# Machine Learning Project Proposal

**The project's domain background — the field of research where the project is derived;**

The domain background for this project is the application of machine learning tools to predict prices for financial securities. For this project I took inspiration from a codepiece uploaded to machine learning platform Kaggle at https://www.kaggle.com/kimy07/eurusd-15-minute-interval-price-prediction/data and written by kimy07.

Our aim is to extend the analysis done in above code to a more granular (sub-second) dataset for EUR/USD with longer history (back to 2000). This should allow me to refine the model used by kimy07 and make better predictions.

From the literature on predicting financial security prices it appears the LSTM model (**L**ong **S**hort **T**erm **M**emory Network) can achieve better prediction capability compared to a simple Recurrent Neural Network and linear regression (http://colah.github.io/posts/2015-08-Understanding-LSTMs/) so we will use this approach. Other attempts at price prediction include bitcoin price prediction (http://trap.ncirl.ie/2496/1/seanmcnally.pdf), general evaluation of reinforcement learning algorithms on the foreign exchange market (https://www.doc.ic.ac.uk/teaching/distinguished-projects/2015/j.cumming.pdf), and stock market price prediction using LSTM specifically (http://ieeexplore.ieee.org/document/7966019/).

The dataset for this project is downloaded from http://www.histdata.com/download-free-forex-data/?/ascii/tick-data-quotes and consists of tick data - that is sub-second price quotes - for the EUR/USD currency pair since the beginning of 2000 up to the present day.

**A problem statement — a problem being investigated for which a solution will be defined;**

The problem we will attempt to solve is price prediction based on historical tick data in the EUR/USD market.

This is a regression problem, as we try to predict future prices based on past information encoded in state variables, and those variables are continuous.

The inputs for the problem are the millisecond date-time stamp, the bid and ask prices, as well as features I will derive from these such as open, close, high, low, bid-offer spread.

**The datasets and inputs — data or inputs being used for the problem;**

The dataset is sourced from http://www.histdata.com/download-free-forex-data/?/ascii/tick-data-quotes. It contains for the EUR/USD currency pair tick by tick bid and ask price data for every day since 2000.

The data will be accessed using a screenscraper to download each monthly ascii file which will be stored in a database for analysis.

I will use date, bid and ask prices for this problem. Date is a millisecond datetime stamp, bid and ask are floating point numbers. Thus all three can be said to be continuous. The dataset has around 126 million rows. The outcome variable is the future price in the next period. Given I have many examples, I will split the data 80-10-10 into training, validation and testing.

To prevent the algorithm from using future information to predict previous prices, I will attach actual future prices as target labels to each historical window I use for training the model. I will experiment with different window ranges as training sets, each being a multiple of 15 minutes.

Regarding class balances, there should not be any price groups in this dataset – it is unlikely that there is a “preferred price” at which the currency pair trades. To stationarize the data, i.e. to give no particular preference to absolute price levels, we will predict returns instead of absolute price values. This also leads to balanced classes, as both positive and negative return between ticks are approximately equally likely. However, there is a slightly fat tail of very negative returns, so I may have to deal with those.

I am aware that attempting to use only past prices to predict future prices relies on the prices being Markov, that is all information necessary for price determination at a given time is already reflected in the price, and therefore price is a useful proxy for all current information in the state space.

However, this fact is fairly well established as the price reflects information held by a large community of market participants in the worldwide, continuously traded, foreign exchange market - the most liquid market in the world, usually trading at tight (0.0005 USD per EUR) bid offer spreads in the G10 currencies.

The risk with this approach is that new information that becomes available to market participants between ticks, will lead to unpredictable price changes that my model cannot capture. Thus a shortcoming of this project will be its focus on extrapolation of past events to predict the future, ignoring changes in an alternate state space that may have stronger predictive power.

**A solution statement — a solution proposed for the problem given;**

Algorithms to try are linear regression and PCA for dimensionality reduction and feature engineering as a preprocessing step for the subsequent neural network. Another approach to find useful feature will be a random forest regressor to help extract feature importances. I will then use LSTM to make the final prediction.

A challenge will be the selection of the lookback period. A 1 tick lookback period may give the best accuracy, given it would use only the most up-to-date information about price. However, a longer lookback period could incorporate patterns like mean reversion, the fact that prices or returns revert to a certain mean value over a period of time. To make the results comparable with the benchmark model, we will use a 15 minute lookback window, or integer multiples of this.

**A benchmark model — some simple or historical model or result to compare the defined solution to;**

As benchmark the Kaggle code created by kimy, a contributor on machine learning platform Kaggle, will be used: https://www.kaggle.com/kimy07/eurusd-15-minute-interval-price-prediction/notebook. The benchmark model attempted price prediction with bid price data on EUR/USD with a fairly coarse 15 minute observation window.

**A set of evaluation metrics — functional representations for how the solution can be measured;**

The success of any solution will be measured by looking at the error of predicted out-of-sample next prices.

Given that the problem is stated as a regression, I will use the mean absolute error and the mean squared error as evaluation metrics. I would hope to achieve a MAE of less than 0.0005.

**An outline of the project design — how the solution will be developed and results obtained.**

To attempt a solution, the dataset is loaded into a sql or mongodb database, probably sql as the syntax is easier. Next data cleanliness is checked, to account for nulls, positive and negative outliers and to create a final, clean dataset.

The dataset will be enriched by creating additional features for my state space, such as:

- returns
- bid-offer spreads
- period high, low, average, median, standard deviation

The tricky bit here is that prediction from the previous tick row is straightforward, as each feature can be used directly. However, using several tick rows to cover a 15 minute lookback window requires a decision how to use the features of each row.

One way would be to group the data into 15 minute intervals, with a number of aggregates to describe price behaviour during this time. We will initially attempt this, and only resort to the tick by tick prediction if error is too high.

This enriched dataset will be fed to various algorithms to predict the next price and return. We use a 15 minute lookback timeframe for prediction, and predict the closing price in 15 minutes, in line with the benchmark model from Kaggle.

Algorithms to try would start out simple, for example Linear Regression and Decision Trees. Next will be Long Short Term Memory Networks, which seem to work best on many timeseries problems: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

For visualisation the final evaluation metrics of each model will be shown where applicable, such as the mean average error, and will explain how they inform next steps.

The coding stack will be Python 3 based, using some or all of the following libraries: jupyter, pandas, numpy, keras, tensorflow, matplotlib, seaborn, sklearn, scipy.

Training will take place on an Intel i7 processor with 16GB of memory, as well as on an Amazon AWS p2.xlarge cloud virtual machine which provides access to a NVidia Tesla K80 card with 11GB of memory which allows faster computation. Given its cost of 1 USD per hour, we will mainly use the Amazon server to refine solutions deemed promising based on initial local testing with only a few epochs to observe how fast evaluation metrics improve.