# Malicious Web Page Prediction

Nadine Safwat - 900212508

Link to model scripts: Machine Learning Project

## INTRODUCTION

To recap, this project is attempting to create a malicious URL detection system using machine learning models. To begin I gathered research on existing systems that use the same technique and chose an appropriate dataset to train my models. Then I preprocessed the data so that it is suitable and allows the models to train and predict fairly. The next step was to test all the models i could to see the scores they would yield so that i could choose the best to use for the detection system

## MODEL CHOICE - RANDOM FOREST

For the parameters I ran a gridsearch to see the best options and it did not yield any better scores than the default. However, due to the imbalance in my label I will add class weights so that the model can oversample the smaller class.

The default parameters include:

- n_estimators=100 which means there quill be 100 trees in the forest
- criterion='gini' which is the function that will measure the quality of the split

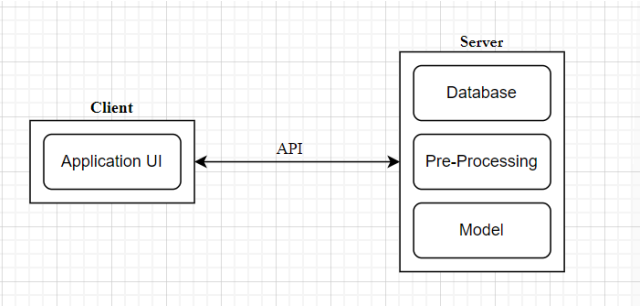
As said in the last milestone this model was chosen because it yielded the highest results and in a reasonably fast amount of time (10-15 minutes) making it the best option.

## UTILITY DESIGN:

The main goal of this system is to provide an easy and reliable way for users to quickly check if a website that they want to visit is malicious or benign.

## ARCHITECTURE:

For scalability reasons I chose a client-server architecture. On the server side, I will have my database that stores previous information in case a multiple users ask about the same website more than once, as well as the input pre-processing model that is required to run the users input on the Machine learning model as well as, ofcourse, the actual model itself. The client side will only host the application UI. Ideally the UI will only ask the user for the url of the website which will be sent over to the server using an API and from this url I can extract all the required data that is needed for the model to work. Of course if the pre-processing code cannot find a specific field a earning can be sent back to the user asking them to fill it themselves or willingly ignore that it cannot be found.




## PRE-PROCESSING MODEL

Most of the features that the machine learning model uses already stem directly from the URL, like TLD and URL_LEN but other features, like JS_LEN, IP_ADD, GEO_LOC and WHO_IS. To solve this issue I have found libraries that can be used to solve this issue. These include:

- requests & BeatuifulSoup for JS_LEN
- socket for IP_ADD
- requests & socket for GEO_LOC
- whois for WHO_IS

## USER INTERACTION

The user will be able to do three things in this malicious URL detection system:

- Enter in the URL to a webpage to detect if it is malicious or not. The user simply needs to enter the URL and it will be sent to the server to be pre processed and predicted.
- Give feedback on the prediction so that the system can know if the model is working well enough or if there are issues in the predictions
- View past predictions that will be stored in the database

**DATA FLOW**

The application will open to the main page that will ask the user to choose between making a new prediction or viewing an old one. If the user asks to view an old prediction they will be sent to a page that shows the most recent predictions and the user should be able to click through to older ones as well. If the user asks to make a new prediction they will be asked for the URL of the page to be predicted and the model will run and a prediction will be sent, after the user will be prompted to enter feedback on the prediction before being sent back to the main page.