**Problem**

Let's say we have 1 million Lyft rider journey trips in the city of Seattle. We want to build a model to predict ETA after a rider makes a Lyft request.

How would we know if we have enough data to create an accurate enough model?

**Solution**

Collecting data can be costly. This question assesses the candidate’s skill in being able to practically figure out if a model needs more data. There are a couple of factors to look into.

- Look at the feature set size to training data size ratio. If we have an extremely high number of features compared to training data, then the model inaccuracy will be prone to overfitting.

- Create an existing model off a portion of the data, the training set, and measure performance of the model on the validation sets, otherwise known as using a holdout set. We hold back some subset of the data from the training of the model, and then use this holdout set to check the model performance to get a baseline level.

- Learning curves. Learning curves help us calculate our accuracy rate by testing data on subsequently larger subsets of data. If we fit our model on 20%, 40%, 60%, 80% of our data size and then cross-validate to determine model accuracy, we can then determine how much more data we need to achieve a certain accuracy level.

For example. If we reach 75% accuracy with 500K datapoints but then only 77% accuracy with 1 million datapoints, then we’ll realize that our model is not predicting well enough with it’s existing features since doubling the training data size did not significantly increase the accuracy rate.

**Problem**

Let's say that you work at a bank that wants to build a model to detect fraud on the platform. The bank wants to implement a text messaging service in addition that will text customers when the model detects a fradulent transaction in order for the customer to approve or deny the transaction with a text response.

1. What kind of model would need to be built?
2. Given the scenario, if you were building the model, which model metrics would you be optimizing for?

**Solution**

1. **Binary classifier**. Given that fraud is binary, there either is a fradulent transaction or there isn't.

2. There are a lot of different ways to analyze model performance but let's take into account what's specified. We know that in binary classification problems there are precision versus recall trade-offs.

**Precision** is defined as the number of true positives divided by model predicted positives. In our example this would be the percentage of correct fradulent transactions out of predicted fradulent transactions.

**Precision** = (True Positive / (True Positive + False Positive))

**Recall (= Sensitivity = True Positive Rate)** is defined as the number of true positives divided by number of actual true positives. In our example this would be the number of correct fradulent transactions out of actual fradulent transactions.

**Recall** = (True Positive / (True Positive + False Negative))

Given these two metrics for evaluating a binary classifier, which metric would a bank prefer to be higher? Low recall in a fradulent case scenario would be a disaster. With low predictive power on false negatives, fradulent purchases would go under the rug with consumers not even knowing they were being defrauded.

Meanwhile if there was low precision, customers would think their accounts would be under fraud all the time. But since the question prompts for a text messaging service, this would be okay since the end customer would just have to approve or deny transactions that were false fraud transactions.

