# 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? |
| We want to understand how classify fraudulent credit card transactions based on a set of transaction features. This is important to reduce the negative impacts of fraud. |
| What inputs (**features**) would you include? Describe **data types** and possible feature **transformations**. |
| A combination of numerical (integer and float) and categorical features would be included including date of transaction, merchant, billed amount, payment history, credit score, monthly income, debt-to-income ratio. We would most likely normalize (i.e., standard scaling) all the features because of the wide range of values that certain features might take on. We would also either embed or one-hot encode the categorical features. |
| What are the potential outputs (**labels**)? |
| Is a particular observation (ie. set of credit card transaction features) a fraudulent transaction? Yes or No 🡪 Binary output 1 or 0. |
| 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. |
| There may be a possibility that a fraudster made multiple fraudulent transactions which exhibit a similar feature footprint. |
| Is this a **regression**, or **classification** (**binary** or **multiclass**)? Explain. |
| Binary classification. We want to be able to predict whether a particular set of CC transaction features indicate a fraudulent transaction. |
| How might you split the observations into **train** and **test** sets? Are there potential **biases** to look out for? |
| We need to address the fact that, in a training set population, the percentage of fraudulent transactions will likely be very low. We will most likely want to map the fraudulent vs. non-fraudulent percentages to the test dataset. |
| What type of **loss** function might be appropriate for quantifying the error of your algorithm? |
| Binary cross-entropy loss. |

# 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**? |
| It might be the case that the misclassified data points are on the boundary between two or more classification groups. To avoid this situation, I would recommend lowering k to something like 5 or 6 and see if the accuracy improves. |
| 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? |
| We use the distance function to determine the distance of a point to its k-nearest neighbors. We will need to **minimize** this distance function to produce the best results for accuracy. |
| **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 determine the k-nearest neighbors for a particular point, we must compute distances from each test point to every training point. This process becomes more intensive and slower as the size of the training set increases. To improve computation speed, we can split the population space into k-d ‘boxes’ and find a point’s k-nearest neighbors in the appropriate box. This eliminates the need to calculate distances to every point in the total population. |
| **Describe** the curse of dimensionality. **Why** does k-NN break down in high-dimensional space? |
| As the number of dimensions increases — that is, as one includes more and more features in a data set — all the data points become more unique. Eventually, the data points become so dissimilar that the approach of finding close neighbors to predict the label of a test point is no longer feasible. |