# Lab 2: Classification and Regression Stock Price

## Objectives

- Design, train, and evaluate neural networks for both regression and classification tasks using stock price time series.
- Experiment with noise-injection techniques to simulate multi-step forecasting uncertainty.
- Curate raw time series data into labeled datasets suitable for classification, justifying your choice of labels and their relevance.
- Critically analyze model results, both quantitatively (metrics, plots) and qualitatively (discussion of behavior).

## Part 1: Time Series Regression Problem

1. Download the `stock-data.zip` file (Referenced from [https://www.kaggle.com/datasets/szrlee/stock-time-series-20050101-to-20171231](https://www.kaggle.com/datasets/szrlee/stock-time-series-20050101-to-20171231). This contains stocks from various companies (1 company per file).
2. Choose a company as provided by one of the company codes.

['MMM', 'AXP', 'AAPL', 'BA', 'CAT', 'CVX', 'CSCO', 'KO', 'DIS', 'XOM', 'GE',
'GS', 'HD', 'IBM', 'INTC', 'JNJ', 'JPM', 'MCD', 'MRK', 'MSFT', 'NKE', 'PFE',
'PG', 'TRV', 'UTX', 'UNH', 'VZ', 'WMT', 'GOOGL', 'AMZN', 'AABA']

Pick one "ticker" / company and a dimension to be used for a time series problem (Open, High, Low, Close or Volume).

3. Create a time series dataset using PyTorch. Split the instance of that dataset to `train` and `validation` dataset instances with 70% and 30% partitions respectively. Make sure to declare the window size, stride and number of outputs. Do not shuffle the datasets.
4. Create a base neural network with the following conditions:

- At least 2 hidden layers
- No activation in the output layer

5. Train the network with the train dataset using the `MSELoss` module and validate it against the validation set. Plot the following in a graph:

- The training set
- The validation set
- The prediction of your model (against the validation set)

6. For a given `n` value, make the model predict `n` timesteps into the future similar to what we did in class.
7. Research on at least 2 methods to add noise to the model and graph `n` values into the future using the added noise to the prediction. The 2 methods have to be different from what was discussed in class where we arbitrarily added a noise value. Make sure to cite related literature

(can be a online article) on how to compute for this noise. Graph all 3 predictions (vanilla, noise method 1, noise method 2) for `n` steps.
8. Answer the guide questions for `Time Series Regression` below.

## Part 2: Time Series Classification Problem

9. Come up with 2 methods that takes the same time series dataset you used in part 1 and curate the data such that it represents a classification problem. You may draw ideas from existing literature. The exercise is to come up with a set of `x` and `y` pairs to be setup for classification.

Example 1: Given a window_size of 5, perform linear regression on an observed sample of 5 consecutive values. If the slope of the linear regression is positive, we label it as "increasing". Else, we label it as "decreasing".

Example 2: Given a window_size of 5, get the average of all the values. If the average is greater than the previous set, label it as "increasing", else label it as "decreasing".

Create 2 Dataset instances to represent your "engineered" features.

***Note 1:*** *Although the examples are very simplistic, come up with a more objective method to self-classify the dataset. The number of labels can just be binary and need not be multi-class.*

***Note 2:*** *The labels need not be as simple as increasing or decreasing. You may come up with more creative labels like "buy" signal vs a "sell" signal in the context of stocks.*

10. Design and train a neural network model of your design with the 2 datasets with a partition configuration of 70% for training and 30% for validation. Compute for both the confusion matrix and `f1` score of the model for each type of dataset.
11. Answer the guide questions for `Classification` below.

## Guide Questions for Time Series Regression

a. How does your choice of window size and stride affect model performance, generalization, and stability of multi-step forecasts?

b. What are potential risks when recursively feeding predictions back as inputs for multi-step forecasting?

c. Compare the performance of the model with and without noise injection. Which method produced the most realistic forecast paths? Explain this both qualitatively and quantitatively.

d. Explain the different parameters used for each noise injection method and how it might model predicting the future.

e. What does the error trend (RMSE vs predicted horizon) tell you about your model's ability to extrapolate?

## Guide Questions for Classification

a. Which engineered features contributed most to a higher F1 score and why do you think this is the case? Since you defined how to compute for a given label, explain this in terms of how a neural network attempts to *learn your method*.

b. How did your labeling scheme affect class balance and therefore F1 performance? Answer this with quantiatve evidence (i.e. class distribution).

c. If F1 was low for one or more classes, which additional features or transformations could you add to improve it?

d. What patterns do you observe in false positives vs. false negatives when F1 is low, and which feature adjustments could reduce these errors?

e. How does modification of features based on your methods (i.e. adding, aggregating or any transformation you applied) affect the F1 score, and what does this reveal about which features are most informative for your classifier?

## List of Items for Submission

- Code for Time Series and Classification Problems
- Graphs required as presented in the time series regression instructions
- Performance table for `f1` score in the classification instructions
- Answer to guide questions