UDACITY MLND CAPSTONE PROPOSAL

Blake Ward
12/07/17

Domain Background
This project is derived directly from my day-to-day work. As a data engineer working on building out a Hadoop data lake, I am responsible for designing jobs that load data from various sources. These jobs run on a specific SLAs and need to be monitored for potential failures.

To date, we have not had a great way to know if a job was having problems. Specifically, we want to know if a job is "long running", indicating it may miss its window for completion. Recently we implemented a very simple algorithm to predict the run time for a job. It is based on average run times and is not very accurate.

Problem Statement
In this project the goal is to improve upon the predicted run time estimate for jobs that are used to load data into a data lake at my company. Our current predictions are not accurate and do not give us the ability to monitor job performance. We have a lengthy history of completed job runs and I believe this problem can be solved by building a better predictive model that will estimate run time at the start of a job.

Datasets and Inputs
The dataset for the project will come from our history of job executions. We have approximately 10 months of data to use. For this project, I will be focusing on jobs that move data from ERP databases to a Hadoop data lake. The type of data that will be included in the data set are things like run date, run time, number of tables ingested, min table size, max table size, average table size, database type, location of source system.

I'm hoping to keep the model agnostic of any specific job. We have 100s of jobs, and I'd like to be able to predict run time for a job that has never run before, meaning I want to base this prediction more on the database type than the specific job. This would remove the ability to use any job specific inputs (min table size, max table size, average run time) and may not be feasible. In that case, a job will need to have run at least once to get a prediction for its next run.

Solution Statement
My plan to solve this problem is to use a machine learning algorithm to output a predicted run time. I would like to be able to predict the final run time for a job at the job's beginning. The output will be a continuous variable in seconds.

Benchmark Model
We are currently implementing a simple algorithm to predict run time. This can serve as the benchmark model to compare the performance of our new model. The current model takes two inputs and outputs a predicted run time. The first input is the average run time – based on all previous runs – for all the tables to be loaded. The second input is the number of tables to

be loaded for the current run. These two inputs are multiplied to get the job's predicted run time in seconds.

Evaluation Metrics
As a regression problem, two good evaluation metrics for goodness of fir are the r-squared score and the standard error of the regression. These both can be applied the benchmark model and our new model for a comparison of performance. The r-squared score measures the fraction by which the variance of the errors is less than the variance of the dependent variable.

Project Design
This project will follow a fairly standard modeling flow of work. The step I will take are outlined below.

1. *Data Gathering*
   I already have the data stored on a MySQL database. It has nearly a year's worth of data on job statistics. I need to write a query to aggregate the data into a single data set for modeling. This should be straightforward

2. *Data Exploration*
   After I have the raw data that I am planning to use, my next step is to do so summary statistics and visualizations to get a better understanding of the distributions and any possible outliers. The raw data is directly from our operational tables, so includes data that might not be representative of "true" job runs that I want to capture. For example, I expect some jobs will have really long run times because of server failures or other technical glitches. I am going to have to give some careful thought removing outliers for this reason as I don't want to put too much unnecessary into my data while also retaining the true range of the data.

3. *Pre-Processing*
   At this step I plan to turn my raw data into clean data by removing any outliers, doing any encoding or standardizing that I need. There are some categorical variables that I know will need one-hot encoding so that is definitely one step I will be taking.

4. *Model Selection*
   The next step will be model selection. I have a feeling that some type of decision tree or ensemble method will prove to be my best option, but I want to explore the results of multiple algorithms. I also want to explore multiple algorithms because I am still learning that it is valuable experience. I plan to run through a number of regression algorithms to get a sense of where I should spend my time optimizing and tuning.

5. *Model Optimization*
   The last step in the workflow, at this point I will have chosen an algorithm based on earlier work and I will be tuning the parameters and any other optimizing that is available.