# Problem-Solving With Machine Learning

**Project Part One: Frame a Machine Learning Problem**

**Instructions:** Think of a problem that you want to solve with machine learning. Frame this problem like a data scientist by answering the following questions. Please limit your answers to 100 words or less.

|  |
| --- |
| What is the **context** of the problem? Why is it important to solve this problem? |
| Cryptocurrencies such as Bitcoin have become popular in recent years. I am interested in predicting Bitcoin’s price in USD.  This problem is important because Bitcoin is the largest cryptocurrency in terms of Market Cap. If the prediction is accurate enough, it has the potential to be applied in trading strategies and gain large profits. |
| What inputs (**features**) would you include? Describe **data types** and possible feature **transformations**. |
| I would like to include the following US macroeconomic indicators and other factors as features of the model:   1. Month-over-month inflation rate (decimal). 2. 1-month US treasury yield (decimal). 3. NASDAQ Index price (decimal). We might need to transform it to the logarithmic return of NASDAQ Index. 4. Bitcoin trading volume in last 24 hours (decimal). 5. Number of Bitcoin ATMs in the US (integer). 6. Number of US publicly traded companies that accept bitcoin (integer). |
| What are the potential outputs (**labels**)? |
| Potential output 1: log return of next hour’s Bitcoin price.  Potential output 2: log return of next 24 hour’s Bitcoin price. |
| Are **observations dependent** anyhow? Often, we have either temporal dependency or multiple observations per patient or device or event. This is important for train/test splitting. |
| Since Bitcoin price is in time series and the log return of the price might be impacted by different factors as time changes, the future log return will likely be impacted by previous log return, and hence the observations should be dependent. |
| Is this a **regression**, or **classification** (**binary** or **multiclass**)? Explain. |
| This is a regression problem due to the nature of price and return. |
| How might you split the observations into **train** and **test** sets? Are there potential **biases** to look out for? |
| If the outputs/labels are the log returns of next hour’s Bitcoin price, I would collect all historical log return data of Bitcoin, and split the observations by 8:1:1 into train, validation, and test sets according to the time series of log return data.  It is possible some selected features have different impact on the Bitcoin log return in different time range. A previously useful feature may no longer be relevant and vice versa. We should be cautious when selecting the features and should ensure that the factors have similar degree of impact on Bitcoin log return across train, validation, and test sets. |
| What type of **loss** function might be appropriate for quantifying the error of your algorithm? |
| The squared loss function might be appropriate for quantifying the error because the label is continuous and we want to avoid extreme prediction discrepancy. |

# Project Part Two: Application and Limitations of k-NN

**Instructions:** Answer the following questions about the k-Nearest Neighbors algorithm. Please limit your answers to fewer than 100 words.

|  |
| --- |
| You have implemented k-NN to learn a decision boundary from a training data set that you know to be noisy. To address the issue of noise, you increase this number of nearest neighbors to nine, which has seemingly reduced the impact of the noise on your decision boundary. However, you notice that significant groups of data points in your training data set are now misclassified by this new decision boundary. **Why** might this happen? **How** can you adjust your algorithm **to** **improve** its **accuracy**? |
| As we increase the value of the hyper-parameter k in the k-NN algorithm, the boundary will become simpler and might be too smooth for truly complex data sets. Therefore, some data points might be misclassified by the new boundary.  A potential way to improve its accuracy is to reduce the number of k to a smaller value, such as 7 or 5. Another way could be using a different distance function that can better measure the ‘similarity’ between data points. |
| When implementing k-NN, you must choose an appropriate distance function. **What** is the role of the distance function and **why** is it important to the accuracy of your machine learning algorithm? |
| The role of the distance function is defining the way of measuring similarity between data points. This is critical to the accuracy and performance of the algorithm because it will determine the neighbors of the test points chosen by the algorithm and thus determine the label of the test points. |
| **Why** does increasing the number of observations also increase the computing **power** and **time** necessary to run k-NN? **What** possible solution is available to help improve the efficiency of k-NN in such a scenario? |
| To find the k nearest neighbors of a test point, the algorithm needs to compute the distances from each test point to every training point. Therefore, increasing the number of observations will increase the power and time of computing the distances.  A possible solution is applying data structures such as k-d trees. K-d trees will find the axis split value for each dimension and store the data. With the stored split values, it will be faster in the testing stage. However, it might be more computationally expensive in training stage. |
| **Describe** the curse of dimensionality. **Why** does k-NN break down in high-dimensional space? |
| The curse of dimensionality appears in the situation that all data points become more unique and farther away from each other as the dimension of data set increase.  Due to the curse of dimensionality, k-NN could become infeasible in high-dimensional space since the distances between points all become larger and less distinguishable. The distance metrics are more susceptible to noise as the dimension of data increases. |