Classification Analysis Report

I. Introduction:

1.1 Problem Statement:

How can eating habits and physical activity help classify obesity and aid in promoting better health decisions?

1.2 Dataset Description:

The dataset used in this analysis is ObesityDatasetEstimation which was obtained from [UCI Machine Learning Repository](https://archive.ics.uci.edu/). The dataset contains different features concerning food habits which helps us deduce whether a person is obese (1) or not (0).

- Features:
    - Gender, Age, Height, Weight
    - Family History: History of obesity in the family.
    - AVC: Binary, frequency of consuming high-caloric food.
    - FCVC: Integer, vegetable consumption frequency.
    - NCP: Continuous, number of main meals daily.
    - CAEC: Categorical, food intake between meals.
    - CH2O: Continuous, daily water intake.
    - FAF: Continuous, frequency of physical activity.
    - TUE: Integer, time spent on tech devices.
    - CALC: Categorical, alcohol consumption frequency.
    - MTRANS: Categorical, usual mode of transportation.
- Target:
    - is_obese: Binary 1/0

1.3 Objective:

Create a classification model to predict our target variable ìs_obesè.

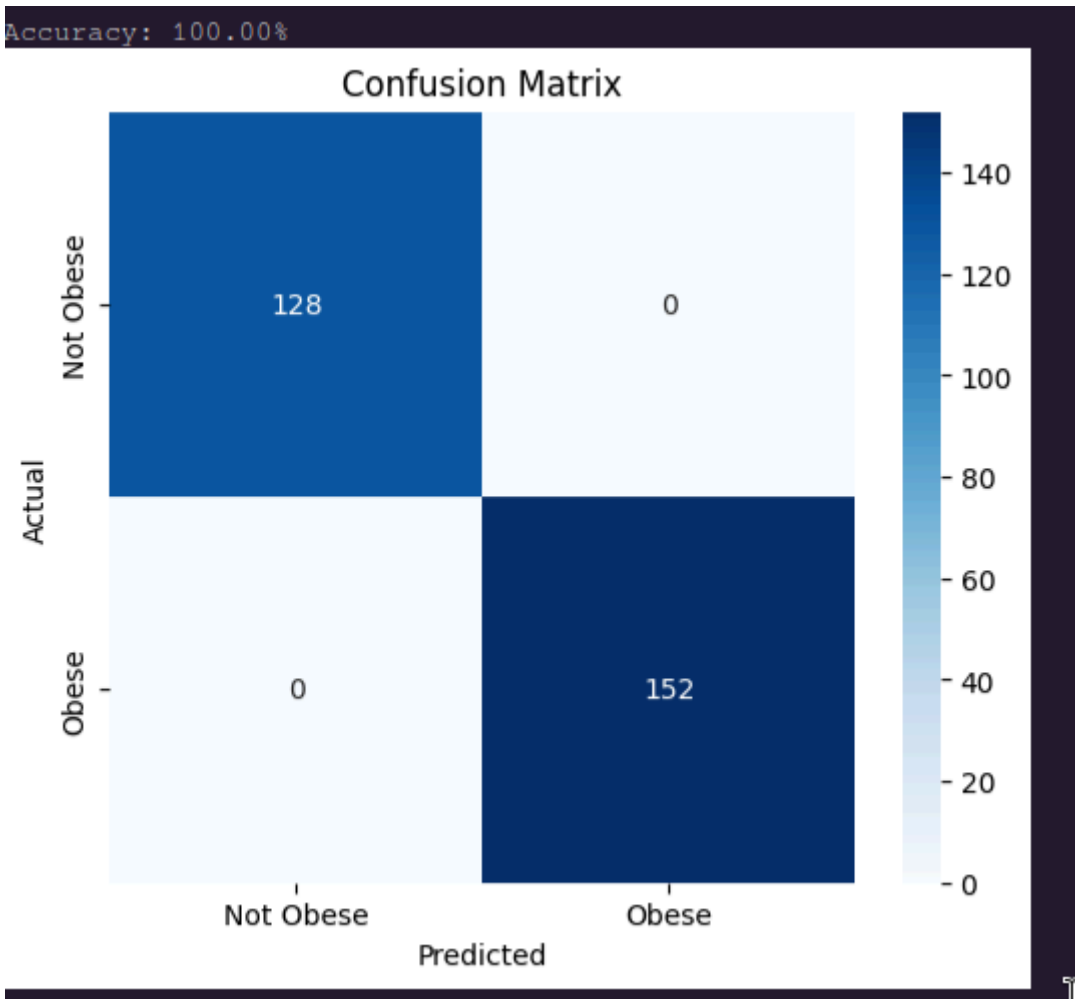
II Methodology

2.1 Data Preprocessing

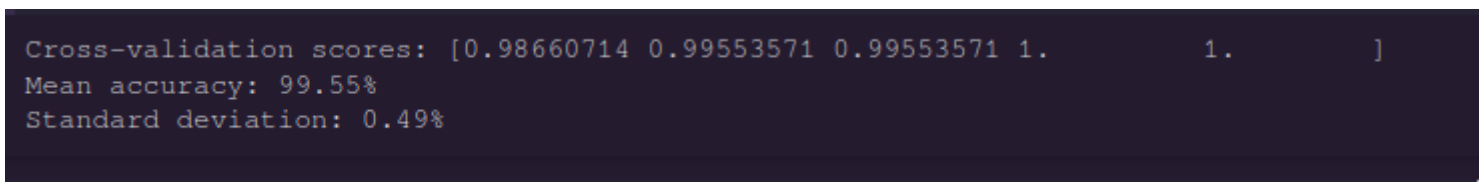
In this analysis, no significant data preprocessing was needed. Looking into the correlation of the features with the target variable, it was clear that the features and the target were not linearly correlated, furthermore there were many categorical features. That insight is the main reason why a decision tree classification model was chosen for this analysis. There were no missing values in the dataset. Although the presence of outliers in certain columns such as (Age and NCP) was noted,

```
Age: 0 outliers
Height: 3 outliers
Weight: 0 outliers
FCVC: 0 outliers
NCP: 273 outliers
CH2O: 0 outliers
FAF: 0 outliers
TUE: 0 outliers
is_obese: 0 outliers
```

Initially, the outliers were dropped, which later resulted in a 100% accuracy score on training , testing splits and cross-validation.



Accuracy Score



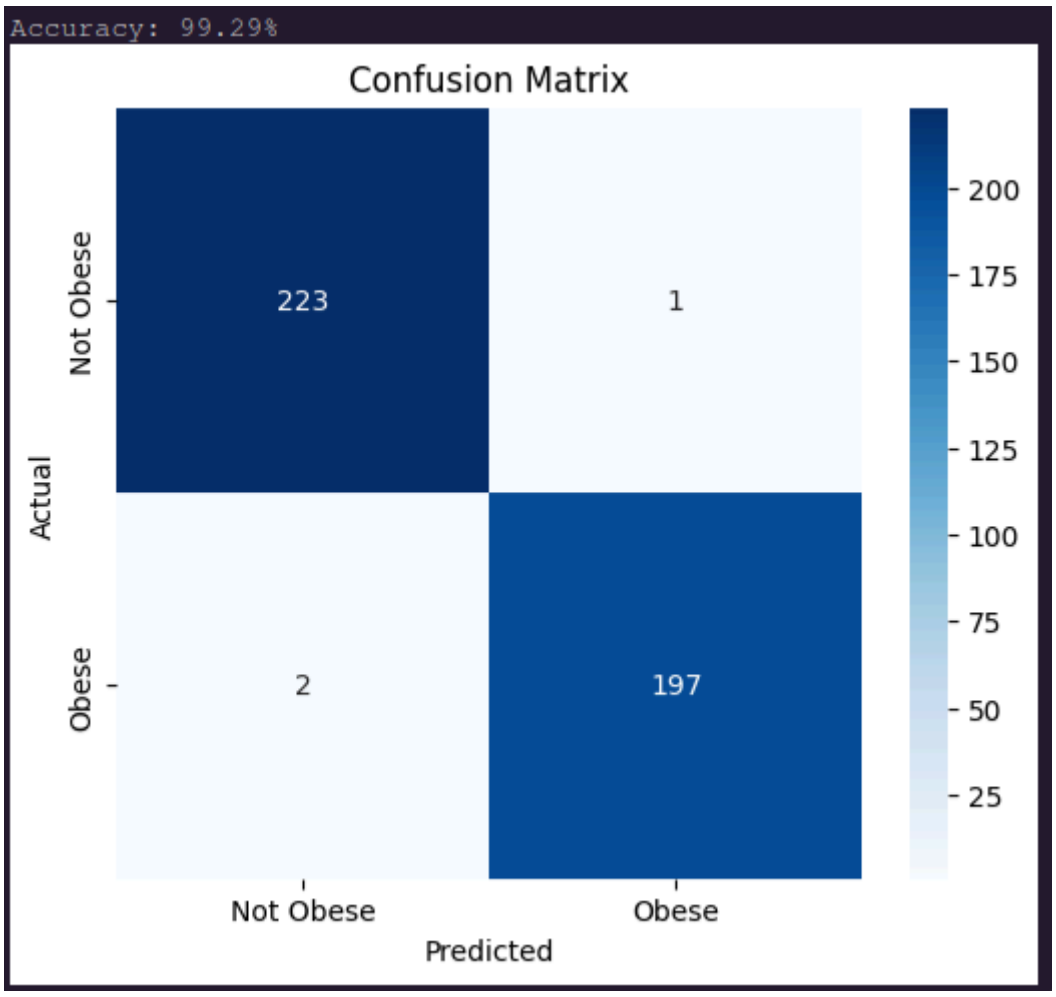
Cross Validation Score

Despite the low standard deviation and the close alignment of test and training accuracy, the possibility of overfitting was still considered. Although, other models performed similarly which indicate that the dataset is easy to classify. To possibly improve generalization, the decision to keep the “outliers” was made. Since the model already performed well with the outliers included, keeping them helped ensure the model generalised more.

2.2 Model Evaluation

The model’s performance was evaluated using the following metrics:
• Accuracy: The proportion of correct predictions over the total predictions.
• Precision: The proportion of positive predictions that are actually correct.
• Recall: The proportion of actual positive cases that are correctly identified.
• F1-Score: The harmonic mean of precision and recall.

Accuracy: 99.29%




```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       224
           1       0.99      0.99      0.99       199

    accuracy                           0.99       423
   macro avg       0.99      0.99      0.99       423
weighted avg       0.99      0.99      0.99       423
```

```
Cross-validation scores: [0.98230088 0.99115044 0.97345133 0.99115044 0.98230088 0.99115044
 0.98230088 0.97345133 0.99107143 0.97321429 0.96428571 0.98214286
 0.99107143 0.99107143 0.97321429]
Mean accuracy: 98.22%
Standard deviation: 0.86%
```

Results of performing performance measure metrics.

The model achieved 99% precision, 99% recall, and 99% F1-score across both classes. The precision indicates that 99% of the predicted positive cases were correct, while the recall shows that 99% of actual positive cases were correctly identified. The F1-score provides a harmonic mean of precision and recall, reflecting a well-balanced performance across both classes.

The model evidently performs very well in those evaluation metrics which raised suspicion. After looking at the tree produced, it was evident that Weight had a significant impact, which although makes sense, it begs the question that **is the model ONLY using weight and ignoring everything else?**

```
is_obese    1.000000
Weight      0.814997
FCVC        0.196686
Height      0.173907
Age         0.160656
CH2O        0.136294
NCP        -0.046326
TUE        -0.156298
FAF        -0.162428
```

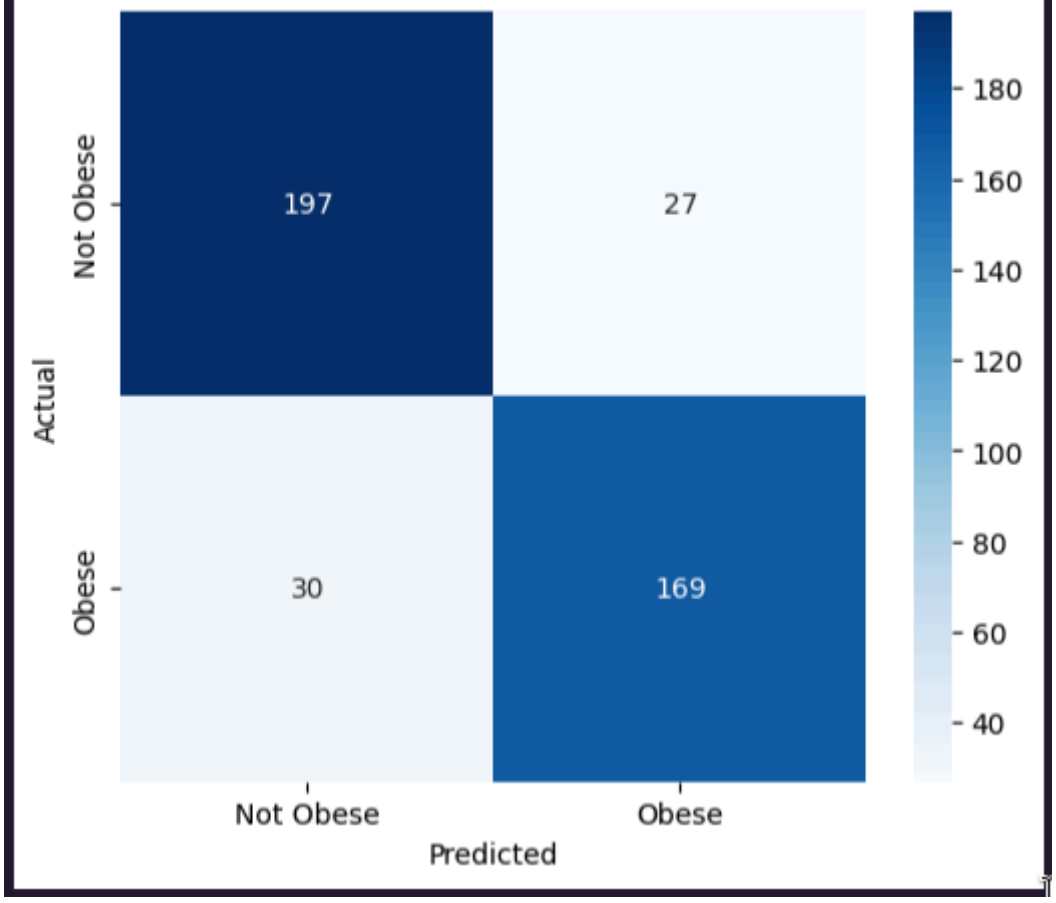
To test this, the weight column was dropped in its entirety which resulted in the following metrics:

```
              precision    recall  f1-score   support

           0       0.87      0.88      0.87       224
           1       0.86      0.85      0.86       199

    accuracy                           0.87       423
   macro avg       0.87      0.86      0.86       423
weighted avg       0.87      0.87      0.87       423
```

Accuracy: 86.52%




```
Cross-validation scores: [0.82300885 0.86725664 0.88495575 0.78761062 0.81415929 0.85840708
 0.84070796 0.77876106 0.84821429 0.84821429 0.8125     0.83035714
 0.83928571 0.89285714 0.79464286]
Mean accuracy: 83.47%
Standard deviation: 3.26%
```

What this tells us is that although `weight` played a significant role, it was not the only metric carrying the entire model as even without the weight column the model still performs well.

2.3 Hyper Parameter Optimization

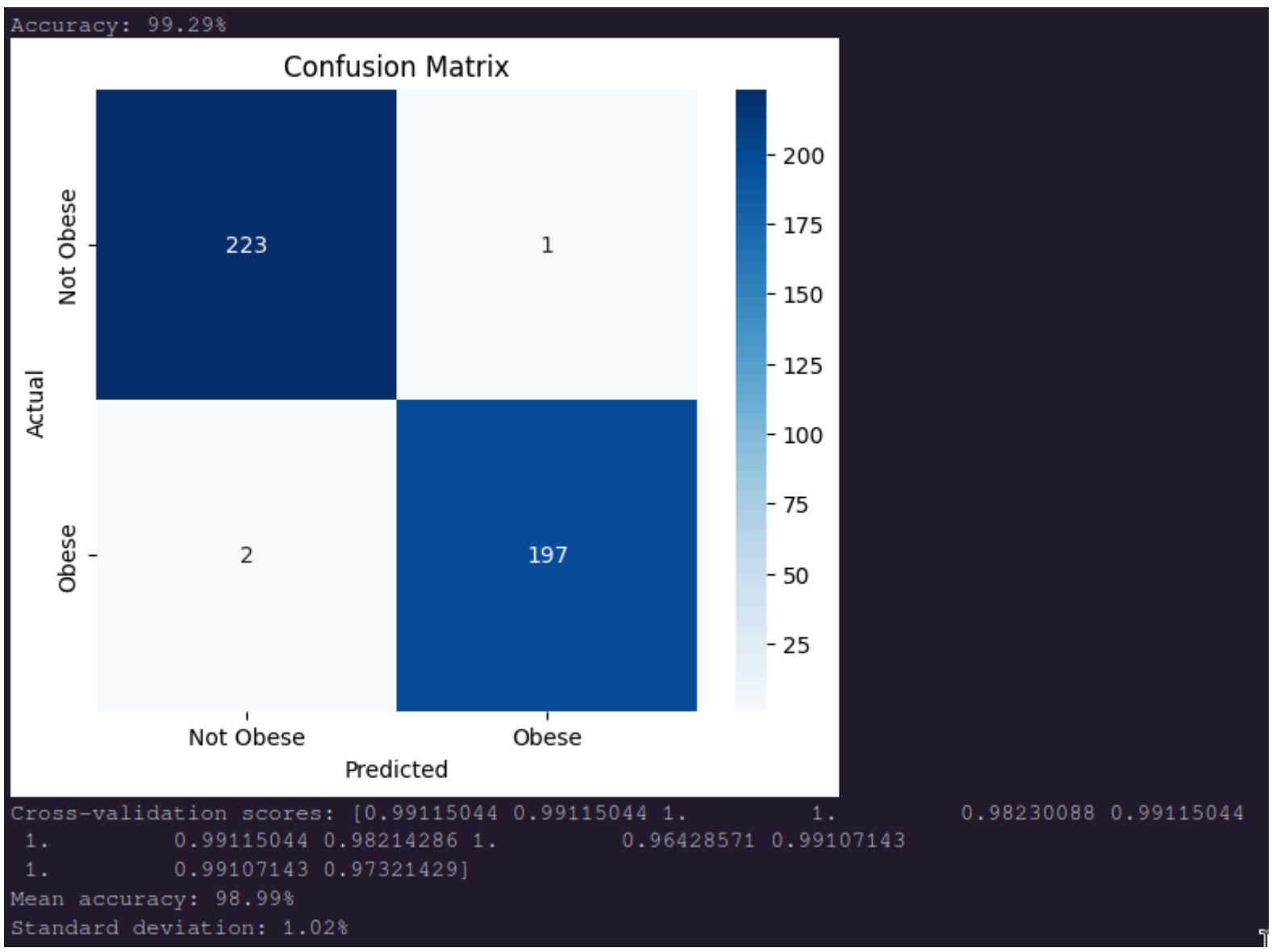
To improve the performance of the model, hyper-parameter optimization was conducted using GridSearchCV. The optimal parameters for the model were found to be:

```
# Best parameters found:  {'criterion': 'entropy', 'max_depth': 10, 'min_samples_split': 2}
```

2.4 Feature Selection

The features were implicitly selected by the decision tree itself. The decision tree algorithm automatically selects the most important features by evaluating how much each feature reduces the impurity (in our case using entropy as suggested by the gridsearch) during the training process. Thus, there was no need for an explicit feature selection technique, as the model itself identified the most informative features.

2.5 Finalised Performance Of Custom Model:



After setting the optimized parameters the above image shows the performance of the model made from scratch.

III Comparison Against Other Models

Before Optimisation:

```
KNN Accuracy: 0.943217665615142
Decision Tree Accuracy: 0.9889589905362776
Logistic Regression Accuracy: 0.9952681388012619
```

After Optimization:

```
KNN Accuracy (Optimized): 0.9589905362776026
KNN Best Cross-Validation Score: 0.9607160035731462
KNN Cross-Validation Standard Deviation: 0.023576834711998066
Decision Tree Accuracy (Optimized): 0.9905362776025236
Decision Tree Best Cross-Validation Score: 0.9932041503470076
Decision Tree Cross-Validation Standard Deviation: 0.014690758918267885
Logistic Regression Accuracy (Optimized): 0.9968454258675079
Logistic Regression Best Cross-Validation Score: 0.9952655809798667
Logistic Regression Cross-Validation Standard Deviation: 0.008423673986919155
```

Conclusion

Logistic Regression is the best model in this case, achieving the highest accuracy and cross-validation score. It outperforms both the Decision Tree and KNN models in terms of overall performance and stability across folds. Furthermore, hyperparameter tuning resulted in better model performance for all algorithms. This reinforces the value of tuning in enhancing model accuracy and robustness.

IV Discussion:

4.2 Custom Model Vs Pre-Built Models

The custom model achieved a mean cross-validation accuracy of 98.99% with a standard deviation of 1.02%. The accuracy across the folds ranged from 1.00 to 0.96, indicating a generally high performance and consistency. In comparison to the pre-built models, the custom model shows comparable or better accuracy in some cases, suggesting it can perform well with the given dataset when properly tuned.

4.3 Challenges

As for challenges related to the dataset, there weren't many. The nature of decision trees made it so that it selected the best features itself, furthermore I happened to pick a dataset with no missing values. The most significant challenge was the model's performance optimization. During initial stages, using pandas for data slicing and manipulation resulted in long training times. The process could take up to 3 minutes, and grid search could take hours. But after realising the fact that numpy was a lot faster and specifically made for operations like these, a switch to numpy based operations was made which drastically reduced the training time, cutting it down to about 20 seconds, which was a major performance improvement.

5.1 Impact of Hyperparameter Tuning and Feature Selection

While hyperparameter tuning and feature selection are generally crucial for improving model performance, in this case, it didn’t yield much improvement. All models benefitted from this optimization but nothing too significant

5.2 Limitations

Despite the good results this model is yielding. One limitation is the relatively small dataset size (~2k rows), which may limit the model's ability to generalize well to unseen data, especially in real-world applications with more diverse data. Furthermore, the lack of external validation, such as using a more varied or larger dataset, limits the robustness of the results. In addition, it was revealed that the model is prioritising the weight column a lot more. Granted that it intuitively makes sense, and the model was performing decently well (86% accurate) even after dropping the weight column, it shows room for improvement.

V. Summary Of Results:

Model	Accuracy	Approach
From Scratch	100%	After Removing Outliers , Expecting Outliers another test was done
From Scratch	86.52%	After dropping the weight column, which I suspected was causing a data leak
From Scratch	98.99%	After Hyper Parameter Optimisation.
KNN Classification	95.88%	“”
Logistic Regression	99.68%	“”
Decision Tree (prebuilt)	99.05%	“”