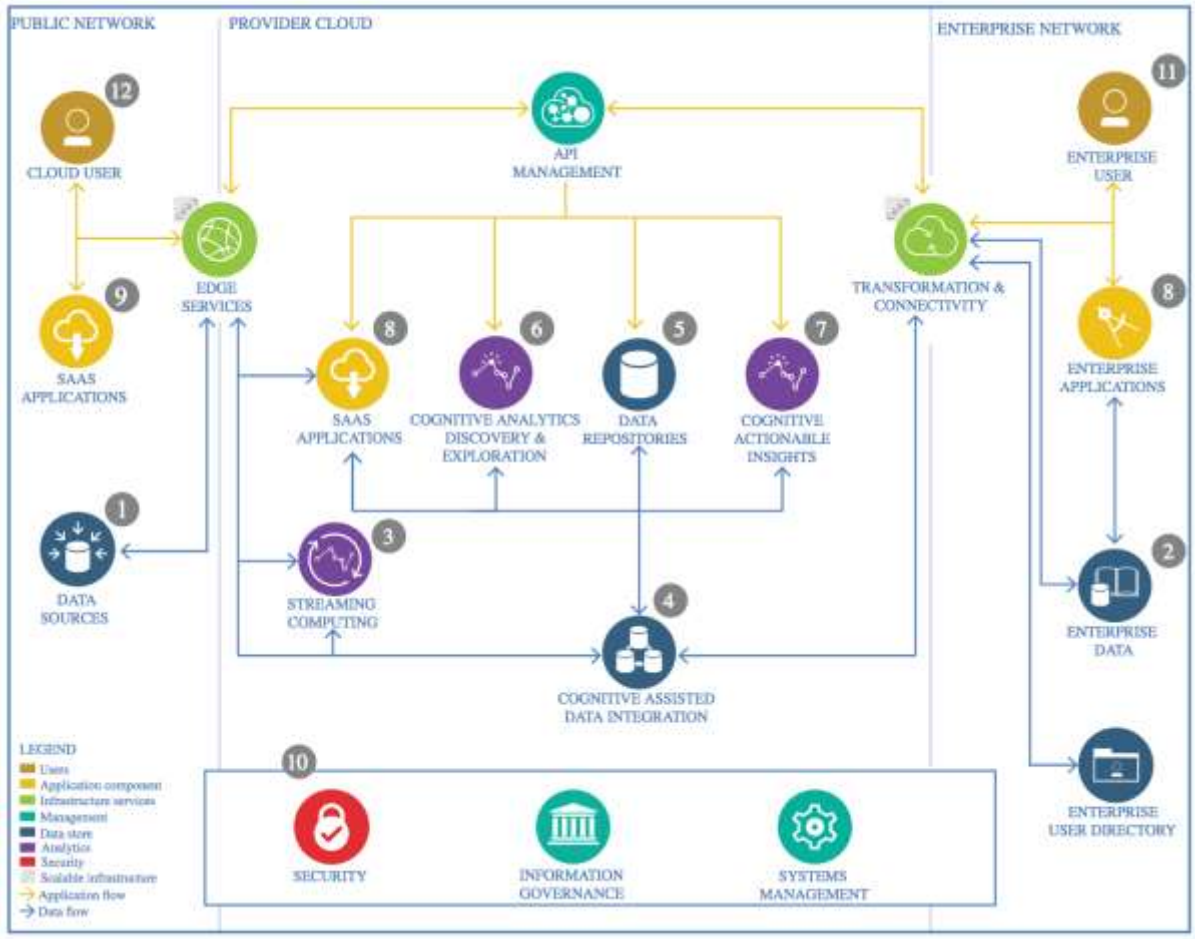
The Lightweight IBM Cloud Garage Method for Data Science

Architectural Decisions Document Template

I am going to predict the propensity to buy a first property by non-property individuals.

I have extracted the data from my company's database and saved it as a CSV file on the IBM cloud storage. I will use this data to develop a propensity model for the Capstone project.

1 Architectural Components Overview



IBM Data and Analytics Reference Architecture. Source: IBM Corporation

1.1 Data Source

1.1.1 Technology Choice

I extracted the data from my company's MS SQL Server database and save it as a CSV file in.

1.1.2 Justification

Please justify your technology choices here.

CSV is a very easy-to-use file format and it's suitable for small to medium size datasets. It can be read easily in many programming languages such as Python.

1.2 Enterprise Data

1.2.1 Technology Choice

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

I used IBM cloud storage to keep the original CSV file, and converted it to Apache Parquet file format for faster operation by Apache Spark framework.

1.2.2 Justification

Please justify your technology choices here.

Apache Spark is accessible in Python by pyspark library. It was a good practice to write the entire modeling code using pyspark library.

1.3 Streaming analytics

1.3.1 Technology Choice

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

1.3.2 Justification

Please justify your technology choices here.

No Streaming analytics was used in this project.

1.4 Data Integration

1.4.1 Technology Choice

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

For the Capstone project I decided to use distributed computing. I used IBM Watson studio, a ready-to-use environment for data scientists, and created Jupyter Notebooks for each modeling stages there. I used Python3.6 with Spark 2.4 runtime to run the code and develop models, and used IBM Cloud Storage to store the files.

1.4.2 Justification

Please justify your technology choices here.

I decided to code in Python since it's a well-developed programming language and has a very rich machine learning libraries and packages. I used Python 3.6 runtime to write a compatible code with most recent versions of Python.

In addition, I used several libraries, mainly Pyspark built-in machine learning (ML) libraries. In order to develop a deep learning model, I used Tensorflow 1.x and Keras. To run Keras model in a distributed environment, I used Elephas library, a Distributed Deep Learning with Keras & Spark library, since the SystemML was only compatible with Python 2.x.

Despite the fact that Elephas worked flawlessly on my local machine (used Spark 2.4.5 locally), I had issues in IBM Watson Studio so couldn't train the Keras model there. As a result, I trained and evaluated the models on my local machine and uploaded the final notebook in IBM Watson Studio.

Regarding the data used, it had some missing values, which were imputed. In addition, there was a highly correlated features pair so one was removed. Moreover, categorical data were transformed by One-Hot Encoding technique.

1.5 Data Repository

1.5.1 Technology Choice

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

I used IBM Cloud Storage to store the files.

1.5.2 Justification

Please justify your technology choices here.

The original data was in a CSV file, but it was less efficient as the Spark task was running slowly. So the cleaned data was stored in Parquet format, which could speed up the tasks.

1.6 Discovery and Exploration

1.6.1 Technology Choice

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

1.6.2 Justification

Please justify your technology choices here.

I created several diagrams by matplotlib and seaborn libraries in the initial data exploration stage. I found outliers using univariate analysis and histograms plots and treated them. I also performed the bivariate and pair correlation analyses and found one highly correlated pair of features.

1.7 Actionable Insights

1.7.1 Technology Choice

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

1.7.2 Justification

Please justify your technology choices here.

The current dataset was rich in features and had various numerical (age, income_code, LSM, etc.) and categorical (gender, directorship, etc.) features. However, I created an age_group feature to squash the wide range of ages in few categories.

By using Pyspark ML library and pipeline, I transformed categorical and numerical variables into Spark vectors and created 45 features from 22 initial features.

Some machine learning algorithm like neural networks are required to have the normalized data; but my main focus was on Logistic Regression and Tree based models, so I didn't include the normalization step in the pipeline. However, I checked the effect of normalization and noticed that it hurt the model performance. As a result, I didn't keep the results in the Notebook to keep it tidy and clean.

In order to develop a baseline model quickly, I chose the Logistic Regression algorithm. Despite its simplicity, it outperforms Decision Tree (which is also a quick algorithm) in terms of prediction power over the test set in several cases, especially for a dataset with high number of features.

I also used Pyspark Gradient-Boosted Trees (also known as XGBoost) algorithm to check if I could have a better model. In addition, I also used Keras framework to develop a deep learning model (feed forward sequential neural network), combined with Elephas library (Distributed Deep Learning with Keras & Spark). I created a three-layer neural network model with 64 and 32 neurons for the first and second layers.

Finally, I allocated 80% of the data for the *training* set and 20% for the *test* set.

The steps taken to reach the final model is described below.

Step 1 – Initial training & evaluation

As mentioned, I developed three models using different ML algorithms. The accuracy score for all models over *training* and *test* sets are shown below.

	Accuracy score	
	Training set	Test set
Logistic Regression	0.8895	0.8847
Gradient-Boosted Trees	0.9031	0.8975
Feed Forward Neural Network	0.8925	0.8870

As can be seen, all models have almost similar performance. However, the complexity of the model so the computation expense increases from top to bottom. So I decided to work more on Logistic Regression model because its development is faster, and it is easier to implement in all programming languages. Since I am using a MS SQL Server database, it is much easier to calculate the propensity to buy score for the entire database by implementing the Logistic Regression model in a SQL query.

Step 2 – Feature importance evaluation

It is so easy to look at Logistic Regression model coefficients and see which features have less or no effect on the model performance. In this regard, I sorted the coefficient and plotted them so found four features had very low or no importance in the model. So I removed months_home_phone, months_mobile_phone, months_work_phone, and contactability_score features, and re-train the Logistic Regression model and got following results.

	Accuracy score	
	Training set	Test set
Logistic Regression	0.8909	0.8871

As can be seen, there was a very slight increase in the model accuracy using 4 less features. So I reduced the features used from 45 to 41, without losing the performance.

Step 3 – Hyperparameter tuning by Gridsearch

At this point, I used Spark tuning library to perform the Gridsearch and Crossvalidation in otder to tune two parameters in the Logistic Regression model: regParam, and threshold. Sometimes, the default threshold of 0.5 does not give the maximum performance so it should

be tuned. By using the *accuracy* score during the Gridsearch, I found the best parameters are as follow:

regParam = 0.01

threshold = 0.5
The best model used these parameters had following performance.

	Accuracy score	
	Training set	Test set
Logistic Regression	0.9002	0.8969

Gladly, the simpler and less complex Logistic Regression model got same performance as the Gradient-Boosted Trees. So I can use and implement it confidently to score all records using a simple SQL query.

Despite being happy with the results, it is worth to perform an additional training/evaluation step.

Step 4 – Dimensionality reduction (PCA)

It was decided to perform principle component analysis to check if the model performance can be increased more, especially for the Gradient-Boosted Trees model. It was found that 4 principle components could explain 97% of variance in the data; but the training and evaluation showed that the developed models had poor performance. As a result, 10 principle components, which explained 98.9% of variance, were used.

Using 10 principle components (features) decreased the training and evaluation time slightly, but the performance did not rich to the previous step. Below is the result.

	Accuracy score	
	Training set	Test set
Logistic Regression	0.8972	0.8944
Gradient-Boosted Trees	0.8991	0.8925

Finally, I decided to use the Logistic Regression modeled developed in the previous step as the final model.

1.8 Applications / Data Products

1.8.1 Technology Choice

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

1.8.2 Justification

Please justify your technology choices here.

In order to deploy this application, I saved the final model in the Parquet format so it can be loaded by Pyspark easily. The data is fed as a CSV file via a Jupyter Notebook and the result is saved as a CSV file too.

1.9 Security, Information Governance and Systems Management

1.9.1 Technology Choice

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

1.9.2 Justification

Please justify your technology choices here.

No special security is required since the data in not confidential.