Model Choice

First Model: Logistic Regression

- Logistic regression is a linear model used for binary or multiclass classification. This model is very computationally efficient, handles multiple dimensions well and has a reduced risk of overfitting, so I thought it would perform well for this task.

Second Model: EarlyStopping Neural Network

- Neural Networks consist of layers of interconnected nodes and are great at observing nonlinear relationships between features. Thus, I thought this model would perform well and even capture some patterns that Logistic Regression might miss, even though the sample size is not that large.

Input to classifier

Some data preprocessing was required before training either of the models.

For both models:

- The dataset had missing values, which is a problem for both of the models. In order to fix this issue, I used an imputer (sklearn SimpleImputer), using the 'mean' strategy, which replaced all the missing values of a missing feature with the overall mean of that specific feature.

Only for the neural network:

- The data differed in dimensionality (some values were much greater than others, which would bias the network weights in a manner that could be inaccurate), so I used a scaler (sklearn StandardScaler) to standardise all features.
- I used One-Hot encoding to convert the class matrices into binary class matrices, in order to help the model understand that the classes are not distinct and not ordinal.
- I used Principal Component Analysis to reduce the dimensionality of the input features. The component numbers I tried were: 12, 11, 10, 9, 8, 7, 6. From experimenting, the best performing number of components was 10, so that was used for the model.

Hyperparameter tuning

*Not all candidates are present in the final coding file, but all were tested.

For Logistic Regression:

- The main hyperparameters I tuned were:
    - C (learning rate)
        - Candidate values: 0.01, 1, 100)
    - Penalties
        - Candidate values: l1, l2
    - Solvers:

        - Candidate values: liblinear, newton-cg, lbfgs

For Neural Network:

- The main hyperparameters I tuned were:
    - Activation function
        - Candidate values: ‘tanh’ and ‘relu’
    - Optimizer
        - Candidate values: Adam, Adadelta, Adagrad, SGD, RMSprop
    - Learning rate
        - Candidate values: 0.001, 0.01, 0.1, 1
    - Batch size
        - Candidate values: 32, 64, 128, 256
    - Number of Layers
        - Candidate values: 2, 4, 6, 8
    - Number of Units
        - Candidate values: 8, 16, 32, 64

Model Training

For Logistic Regression:

- GridSearch was used to find the best hyperparameters, along with a 5 split kfold cross-validation.
- Best settings found:
    - C: 1
    - Penalty: l1
    - Solver: liblinear

For Neural Network:

- Nested loops were used to find the best hyperparameters, along with manually splitting the data for validation.
- Best settings found:
    - Activation function: tanh
    - Optimizer: RMSprop
    - Learning rate: 0.001
    - Batch size: 32
    - Number of layers: 2
    - Number of Units: 32

Model Evaluation

For Logistic Regression:

- Training accuracy: 72.61%
- Validation accuracy: 70.44%

For Neural Network
- Training accuracy: 78.95%
- Validation accuracy: 73.86%
-

Comparison of Models

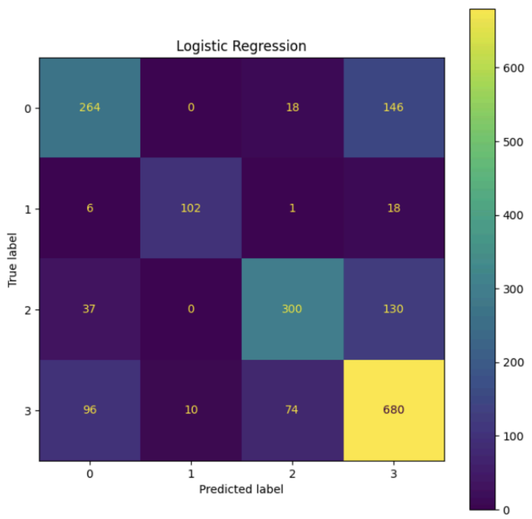
The Neural Network performed better accuracy-wise in both training and validation, but took significantly longer to train and test.

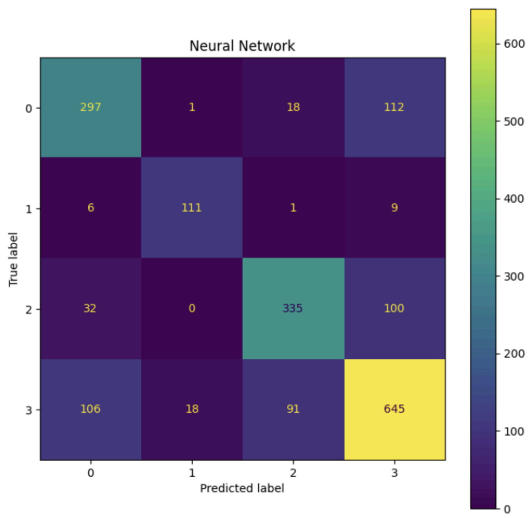
Training time:
- Logistic Regression: ~0.5 seconds
- Neural Network: ~35 seconds

Final Test Accuracy

Logistic Regression: 71.52%
Neural Network: 73.75%

Final Results





The first diamond type (Lab-grown, label 0) was most difficult to classify and the second (Zirconia, label 1) was easiest. It was common for Lab-grown (label 0) to be confused with Natural (label 3), and for Moissanite (label 2) to be confused with Natural (label 3) as well.