Machine Learning Project

# Introduction

To try to achieve the highest possible accuracy across models, I decided to try many different models. This was relatively easy, as Python comes with built-in functions that easily allow to change models. The two models with which I was able to achieve the highest accuracies were the Random Forest classifier (75%) and SVM (74%). However, in this report I will be covering the Random Forest Classifier and the KNN models only, as these are the ones where I made informed decisions when it comes to changing and testing out different hyperparameter values.

My general workflow to find the optimal hyperparameters was to implement a Grid Search approach, testing all possible combinations of parameters that the sci-kit classes offered. By doing this it is possible to append all the results to a list and return the highest number. The main reason why I decided to use this approach is because of its simplicity, the dataset provided is relatively small and I have the computational resources to implement this system. Preferably, it would be better to use a method such as Bayesian optimization, but I felt that I was not familiar enough to implement it.

# Model 1 - K-Nearest Neighbors

## Choice

Random Forest Classifiers are a good option for…

## Input to classifier

How do you represent your data? Have you applied any preprocessing such as changing or transforming the data?

## Hyperparameters

2+ hyperparameters tuned with 2+ values each. How and why were they tuned and what was their impact on performance?

## Training

Give any additional information about training. Provide the best settings you found for the model.

## Evaluation

# Model 2 – Random Forest Classifier

## Choice

## Input to classifier

How do you represent your data? Have you applied any preprocessing such as changing or transforming the data?

## Hyperparameters

Again, for the Random Forest Classifier model, I looked at the scikit learn documentation to see which arguments the RandomForestClassifier class has. n\_estimators=500,

max\_depth=20,

max\_features='log2'

## Training

## Evaluation

# Comparison of the two models

**Performance:**

**Training Speed:** To test the speed of training the two models, I used the time module in the Python standard library. Although the time to train varies from machine to machine based on computer power, here are the times it took my laptop to train both models.

# Classification Reports and Confusion Matrices

Both Classification Reports and Confusion Matrices are included in the code and are therefore provided in the output of the scripts respectively.