Big Data Project Proposal - By Luis Severien Marcilla and Jelena Stankovic

High level requirements: Motivation and goal

In big cities such as New York one is overwhelmed by the traffic and just happy to get safely and fast from point A to point B. Especially when it comes to hailing a cab there are other worries than calculating prices. Still, there are big differences in fares which the consumer might want to take into account when deciding about if he is going to travel by a NYC Cab, Uber or just may take the metro.

The most prominent way to travel New York is in a typical yellow NYC Taxi. The fares of these taxis can easily be calculated retrospectively (the composition of the price will be presented in the analytics part and can be looked up here: de.wikipedia.org/wiki/New_York_City_Taxi_Cabs), but a prediction turns out to be difficult due to uncertainty of travelling time through NYC traffic and the actual distance. Some tourists might also not have in mind that there are extra fees charged for rides at night and during rush hours.

To create some transparency, we are going to create an application for the consumer, which will enable him or her to foresee the NYC Cab price for a future ride. The application uses basic information, such as the time of the day, the pick-up and the drop-off coordinates and forecasts the corresponding prices of the NYC TLC (Taxi and Limousine Commission) company. Thanks to the application the user will be able to do an improved assessment of transportation options, since it reveals the fare amount of a cab ride as well as its expected time of no or little velocity of motion during the ride for a certain route. Due to the latter, the consumer could not only compare prices, but also do own estimations of total travel time for the requested taxi ride and also examine the durations of transportation alternatives in order to select the option, which fits the consumer's needs best.

The main requirements for a solid base to build the just described application on are a data set comprising a sufficient amount of travel data and a continuously updating as well as improving model for the price estimation. The used data set does not only need to include the relevant information (date, time, pick up location, drop off location, fare amount, distance travelled), but also needs to contain a high density of data tuples in order to use it to simulate a real time data pipeline (as described in the analytics part). The estimation model shall take into account the user inputs from the application and return the requested price and waiting time information or the calculation components for just those.

In the following we will use and refer to the NYC taxi dataset, which is publicly available (www.nyc.gov/html/tlc/html/about/trip_record_data.shtml) and comprises data covering more than one billion taxi rides from 2009 to 2017. These huge amount of past data will enable us to create a predictive price model for future rides.

Implementation requirements:

Analytics

We will use *scikit-learn*, Python's machine learning library for clustering and regression analysis, as predicting the taxi price is a regression problem (supervised learning problem). In a first step, before examining the possible regression variables, we use the general information which is already given:

The taxi price is calculated on the basis of several variables. One has to add together the basic price (2.60$), the NY state tax per ride (0.55$) and the surcharges for the night shift (0.80$ for rides between 20:00h-6:00h) or the rush hour (1.00$ for rides between 16:00h-20:00h on weekdays). Of course the price also depends on the miles driven (0.60$ per 1/5 mile) and on the time in case of driving slow or standing in traffic (0.50$ per 60 seconds). The last variable brings uncertainty into the price calculation, which offers room for predictions. The formula for calculating the fare is to follow:

$$Y = 3.15 + 0.8d1 + 1d2 + 0.6 * 5 * x1 + 0.5 * x2$$

Y: Price in $

d_1: Dummy parameter for the night shift

d_2: Dummy parameter for the rush hour

x_1 : Distance travelled in miles

x_2: Waiting time in minutes

We can determine all variables with certainty except for the waiting time. For each tuple of data we know the distance travelled and the duration of the ride, so we can decompose the final fare by subtracting all the factors, until we are left with the waiting time x_2 (in other words, solve the equation for x_2). This is the dependent variable in our regression model that we want to estimate. We assume that the time of the day and the pick-up and drop-off spots are the independent variables which will have an impact on the waiting time. These variables should give us a clue about the typical traffic conditions between the neighborhoods at different times of the day. The exact coefficients will be determined with a regression and adjusted by additionally incoming data. Finally, we will recompose the price after estimating the waiting time, as this is our final output for the consumer.

Prior to the regression we will have to cluster the drop-off and pick-up coordinates into small neighborhoods, as working with exact latitudes and longitudes would not deliver any further insights. One can as well treat the locations in a certain radius as the same pick-up or drop-off spot as the taxi fare should not change within this radius. We expect to have about 300 clusters, similarly to the zones contained in the datasets from 2016 on.

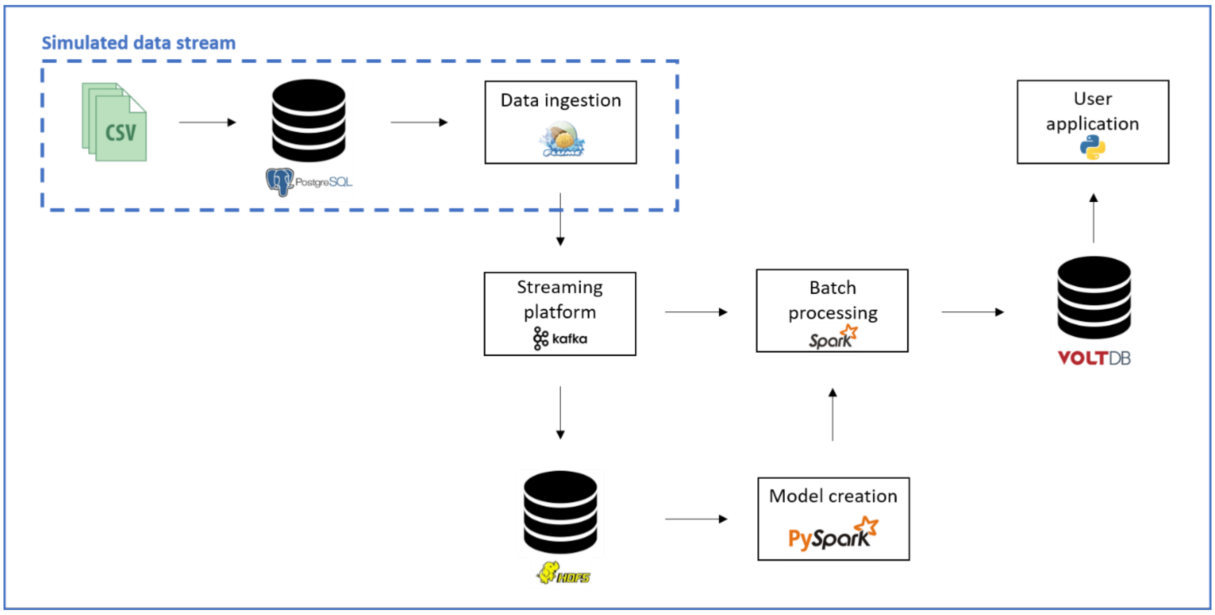
For the project we will use the datasets from 2009 until 2015. The data from 2009 until 2012 will be our training data, which we will use for modeling. Data from 2013 will serve as tuning data, so we can tune the parameters and eventually check the accuracy of our first predictions with the test data from 2014. We can evaluate our preliminary model by calculating its goodness of fit. There are various statistical coefficients for this purpose, like the R^2 or the RMSE. Nevertheless, the model is supposed to work and improve with streaming data, so we will use the 2015 dataset as a simulated data stream and adjust the parameters on a daily basis.

Architecture

The data set exists only as a collection of CSV-files which we will comprise inside a PostgreSQL database in order to provide a single point of access to the data source. As this is not a data stream we will have to simulate just this by ingesting it part by part into the streaming platform within a prespecified time frame. The ingestion will be done with batch jobs via Apache Flume using the datetime package to access the current time and to create the timestamps during the streaming. This construct allows us to artificially replicate a near real-time data stream.

The streaming platform into which the data is ingested to is Apache Kafka. Once the data arrives as a data stream in Kafka, the data will continuously be stored to a repository in HDFS as well as forwarded to Apache Spark. The storage in HDFS serves as an access point for the creation of the data model, which will be programmed in Python using the Apache Spark environment via the PySpark-API. Firstly, RDDs will be created from the data files in Spark in order to cut off unnecessary data, which will not be used in the model. Subsequently, the model picks up the remaining data in order to execute the analyses described in the analytics part. Once the model is established and tested, it will be passed to Apache Spark to be fed further streaming data from Kafka, which also will be pre-filtered through RDDs in Spark and adjust itself based on real-time data.

The data models relevant parameters and components will be stored on a VoltDB instance, which provides quick storage capacities and allows several applications to run analyses over the same data. This will be necessary, as the user application is also connected to this database and will pull the just mentioned parameters in order to calculate and display a result to the user. The user application will also be programmed in Python using the standard user interface libraries.



Basic project architecture