Application of Machine Learning Algorithms to Predict Obesity Levels

By Joseph Gross
Prepared for Dr. Chris Nikolopoulos

Abstract: Predicting obesity from lifestyle and biometric data is a critical task in healthcare, offering insights for prevention and intervention. The aim of this project is to demonstrate and compare 10 machine learning algorithms on a dataset consisting of health and lifestyle factors in order to find the most effective model to predict obesity classification on a given dataset from the UC Irvine Repository. WEKA Explorer was used as the tool to automatically preprocess and test the data set.

1. INTRODUCTION

Machine learning is capable of transforming the healthcare industry, enabling predictions and insights that find key risks and allow preventative measures or intervention to be taken earlier. Obesity is a growing global health concern, and can significantly benefit from such predictive models, as they provide actionable insights for individuals. The tool utilized for this project is WEKA, a comprehensive open-source software that provides tools for data preprocessing, classification, forecasting, and visualization. This project aims to leverage WEKA's capabilities to highlight the applications of machine learning in addressing public health challenges like obesity.

1.1 Definitions

Before we begin an investigation into individual models, we should note that there are 3 broader blocks of algorithms that will be used to divide this project's methods: classification, association, and clustering. The majority of applied algorithms will be classification. These are supervised, learning from labeled data to predict on unseen inputs. Classification algorithms find a predicted nominal class and their sub-type, regressions, provide a numeric result which is then binned to our target class encoded to a range. These will do our actual prediction after unsupervised algorithms are used for the sake of demonstration and exploration of the data. In both cases the predicted and actual values of unseen test data can then be compared. The unsupervised association and clustering algorithms are able to work without labels, and attempt to find patterns and structures without explicit targets. Association algorithms like APRIORI will be used for the sake of exploration to discover relationships and meaningful rules within the dataset (if X then y). Finally, clustering algorithms will attempt to separate the data into meaningful groups

1.2 Algorithms

This paper will evaluate over 10 algorithms including: ZeroR, Logistic regression, Support Vector Machine Classification, Decision Trees, K-Nearest Neighbors, XGBoost, Random Forest, Naive Bayes, and a Perceptron Model. for classification learning, K-Means for clustering learning, and APRIORI for association learning. The performance of these algorithms is to be validated and compared to determine the most effective approach for

obesity prediction. We will also attempt hybrid and aggregate models involving multiple algorithms to assess their potential for boosting predictive accuracy. The *ZeroR* algorithm will serve as the baseline estimate for the other models to compare against. It simply builds a frequency table and selects the majority class for all predictions itself having no predictive power.

Logistic Regression in Weka is a simple algorithm that first computes a linear combination of the input features and then applies a softmax function to the data to determine the probability of each of the output classes. It trains by minimizing a cross-entropy loss function and outputs the class with the highest probability for each instance.

Support Vector Machines is another supervised classification and regression algorithm. It looks to find the hyperplane between the feature space that maximizes the margin between the features. Kernel functions allow it to handle non-linearly separable data (nominal) by transforming their feature space. SVMs also work well for high dimension and computationally intensive data sets by focusing primarily on a small subset of the points that define the decision boundary (called Support Vectors).

K-Nearest Neighbors is a lazy algorithm that compares each new instance against the closest k neighbors in the training set by either euclidean or manhattan distance. The class label of the new instance is determined by majority vote of the neighbors.

J48 is an implementation of the C4.5 Decision Tree Algorithm. It builds a decision tree by recursively splitting the data by the attribute that provides the most information gain. This creates a tree of rules that continuously divides the data until a final output class is reached as a tree leaf. Decision trees can also be pruned to prevent overfitting to noise by allowing a margin of error on decisions or in the removal of branches that only increase complexity without accuracy after the tree is made.

XGBoost (Extreme Gradient Boosting) is a more advanced implementation of gradient boosting designed for efficiency. Gradient boosting builds an ensemble of weak models like decision trees where each is trained to minimize the residual errors of the current model. Each iteration, a new tree is trained to predict on the difference between the actual value and the predicted value from the previous generation and the new predictions (with a learning rate coefficient) are added to the current ones. XGBoost further includes several optimizations over regular gradient boosting such as adding regularization terms to the loss function and max tree depth/pruning to prevent overfitting.

Random Forest in Weka is an ensemble method that builds many full depth decision trees without pruning (this is to help reduce overfitting). Each tree is made from a random subset of the data and a random subset of features. The prediction class is taken by majority voting of the entire forest.

Naive Bayes is a simple probabilistic classifier that applies Bayes Theorem to calculate the probability of a class given the input features by multiplying the input probabilities. It uses a Gaussian feature distribution for numeric data. The classifier predicts whichever class has the highest probability.

Perceptron models implement a basic neural network inspired by biological neurons. Each neuron sums weighted inputs, adds a bias, and performs an activation function to bring the output to binary. The model iteratively adjusts its weights as it samples the training data to minimize misclassifications. They often work

better on linear data due to their simplicity. Due to having multiclass data, multilayer perceptrons could be attempted but not a voted perceptron.

k-Means-Clustering is implemented in Weka as simpleKMeans; it begins by randomly initializing k clusters just given feature similarity and proceeds to assign each instance to the nearest cluster. Centroids (cluster centers) are recalculated as the mean of all points in the cluster and the process is repeated until stability or a maximum number of iterations is reached.

APRIORI is used to generate rules to define the relationships within a dataset. These rules are based on counts of how often attributes are shared in instances (support) and how often the rule is correct (confidence). APRIORI adds items iteratively, adding one item at a time to itemsets and pruning combinations that fail to meet a minimum support threshold.

1.3 Prior Research

This dataset was developed and published by Palechor and Manotas, as described in their paper "Dataset for Estimation of Obesity Levels Based on Eating Habits and Physical Condition in Individuals." [1] Previous studies using the same dataset have demonstrated the effectiveness of hybrid models and neural networks in obesity prediction. This project builds upon these findings by applying a broader range of algorithms and exploring additional hybrid models, aiming to provide a comprehensive evaluation of predictive techniques and a general demonstration of many algorithms.

2. DATA SET

This project will analyze a dataset with 17 features and 2111 instances retrieved from the UC Irvine repository. Each instance (row) consists of independent physical and lifestyle factors and a dependent obesity classification on a 7 class scale. 23% of the data was collected directly from web survey participants (from the regions of Columbia, Peru, and Mexico as noted by the dataset's introductory paper).[1] The remaining 77% of the UC Irvine dataset was previously generated synthetically using the SMOTE tool from WEKA (See Appendix fig. 1 and 2). The Synthetic Minority Oversampling Technique (SMOTE) is used to address class imbalances and finds instances of the minority class, applying K Nearest Neighbors and creating new instances by interpolating between each instance and its nearest neighbor. This was used to ensure each of the output classes has a similar number of instances.

2.1 Exploratory Data Analysis

The independent X of the dataset consists of 16 nominal, binary, and continuous numeric features. Each instance includes: age, gender, height, weight, family history of obesity (binary), and lifestyle questions such as their kind of daily transportation and whether the participant eats high calorie food, vegetables, smokes, drinks, exercises and similar behaviors. The target value, obesity, has 7 class values: insufficient weight, normal weight, overweight types I and II, and obesity types I, II and III. The target class, mapped to the numeric range 1–7, shows the following correlations with the numeric and binary features: Weight (0.91), Age (0.28), Vegetables with Meals (0.23), Height (0.13), Daily Water Intake (0.13), Daily Number of Meals (0.03), Tech. Usage (-0.11), and Exercise Frequency (-0.20).

---

[1] https://www.semanticscholar.org/paper/Dataset-for-estimation-of-obesity-levels-based-on-Palechor-Manotas/35b40bacd2ffa9370885b7a3004d88995fd1d011

2.2 Pre-Processing

UC Irvine provides high quality cleaned test data in its repository, but this project aims to demonstrate dealing with holes. Weka's ReplaceWithMissingValues filter was used to artificially create holes in the data at 0.2% of the values. Because KNN imputation cannot be found in Weka, the data was exported to Python, the nominal values were filled with the mode of the feature and the numeric values were filled using the KNN Imputation algorithm before the nominal and numeric features were concatenated back together. K-Nearest Neighbors Imputation takes the mean of the closest neighboring data points in distance to fill holes (in this case, the closest 49 neighbors, the square root of the total number of instances is often optimal).

3. CLASSIFICATION

We applied several classification algorithms to the dataset using the nominal *nobeyesdad* attribute as the target class. The output has been displayed below each algorithm, in this case we are looking primarily at accuracy in the top right corner for the total percentage of correct cases and the weighted average of recall for the number of False Negatives, which would represent present risks not detected by the algorithm. 10 fold cross validation was used with each model in Weka (XGBoost being an exception) to automatically partition the train and test data and attempt each of the 10 blocks as the test case. Only the summary results for each model are shown for brevity; all models were constructed in under 5 seconds, the vast majority under 1 second. Batch size and seeds were left to to the default 100 and 1 respectively

3.0 ZeroR

ZeroR was run with the nobeyesdad class to create a frequency table as the control for the other algorithms. Because the 7 class values were balanced, it performed rather poorly as expected. The results are shown in Figure 3 below.

```
Correctly Classified Instances         351               16.6272 %
Incorrectly Classified Instances      1760               83.3728 %
Kappa statistic                          0
Mean absolute error                      0.2446
Root mean squared error                  0.3497
Relative absolute error                100      %
Root relative squared error            100      %
Total Number of Instances             2111

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  I
                 0.000    0.000    ?          0.000    ?
                 0.000    0.000    ?          0.000    ?
                 1.000    1.000    0.166      1.000    0.285
                 0.000    0.000    ?          0.000    ?
                 0.000    0.000    ?          0.000    ?
                 0.000    0.000    ?          0.000    ?
                 0.000    0.000    ?          0.000    ?
Weighted Avg.    0.166    0.166    ?          0.166    ?

=== Confusion Matrix ===

   a   b   c   d   e   f   g   <-- classified as
   0   0 272   0   0   0   0 |   a = Insufficient_Weight
   0   0 287   0   0   0   0 |   b = Normal_Weight
   0   0 351   0   0   0   0 |   c = Obesity_Type_I
   0   0 297   0   0   0   0 |   d = Obesity_Type_II
   0   0 324   0   0   0   0 |   e = Obesity_Type_III
   0   0 290   0   0   0   0 |   f = Overweight_Level_I
   0   0 290   0   0   0   0 |   g = Overweight_Level_II
```

Figure 3: ZeroR Results

3.1 Logistic Regression

The logistic regression algorithm was run with the nobeyesdad classes. Weka first computes a linear combination of the input features and then applies a softmax function to the data to determine the probability of each of the output classes. It trains by minimizing a cross-entropy loss function and outputs the class with the highest probability for each instance. The results are shown in Figure 4 below:

```
Correctly Classified Instances        1982               93.8892 %
Incorrectly Classified Instances       129                6.1108 %
Kappa statistic                          0.9286
Mean absolute error                      0.0209
Root mean squared error                  0.1267
Relative absolute error                  8.5479 %
Root relative squared error             36.2199 %
Total Number of Instances             2111

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  I
                 0.978    0.004    0.974      0.978    0.976      (
                 0.864    0.014    0.908      0.864    0.886      (
                 0.954    0.013    0.938      0.954    0.946      (
                 0.956    0.003    0.979      0.956    0.968      (
                 0.994    0.001    0.994      0.994    0.994      (
                 0.897    0.024    0.858      0.897    0.877      (
                 0.921    0.013    0.918      0.921    0.919      (
Weighted Avg.    0.939    0.010    0.939      0.939    0.939      (

=== Confusion Matrix ===

   a   b   c   d   e   f   g   <-- classified as
 266   6   0   0   0   0   0 |   a = Insufficient_Weight
   7 248   0   0   0  30   2 |   b = Normal_Weight
   0   0 335   5   1   2   8 |   c = Obesity_Type_I
   0   0  10 284   1   0   2 |   d = Obesity_Type_II
   0   0   1   1 322   0   0 |   e = Obesity_Type_III
   0  18   0   0   0 260  12 |   f = Overweight_Level_I
   0   1  11   0   0  11 267 |   g = Overweight_Level_II
```

3.2 Support Vector Machine

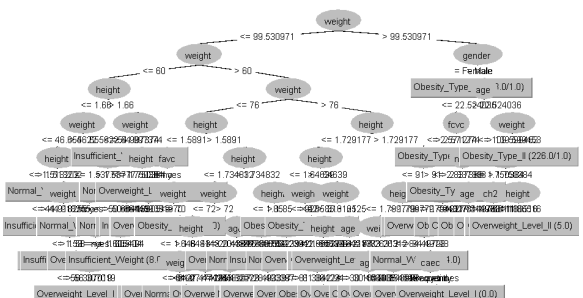
The support vector machine classifier was run with the specified classes. The best performing parameters were found to be a C value of 100 and a polynomial kernel, which maps the data into higher dimensional space using polynomial relationships between the input features, essentially allowing more complex decision boundaries than linear kernels. The results are shown in Figure 5 below:

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        2027               96.0208 %
Incorrectly Classified Instances        84                3.9792 %
Kappa statistic                          0.9535
Mean absolute error                      0.2047
Root mean squared error                  0.3021
Relative absolute error                 83.6801 %
Root relative squared error             86.3859 %
Total Number of Instances             2111

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure
                 0.989    0.007    0.954      0.989    0.971
                 0.941    0.008    0.951      0.941    0.946
                 0.969    0.005    0.974      0.969    0.971
                 0.980    0.004    0.977      0.980    0.978
                 0.994    0.001    0.997      0.994    0.995
                 0.910    0.010    0.933      0.910    0.921
                 0.934    0.012    0.928      0.934    0.931
Weighted Avg.    0.960    0.006    0.960      0.960    0.960

=== Confusion Matrix ===

   a   b   c   d   e   f   g   <-- classified as
 269   3   0   0   0   0   0 |   a = Insufficient_Weight
  13 270   0   0   0   4   0 |   b = Normal_Weight
   0   0 340   5   0   0   6 |   c = Obesity_Type_I
   0   0   5 291   1   0   0 |   d = Obesity_Type_II
   0   0   0   2 322   0   0 |   e = Obesity_Type_III
   0  11   0   0   0 264  15 |   f = Overweight_Level_I
   0   0   4   0   0  15 271 |   g = Overweight_Level_II
```

3.3 J48 Decision Tree

The J48 decision tree algorithm was run with the specified classes with solid results. Decision tree splits features based on information gain, maximizing entropy of the majority class. We applied a confidence factor of 0.25, a seed of 1, and automatic pruning. The tree reached a size of 115 with 59 leaf nodes.

Figure 6: J48 Tree




```
Correctly Classified Instances        2010               95.2155 %
Incorrectly Classified Instances       101                4.7845 %
Kappa statistic                          0.9441
Mean absolute error                      0.0163
Root mean squared error                  0.1148
Relative absolute error                  6.6483 %
Root relative squared error             32.8273 %
Total Number of Instances             2111

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure
                 0.978    0.007    0.953      0.978    0.966
                 0.906    0.009    0.939      0.906    0.922
                 0.963    0.005    0.974      0.963    0.968
                 0.970    0.007    0.957      0.970    0.963
                 0.991    0.003    0.982      0.991    0.986
                 0.921    0.014    0.911      0.921    0.916
                 0.931    0.009    0.941      0.931    0.936
Weighted Avg.    0.952    0.008    0.952      0.952    0.952

=== Confusion Matrix ===

   a   b   c   d   e   f   g   <-- classified as
 266   6   0   0   0   0   0 |   a = Insufficient_Weight
  13 260   0   0   0  14   0 |   b = Normal_Weight
   0   0 338   6   2   0   5 |   c = Obesity_Type_I
   0   0   5 288   4   0   0 |   d = Obesity_Type_II
   0   0   0   3 321   0   0 |   e = Obesity_Type_III
   0  11   0   0   0 267  12 |   f = Overweight_Level_I
   0   0   4   4   0  12 270 |   g = Overweight_Level_II
```

Figure 7: J48 Summary Results

3.4 K-Nearest Neighbors

KNN was performed on the target class as *Instance-Based k* in Weka. The optimal k parameter for the number of neighbors was found to be 3 and a LinearNN search kernal was used to simply brute force the distance search.

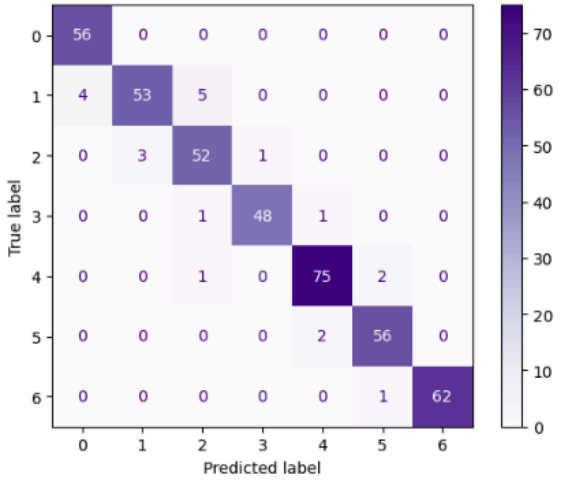
Figure 8: KNN Summary Ressults

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1694               80.2463 %
Incorrectly Classified Instances       417               19.7537 %
Kappa statistic                          0.7692
Mean absolute error                      0.0651
Root mean squared error                  0.2059
Relative absolute error                 26.6192 %
Root relative squared error             58.8828 %
Total Number of Instances             2111

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure
                 0.919    0.049    0.733      0.919    0.816
                 0.467    0.048    0.606      0.467    0.528
                 0.846    0.045    0.788      0.846    0.816
                 0.929    0.018    0.896      0.929    0.912
                 0.991    0.003    0.985      0.991    0.988
                 0.714    0.032    0.781      0.714    0.746
                 0.721    0.035    0.766      0.721    0.742
Weighted Avg.    0.802    0.033    0.798      0.802    0.797

=== Confusion Matrix ===

   a   b   c   d   e   f   g   <-- classified as
 250  15   1   0   0   4   2 |   a = Insufficient_Weight
  67 134  21   3   1  27  34 |   b = Normal_Weight
   4  12 297  18   3   7  10 |   c = Obesity_Type_I
   1   3   9 276   0   1   7 |   d = Obesity_Type_II
   0   1   0   1 321   0   1 |   e = Obesity_Type_III
```

3.5 XGBoost

The XGBoost algorithm was run with the specified classes encoded to the range 1-7. Because Weka cannot perform gradient boosting natively, XGBoost had to be deployed manually in Python. To enable categorical data we used XGBoost's low level DMatrix API and performed 100 rounds of boosting. The results are shown in Figure 9 below:

```
confusion matrix:
 [[56  0  0  0  0  0  0]
 [ 4 53  5  0  0  0  0]
 [ 0  3 52  1  0  0  0]
 [ 0  0  1 48  1  0  0]
 [ 0  0  1  0 75  2  0]
 [ 0  0  0  0  2 56  0]
 [ 0  0  0  0  0  1 62]]
classification report:
              precision    recall  f1-score   support

           0       0.93      1.00      0.97        56
           1       0.95      0.85      0.90        62
           2       0.88      0.93      0.90        56
           3       0.98      0.96      0.97        50
           4       0.96      0.96      0.96        78
           5       0.95      0.97      0.96        58
           6       1.00      0.98      0.99        63

    accuracy                           0.95       423
   macro avg       0.95      0.95      0.95       423
weighted avg       0.95      0.95      0.95       423

Mean Squared Error (MSE): 0.0567
Mean Absolute Error (MAE): 0.0520
R² Score: 0.9857
```



3.6 Random Forest

The Random Forest algorithm was run for the target class. The most effective parameters proved to be an ensemble of 200 tree iterations with unlimited max depth. The results are shown in Figure 10 below.

```
Correctly Classified Instances        2028               96.0682 %
Incorrectly Classified Instances        83                3.9318 %
Kappa statistic                          0.9541
Mean absolute error                      0.0444
Root mean squared error                  0.1146
Relative absolute error                 18.1469 %
Root relative squared error             32.7556 %
Total Number of Instances             2111

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure
                 0.945    0.002    0.988      0.945    0.966
                 0.955    0.027    0.848      0.955    0.898
                 0.977    0.004    0.980      0.977    0.979
                 0.987    0.002    0.990      0.987    0.988
                 0.994    0.001    0.997      0.994    0.995
                 0.907    0.006    0.960      0.907    0.933
                 0.952    0.005    0.968      0.952    0.960
Weighted Avg.    0.961    0.006    0.963      0.961    0.961

=== Confusion Matrix ===

   a   b   c   d   e   f   g   <-- classified as
 257  15   0   0   0   0   0 |   a = Insufficient_Weight
   3 274   0   0   0   9   1 |   b = Normal_Weight
   0   3 343   1   0   0   4 |   c = Obesity_Type_I
   0   1   2 293   1   0   0 |   d = Obesity_Type_II
   0   0   0   2 322   0   0 |   e = Obesity_Type_III
   0  23   0   0   0 263   4 |   f = Overweight_Level_I
   0   7   5   0   0   2 276 |   g = Overweight_Level_II
```

3.7 Naive Bayes

The Naive Bayesian algorithm performed rather poorly when applied to the dataset. This however, could be expected as there are 15 input features that likely correlate where Naive Bayes assumes independence and 7 possible output classes. The results are shown in Figure 11 below.

```
Correctly Classified Instances        1425               67.5036 %
Incorrectly Classified Instances       686               32.4964 %
Kappa statistic                          0.6205
Mean absolute error                      0.1089
Root mean squared error                  0.2542
Relative absolute error                 44.5319 %
Root relative squared error             72.6807 %
Total Number of Instances             2111

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure
                 0.897    0.046    0.742      0.897    0.812
                 0.488    0.056    0.579      0.488    0.529
                 0.516    0.082    0.557      0.516    0.536
                 0.835    0.038    0.782      0.835    0.808
                 1.000    0.029    0.862      1.000    0.926
                 0.514    0.047    0.637      0.514    0.569
                 0.479    0.082    0.483      0.479    0.481
Weighted Avg.    0.675    0.055    0.663      0.675    0.666

=== Confusion Matrix ===

   a   b   c   d   e   f   g   <-- classified as
 244  22   0   0   1   5   0 |   a = Insufficient_Weight
  76 140   0   0  18  37  16 |   b = Normal_Weight
   0   1 181  63  15  25  66 |   c = Obesity_Type_I
   0   1  43 248   5   0   0 |   d = Obesity_Type_II
   0   0   0   0 324   0   0 |   e = Obesity_Type_III
```

3.8 Multilayer Perceptron

The multilayer perceptron model was run on the target class. It made use of a GUI to visualize the model which may be found at Fig. 13 in the appendix. For multilayer, we applied no decay, a learning rate of 0.3, momentum of 0.2, exponent of 1.0, and a training time of 500. (Any more iterations had what felt like indefinite build times). This was the only algorithm in which the seed used was 0 and not 1 by default.

Figure 14: Perceptron Summary Results

```
Correctly Classified Instances        1970               93.3207 %
Incorrectly Classified Instances       141                6.6793 %
Kappa statistic                          0.922
Mean absolute error                      0.0233
Root mean squared error                  0.1259
Relative absolute error                  9.5049 %
Root relative squared error             35.9927 %
Total Number of Instances             2111

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure
                 0.963    0.012    0.923      0.963    0.942
                 0.875    0.014    0.906      0.875    0.890
                 0.966    0.004    0.980      0.966    0.973
                 0.976    0.003    0.980      0.976    0.978
                 0.991    0.001    0.997      0.991    0.994
                 0.841    0.022    0.859      0.841    0.850
                 0.907    0.021    0.871      0.907    0.889
Weighted Avg.    0.933    0.011    0.933      0.933    0.933

=== Confusion Matrix ===

   a   b   c   d   e   f   g   <-- classified as
 262  10   0   0   0   0   0 |   a = Insufficient_Weight
  22 251   0   0   0  14   0 |   b = Normal_Weight
   0   0 339   4   0   1   7 |   c = Obesity_Type_I
   0   0   5 290   1   0   1 |   d = Obesity_Type_II
   0   0   0   2 321   0   1 |   e = Obesity_Type_III
   0  16   0   0   0 244  30 |   f = Overweight_Level_I
   0   0   2   0   0  25 263 |   g = Overweight_Level_II
```

4. CLUSTERING AND ASSOCIATION

We will be applying the *k-means clustering* algorithm to attempt to divide the dataset into meaningful clusters. The algorithm in Weka will be set with parameters a maximum of 100 canopies, a maximum canopy density of 2.0, a seed of 10, and a pruning rate of 10000. Manhattan distance was used due to the number of nominal and binary features. To reach the point where each of the output classes represented a majority in at least one cluster, 10 clusters were used. All features served some relevance with even minor correlations and all were retained. The attribute prevalence can be found in Figures 15 and 16 in the appendix.

The *APRIORI* algorithm was additionally utilized to devise association rules for the data set. We looked for 30 rules with the delta as 0.05, the minimum metric as 0.9, the minimum support between 0.1 and 1.0. Because the APRIORI algorithm finds association rules based on occurrences (support) it requires nominal data. Weka's Discretize filter was used to split numeric features into 10 bins within the range of the minimum and maximum of each feature. The resulting list of rules can be found in Figure 17 in the appendix. The minimum amount of support was found to be 0.7 for the dataset.

5. BEST MODEL

Of the original evaluations of classification models, the highest accuracy, recall value, and kappa statistic and the lowest Root Mean Squared Error belonged to the J48 decision tree, Random Forest, and SVM in order of RMSE. To confirm this order, Weka's Estimator was used to evaluate the models for 10 iterations with 0.1 significance looking for Mean Absolute Error, again ranking J48 the highest but swapping the other two.

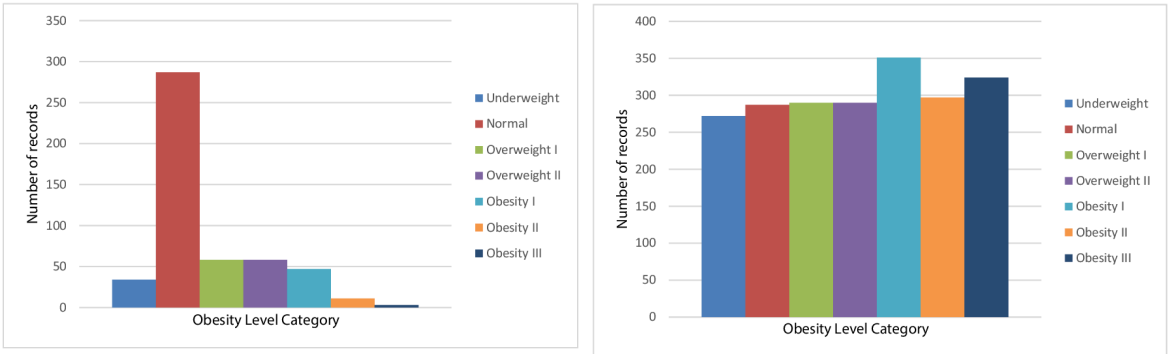
Figure 18: Estimator Results

```
Analysing:  Mean_absolute_error
Datasets:   1
Resultsets: 3
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       12/8/24, 6:45 PM


Dataset                   (1) trees.J | (2) func (3) tree
----------------------------------------------------------
imputed_dataset          (100)   0.02 |   0.20 v   0.04 v
----------------------------------------------------------
                              (v/ /*) |  (1/0/0)  (1/0/0)


Key:
(1) trees.J48 '-C 0.25 -M 2' -217733168393644444
(2) functions.SMO '-C 100.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K \"functions.supportVector.PolyK
(3) trees.RandomForest '-P 100 -I 200 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1' 1116839470751428698
```

6. CONCLUSION

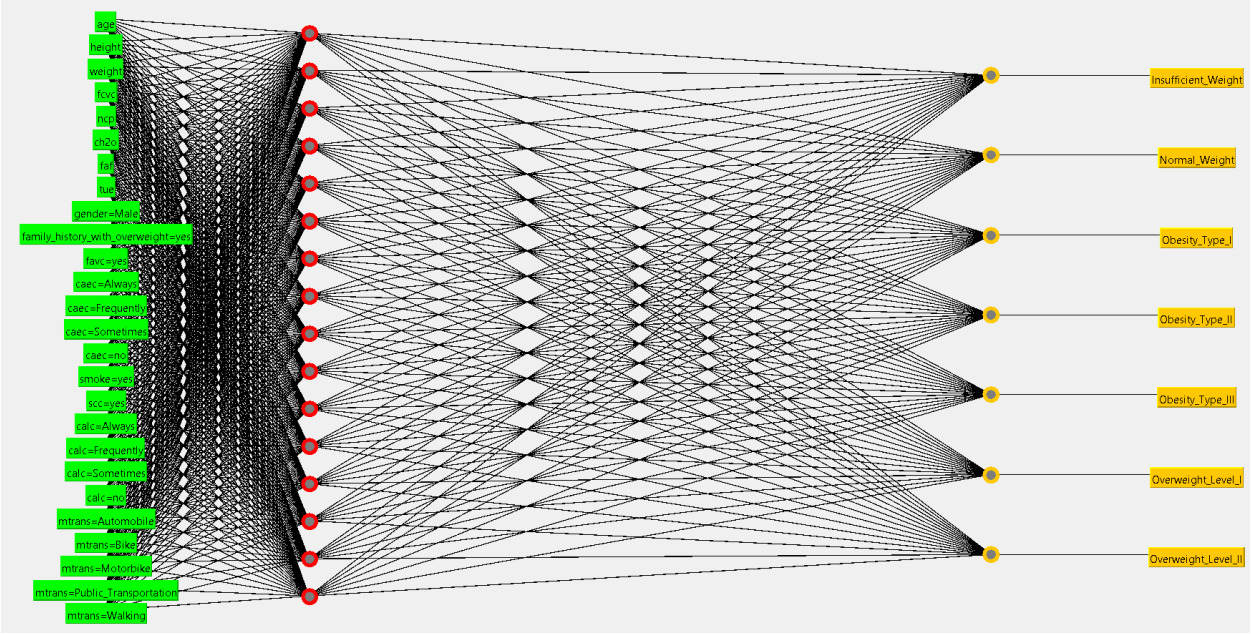
In this paper we demonstrated the use of several machine learning algorithms to predict a classification of obesity. As emphasized in the introduction, machine learning offers immense potential for the field of healthcare and the strong performance of models like J48, Random Forest, and SVM highlights their potential in it. Considering the expectation of noise in such a large dataset collected via survey, the models performed far better than anticipated. At first one would assume overfitting, but this is with ten-fold cross validation. Weka proved to be an invaluable tool for preprocessing, visualization, and testing and we hope this paper showcases its potential and a general understanding of the algorithms it can apply.

7. APPENDIX



Figures 1 and 2: Unbalanced distribution of data(left) and balanced distribution after applying SMOTE [1]

Figure 13: Perceptron Neural Network:



Figures 15 and 16: Attribute prevalence in k-means clustering

Attribute	Full Data (2111.0)	Cluster# 0 (113.0)	1 (336.0)	2 (75.0)	3 (444.0)	4 (205.0)
age	22.7894	19.9317	24.9105	20	25.0212	21.3928
height	1.7002	1.5995	1.6687	1.72	1.7684	1.6304
weight	83	45	111.946	65	111.9752	67
fcvc	2.3874	3	3	2	2.1873	2.4575
ncp	3	3	3	3	3	3
ch2o	2	1.527	2.4457	2	2	2
faf	1	1.0678	0.2191	2	0.9929	1
tue	0.6253	0.4037	0.658	1	0.4288	1
gender	Male	Female	Female	Male	Male	Female
family_history_with_overweight	yes	no	yes	no	yes	yes
favc	yes	no	yes	yes	yes	yes
caec	Sometimes	Frequently	Sometimes	Frequently	Sometimes	Sometimes
smoke	no	no	no	no	no	no
scc	no	no	no	no	no	no
calc	Sometimes	Sometimes	Sometimes	no	Sometimes	no
mtrans	Public_Transportation	Public_Transportation	Public_Transportation	Public_Transportation	Public_Transportation	Public_Transportation
nobeyesdad	Obesity_Type_I	Insufficient_Weight	Obesity_Type_III	Normal_Weight	Obesity_Type_II	Obesity_Type_I

Attribute	5 (147.0)	6 (93.0)	7 (196.0)	8 (212.0)	9 (290.0)
age	37.4962	21	27.4499	21.3433	21
height	1.6478	1.72	1.7993	1.7349	1.62
weight	79.6973	68	85.47	97.2679	68.9448
fcvc	2.0117	3	2.4945	2	2
ncp	3	3	3	3	3
ch2o	1.8446	2	2.0133	2	2
faf	0.1803	1	2	1	0.9514
tue	0	0	0.3087	1.1638	0.9771
gender	Female	Male	Male	Male	Female
family_history_with_overwei	yes	yes	yes	yes	yes
favc	yes	yes	yes	yes	yes
caec	Sometimes	Frequently	Sometimes	Sometimes	Sometimes
smoke	no	no	no	no	no
scc	no	no	no	no	no
calc	no	Sometimes	Sometimes	no	Sometimes
mtrans	Automobile	Public_Transportation	Automobile	Public_Transportation	Public_Transportation
nobeyesdad	Overweight_Level_II	Normal_Weight	Overweight_Level_I	Obesity_Type_I	Overweight_Level_I

Figure 17: APRIORI top 30 rules

1. favc=yes caec=Sometimes 1608 ==> smoke=no 1585 <conf:(0.99)> lift:(1.01) lev:(0) [10] conv:(1.4)
2. family_history_with_overweight=yes caec=Sometimes smoke=no 1520 ==> scc=no 1498 <conf:(0.99)> lift:(1.03) lev:(0.02) [46] conv:(2.97)
3. favc=yes caec=Sometimes scc=no 1568 ==> smoke=no 1545 <conf:(0.99)> lift:(1.01) lev:(0) [9] conv:(1.36)
4. family_history_with_overweight=yes caec=Sometimes 1548 ==> scc=no 1524 <conf:(0.98)> lift:(1.03) lev:(0.02) [45] conv:(2.79)
5. family_history_with_overweight=yes favc=yes scc=no 1552 ==> smoke=no 1526 <conf:(0.98)> lift:(1) lev:(0) [6] conv:(1.2)
6. caec=Sometimes scc=no 1712 ==> smoke=no 1683 <conf:(0.98)> lift:(1) lev:(0) [6] conv:(1.19)
7. family_history_with_overweight=yes caec=Sometimes scc=no 1524 ==> smoke=no 1498 <conf:(0.98)> lift:(1) lev:(0) [5] conv:(1.18)
8. favc=yes scc=no 1809 ==> smoke=no 1778 <conf:(0.98)> lift:(1) lev:(0) [6] conv:(1.18)
9. scc=no mtrans=Public_Transportation 1507 ==> smoke=no 1481 <conf:(0.98)> lift:(1) lev:(0) [5] conv:(1.16)
10. family_history_with_overweight=yes favc=yes smoke=no 1553 ==> scc=no 1526 <conf:(0.98)> lift:(1.03) lev:(0.02) [42] conv:(2.5)
11. caec=Sometimes 1765 ==> smoke=no 1734 <conf:(0.98)> lift:(1) lev:(0) [5] conv:(1.15)
12. family_history_with_overweight=yes caec=Sometimes 1548 ==> smoke=no 1520 <conf:(0.98)> lift:(1) lev:(0) [4] conv:(1.11)
13. favc=yes 1866 ==> smoke=no 1832 <conf:(0.98)> lift:(1) lev:(0) [4] conv:(1.11)
14. family_history_with_overweight=yes favc=yes 1582 ==> smoke=no 1553 <conf:(0.98)> lift:(1) lev:(0) [3] conv:(1.1)
15. family_history_with_overweight=yes favc=yes 1582 ==> scc=no 1552 <conf:(0.98)> lift:(1.03) lev:(0.02) [41] conv:(2.3)
16. mtrans=Public_Transportation 1580 ==> smoke=no 1550 <conf:(0.98)> lift:(1) lev:(0) [2] conv:(1.06)
17. scc=no 2016 ==> smoke=no 1977 <conf:(0.98)> lift:(1) lev:(0) [3] conv:(1.05)
18. family_history_with_overweight=yes scc=no 1683 ==> smoke=no 1650 <conf:(0.98)> lift:(1) lev:(0) [2] conv:(1.03)
19. family_history_with_overweight=yes 1729 ==> smoke=no 1691 <conf:(0.98)> lift:(1) lev:(-0) [-1] conv:(0.92)
20. family_history_with_overweight=yes smoke=no 1691 ==> scc=no 1650 <conf:(0.98)> lift:(1.02) lev:(0.02) [35] conv:(1.81)
21. favc=yes caec=Sometimes 1608 ==> scc=no 1568 <conf:(0.98)> lift:(1.02) lev:(0.02) [32] conv:(1.76)
22. favc=yes caec=Sometimes smoke=no 1585 ==> scc=no 1545 <conf:(0.97)> lift:(1.02) lev:(0.01) [31] conv:(1.74)
23. family_history_with_overweight=yes 1729 ==> scc=no 1683 <conf:(0.97)> lift:(1.02) lev:(0.02) [31] conv:(1.66)
24. caec=Sometimes smoke=no 1734 ==> scc=no 1683 <conf:(0.97)> lift:(1.02) lev:(0.01) [27] conv:(1.5)
25. favc=yes smoke=no 1832 ==> scc=no 1778 <conf:(0.97)> lift:(1.02) lev:(0.01) [28] conv:(1.5)
26. caec=Sometimes 1765 ==> scc=no 1712 <conf:(0.97)> lift:(1.02) lev:(0.01) [26] conv:(1.47)
27. favc=yes 1866 ==> scc=no 1809 <conf:(0.97)> lift:(1.02) lev:(0.01) [26] conv:(1.45)
28. family_history_with_overweight=yes caec=Sometimes 1548 ==> smoke=no scc=no 1498 <conf:(0.97)> lift:(1.03) lev:(0.02) [48] conv:(1.93)
29. family_history_with_overweight=yes favc=yes 1582 ==> smoke=no scc=no 1526 <conf:(0.96)> lift:(1.03) lev:(0.02) [44] conv:(1.76)
30. favc=yes caec=Sometimes 1608 ==> smoke=no scc=no 1545 <conf:(0.96)> lift:(1.03) lev:(0.02) [39] conv:(1.59)

8. WORKS CITED

[1] Palechor, Fabio Mendoza, and Alexis De la Hoz Manotas. "Dataset for Estimation of Obesity Levels Based on Eating Habits and Physical Condition in Individuals from Colombia, Peru and Mexico." Data in Brief, vol. 25, 2019, p. 104344.