## Introduction – Phishing Detection

* Introduce what phishing is in one sentence
* Some stats to throw out about phishing:
  + Phishing attacks contributed to a net loss of $1.7 billion USD in 2022 within the Asia Pacific Region
  + A global average of 84% of businesses have been successfully hacked (Australia leads with 94%)
  + Phishing rates are increasing, with Singapore receiving a 25% rise in 2023
  + The complexity of phishing schemes is improving
* So phishing is costing the planet billions and getting worse every year, and manual methods of phishing detection are struggling to keep up with the rising demand. That’s why we employ ML for phishing detection
* Introduce the problem, what we’re going to cover in today’s exciting video

## The Dataset

* The “Web Page Phishing Detection Dataset” posted on Kaggle by Tiwari et al. provides an excellent training ground for developing ML models.
* It features a dataset of over 11’000 URLs, some of which are fraudulent and the others legitimate. For this reason, it’s set up as a supervised binary classification dataset.
* Dataset is pre-loaded with comprehensive feature selection. It contains over 80 features that are correlated with phishing sites, including URL length and the use of IP addresses
* It’s ideal in that feature selection has already been integrated; we can focus solely on data cleaning and ML model creation

## Genetic Algorithm

* Genetic algorithms (GAs) are powerful AutoML techniques capable of finding the optimal feature selection, data cleaning and hyperparameter tuning
* Based on the concept of natural evolution
* Give the smartest human analogy
* We also implemented K-fold cross-validation to give more accurate results

## Data Cleaning and Statistical analysis

* The first design decision was to reduce the dimensionality of the input data. Although this would remove some of the model’s accuracy and prevent some of the non-linear relationships between inputs to be exploited, it would vastly improve training time.
* For this reason, the Pearson Correlation Coefficients for each of the input features was generated wr/ to the URL’s label. Most had little to no correlation. Through eyeballing, we kept all features that had a correlation co-efficient of > 0.2 (absolute value). This left a reduced dataset with 24 input features
* In analysing a heatmap of the remaining Pearson Correlation Coefficients, a high degree of correlation was present between the remaining inputs. This suggests a degree of dependence, making ML models such as Naïve Bayes unsuitable for the task
* But all remaining inputs have relatively high correlation with the output labels
* All other data cleaning was handled by GA. GA was capable of further refining the feature selection as well as selecting the best scaling process
* GA could select between two finely-grained feature selection process; a wrapper based feature mask and a filter-based Principal Component Analysis (PCA).
* PCA operates by redefining the vectors of the input matrix to fit along orthogonal vectors known as principal components. They’re able to vastly reduce the dimensionality of data whilst preserving the maximum amount of accuracy. They’re a faster method of dimensionality reduction, but tend to reach a lower level of final accuracy
* Feature masks are able to iteratively select and deactivate the presence of individual columns, in order to find the optimal combination of features. Although the final result is often more accurate than filter based methods, it tends to take far more iterations to reach the perfect solution.
* After the finely-grained feature selection, the GA was able to select the optimum scaler to apply to the input data. It’s able to select between robust, normalizing and standard scaling, or having no scaling whatsoever

## ML Models

* First method was KNN:
  + KNN is able to classify input data based upon the class of the K neighbours that are nearest to it. It takes a majority vote of the classes, and classifies the input data accordingly
  + The nearest neighbours are decided according to their Euclidean distance to the input point
  + It’s popular because it requires no training time, as the data points are simply loaded into the model. However, it suffers from computationally expensive testing time. It’s hypothesized that the KNN will have considerably faster training times, at the expense of slower testing times
  + Furthermore, it suffers from distancing issues. As such, it’s extremely sensitive to higher dimensionality inputs and different degrees of magnitude. It’s hypothesized that the GA will heavily reduce the dimensionality of the input data (either through PCA or feature masking), as well as apply scaling to vastly reduce difference in feature’s degrees of magnitude
* Next came decision trees:
  + Decision trees categorise data by splitting it into separate groups that are decided via threshold values of input data
  + The decision tree determines which input to split on based upon whichever feature will contribute to the most entropy. As such, they prefer higher-dimensionality datasets, as they have more features to select from. This will likely be reflected through minimal dimensionality reduction
  + Since thresholds for splitting are determined based on the averages of input features, decision trees are insensitive to scaling methods. It’s likely that the scaler method being used by the decision tree will be completely arbitrary.
  + Decision trees are famously prone to overfitting data. For this reason, K-fold cross-validation was implemented to reduce this effect. There should be minimal difference between testing and training accuracy
  + It’s a lightweight but incredibly accurate algorithm. The training and testing times should be lightning fast with little difference between them. However, its accuracy tends to be lower than ensemble-based ML methods
* Last is Random Forest:
  + Ensemble version of decision trees
  + Creates a ‘forest’ of trees, each trained on separate subsets of input features and data. It then classifies outputs based upon a majority consensus of the tree’s decisions
  + Due to the majority consensus, it removes all issues with overfitting. Furthermore, it tends to be far more accurate than decision trees. However, training those extra models takes a long time, and it will likely be the slowest model both in terms of training and testing time
  + Like decision trees, it prefers higher dimensional inputs for the greater feature range. For this reason, dimensionality reduction via PCA and feature masking will likely be minimised
  + Similarly, scaling methods will likely be arbitrary due to the nature of thresholding. Scaling may be present, but it’s likely not having any effect

## Results and Conclusion

* Yada yada yada fucked if I knew haven’t run the simulation yet aye