# Computer Science Department CS677 –Machine Learning (CRN: 22150) – Spring 2024 Professor Enze Bai Project #1 | Due: February 20, 2024

The goal of this assignment is to understand the logic, methods, and types available to implement machine learning classifiers. Classification is the process of predicting the class of given data points. Classes are sometimes called targets, labels, or categories.

A classifier is the algorithm itself – the rules used by machines to classify data. A classification model, on the other hand, is the end result of your classifier's machine learning. The model is trained using the classifier, so that the model, ultimately, classifies your data.

In this project, we'll deal with specific types of data input – financial data. Specifically, we'll try to build a trading (security) strategy for future stock purchases based on historical values!

You should get the historical data (this is real/production data) from a credible source, see below. Prior commencing your efforts on coding, you must install/utilize the following libraries:

- Yahoo Finance, or
- GoogleFinance API

https://pypi.org/project/yfinance/
https://support.google.com/docs/answer/3093281

There are many more finance (stock prices) alternative data sources such as Quandl, Interactive Brokers, Alpha Vantage, TradingView, IEX, etc…

I would highly recommend using the Yahoo Finance (yfinance) Python library.

Get data for any security of your choice (AAPL, IBM, GOOG, or the S&P500 index SPY) from beginning of 2015 until the end of last year. For example, to download data for 'SPY' execute:
**spy_data = yf.download("SPY", start="2015-01-01", end="2023-12-31")**

Perform **two (2) distinct simple trading strategies in Python,** which will be based on Machine Learning **Classification modeling** approaches. The strategies will be such that you will call for a 'buy' or 'sell' signal of a certain security. The signal will be the target/label variable used by the ML classifiers. Namely, the **target variable** would be **either buying or selling the stock based on feature variables.** The target variable content should be constructed as follows: Store the value of +1 for the buy signal and -1 for the sell signal, respectively.

**strategy_1**: If the next trading day's close price is greater than today's close price then the signal is 'buy', otherwise 'sell'

**strategy_2:** Utilize the 50-day moving average vs the 200-day moving average. A golden cross (or golden crossover) is a chart pattern that involves a short-term moving average crossing above a long-term moving average. Typically, the 50-day MA is used as the short-term average, and the 200-day MA is used as the long-term average. This is an indicator of bullish (buying) signal. (See Figure 1 below)

A death cross is basically the opposite of a golden cross. It's a chart pattern where a short-term MA crosses below a long-term MA. For example, the 50-day MA crosses below the 200-day MA. As such, a death cross is typically considered to be a bearish (selling) signal.


*Figure 1: Golden Cross (Bullish Signal)*

NumPy library provides a plethora of building methods to handle current (today's) price, yesterday's price, Moving Averages (MAs), Roll, etc.

For instance, to calculate the values of the target variable for Strategy-1, use this NumPy call:
**y = np.where(df['Stock_Price'].shift(-1) > df['Stock_Price'], 1, -1)**

For Moving Averages (strategy_2) you should come up with a similar formula.

The following steps, generating the ML classifiers, should be performed:

- Pre-process and clean the data
- Define the Feature Variable 'X', and the Label/Target variable 'y'
- Spilt the data into training and test datasets (use the 80/20 percent ratio)
- Choose a Classifier and fit it on the training dataset (take its default parameters)
- Evaluate the Classifier on the test dataset

Utilize the following ML Classifier Models:

**1)** K-Nearest Neighbors (KNN) **from sklearn.neighbors import KNeighborsClassifier**
**2)** Random Forest Classifier (RF) **from sklearn.ensemble import RandomForestClassifier**
3) Gradient Boosting Classifier (GB) **from sklearn.ensemble import GradientBoostingClassifier**
**4)** Support Vector Machines (SVMs) **from sklearn.svm import SVC**
**5)** XGBoost Classifier
(Note: You don't import it from scikit-learn module!)
(You need to install it, it not available: **pip install xgboost)**
**from xgboost import XGBClassifier**

Write **Python** scripts to complete the above tasks along with their output. All work should be done and submitted in a single **Jupyter (or Colab) Notebook.**

Extra Credit: Try to tune some classifiers (you decide which ones, anywill do), by changing their default parameters.