Two different machine learning algorithms have been used for analysis on the data, multi-linear regression and a deep neural network (DNN). Multi-linear regression was the initial model that was chosen to be used – both the sklearn and statsmodel.api libraries were used to train and test the data. Initial results showed that the linear regression model worked relatively well for the US dataset, which it was trained on.

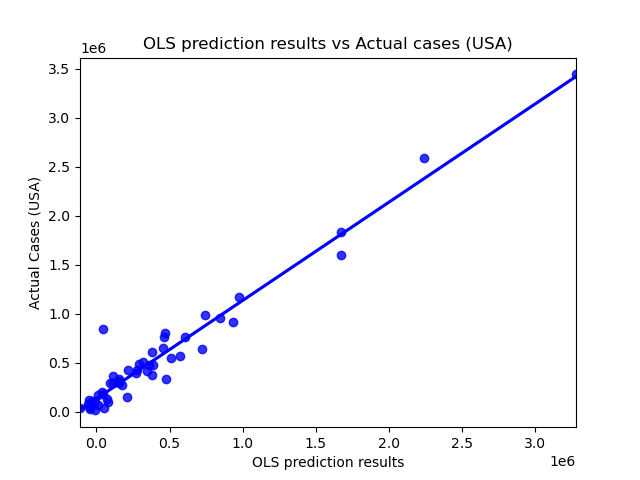


Figure 1. OLS predictions vs Actual Cases USA

However, when performing predictions on the EU dataset the results came in two distinct clusters at a right-angle to one another, indicating a low level of accuracy on unseen data and possible overfitting.

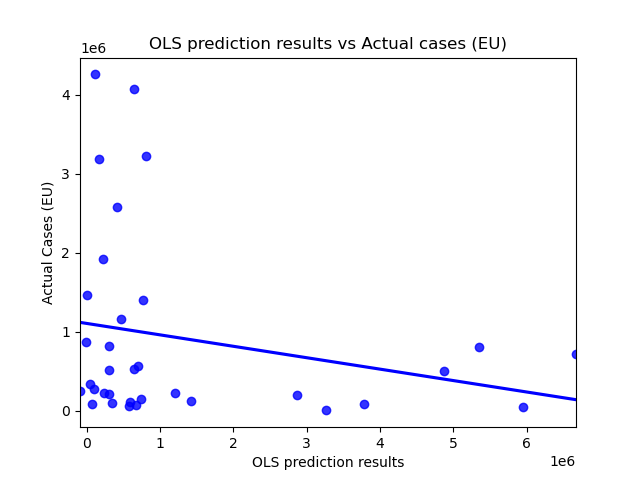


Figure 2. OLS predictions vs Actual cases EU, with the two ovals indication the separate clusters

The neural network code operates Keras architecture from the Tensorflow library to construct the model. The DNN utilises 1 output layer, 1 input layer and 2 dense hidden layers, visualised in the following figure:

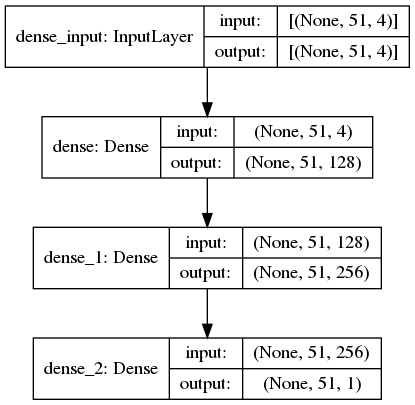
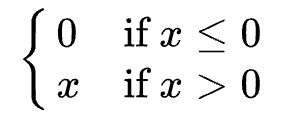


Figure 3. DNN Architecture

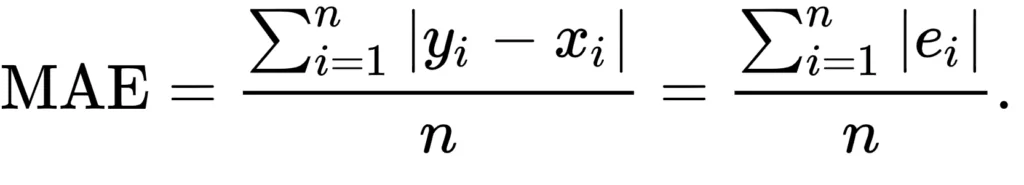
Both dense layers utilised the Rectified Linear Unit (ReLU) activation function, which is as follows:



Equation 1. https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

The slope is always 0 for negative inputs and always 1 for positive inputs. ReLU was used as it is computationally less intensive and faster than most other activation functions, such as sigmoid and tanh. However, the ReLU function is **not** differentiable because its derivative is 0 for any negative input, and it can have issues calculating values close to 0.

The mean absolute error (MAE) function is used to calculate loss in the current iteration of the neural network. This function takes the absolute error of all points and calculates their mean. MAE is calculated via the following equation:



Equation 2. <https://www.machinecurve.com/index.php/2019/10/04/about-loss-and-loss-functions/>

MAE was used because it is a commonly used metric and relatively robust to outliers, which our data contains.

The data collection team was able to collect a variety of different features for potential usage within both machine learning models. The currently utilised features are 'Population', 'Tests', 'Gini', and ‘% urban population'. All of these features were independently plotted against the actual number of cases, to observe their collinearity.

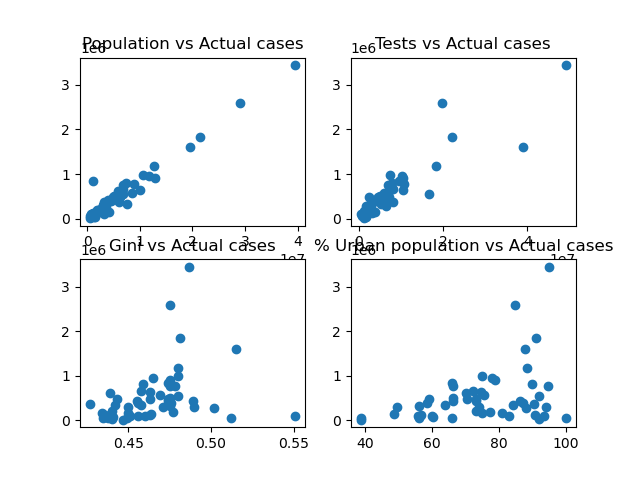


Figure 4. Plots of individual features against Actual Cases

As can be seen, while both population and tests were observed to have a general strong, positive relationship with actual cases, the gradients for Gini and urban population are much less clear. It was important to account for this when analysing the results of this iteration.

Some pre-processing steps had to be taken to clean the data before it could be used for the machine learning algorithm. Firstly, the relevant features and information were extracted from the .csv file where the data is stored, whereupon all commas were removed from individual datapoints to make sure python could parse them correctly. The data was then normalised via a min-max-scaler, which places all data points between 0 and 1. For each data point in a feature, the MinMaxScaler deducts the smallest value in the feature and then divides this answer by the range, which is the difference between the original maximum and original minimum. The MinMaxScaler retains the original shape of the distribution, thus preserving the information embedded into the initial data set. However, it is important to note that this also means that the MinMaxScaler does **not** reduce the importance of outliers. Finally, the pre-processing procedure was completed by dropping all NA rows from the data frame. This is to make sure that all data can actually be used for training the model, as missing values can cause errors and unwanted variations within the procedure.

The neural network contains a number of hyperparameters that had to be set manually before training could begin. These hyperparameters helps determine how the network is structured and how it is trained. As mentioned earlier, the choice of the ReLU activation function and the current number hidden layers in the network are both examples of hyperparameters. The decision to use 128 neurons in the first layer and 256 in the second is also a hyperparameter. The train-test split is set at 60%/40%, and the dataset is always trained for a set number of 100 epochs.

The R-squared coefficient is a statistical measure that can be used to determine the proportion of variance in the dependent features that can be explained by the independent variable. Basically, R-squared shows how well the data fits the model, where a higher R-squared score indicates a better fit for the model. R-squared is calculated via the following equation:

R^2 = 1 - \frac{SSE}{SST}

Equation 3. https://www.codinground.com/calculating-r-squared-python/

where SST (total sum of squares) is:

SST =\sum_i (y_i - \bar{y})^2

Equation 4. https://www.codinground.com/calculating-r-squared-python/

These are the squared differences between the actual y values and the mean y values.

SSE (residual sum of squares) is:

SSE =\sum_i (y_i - \hat{y}_i)^2

Equation 5. https://www.codinground.com/calculating-r-squared-python/

This equation indicates the summed squared differences between the regression line (mx+b) and the predicted y values.

While this statistical measure can provide useful insights in regards to the regression model, it does not disclose information about the causation relationship between the independent and dependent variables. In addition, it does not necessarily indicate the correctness of the regression model. Sometimes, a high R-squared score can indicate problems with a regression model, while a good model may also occasionally show a small value. It is essential to remember that there is no universal rule on how exactly to incorporate a statistical measure in evaluating a model, and that the contextual framework of a dataset is highly important in understanding how the insights of a metric can vary.