# 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 A to 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 [wiki/New\_York\_City\_Taxi\_Cabs](https://de.wikipedia.org/wiki/New_York_City_Taxi_Cabs" \l "Boro-Taxis)), 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 use the NYC taxi dataset and create an application for the consumer, which will enable him to foresee the NYC Cab price for a future ride. The application takes the time of the day, the pick-up and the drop-off coordinates and forecasts the prices of the NYC TLC (Taxi and Limousine Commission) company.

The publicly available dataset ([www.nyc.gov/html/tlc/html/about/trip\_record\_data.shtml](http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml)) comprises data covering more than 1 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.

High Level requirements:

A dataset comprising travel data

## 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). But before examining the possible regression variables, we use the information we have:

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: Price in $

d1: Dummy for the night shift

d2: Dummy for the rush hour

x1 : Distance travelled in miles

x2: Waiting time in minutes

We can determine all variables with certainty except for the waiting time. For each dataset we know the distance travelled and the time of the ride, so we can decompose the final fare by subtracting all the factors, until we get the waiting time x2 (in other words, solve the equation for x2). 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. Of course we have to 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 R2 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

As the given data source is not a data stream we will have to simulate just this by ingesting it part by part within a prespecified time frame (maybe even simulate a real data stream with timestamps from the previous year) into our data pipeline. The ingestion will be done with flume.

* Use the datetime package to access the current time and create the timestamps for streaming simulation
* Ingest data with Flume
* Connect to Spark and process with an API for Python

As the given data source is not a streaming data source we will have to simulate a stream by ingesting it part by part within some time frame (maybe even simulate a real data stream with timestamps that match the previous year).

1. The ingestion will be done with Apache Flume, the distributed Hadoop service for ingesting streaming data. As we only have data from one source and in one single format this should be relatively easy, so we refrain from using more complicated services like Kafka.

* save files on a local HDFS directory according to the timestamp. This local directory will be used by flume as the source (spooldir).
* Ingest data with Flume and write it on to HDFS (sink another HDFS directory or HBase).
* Connect to Spark (pick data from HDFS) and process with an API for Python – use pyspark
* Create RDDs in Spark from data files on HDFS. From those extract the values we need for the analytics part. Perform transformations on the RDDs so only the needed values are left (filter, then action “collect” which will return an array – store these arrays in variables which will be used to improve the model)