# University of Cincinnati

**Date: 7/12/2023**

**I, Hanniel  Shih, hereby submit this original work as part of the requirements for the degree of Master of Science in Computer Science.**

It is entitled:

**Anomaly Detection in Irregular Time Series using Long Short-Term Memory with Attention**

Student's name: **Hanniel  Shih**

This work and its defense approved by:

**Committee chair: Raj Bhatnagar, Ph.D.**

**Committee member: Vikram Ravindra, Ph.D.**

**Committee member: Danny T. Y. Wu, PhD**

46688

# Anomaly Detection in Irregular Time Series using Long Short-Term Memory with Attention

by

Hanniel, Shih

Thesis submitted to the
Graduate School
of the University of Cincinnati
in partial fulfillment of the
requirements for degree of

**Master of Science**

in the Department of Computer Science
of the College of Engineering and Applied Science

## Abstract

Anomaly Detection in Irregular Time Series is an under-explored topic, especially in the healthcare domain. An example of this is weight entry errors. Erroneous weight records pose significant challenges to healthcare data quality, impacting clinical decision-making and patient safety. Existing studies primarily utilize rule-based methods, achieving an Area Under the Receiver Operating Characteristic Curve (AUROC) ranging from 0.546 to 0.620. This thesis introduces a two-module method, employing bi-directional Long Short-Term Memory (bi-LSTM) with Attention Mechanism, for the prospective detection of anomalous weight entries in electronic health records. The proposed method consists of a predictor and a classifier module, both leveraging bi-LSTM and Attention Mechanism. The predictor module learns the normal pattern of weight changes, and the classifier module identifies anomalous weight entries. The performance of both modules was evaluated, exhibiting a clear superiority over other methods in distinguishing normal from anomalous data points. Notably, the proposed approach achieved an AUROC of 0.986 and a precision of 9.28%, significantly outperforming other methods when calibrated for a similar sensitivity. This study contributes to the field of entry error detection in healthcare data, offering a promising solution for real-time anomaly detection in electronic health records.

## Acknowledgements

I would like to express my sincere gratitude and appreciation to all those who supported me throughout my pursuit of a Master of Science degree at the University of Cincinnati.

Firstly, I would like to thank my parents for providing support for me to realize my dream of studying in the US. It is their support that allows me to have a memorable 2 years of education with the peace of mind that they always have my back.

Secondly, I would like to thank Prof. Danny Wu, who has been mentoring me for the majority of my journey. Under his guidance, I have had the opportunity to apply my skills in medical research and publish my work. His support and guidance have been instrumental in my professional growth, and I am immensely grateful for the experience and knowledge he has imparted to me.

I would also like to thank my thesis committee, Prof. Raj Bhatnagar, Prof. Danny Wu, and Prof. Vikram Ravindra, for taking the time to review my work and provide valuable feedback.

I also extend my heartfelt gratitude to everyone who has helped or inspired me during my Master's program at the University of Cincinnati.

Sincerely,

Hanniel Shih

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Background and Motivation

Time series data is essential in various domains, providing valuable insights into temporal information and supporting decision-making processes. While the detection of anomalies in time series data has been extensively studied, the identification of incorrectly recorded data entries has received limited attention. Detecting erroneous values in time series data is a critical task that can have significant implications for data cleaning, analysis, and decision-making processes.

One specific area where the detection of entry errors is of paramount importance is in Electronic Health Records (EHRs), particularly in the context of pediatric healthcare. Erroneous data entries in EHRs can have severe consequences, potentially leading to incorrect diagnoses, treatment plans, and patient outcomes. Inaccurate weight entries, for example, can mislead healthcare providers into making incorrect and potentially life-threatening judgments, particularly in weight-based dosing practices commonly employed in pediatric care.

Addressing the challenge of detecting entry errors in time series data can offer valuable solutions for data cleaning and improving the integrity of medical records. Prospective entry error detection, in particular, can pave the way for the development of real-time alert systems at the point of data entry, prompting users to review and double-check their input if the system detects implausible values. Such systems can significantly enhance data accuracy and patient safety in healthcare settings.

While existing approaches in informatics have proposed rule-based methods to identify abnormal values based on empirical thresholds, they often lack the required detection capacity, particularly for complex time series data like weight growth charts. These rule-based approaches rely on fixed thresholds and do not effectively capture the dynamic nature of time series data. Therefore, there is a need for more sophisticated and accurate methods that can capture the complex patterns and dynamics inherent in time series data.

By harnessing the advancements in deep learning and specifically employing bidirectional Long Short-Term Memory (bi-LSTM) models with Attention Mechanism, this study aims to develop an effective approach for detecting prospective entry errors in time series data, with a particular focus on weight entries in pediatric EHRs. Both static patient information, such as gender and age, and dynamic weight data will be leveraged to create a comprehensive model capable of identifying erroneous entries at the time of data entry.

The significance of this research lies in the potential to improve the quality and reliability of time series data in healthcare. By accurately detecting and preventing entry errors, data integrity and patient outcomes can be enhanced and improved. The proposed approach has the potential to contribute to the fields of deep learning and patient safety and make an impact on the healthcare industry.

## 1.2 Research Objectives

The main objective of this study is to detect prospective weight entry errors in pediatric EHRs, with the aim of preventing erroneous data from being recorded. The specific research objectives are as follows:

1. Develop a novel approach for detecting entry errors in time series data, focusing on prospective and real-time error detection.

2. Design and implement a bidirectional Long Short-Term Memory (bi-LSTM) model with Attention Mechanism and incorporating Self-Organizing Map - Minimum Quantization Error (SOM-MQE) for detecting erroneous weight entries.

3. Investigate the effectiveness of combining static data, such as gender, with dynamic time-varying data, like weight, for improved error detection performance.

4. Validate the proposed approach using a comprehensive dataset obtained from the Cincinnati Children's Hospital Medical Center (CCHMC), consisting of de-identified pediatric weight entry data.

5. Assess the performance and effectiveness of the proposed approach through experimentation and comparison with existing methods.

6. Provide insights into the potential implications and applications of the developed approach in enhancing data quality and improving patient safety.

## 1.3 Structure of the Thesis

This thesis is organized into several chapters to provide a comprehensive understanding of the research conducted and the contributions made. The structure of the thesis is as follows:

- Chapter 1 Introduction

  Presents the background, motivation, research objectives, and the thesis structure.

- Chapter 2 Literature Review

  Provides an extensive review of related works in the fields of entry error detection, time series analysis, deep learning, and other relevant methodologies. It explores existing approaches, techniques, and challenges in anomaly detection in time series data.

- Chapter 3 Methodology

  Describes the dataset used in this study and provides a detailed explanation of the proposed approach, including the predictor and classifier module using bidirectional LSTM with Attention Mechanism and the SOM-MQE method for feature extraction. It also discusses the rationale behind the design choices and the integration of static and dynamic data.

- Chapter 4 Experiments

  Presents the experimental setup and performance evaluation metrics. It includes a comprehensive overview of the training process and parameters used.

- Chapter 5 Results

  Provides a detailed presentation of the findings from the experiments. It includes the quantitative results, performance metrics, relevant visualizations or statistical analyses, and comparative results against existing methods.

- Chapter 6 Discussion

  Presents a detailed discussion of the results obtained based on the observations for individual components of the proposed method and addresses limitations.

- Chapter 7 Conclusion

  Summarizes the conclusions based on the findings and contributions of the research. It also outlines potential avenues for future research and development in the field of detecting entry errors in time series data.

# Chapter 2

# Literature Review

The detection of anomalous weight entries in electronic health records (EHRs) is an ongoing challenge within the healthcare research community. Traditional approaches to addressing this problem have predominantly relied on rule-based algorithms, with varying degrees of success.

Daymont et al. (2017)[1] pioneered an innovative automated method for identifying implausible pediatric growth data within EHRs. Their technique involved expressing each measurement as a standard deviation score and then comparing it against a weighted moving average of a child's other measurements. Applied to a substantial dataset of over 2 million growth measurements from 280,610 patients aged between 1 to 21 years, this method flagged 3.8% of weight and 4.5% of height values as implausible or otherwise excluded.

Building on this foundational work, Liu et al. (2019)[2] ventured beyond rule-based methods and instead leveraged machine learning and deep learning techniques to tackle this challenge. Their innovative study applied a bidirectional Long Short-Term Memory (bi-LSTM) model to identify abnormal weight entries within pediatric patient records. Tested and validated on a comprehensive dataset of 15,000 pediatric patients from Cincinnati Children's Hospital Medical Center, Liu's approach was an important milestone in its application of using bi-LSTM for anomaly detection within a healthcare context. However, their use of bi-LSTM enables the model to observe both preceding and following data points when making a prediction. Therefore, it can does not provide a solution to prospective weight entry error detection.

Since the publication of ”Attention is all you need” by Vaswani et al.[3], Attention Mechanism has seen widespread popularity in Deep Learning, particularly in Natural Language Processing (NLP). Attention Mechanism is capable of interpreting sequential data by paying more ”attention” to relevant information by assigning more weight to the most helpful information. Attention Mechanism has three inputs, namely, Query (Q), Key (K), and Value (V). Although primarily utilized in NLP, Attention Mechanism can also be utilized for Time Series Analysis since Time Series can be considered a type of sequential data. Fig 2.1 shows an Illustration of Multihead Attention described by Vaswani et al.

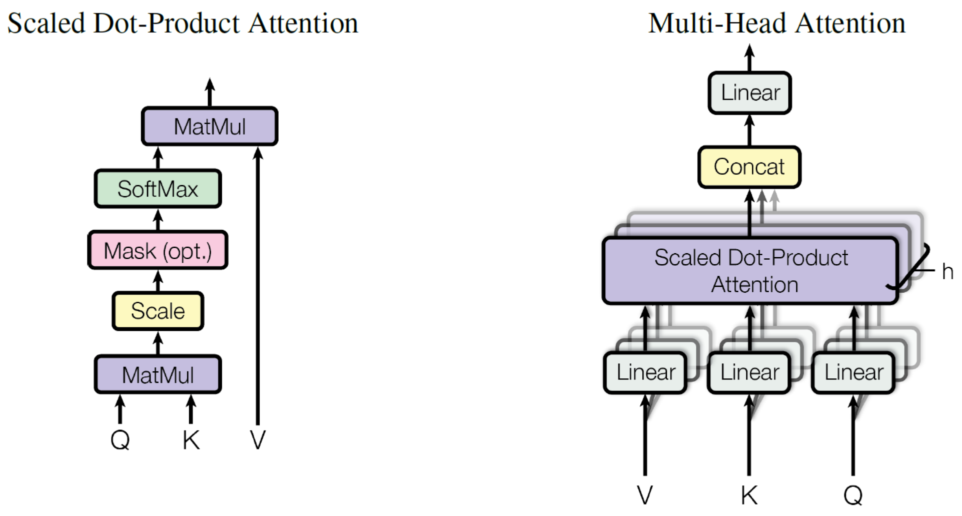


Figure 2.1: Illustration of Attention Mechanism

More recent research has further pushed the boundaries by exploring the potential of machine learning technologies for anomaly detection. Kong et al. (2023)[4], for example, proposed an integrated generative model that combined bi-LSTM and an Attention Mechanism. Primarily targeted at industrial anomaly detection, Kong’s model demonstrated an ability to accurately and promptly identify anomalies within time-series data, highlighting the potential for such a model in a healthcare context.

Separately, Self-Organizing Map - Minimum Quantization Error (SOM-MQE) has been established as a great indicator for anomaly detection tasks[5]. Due to its ability to represent known data, when trained using normal data only, the minimum quantization error can serve as a great indication of how dissimilar a given data is to previously observed data. Even though the value of SOM-MQE in anomaly detection tasks has been shown extensively, it lacks the ability to incorporate temporal information. Consequently, on its own, SOM-MQE can not match the performance of Deep Learning algorithms that can interpret temporal information, such as LSTM.

Despite these advances, three key areas warranting further investigation were identified during the literature review. Firstly, while studies such as Kong's[4] have shown the potential of Attention Mechanisms in anomaly detection, there is currently a lack of research exploring their integration within healthcare data. Secondly, the majority of existing studies focus predominantly on retrospective error detection, a process that uses both preceding and following data points. This approach, although valuable in data cleaning, fails to address the need for real-time alerts that could prevent erroneous entries from being recorded in the first place. Lastly, the utilization of SOM-MQE for anomaly detection in time-series data, specifically in electronic health records (EHRs), has not gained widespread popularity.

To address these gaps, this study proposes a novel two-module method that integrates bi-LSTM, Attention Mechanism, and SOM-MQE for the prospective detection of erroneous weight entries in EHRs. By incorporating bi-LSTM, which excels at embedding temporal information, Attention Mechanism, which further focuses on specific parts of the temporal information, and SOM-MQE, which provide valuable insight into the level of abnormality of each data point, it is hypothesized that the resulting model can surpass the performance of previous methods. The model is specifically designed for prospective entry error detection, which relies solely on preceding data points for anomaly detection. By focusing on prospective error detection, this study aims to evaluate the performance and practicality of deep learning models as real-time alert systems at the point of data entry. Furthermore, by segregating the task into two distinct parts—learning the pattern of weight changes and detecting anomalous entries—this study aims to create a model that can provide the expected weight in addition to detecting prospective weight entry errors with greater accuracy and sensitivity.

# Chapter 3

# Methodology

The input data was first put thru Self-Organizing Map - Minimum Quantization Error (SOM-MQE), where individual data points were analyzed and assigned an anomaly score, and then organized into sequences for the deep learning modules. The two modules-Predictor Module and Classifier Module-both utilizes bi-LSTM[6] with attention[3], which takes in and interprets temporal data and puts focus on relevant information. The use of bi-LSTM, however, inherently includes information both prospectively and retrospectively, which conflicts with the aim of prospective error detection. Therefore, a real-time simulation process was introduced to prohibit such information leaks. Fig 3.1 shows a diagram of the proposed method.

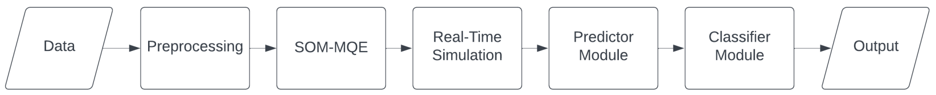


Figure 3.1: Overview of Proposed Method

## 3.1 SOM-MQE

The Self-Organizing Map (SOM), introduced by Teuvo Kohonen[7], is a type of artificial neural network trained using unsupervised learning to produce a low-dimensional, discretized representation of the input space of the training samples, and is widely used for clustering and visualization. SOMs operate through competition among neurons. During the training process, SOM iteratively adjusts the weight vectors of neurons to resemble the input vectors.

The weight update rule in SOM can be mathematically represented as:

$$w_j(t+1) = w_j(t) + \alpha(t) \cdot h_{ji}(t) \cdot (x(t) - w_j(t)) \tag{3.1}$$

where $w_j$ is the weight vector of the $j$th neuron, $\alpha(t)$ is the learning rate at time $t$, $x(t)$ is the input vector, and $h_{ji}(t)$ is the neighborhood function at time $t$.

The neuron whose weight vector is most similar to the input vector is termed the Best Matching Unit (BMU). The BMU's weight is adjusted to be more similar to the input vector.

In this study, Self-Organizing Map - Minimum Quantization Error (SOM-MQE) is used to capture the representation error of each weight data sample. MQE quantifies the distance between a given sample and the weight of its Best Matching Unit (BMU) in the SOM. High MQE values indicate that the data sample is not well-represented by the map, and thus it can be considered as an anomaly.

Mathematically, the MQE of a data sample $x$ is given by:

$$\text{MQE}(x) = \|x - w_{\text{BMU}}\| \tag{3.2}$$

where $w_{\text{BMU}}$ is the weight vector of the BMU for the data sample $x$, and $\|\cdot\|$ represents the

Euclidean distance.

MQE values serve as additional features that provide insights into how dissimilar each data sample is to the typical patterns in the dataset, which can be instrumental in effectively identifying anomalous weight records.

## 3.2 Real-Time Simulation

A real-time simulation procedure was implemented to generate the dataset used for training and evaluation. The primary aim of this process is to simulate the scenario where previous weight entries were recorded correctly over time, establishing a baseline for each patient, and examining a potentially erroneous new entry as it is being recorded.

During this process, the records were divided into multiple sequences using a sliding window of size five. Each sequence is composed of five weight entries, with the fifth entry being the one whose correctness is to be predicted. The selection criteria for the entries depends on the module being trained: for the classifier module, the last entry of a generated sequence can be either normal or abnormal weight entries, whereas, for the predictor module, all weights in the generated sequence must be normal.

By simulating real-time weight entries, the model can be trained to detect erroneous weight values at the time of entry, which provides an opportunity to prevent incorrect data from being recorded. This real-time simulation approach enhances the practical application of the proposed method in detecting and preventing entry errors in time-series data. This real-time simulation process also enables the use of bidirectional Long Short-Term Memory without temporal data leakage, as the model is always predicting the correctness of the last entry in the sequence.

The process of generating the sequences for the real-time simulation is shown in Algorithm 1 for the classifier module, and in Algorithm 2 for the predictor module.

**Algorithm 1** Real-Time Simulation Sequence Generation

**Input**: Patient records *records*, Window size *window_size*
**Output**: Sequences of patient records

```
1: $sequences \leftarrow \emptyset$
2: $normal\_records \leftarrow \emptyset$
3: for each $record$ in $records$ in order do
4:     if number of $normal\_records = window\_size - 1$ then
5:         create $sequence$ with $normal\_records$ and current $record$
6:         append $sequence$ to $sequences$
7:         remove the oldest record from $normal\_records$
8:     end if
9:     if $record.label = normal$ then
10:        append $record$ to $normal\_records$
11:    end if
12: end for
13: return $sequences$
```

**Algorithm 2** Real-Time Simulation Sequence Generation for Predictor Module

**Input**: Patient records *records*, Window size *window_size*
**Output**: Sequences of patient records

```
1: $sequences \leftarrow \emptyset$
2: $normal\_records \leftarrow \emptyset$
3: for each $record$ in $records$ in order do
4:     if $record.label = normal$ then
5:         append $record$ to $normal\_records$
6:     end if
7:     if number of $normal\_records = window\_size$ then
8:         create $sequence$ with $normal\_records$
9:         append $sequence$ to $sequences$
10:        remove the oldest record from $normal\_records$
11:    end if
12: end for
13: return $sequences$
```

## 3.3 Model Architecture

### 3.3.1 Predictor Module

The Predictor Module is responsible for predicting the weight of a patient based on input features such as age and historical weight data. This module utilizes a single-layer bidirectional LSTM network, coupled with Attention Mechanism, to capture sequential patterns and temporal dependencies in the data for accurate weight prediction.

The architecture of the Predictor Module includes the following components:

- **Bidirectional LSTM Layer:** The module employs a single layer bidirectional LSTM with a hidden size of 256. This results in an output dimension of 512 due to bidirectionality. The bi-LSTM layer captures temporal dependencies in the data and encodes this sequential information.
- **Batch Normalization:** After the LSTM layer, batch normalization[8] is applied to improve the stability and generalization of the model by normalizing the activations within each mini-batch.
- **Age Embedding:** The age of the patient is embedded into a higher-dimensional representation using a single linear layer, allowing it to be passed to the Multihead Attention.
- **Multihead Attention:** A multihead attention[3] with 4 heads is utilized. The embedded age is passed as the query, and the LSTM output is passed as the key and value to the attention mechanism, allowing the model to focus on relevant information by weighting different parts of the input data.
- **Dropout:** Dropout regularization[9] is employed with a rate of 0.5 to prevent overfitting by randomly setting a fraction of input units to 0 during training.

- **Linear Layers:** Two linear layers with activation functions process the combined information from the attention output. The output dimensions are 20 and 1, respectively. The first layer uses a Leaky ReLU activation function and the second layer employs an identity function, outputting the estimated weight.

- **Layer Normalization:** Layer normalization[10] is applied after the first linear layer to normalize activations across the feature dimension, which aids in model generalization. This is used in conjunction with dropout and batch normalization to improve the model's generalization.

By employing bi-LSTM, Attention Mechanism, and subsequent linear layers, the Predictor Module estimates the weight of a patient by capturing the relevant temporal patterns and dependencies in the input data.

### 3.3.2 Classifier Module

The Classifier Module is responsible for making the final prediction of weight abnormality based on the inputs and features extracted from the dataset. This module utilizes a two-layer bidirectional LSTM with an attention architecture to capture sequential patterns and incorporate contextual information for accurate classification.

The architecture of the Classifier Module includes the following components:

- **Bidirectional LSTM Layer:** The module begins with two layers of bidirectional LSTM that process the sequential input data. Each layer has a hidden size of 256, resulting in an output dimension of 512 due to bidirectionality. These layers capture temporal dependencies and encode the sequential information into a meaningful representation.

- **Batch Normalization:** Batch normalization[8] is applied after the LSTM layers to improve the stability and generalization of the model. It normalizes the activations

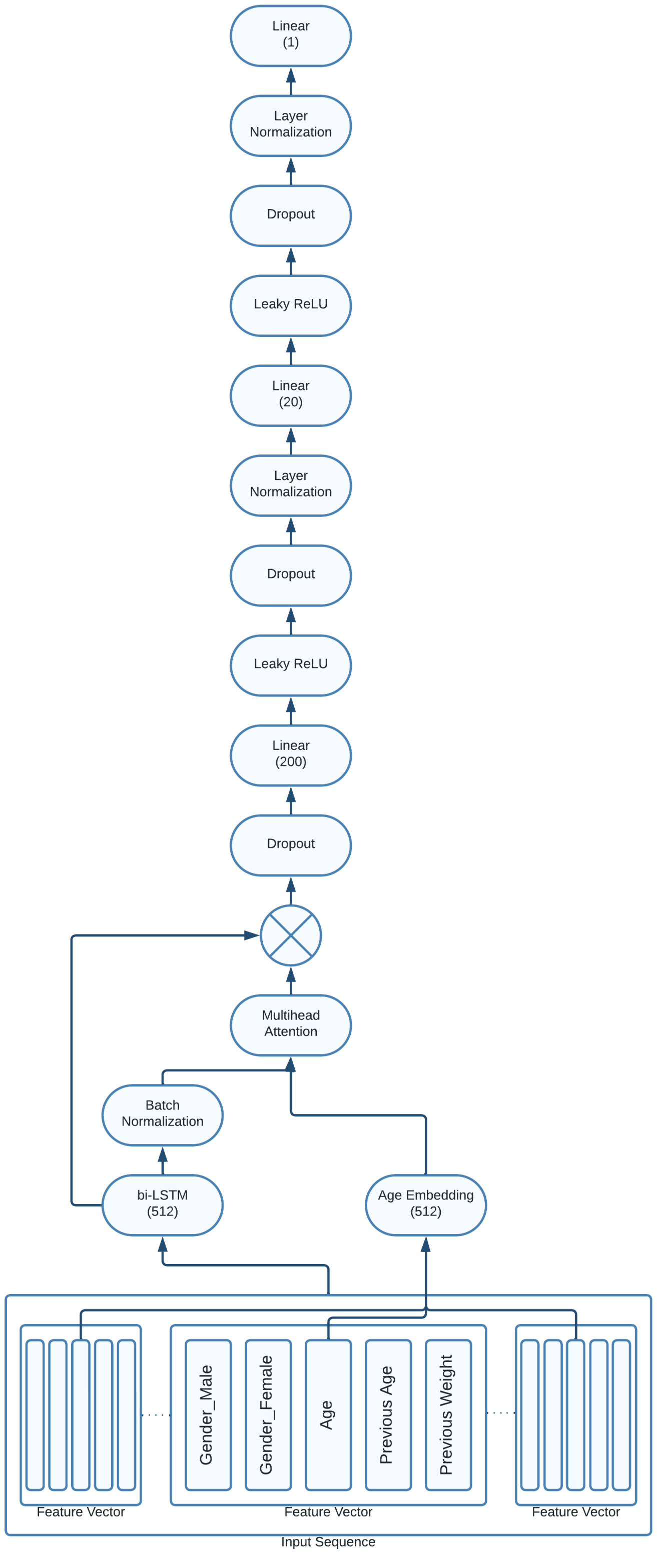


Figure 3.2: Architecture of the Predictor Module

of the hidden units within each mini-batch, reducing the internal covariate shift and speeding up the training process. This is used in conjunction with dropout and layer normalization to improve the model's generalization.

- **Age and Weight Embedding:** The age and both the entered and predicted weight from the Predictor Module are embedded into higher-dimensional representations, which can then be passed to the attention mechanism.

- **Multihead Attention:** The Classifier Module also utilizes a four-headed attention mechanism[3]. The embedded age, LSTM output, and embedded weight are passed as query, key, and value, respectively, to the attention mechanism. This allows the module to focus on relevant information and assign different weights to different parts of the input data based on their importance.

- **Dropout:** Dropout regularization[9] is applied to mitigate overfitting. A dropout rate of 0.9 is used in the attention mechanism, 0.7 after the first linear layer, and 0.5 after the second linear layer.

- **Linear Layers:** Three linear layers with activation functions are employed after the attention mechanism. These layers have shrinking output dimensions from 200, 20, to 2. Leaky ReLU is used for the activation function of the first two linear layers. Layer normalization is used after each of the first two linear layers. The final layer uses a sigmoid activation function to provide probabilistic outputs for the two classes: normal and abnormal.

- **Layer Normalization:** Layer normalization[10] normalizes the activations of the hidden units across the feature dimension and is applied after each linear layer except the output layer.

By utilizing two layers of bidirectional LSTM, attention mechanism, and subsequent linear layers, the Classifier Module effectively captures sequential patterns, attends to relevant information, and produces probabilistic predictions of weight abnormality. The combination of these components enables the module to make reliable predictions based on the given input data.

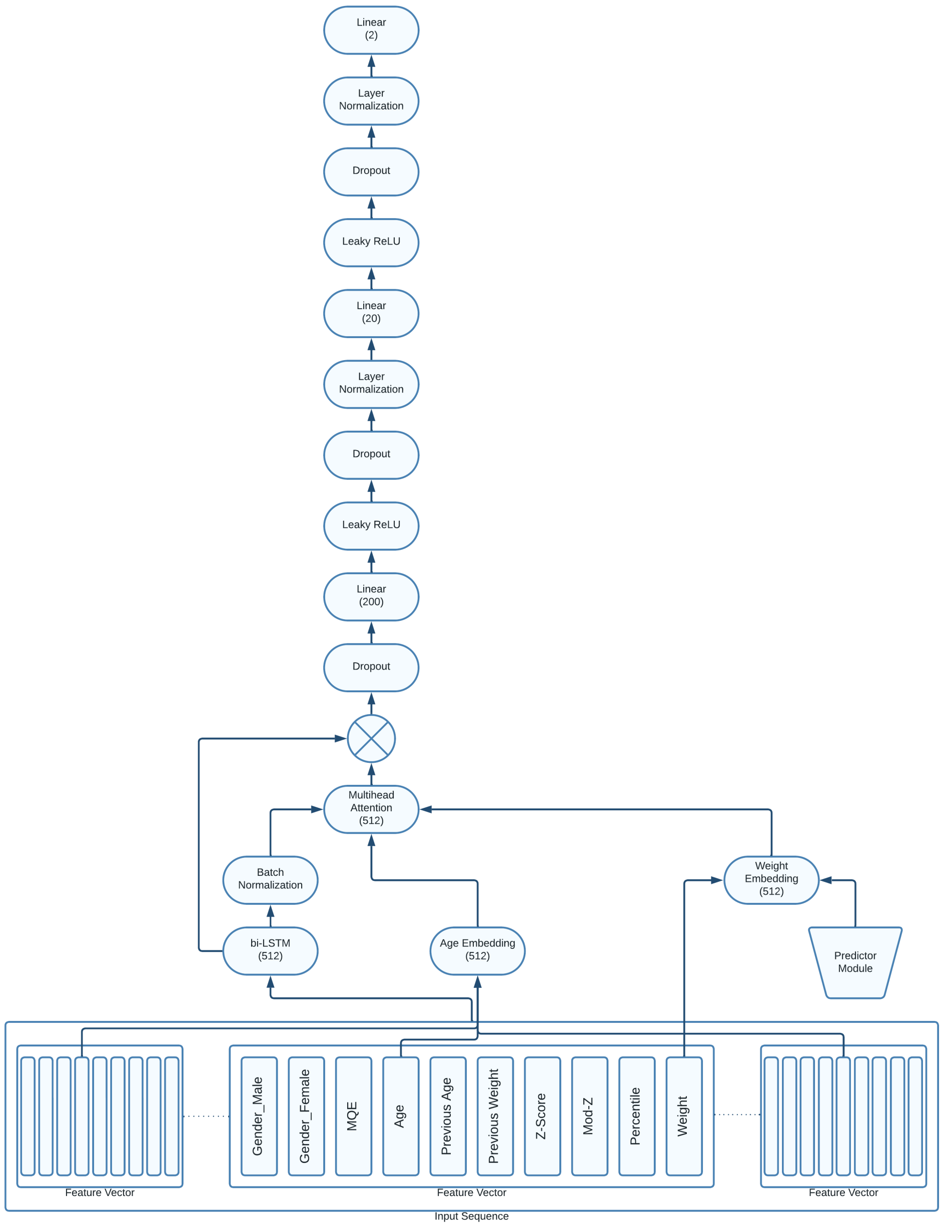


Figure 3.3: Architecture of the Classifier Module

# Chapter 4

# Experiments

## 4.1 Dataset

### 4.1.1 Main Dataset

#### Data Collection

The Main dataset was obtained from the Electronic Health Records (EHRs) of Cincinnati Children's Hospital Medical Center (CCHMC) for the period between 2010 and 2018 as part of the previous study by Liu et al.[2]. In that study, 4.3 million weight points were collected for 347,056 patients.

In the previous study[2], certain criteria were applied to exclude data points:

1. Weight points documented within the 2 years of age.
2. Weight points recorded after 20 years of age.
3. All weight points from patients with less than four weight measurements.

A secondary polynomial regression model was applied to the remaining data to identify charts with potentially abnormal weight values. This regression model identified 107,336 candidate charts, from which 15,000 charts were randomly sampled to create what was referred to as the MAIN dataset[2].

#### Annotation Process

In the previous study[2], 18 domain experts were recruited to annotate the 15,000 weight charts for abnormal weight values. The annotation guidelines stipulated that weight points should be evaluated retrospectively, and labeled as abnormal based on clinical importance and potential risk to the patient.

A scoring system was employed where abnormal weight points with high clinical importance scored two, potentially abnormal weight points scored one, and normal weight points scored zero. A weight point was classified as abnormal if the sum of scores from three annotators exceeded two[2]. The annotation process was performed as part of the previous work, and the annotations were utilized as-is in this study.

### 4.1.2 Auto Dataset

In addition to the main dataset described in 4.1.1, a user-corrected weight dataset from the year 2019 was collected from the CCHMC EHRs (206,330 charts). Each record in this dataset came with a binary flag indicating if the weight of this record was edited by the user who entered it. Other than the addition of the flag and the absence of the expert annotation, the user-corrected dataset contained the same variable as the expert-annotated dataset. Entries that were changed to a weight that's different from the original value entered are considered erroneous. Other entries were considered normal.

### 4.1.3 Dataset Statistics

The dataset resulting from the real-time simulation process described in Section 3.2 and the original dataset have different distributions of the number of charts containing errors.

Table 4.1 presents the breakdown of the number of charts containing varying numbers of errors, as well as the total number of charts for the datasets with and without the application of real-time simulation.

| Dataset | Real-time Simulation | #charts with no error | #charts with 1 error | #charts with 2 errors | #charts with 3 errors | #charts with 4 errors | Total #charts |
|---|---|---|---|---|---|---|---|
| Main | No | 13,432 | 1,294 | 102 | 13 | 3 | 14,844 |
| Main | Yes | 197,169 | 978 | 0 | 0 | 0 | 198,147 |
| Auto | No | 204,580 | 1,677 | 63 | 10 | 0 | 206,330 |
| Auto | Yes | 172,912 | 748 | 0 | 0 | 0 | 173,660 |

Table 4.1: Statistical Summary of Datasets

### 4.1.4 Feature Extraction

Feature extraction is an essential step in the data processing pipeline, which involves converting raw data into a set of features or attributes that can be effectively utilized by machine learning algorithms.

In the previous study[2], nine variables were extracted for each weight data point to capture weight characteristics and growth dynamics:

1. Subject weight in kilograms
2. Subject age in years
3. Subject sex
4. The Box-Cox transformation, median, and generalized coefficient of variation-based z-score according to subject sex and age, denoted as Z-Score
5. Modified LMS-based z-score using the weight-for-age data from a reference population provided by Centers for Disease Control (CDC) to identify extreme weight values, denoted as Modified Z-Score

6. Percentage of the population below a weight value, or percentile
7. Absolute age difference from the immediate previous weight point
8. Absolute weight difference from the immediate previous weight point
9. Absolute z-score difference from the immediate previous weight point

In this study, features 1-6 from the previous study were used. Features 7-9 were excluded as LSTM networks employed in this study can inherently capture temporal information, such as differences over time. However, the weight value and age at the previous measurement were extracted and utilized.

## 4.2 Preprocessing

Data preprocessing is a crucial step in machine learning pipelines. It involves transforming raw data into a format that can be effectively processed by machine learning algorithms. In this study, the preprocessing steps included one-hot encoding for categorical variables and standardization for numerical features.

#### One-hot encoding

Categorical features such as gender are non-numeric, and most machine learning algorithms require numerical input. Additionally, using a single numerical value to represent categories can imply ordinal relationships that may not exist among categories. To avoid these issues, one-hot encoding was used to convert the categorical variables into a format that can be better understood by the algorithms.

In one-hot encoding, each category of a categorical feature is converted into a new binary feature (0 or 1). For example, the gender feature, which has categories 'male' and 'female', is represented using two binary features: 'gender_male' and 'gender_female'.

### Standard Scaling

Standardization of numerical features is important for ensuring that they are on the same scale. This is particularly critical for algorithms that are sensitive to the scale of the input features, such as distance-based methods and methods that use gradient descent for optimization. The standard scaling process is a common practice for this task, where numerical features are standardized to have a mean of zero and a standard deviation of one. The standard scaling process is mathematically represented as:

$$z = \frac{x - \mu}{\sigma} \tag{4.1}$$

where $z$ is the standardized value, $x$ is the original value, $\mu$ is the mean of the feature, and $\sigma$ is the standard deviation of the feature.

This process ensures that the numerical features have comparable scales, which can improve the convergence speed of the learning algorithm and potentially lead to better model performance. However, this approach has an inherent downside of being sensitive to outliers, as outliers can dramatically change both the mean and the standard deviation.

### Robust Scaling

In this study, robust scaling was used instead of standard scaling described in 4.2 to minimize the effects of outliers. Robust scaling uses the Interquartile Range (IQR), which is more robust to outliers than the mean and standard deviation used in standard scaling.

$$X_{robust} = \frac{X - Q1(X)}{Q3(X) - Q1(X)} \tag{4.2}$$

Where $X_{robust}$ is the robustly scaled feature, $X$ is the original feature, $Q1(X)$ is the first quartile of the feature, and $Q3(X)$ is the third quartile of the feature.

## 4.3 Training

### 4.3.1 SOM

The Self-Organizing Map (SOM) is utilized in this study as a feature extraction method. It was trained using normal data points and implemented using the MiniSom Python library[11].

- **Initialization:** Principal Component Analysis (PCA)[12] initialization was used to set the initial weights of the neurons in the SOM. This method projects the input data into the principal components space to set the initial positions of the neurons, leading to faster convergence compared to random initialization.

- **Competitive Learning:** The training process involves competitive learning. For each input vector, the neuron with weights most similar to the input is identified as the Best Matching Unit (BMU). The weights of the BMU and its neighbors are adjusted towards the input vector.

- **Learning Rate:** An initial learning rate of 0.5 was used, which decays over time according to the formula:

$$\text{learning\_rate}(t) = \frac{\text{learning\_rate}}{1 + \frac{t}{T}} \tag{4.3}$$

  where $t$ is the iteration number, and $T$ is half the number of iterations.

- **Neighborhood Size:** An initial sigma value of 1.0 was used to control the size of the neighborhood around the BMU, which also decays over time, similarly to the learning rate. The neighborhood size is given by:

$$\text{neighborhood\_size}(t) = \frac{\text{neighborhood\_size}}{1 + \frac{t}{T}} \tag{4.4}$$

where $t$ is the iteration number, and $T$ is half the number of iterations.

- **Neighborhood Function:** The Gaussian neighborhood function is used. The Gaussian neighborhood function is given by:

$$h_{ij}(t) = \exp\left(-\frac{d_{ij}^2}{2\sigma^2(t)}\right) \tag{4.5}$$

  where $h_{ij}(t)$ is the neighborhood function, $d_{ij}$ is the distance between the BMU and neuron $j$, and $\sigma(t)$ is the width of the neighborhood at iteration $t$. In the case where Euclidean distance is used, $d_{ij}$ is equivalent to $||r_j - r_i||$, where $r_j$ and $r_i$ are the positions of neurons $j$ and $i$ respectively[13].

- **Topology:** A rectangular topology was used for arranging the neurons in the SOM. In a rectangular topology, the neurons are arranged in a grid, where each neuron is connected to its adjacent neighbors in the grid.

- **Distance Function:** The Euclidean distance was used to calculate the activation distance between input vectors and neuron weights.

- **Convergence:** The SOM was trained for 100,000 iterations to ensure convergence.

The training process of SOM involves three main stages: competition, cooperation, and adaptation.

- **Competition**: Each input vector is presented to the SOM, and the neurons in the network compete to be the best matching unit (BMU). The BMU is the neuron whose weight vector is closest to the input vector in terms of Euclidean distance.

- **Cooperation**: After the BMU is determined, a neighborhood around the BMU is established. The Gaussian neighborhood function is utilized to determine the extent to which neighboring neurons participate in the learning process based on their distance from the BMU.

- **Adaptation**: The weights of the BMU and its neighboring neurons are updated to be more similar to the input vector, effectively adapting the network to the data.

### 4.3.2 Predictor Module

The predictor module was trained using normal data points, with each step of the sequence containing the gender and age at the measurement of the patient, along with the age and weight of the previous measurement.

- **Optimizer:** The AdamW optimizer[14] was used, with a learning rate of $5 \times 10^{-5}$ and weight decay of $1 \times 10^{-1}$.
- **Warmup:** An exponential warmup described by Ma et al[15] with a warmup period of 15 epochs was implemented using the pytorch_warmup Python library[16]. The warmup period allows the learning rate to gradually reach its intended value, reducing the chance of divergence at the start of training.
- **Learning Rate Scheduler:** After the warmup period, an exponential learning rate scheduler was implemented using the ExponentialLR module from PyTorch's[17] optim library with a gamma of 0.85. This scheduler gradually reduces the learning rate over time, providing a form of learning rate annealing that helps improve the stability of the training process and the performance of the model[18]. Through trial and error, it was found that the combination of warmup and exponential learning rate decrease improved the model's performance and stability during training.
- **Batch Size and Epochs:** The model was trained for 30 epochs with a batch size of 1000. The use of a relatively large batch size enables the efficient use of GPU resources, while the number of epochs was chosen to balance the training time and performance.

- **Loss Function:** The Mean Squared Error (MSE)[19] loss function was used for the predictor module to measure the difference between the predicted and actual weight values.
- **Model Selection:** At the end of each epoch, a checkpoint of the model was saved. The final model was selected from these checkpoints based on the best test set Mean Squared Error (MSE).

### 4.3.3 Classifier Module

The classifier module was trained using normal and anomalous data points, with each step of the sequence containing the gender and age at the measurement of the patient, along with the age and weight of the previous measurement.

- **Data Sampling:** Due to the highly imbalanced nature of the dataset, weighted random sampling was implemented during training. Each class was weighted by the inverse of its prevalence, ensuring that the model was trained with approximately equal numbers of normal and abnormal samples. This strategy ensured that the classifier was not overwhelmed by the majority class and had a balanced view of the classes during training.
- **Optimizer:** Similar to the predictor module, the AdamW optimizer was used. However, the learning rate was set to $1 \times 10^{-3}$ while the weight decay remained at $1 \times 10^{-1}$[14].
- **Warmup:** As with the predictor module, an exponential warm up was implemented, but with a warm up period of 20 epoches.
- **Learning Rate Scheduler:** After the warmup period, an exponential learning rate scheduler was implemented with a gamma of 0.75. This scheduler gradually reduces

the learning rate over time, similar to the predictor module, though with a different gamma value.

- **Batch Size and Epochs:** The model was trained for 30 epochs with a batch size of 1000, same as the predictor module.

- **Loss Function:** The Cross-Entropy[20] loss function was used for the classifier module to differentiate between normal and anomalous data points.

- **Model Selection:** At the end of each epoch, a checkpoint of the model was saved. The final model was selected from these checkpoints based on the best test set Area Under the Receiver Operating Characteristic Curve (AUROC)[21].

### 4.3.4 Evaluation

For the evaluation of the proposed method, several performance metrics were employed. The primary metric was the Area Under the Receiver Operating Characteristic Curve (AUROC)[21], which is especially useful for binary classification problems. It provides an aggregate measure of performance across all possible classification thresholds regardless of sensitivity and specificity.

In addition to AUROC, precision and recall were also reported. Precision, or the positive predictive value, quantifies the number of correct positive predictions made, while recall, also known as sensitivity, measures how many actual positives the model correctly captured. In clinical settings, sensitivity is usually considered to be much more important than precision, as an illness not diagnosed creates way more risk for the patient compared to a healthy patient incorrectly diagnosed to have it. Therefore, Liu et al.[2] manually set sensitivity to 90% or higher, which is considered the requirement for practical use [22].

Additionally, to evaluate the effectiveness of SOM-MQE, AUROC using only the MQE value was reported and compared to other features. For the evaluation of the predictor module, the Mean Square Error is reported.

For cross-validation, a stratified train-test split was performed, with 70% of the data used for training the model and the remaining 30% used for testing. This approach ensures that the proportion of normal and anomalous data points is consistent in both the training and test sets, facilitating a more fair and reliable evaluation.

## 4.4 Hardware Configuration

The proposed approach was implemented on a computer system with the following hardware configuration:

- CPU: Intel i7-13700k
- GPU: NVIDIA RTX 2080 Ti (11GB GDDR6)
- RAM: 32GB DDR5
- OS: Windows 11

# Chapter 5

# Result

## 5.1 Main Data

### 5.1.1 SOM-MQE

The AUROC of the MQE values was noticeably higher than that of Z-Score, Modified Z-Score, and Percentile. The value is shown in Table 5.1.

| | Z-Score | Modified Z-Score | Percentile | SOM-MQE |
|---|---|---|---|---|
| AUROC | 0.5139 | 0.5140 | 0.5139 | **0.5446** |

Table 5.1: AUROC of extracted features.

### 5.1.2 Predictor Module

For the predictor module, the mean squared error (MSE) across all data points in the test set was $9.97 \times 10^{-5}$. Notably, there was a significant difference between the MSE for normal data points ($1.04 \times 10^{-4}$) and the MSE for anomalous data points ($1.31 \times 10^{-3}$). The difference between the predicted weight and the weight record was also compared. For this regression comparison, the output of the Predictor Module went through the inverse of the scaling described in 4.2, effectively restoring the values back to kilograms. A histogram of these difference is shown in 5.1. A significant tendency of abnormal weight records to produce bigger differences can be observed. This indicates that normal and abnormal data points have different patterns. This information can help the classifier module to better differentiate the two, as the difference between the predicted weight and the actual weight entered will be much bigger for abnormal data points.

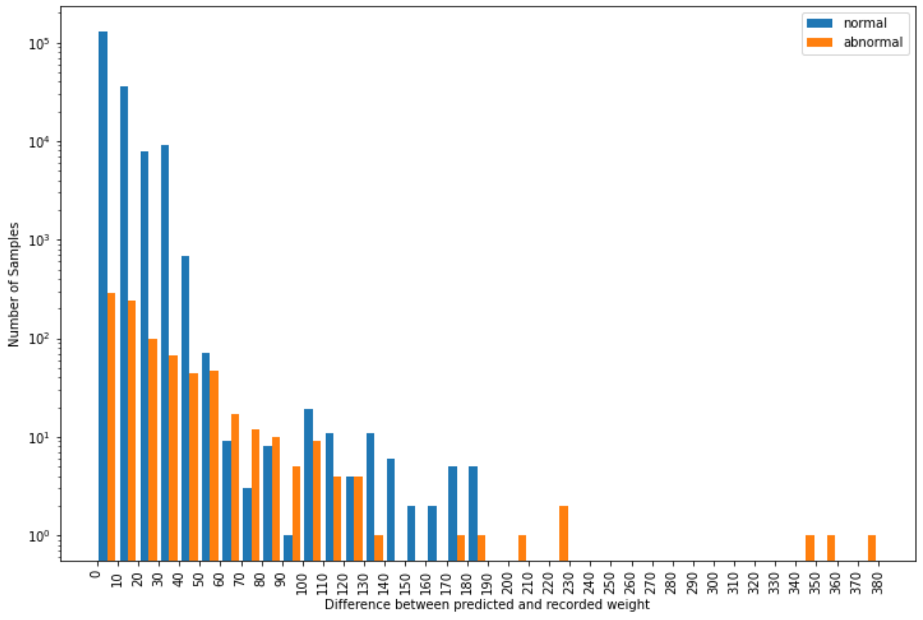


Figure 5.1: Differences between predicted weight and weight recorded.

### 5.1.3 Classifier Module

Figure 5.2 shows the Cross Entopy Loss of the classifier module during training. The loss decreases over epoch, showing good convergence. The Area Under the Receiver Operating Characteristic Curve (AUROC) during training is shown in Figure 5.3, with the epoch where max AUROC was achieved marked. The Receiver Operating Characteristic Curve and Precision Recall Curve are shown in Figure 5.4 and Figure 5.5, respectively.

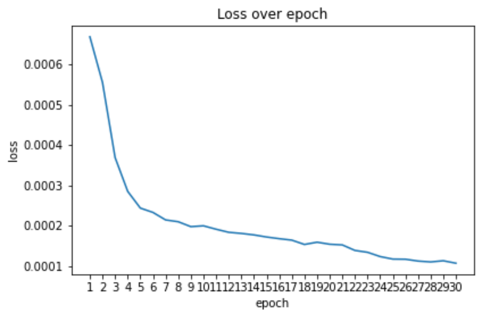


Figure 5.2: Loss over Epoch of the Classifier Module

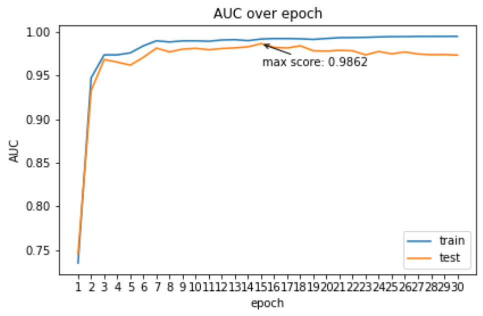


Figure 5.3: AUROC over Epoch of the Classifier Module

The classifier module, with the help of the predictor module, achieved an AUROC of 0.986, indicating excellent performance. The precision, or positive predictive value, was 9.28%. The confusion matrix is shown in Figure 5.6. Due to the big number of samples, a percentage-based confusion matrix is also shown in 5.7.

For comparison, the performance of several rule-based baseline approaches and Machine Learning-based approaches reported by Liu et al.[2] are included. As mentioned in 4.3.4,

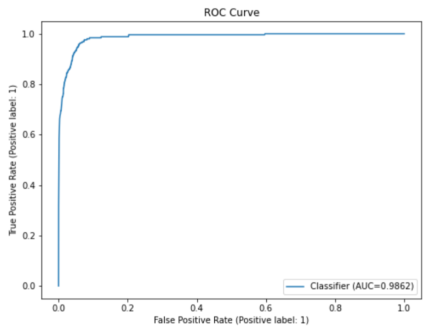


Figure 5.4: Receiver Operating Characteristic Curve

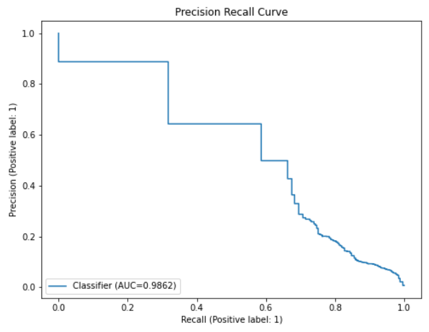
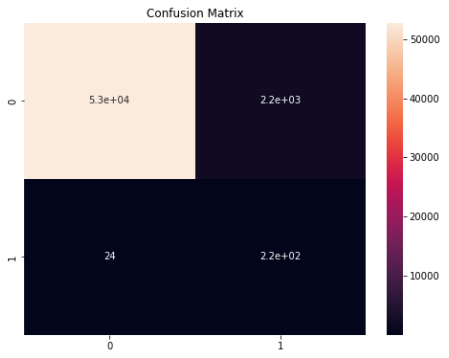


Figure 5.5: Precision Recall Curve



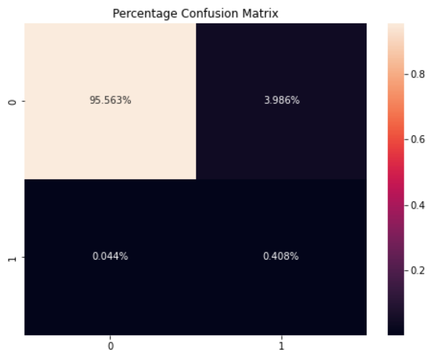


Figure 5.6: Confusion Matrix

Figure 5.7: Percentage Confusion Matrix

all methods presented were calibrated to achieve a sensitivity of 90% or higher. The results are summarized in Table 5.2.

| Type of Approach | Method | AUROC | Precision (%) | Sensitivity (%) |
|---|---|---|---|---|
| Baseline | CDC[2] | 0.546 | 0.52 | 90.11 |
| | RED[2] | 0.578 | 0.52 | 90.08 |
| | CHOP[2] | 0.620 | 0.64 | 90.11 |
| Machine Learning | LSTM[2] | 0.938 | 3.47 | 90.11 |
| | Random Forest[2] | 0.961 | 4.66 | 90.11 |
| | Proposed Approach | **0.986** | **9.28** | 90.36 |

Table 5.2: Performance of the proposed approach compared to other methods.

The proposed approach demonstrates high performance across several metrics, especially in terms of precision. Compared to other methods, the proposed approach shows a clear advantage in distinguishing between normal and anomalous data points.

## 5.2 Auto Data

### 5.2.1 SOM-MQE

Similarly to the result of the Main Data shown in Section 5.1.1, the AUROC of the MQE value was significantly higher than that of Z-Score, Modified Z-Score, and Percentile. However, the overall AUROCs are lower, indicating that Auto Data may be a more challenging dataset. The value is shown in Table 5.3.

| | Z-Score | Modified Z-Score | Percentile | SOM-MQE |
|---|---|---|---|---|
| AUROC | 0.4632 | 0.4590 | 0.4633 | **0.5175** |

Table 5.3: AUROC of extracted features in Auto Data.

### 5.2.2 Predictor Module

The mean squared error (MSE) across all data points in the test set was $9.03 \times 10^{-5}$. Although a meaningful difference between the MSE for normal data points ($9.05 \times 10^{-5}$) and the MSE for anomalous data points ($4.36 \times 10^{-4}$) was observed, it was noticeably smaller than what was obtained from Main Data. A histogram of the difference between the predicted weight and the weight record is shown in 5.8. Compared to the result observed in Main Data, there was considerably less tendency for the abnormal data points to have a bigger discrepancy.

### 5.2.3 Classifier Module

Figure 5.9 shows the Cross Entopy Loss of the classifier module during training. The loss decreases over epoch, indicating convergence. The AUROC during training is shown in Figure 5.10, with the epoch where max AUROC was achieved marked. The Receiver Operating Characteristic Curve and Precision Recall Curve are shown in Figure 5.4 and Figure 5.5, respectively.

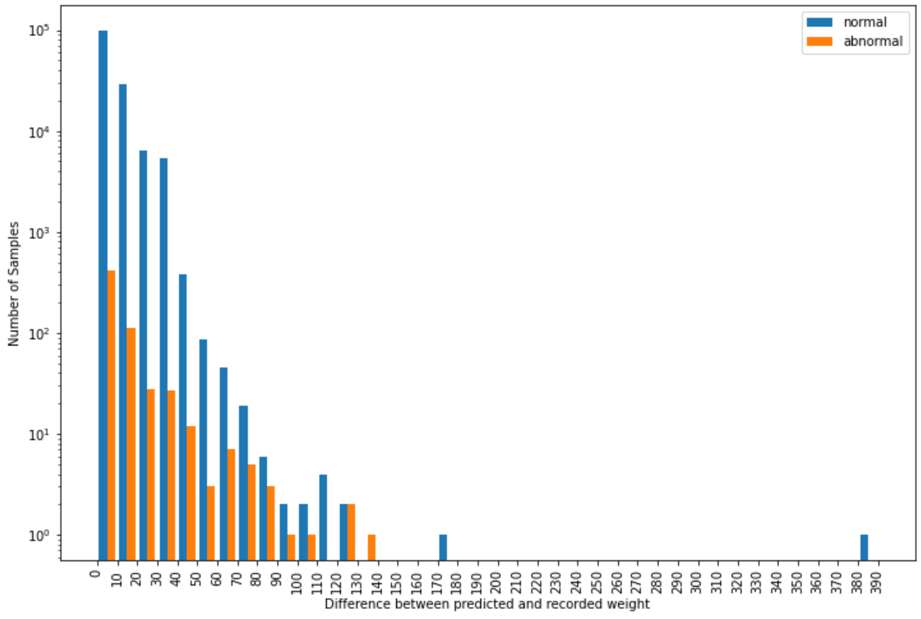


Figure 5.8: Differences between predicted weight and weight recorded in Auto Data.

The proposed approach achieved an AUROC of 0.675 in Auto Data, inferior to the performance in Main Data. The precision of 0.498% is also less ideal. The confusion matrix and percentage-based confusion matrix are shown in Figure 5.13 and Figure 5.14, respectively.

Several Machine Learning algorithms were trained on the same data set for comparison. All methods were calibrated to achieve a sensitivity of 90% or higher as described in 4.3.4. The results are summarized in Table 5.4.

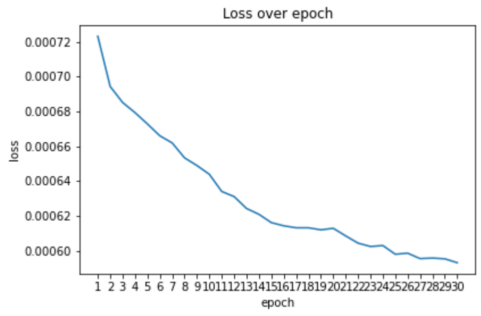


Figure 5.9: Loss over Epoch of the Classifier Module in Auto Data

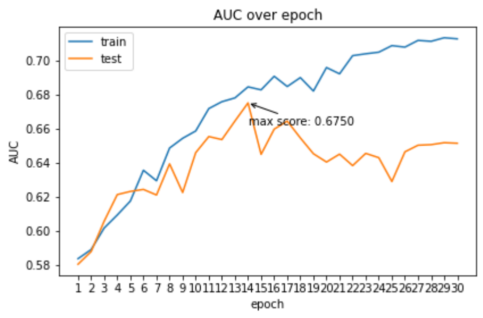


Figure 5.10: AUROC over Epoch of the Classifier Module in Auto Data

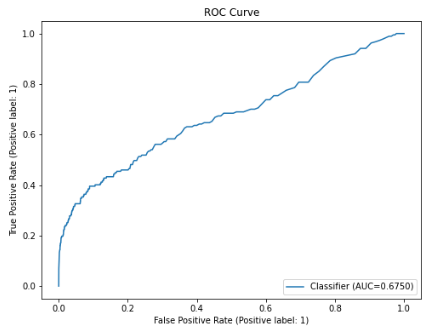


Figure 5.11: Receiver Operating Characteristic Curve in Auto Data

Although the performance of the proposed approach in Auto Data is less impressive compared to the performance in Main Data, it is still clearly advantageous to Machine Learning algorithms, showing the value of the bi-LSTM with Attention approach.

| Method | AUROC | Precision (%) | Sensitivity (%) |
|---|---|---|---|
| Decision Tree | 0.540 | 0.427 | 97.30 |
| Logistic Regression | 0.558 | 0.422 | 90.27 |
| Gradient Boosting | 0.585 | 0.427 | 90.27 |
| Random Forest | 0.617 | 0.441 | 90.27 |
| **Proposed Approach** | **0.675** | **0.498** | 90.37 |

Table 5.4: Performance of the proposed approach compared to other methods.

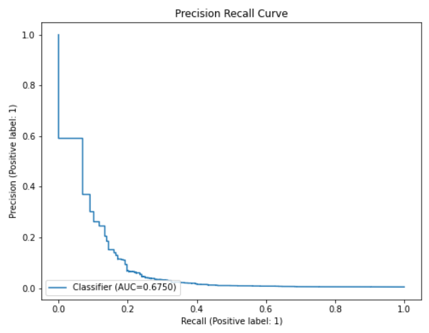
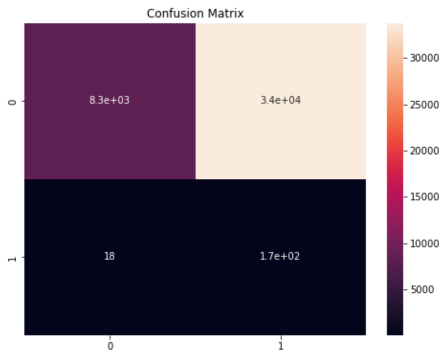


Figure 5.12: Precision Recall Curve in Auto Data



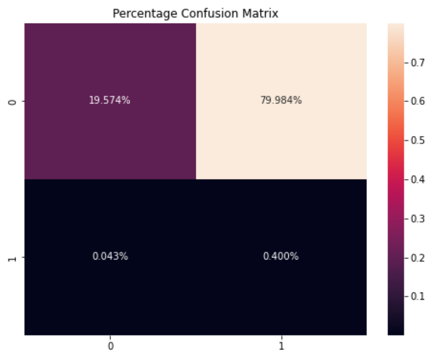


Figure 5.13: Confusion Matrix in Auto Data

Figure 5.14: Percentage Confusion Matrix in Auto Data

# Chapter 6

# Discussion

The SOM-MQE proved to be a very effective feature extraction method in the task of prospective weight entry error detection. As shown in Table 5.1 and Table 5.3, when used individually, the MQE values yielded meaningfully higher AUROC than all of the other extracted features. In fact, in the case of the Main Data, the MQE values alone can already achieve AUROC very close to the CDC method reported by Liu et al.[2]. This result provides evidence that SOM-MQE can provide valuable information for Anomaly Detection in the medical domain.

The predictor module showed to be competent in predicting the weight of patients. More importantly, it produced a clear level of difference between normal and abnormal data points in how much the predicted weight differs from the actual weight entered, although this difference is mainly observed in the Main Data. It is likely that some data points in the Auto Data with big differences in predicted and entered weight are, in fact, erroneous but were not caught by the people who entered them. Therefore, they were not edited and subsequently labeled as normal data.

The hyperparameters were determined by trial and error using the Main Data, which were then applied to the Auto Data with no additional hyperparameter tuning. This has likely contributed to the relatively low performance of Auto Data.

## Limitations

This study is limited to the quality of the annotations. In the case of the Main Data, although data points were manually annotated by domain experts, the annotation process is still susceptible to human errors and the subjectivity of the annotators. The Fleiss' Kappa of 0.644 that Liu et al.[2] reported for inter-rater agreement of the Main Data indicates that annotators do not completely agree with each other. Furthermore, the proposed real-time simulation procedure imposes a requirement for at least four correct preceding weight entries. This requirement can also be compromised by incorrectly labeled data points, especially in the case of Auto Data. Additionally, the two datasets may not be enough to evaluate the effectiveness of the proposed approach and evaluation on more datasets is desired. The result of this study may potentially be overly optimistic due to the lack of a validation set for hyperparameter tuning and model checkpointing.

# Chapter 7

# Conclusion

This study proposed a two-module method utilizing bi-LSTM and Attention Mechanism for the prospective detection of anomalous weight entries in electronic health records. The proposed approach, consisting of a predictor and a classifier module, showed a clear advantage in distinguishing between normal and anomalous data points compared to previous methods.

The predictor module's performance, indicated by the low value of MSE, shows the feasibility of the proposed method. Moreover, the difference between the predicted weight and the actual weight entered was much bigger for abnormal data points, further indicating that the predictor module can provide valuable information for distinguishing erroneous weight entries.

The classifier module's performance was evaluated using several metrics, with a notable AUROC of 0.986 and a precision of 9.28% in Main Data, both superior to the previous methods. The proposed method hence provides a promising solution to the critical problem of detecting anomalous weight records in real-time in a healthcare setting.

The inferior performance in Auto Data compared to Main Data suggests that properly annotated datasets may still be critical in anomaly detection in irregular time series. Using the expert-annotated Main Data, the proposed approach showed significantly better performance. This performance difference is also observed in other Machine Learning-based methods.

This study contributes to the research on utilizing machine learning for error detection in healthcare data. For future work, this method can be applied to and evaluated on other types of healthcare data, and additional ways to improve its performance can be explored. Designs other than bi-LSTM can be experimented with. Including other features, such as height, can enable better performance. Incorporating a validation set will also allow a more accurate evaluation of the proposed approach. Furthermore, integrating this method with electronic health record systems can further assist healthcare providers in ensuring data quality and patient safety.

# References

[1] Carrie Daymont, Michelle E Ross, A Russell Localio, Alexander G Fiks, Richard C Wasserman, and Robert W Grundmeier. Automated identification of implausible values in growth data from pediatric electronic health records. *Journal of the American Medical Informatics Association*, 24(6):1080–1087, 2017.

[2] Lei Liu, Danny Wu, S. Andrew Spooner, and Yizhao Ni. Development and evaluation of an automated approach to detect weight abnormalities in pediatric weight charts. In *AMIA Annual Symposium Proceedings*, 2019.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[4] Fanhui Kong, Jianqiang Li, Bin Jiang, Huihui Wang, and Houbing Song. Integrated generative model for industrial anomaly detection via bidirectional lstm and attention mechanism. *IEEE Transactions on Industrial Informatics*, 19(1):541–550, 2023.

[5] Runqing Huang, Lifeng Xi, Xinglin Li, C. Richard Liu, Hai Qiu, and Jay Lee. Residual life predictions for ball bearings based on self-organizing map and back propagation neural network methods. *Mechanical Systems and Signal Processing*, 21(1):193–207, 2007.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 12 1997.

[7] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.

[8] Sergey Ioffe and Szegedy Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *ICML'15*, pages 448–456. JMLR.org, 2015.

[9] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[10] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[11] Giuseppe Vettigli. Minisom: minimalistic and numpy-based implementation of the self organizing map, 2018.

[12] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[13] Simon S. Haykin. *Neural networks and learning machines*. Prentice Hall, 2009.

[14] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[15] Jerry Ma and Denis Yarats. On the adequacy of untuned warmup for adaptive optimization, 2021.

[16] Tony Y. Pytorch_warmup: Learning rate warmup in pytorch, Oct 2019.

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[18] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472, 2017.

[19] Rob J Hyndman and Anne B Koehler. Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4):679–688, 2006.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[21] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[22] Douglas G Altman and J Martin Bland. Statistics notes: Diagnostic tests 1: Sensitivity and specificity. *BMJ*, 308(6943):1552–1552, 1994.

# APPENDICES

# Appendix A

# Example of Growth Chart with Model Prediction

## A.1 False Positive

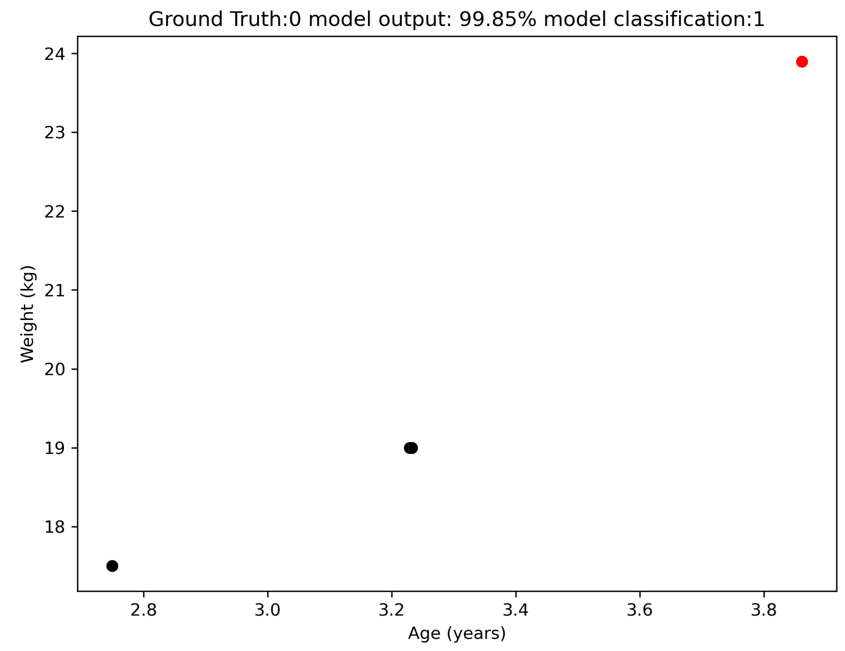


Figure A.1: False Positive Example 1

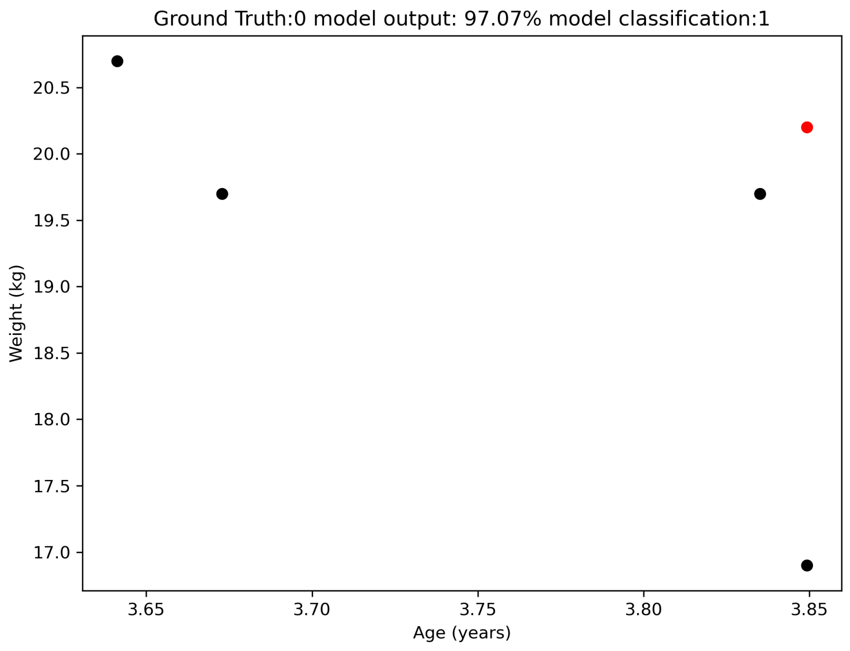


Figure A.2: False Positive Example 2

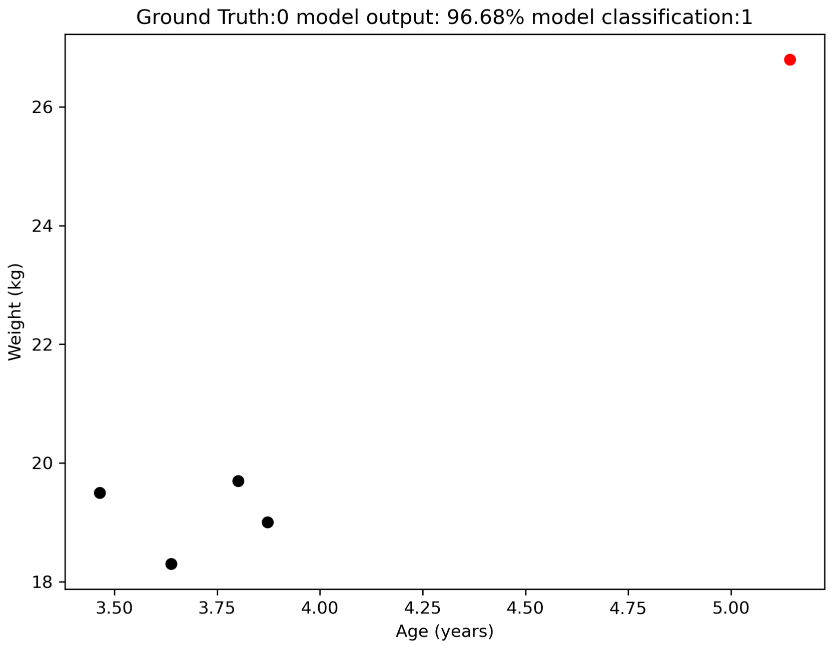


Figure A.3: False Positive Example 3

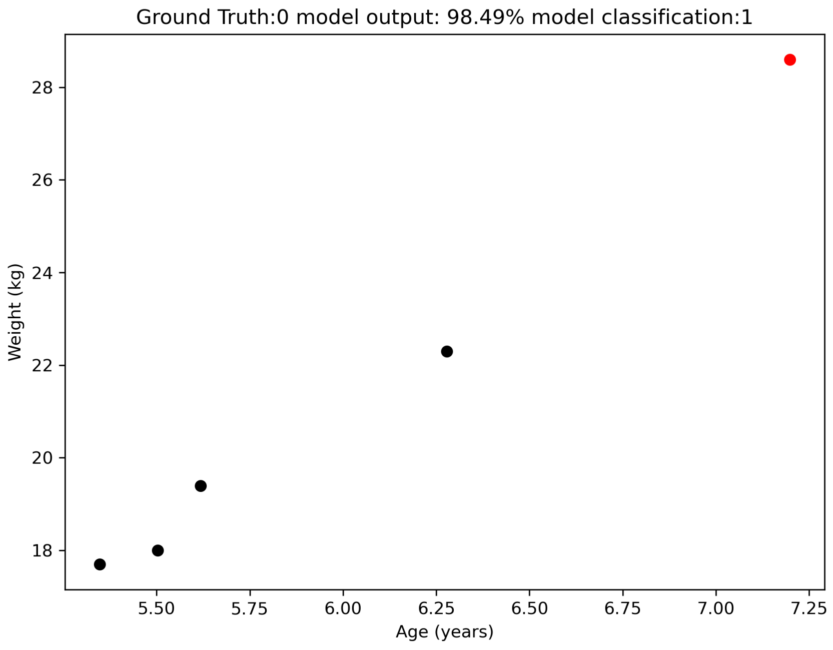


Figure A.4: False Positive Example 4

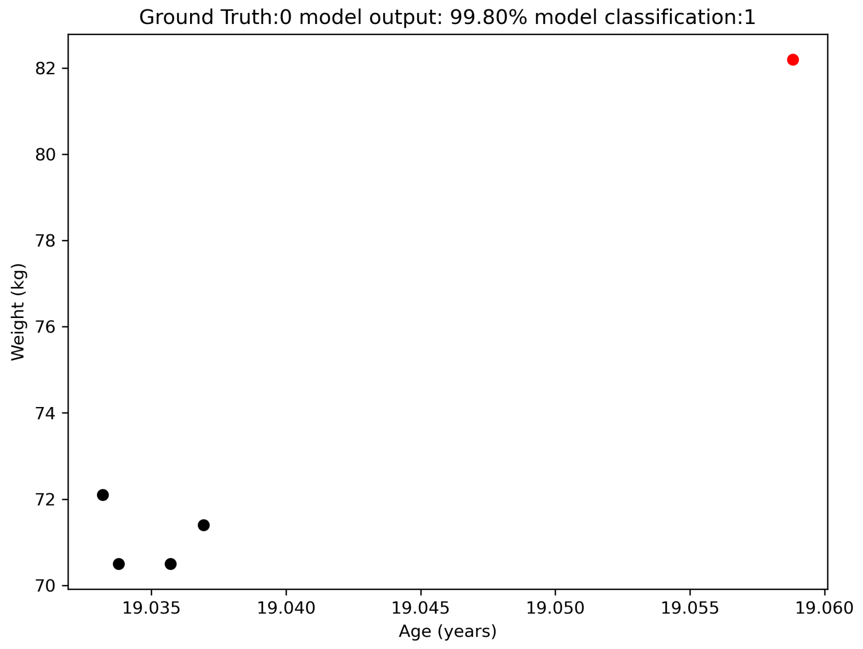


Figure A.5: False Positive Example 5

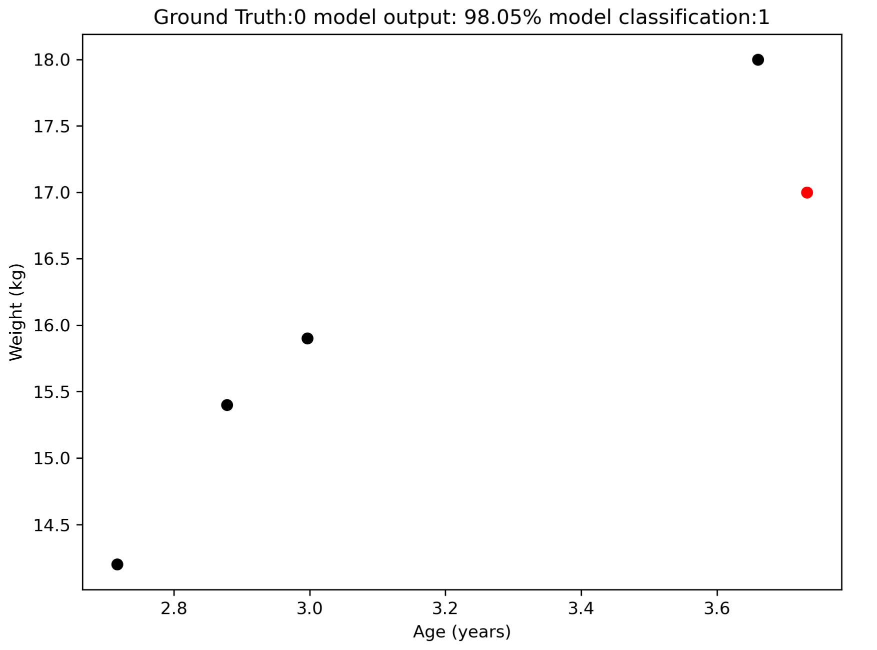


Figure A.6: False Positive Example 6

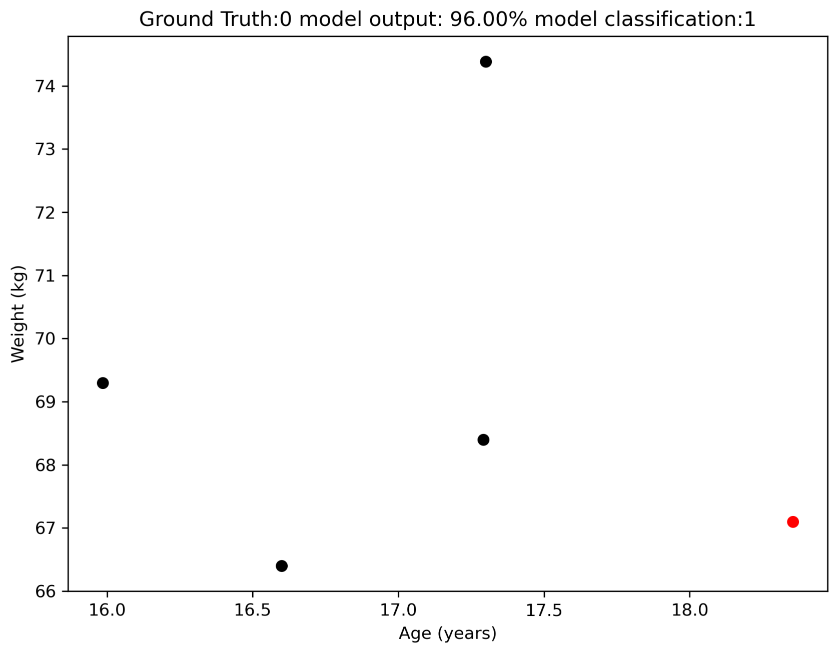


Figure A.7: False Positive Example 7

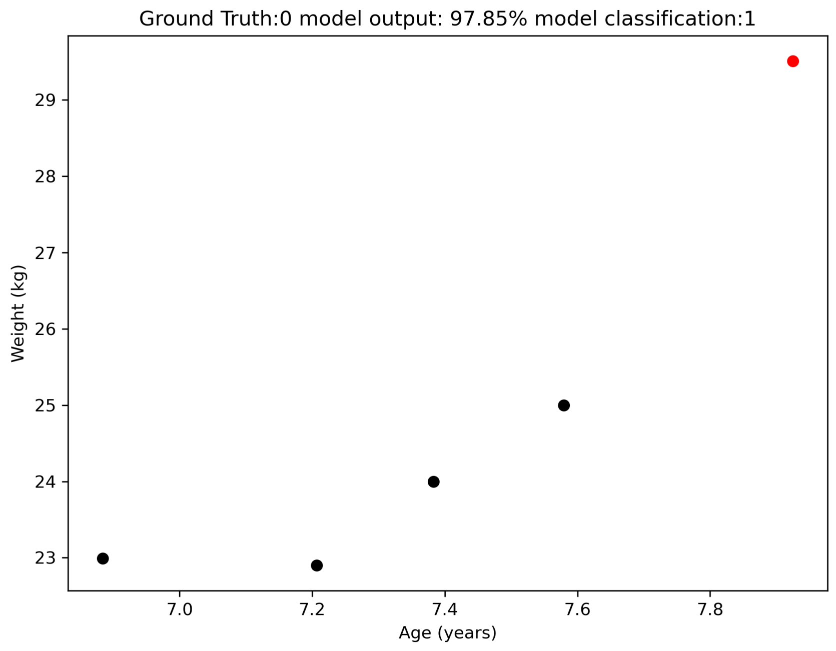


Figure A.8: False Positive Example 8

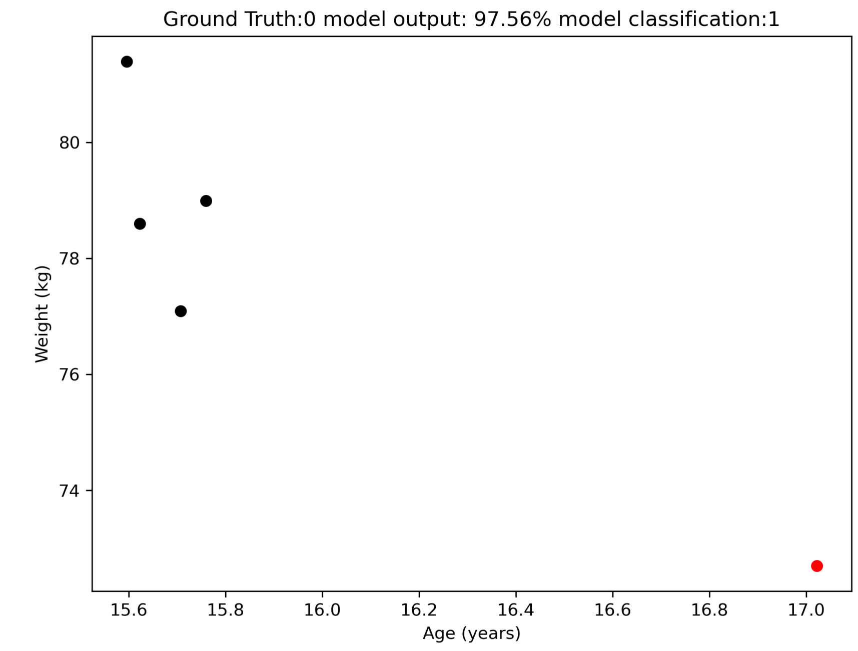


Figure A.9: False Positive Example 9

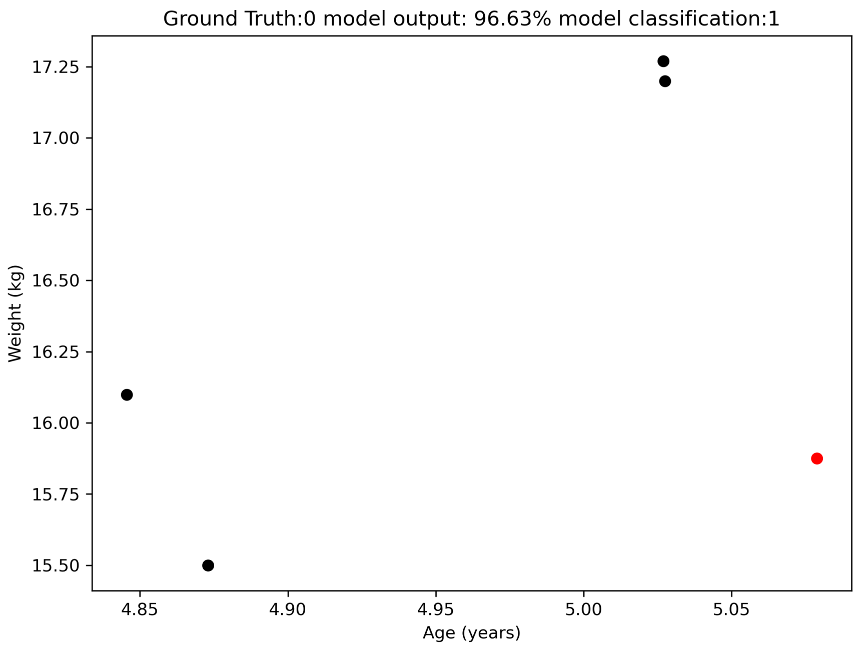


Figure A.10: False Positive Example 10

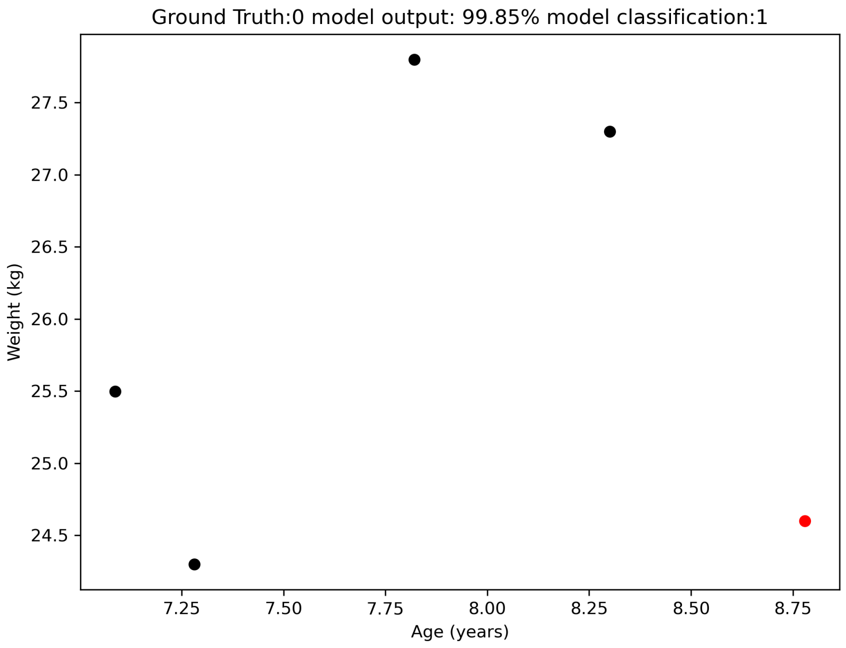


Figure A.11: False Positive Example 11

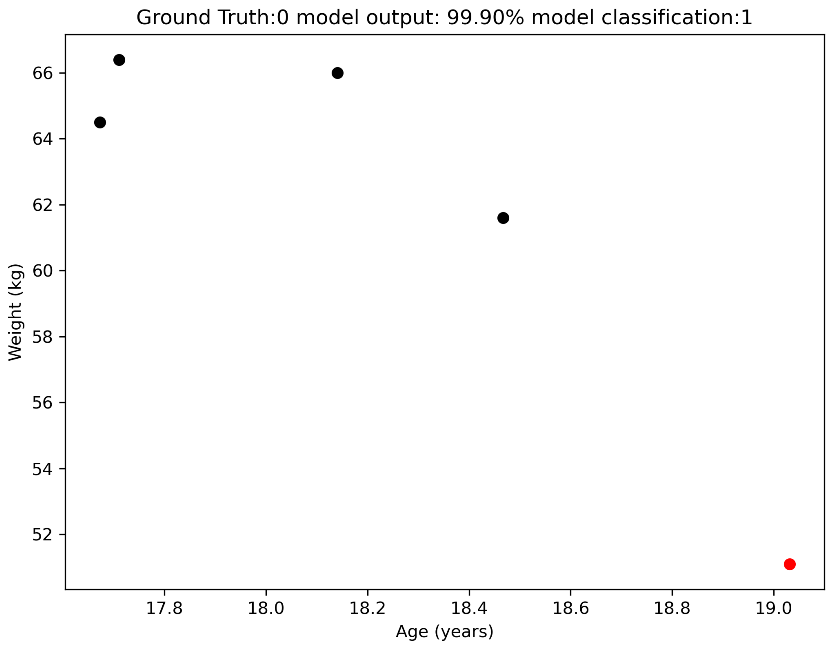


Figure A.12: False Positive Example 12

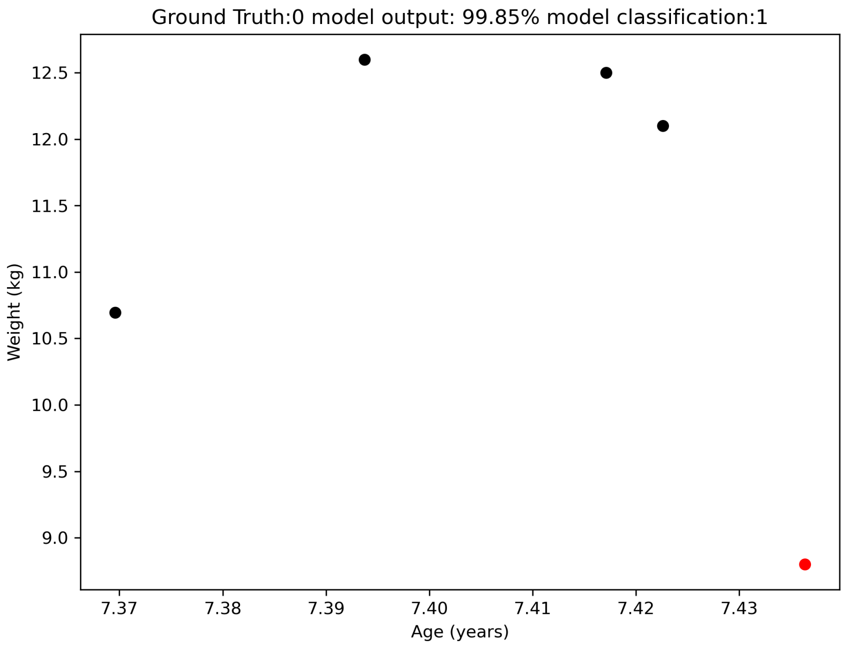


Figure A.13: False Positive Example 13

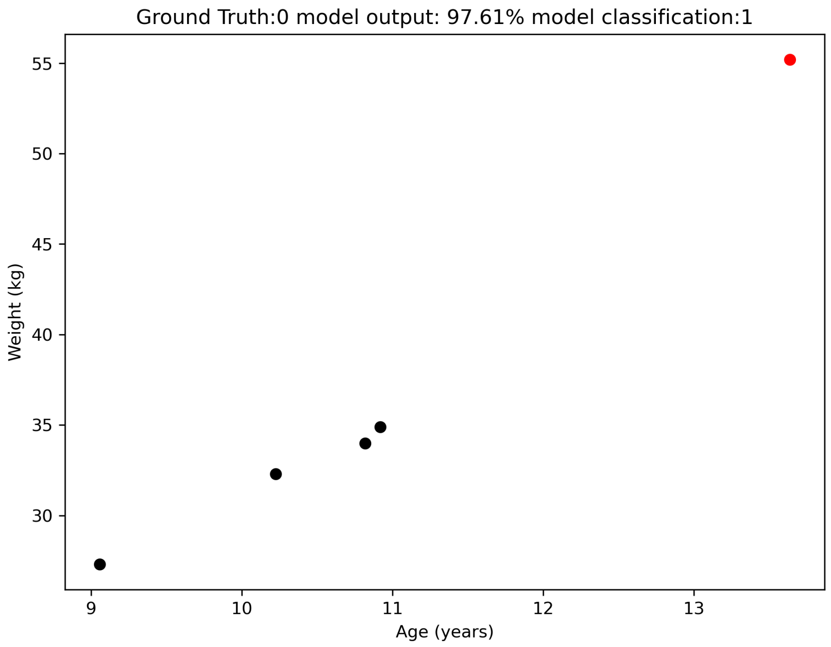


Figure A.14: False Positive Example 14

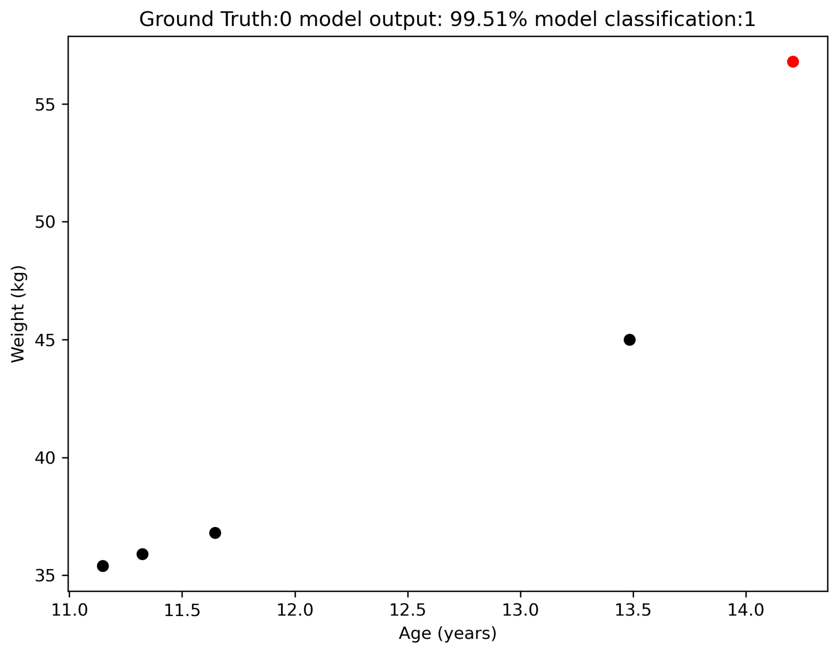


Figure A.15: False Positive Example 15

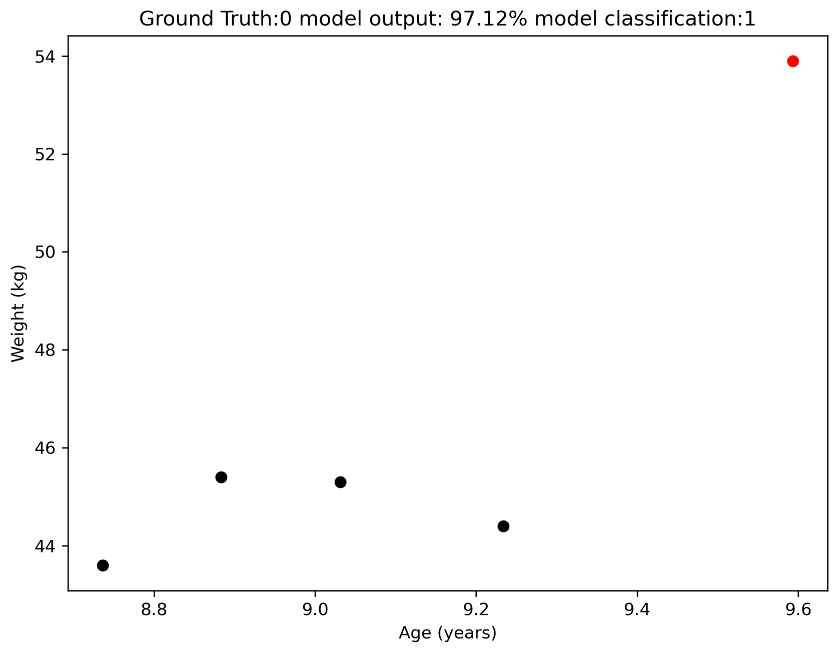


Figure A.16: False Positive Example 16

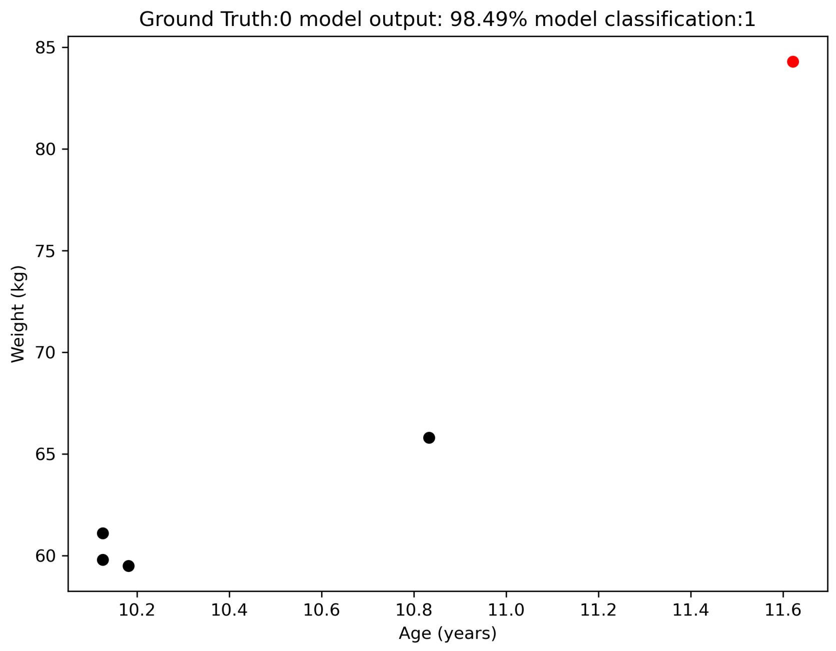


Figure A.17: False Positive Example 17

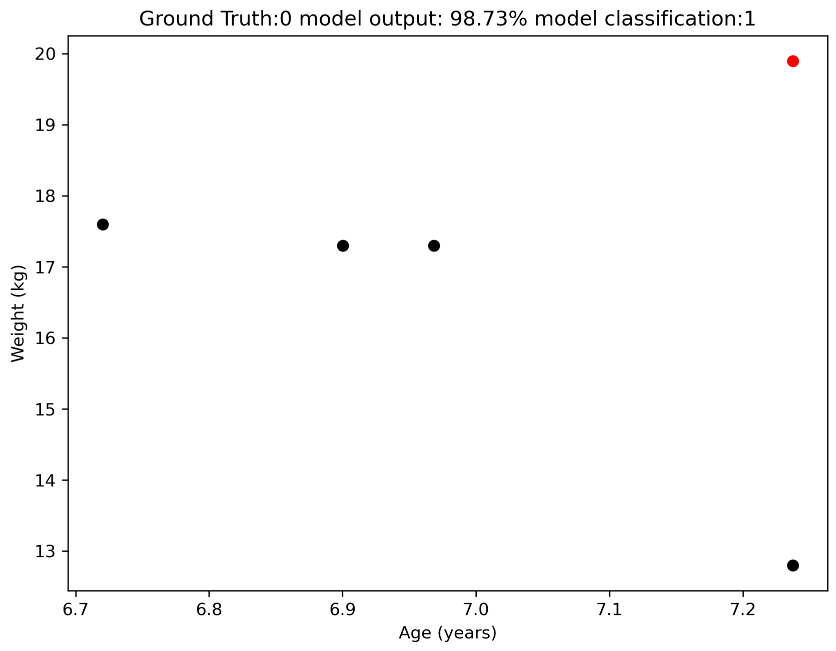


Figure A.18: False Positive Example 18

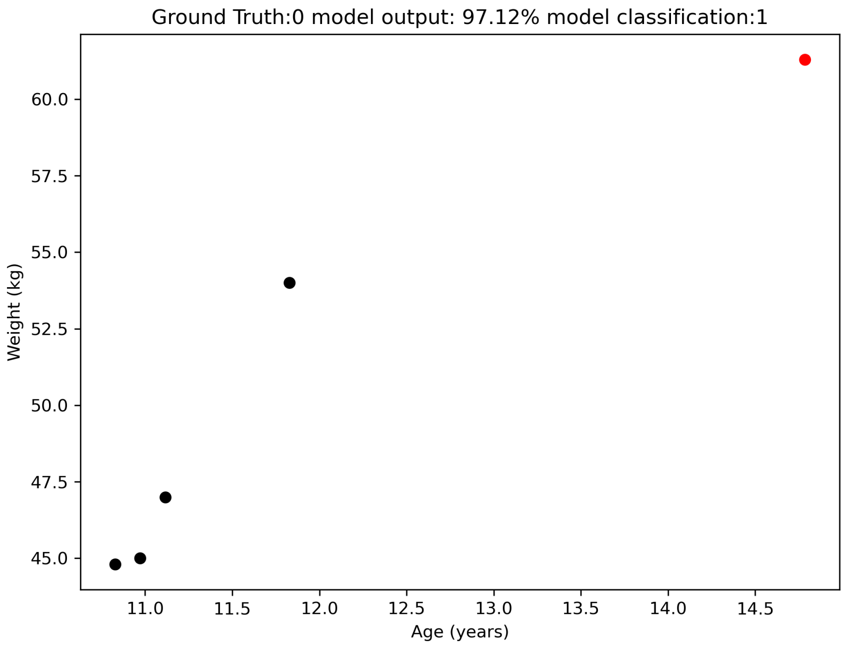


Figure A.19: False Positive Example 19

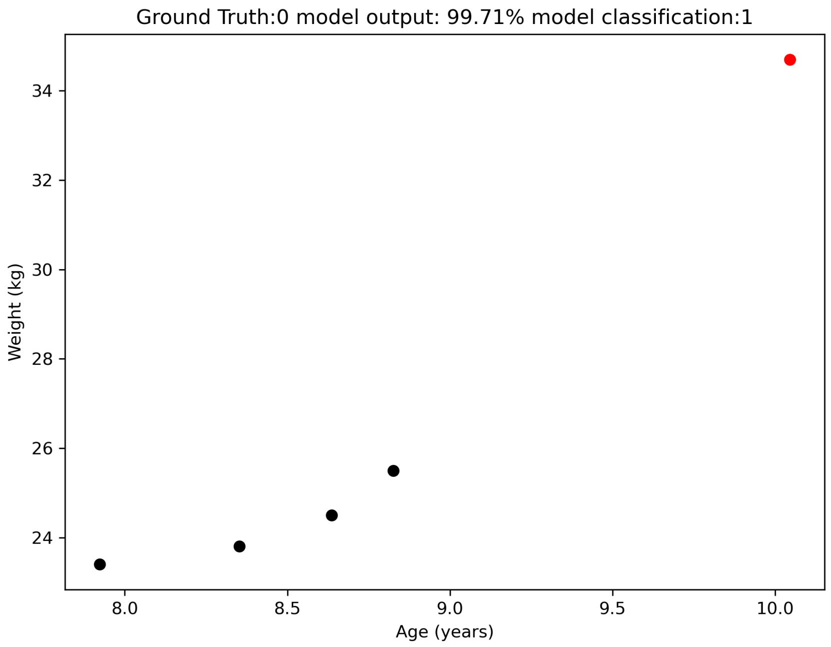


Figure A.20: False Positive Example 20

## A.2 False Negative

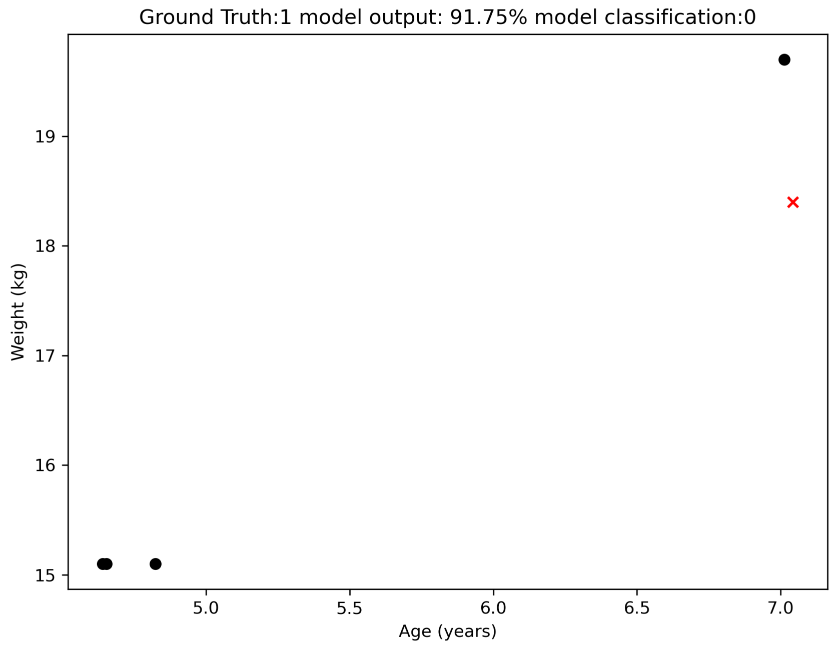


Figure A.21: False Negative Example 1

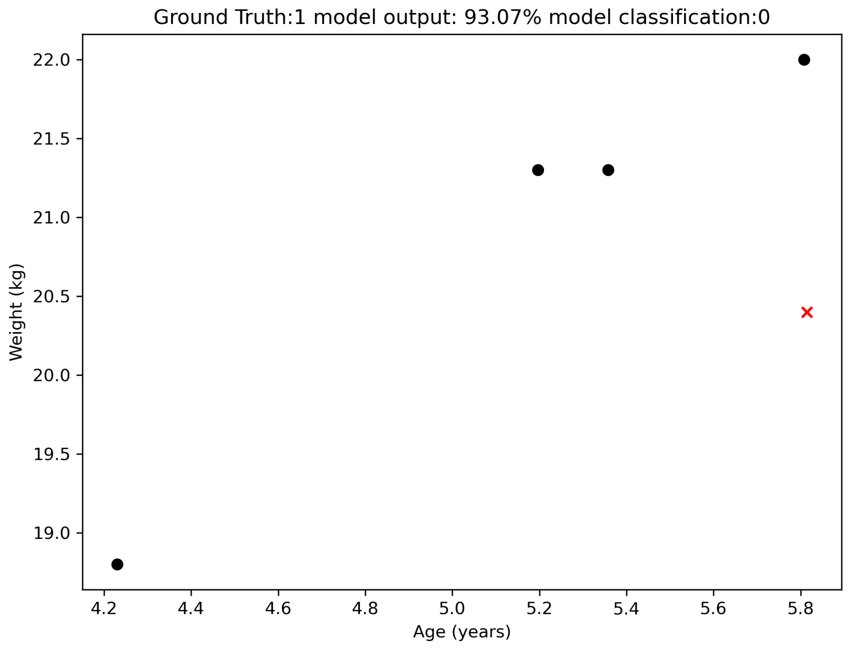


Figure A.22: False Negative Example 2

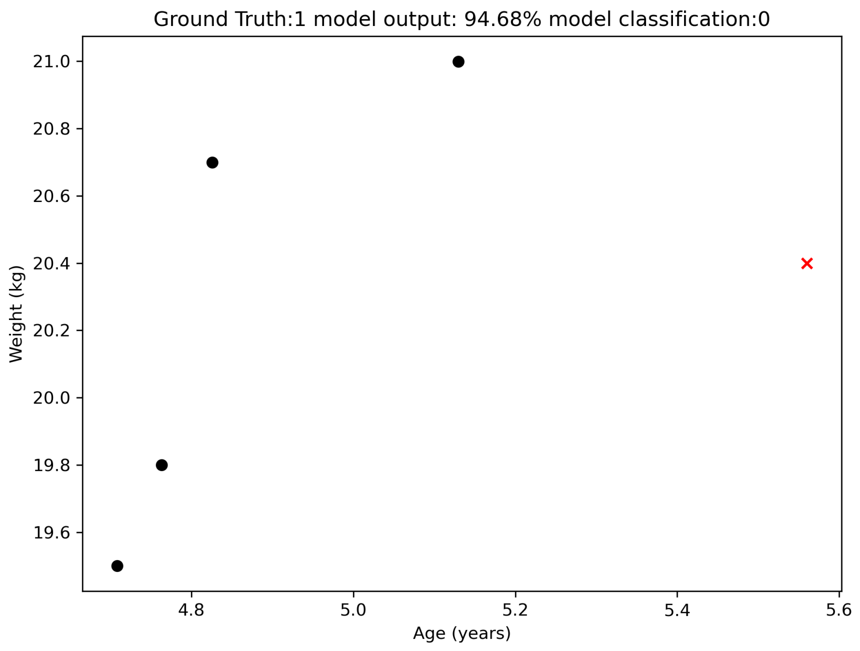


Figure A.23: False Negative Example 3

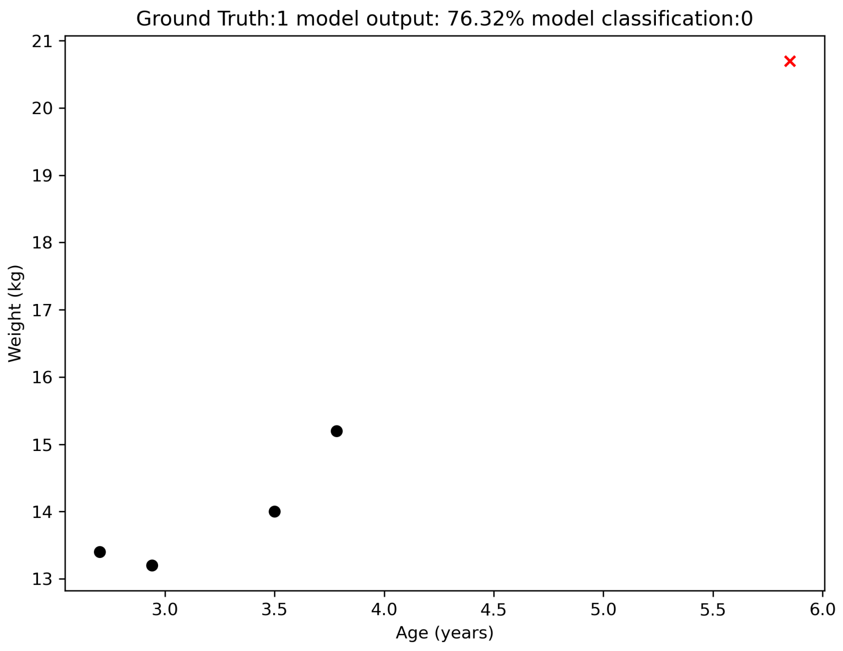


Figure A.24: False Negative Example 4

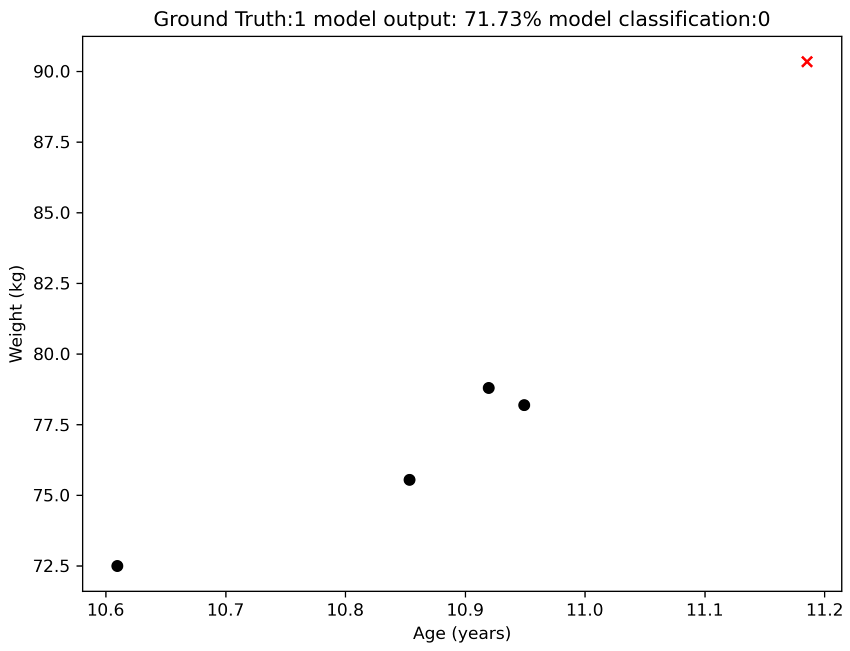


Figure A.25: False Negative Example 5

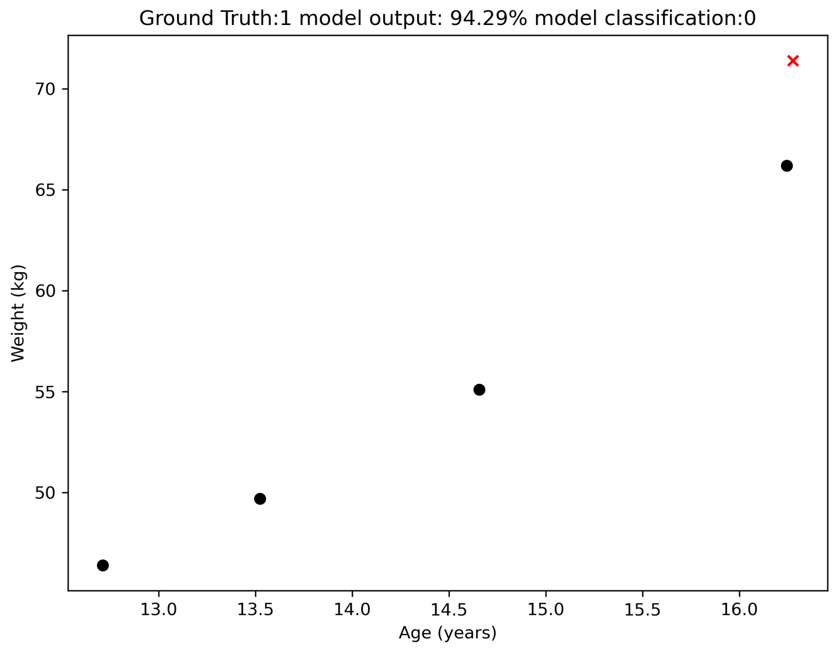


Figure A.26: False Negative Example 6

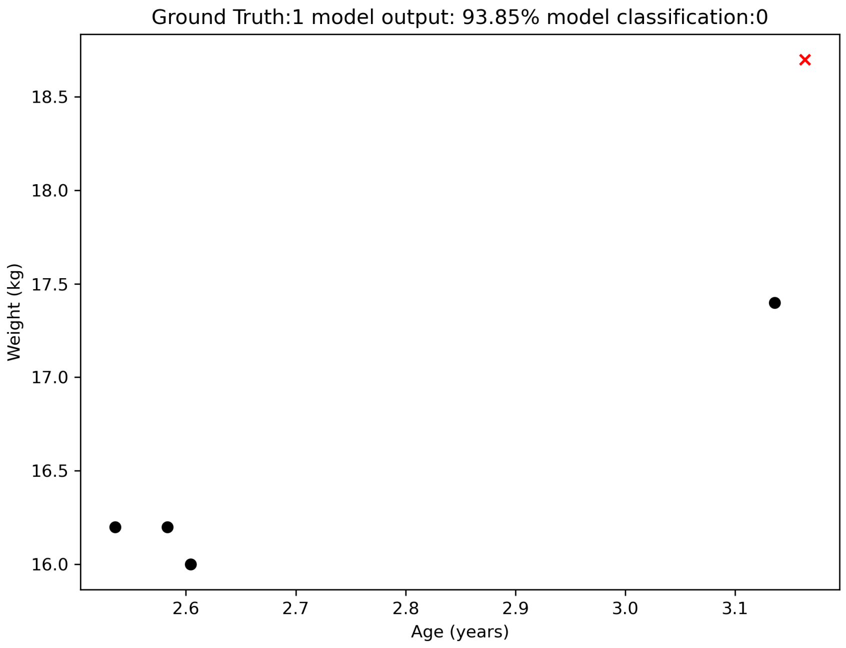


Figure A.27: False Negative Example 7

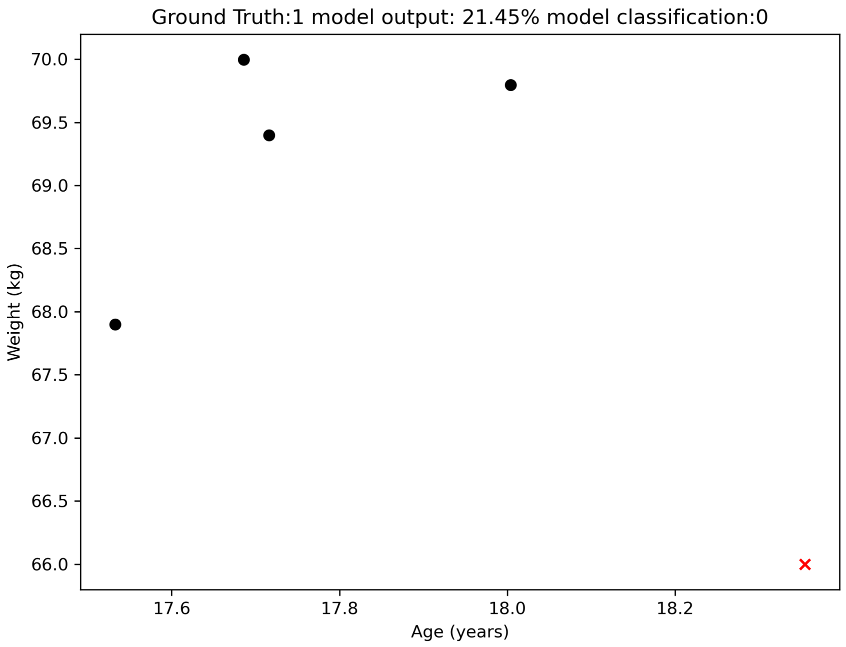


Figure A.28: False Negative Example 8

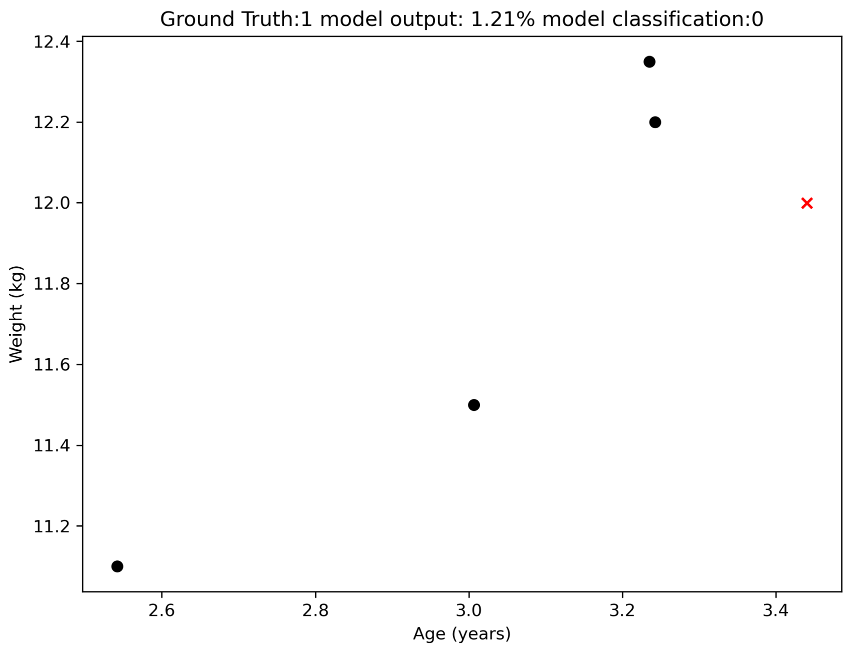


Figure A.29: False Negative Example 9

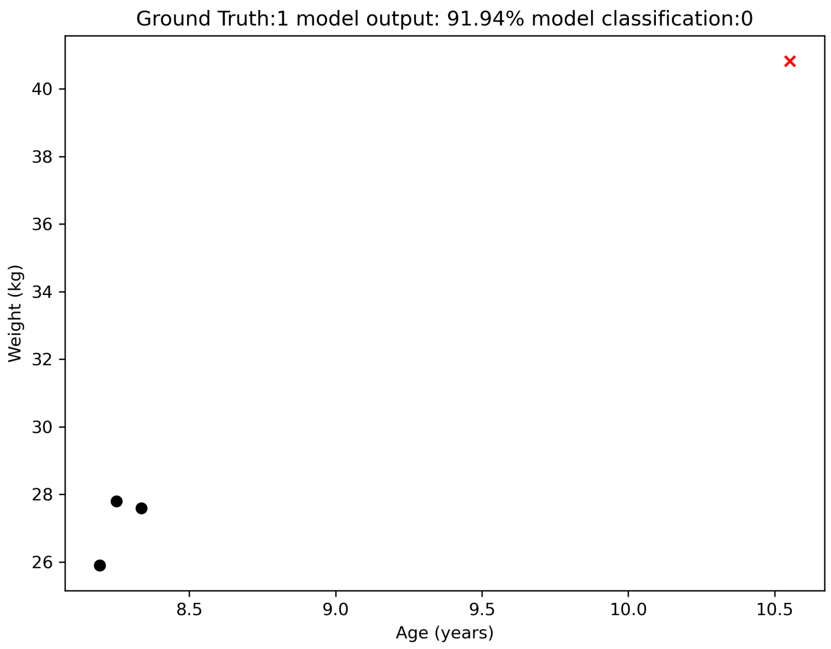


Figure A.30: False Negative Example 10

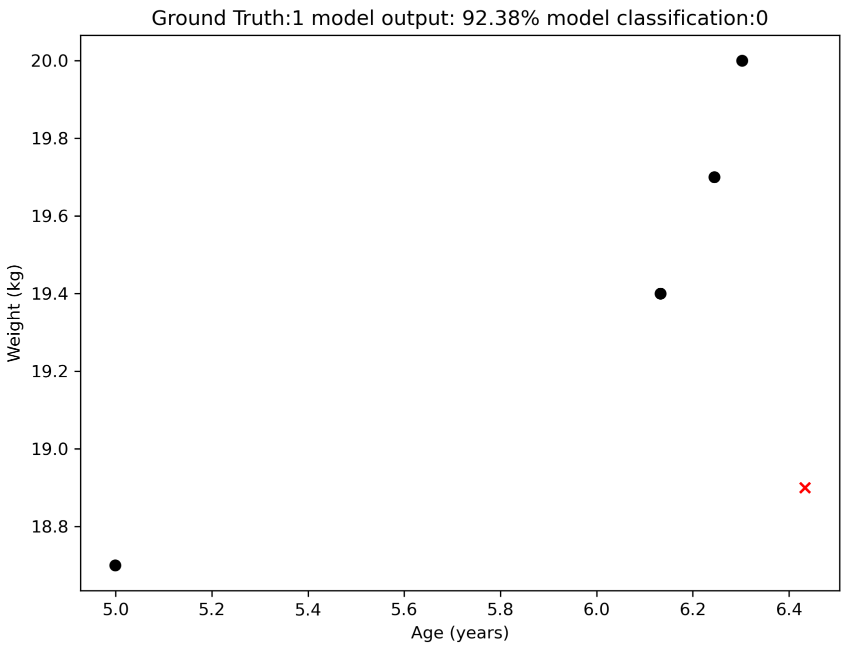


Figure A.31: False Negative Example 11

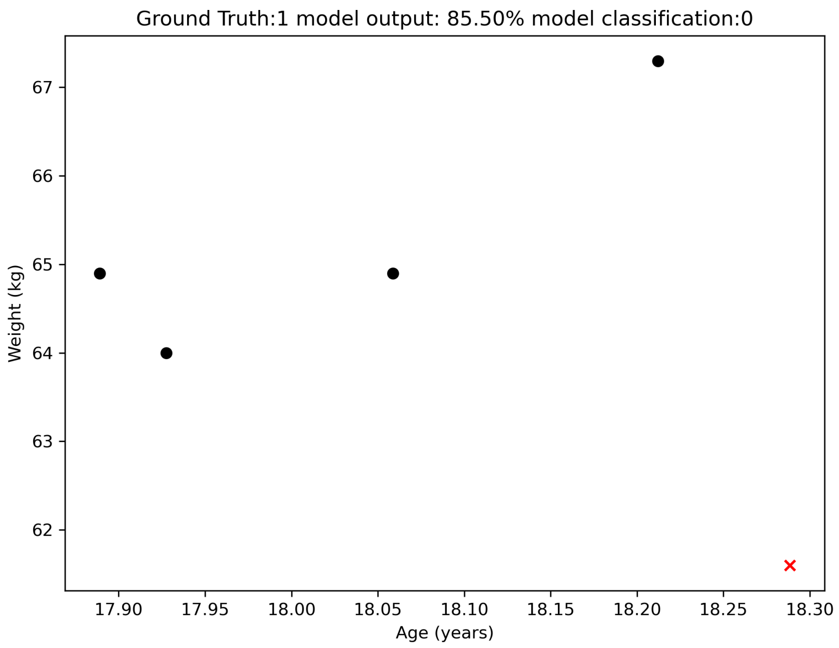


Figure A.32: False Negative Example 12

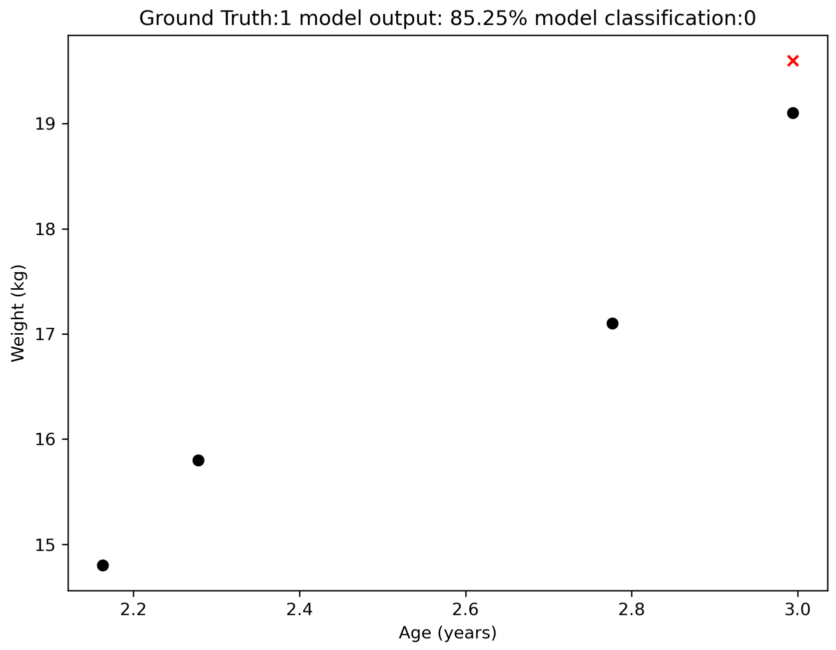


Figure A.33: False Negative Example 13

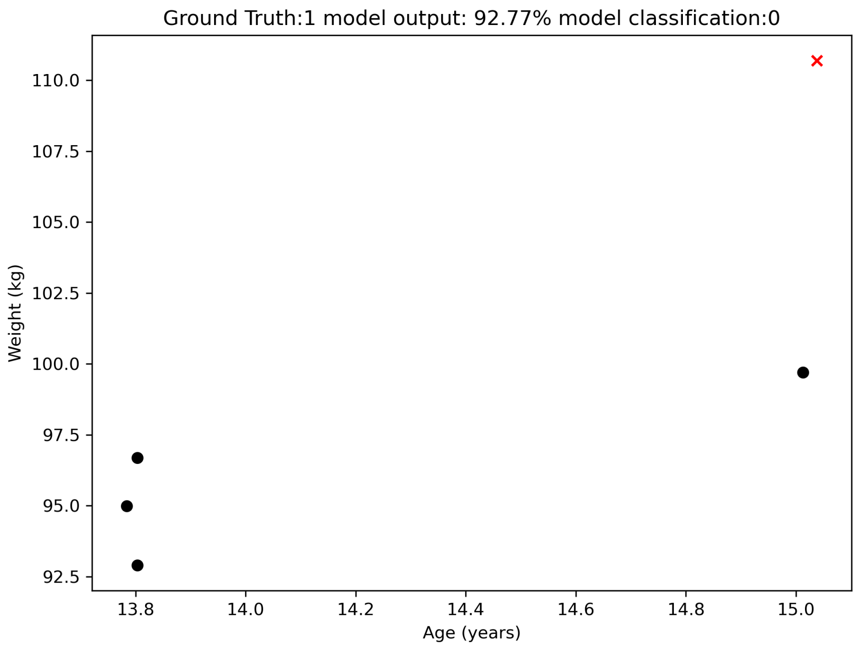


Figure A.34: False Negative Example 14

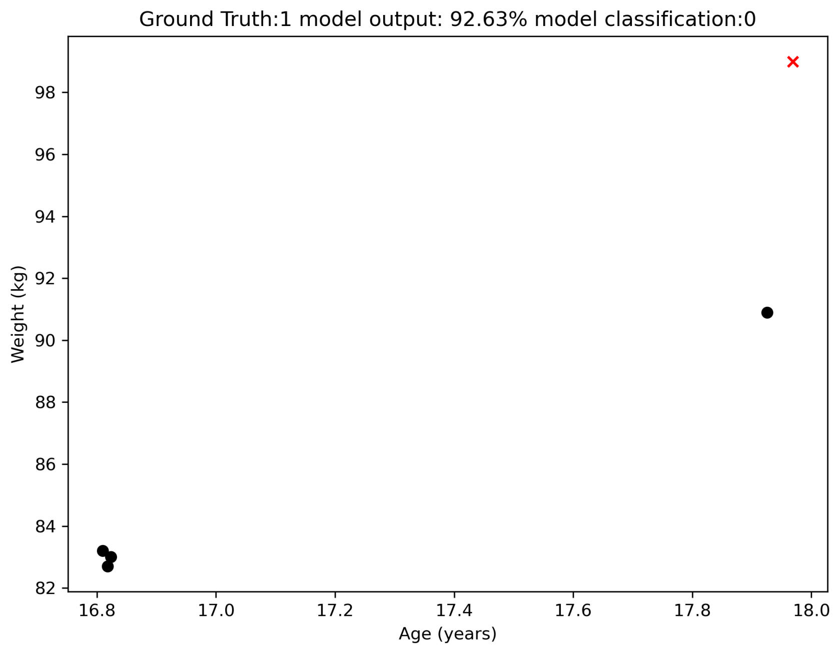


Figure A.35: False Negative Example 15

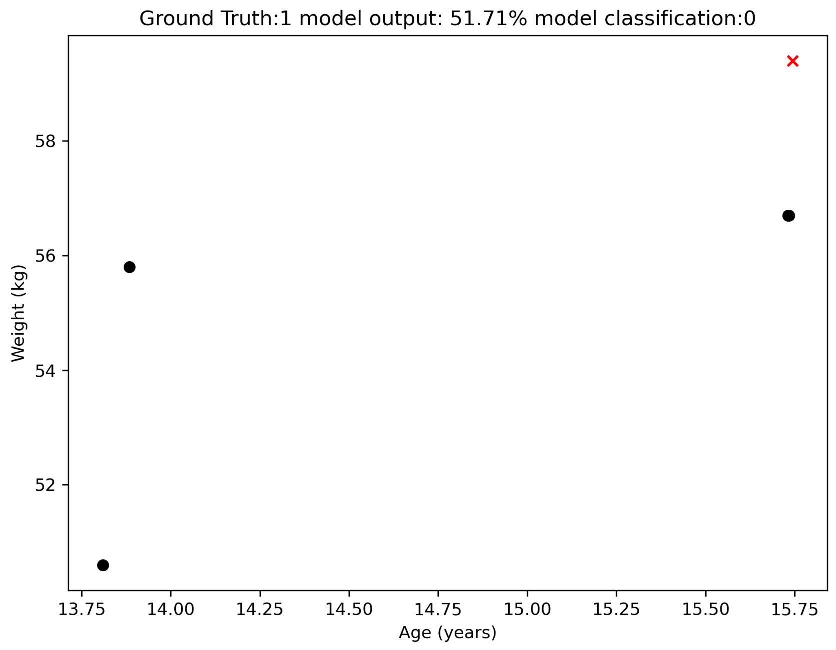


Figure A.36: False Negative Example 16

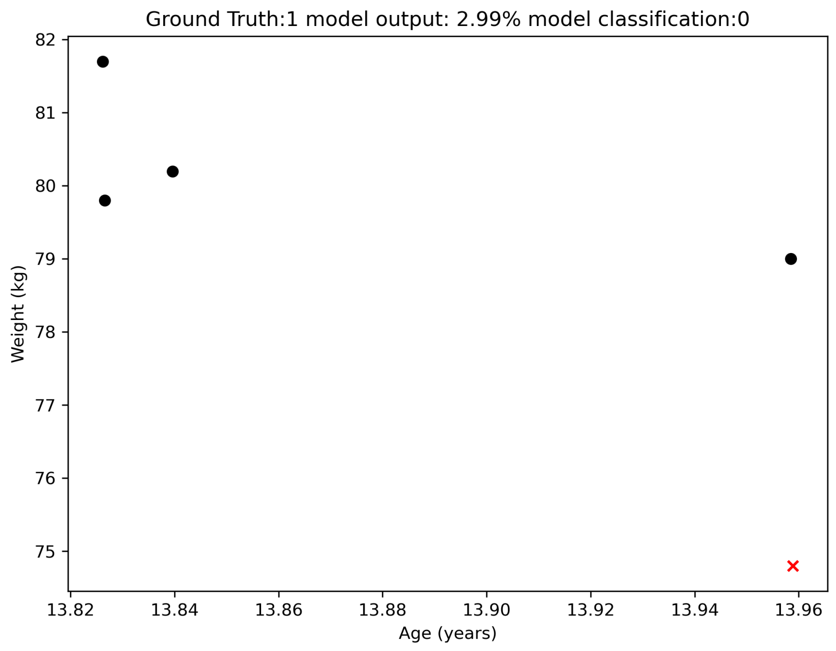


Figure A.37: False Negative Example 17

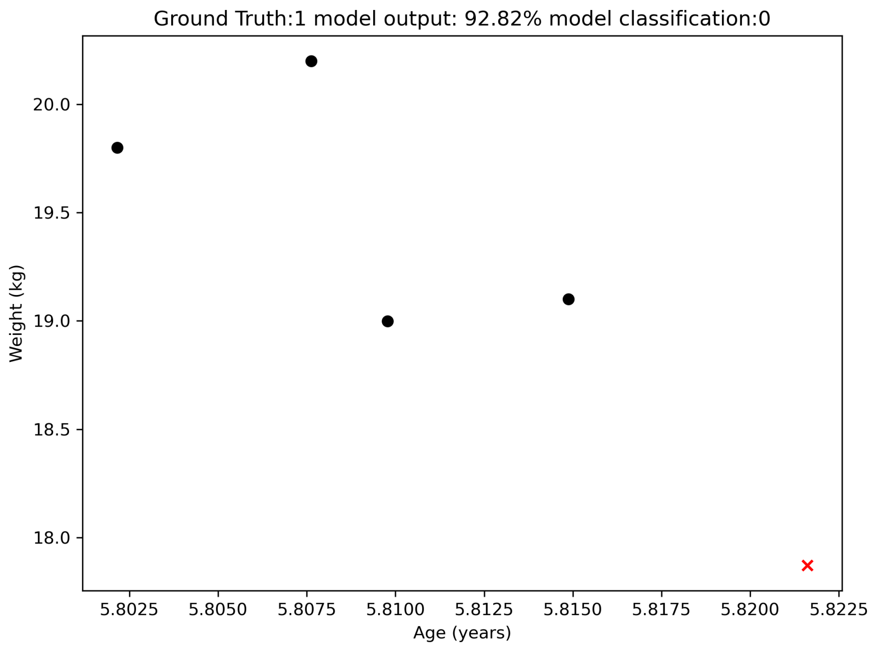


Figure A.38: False Negative Example 18

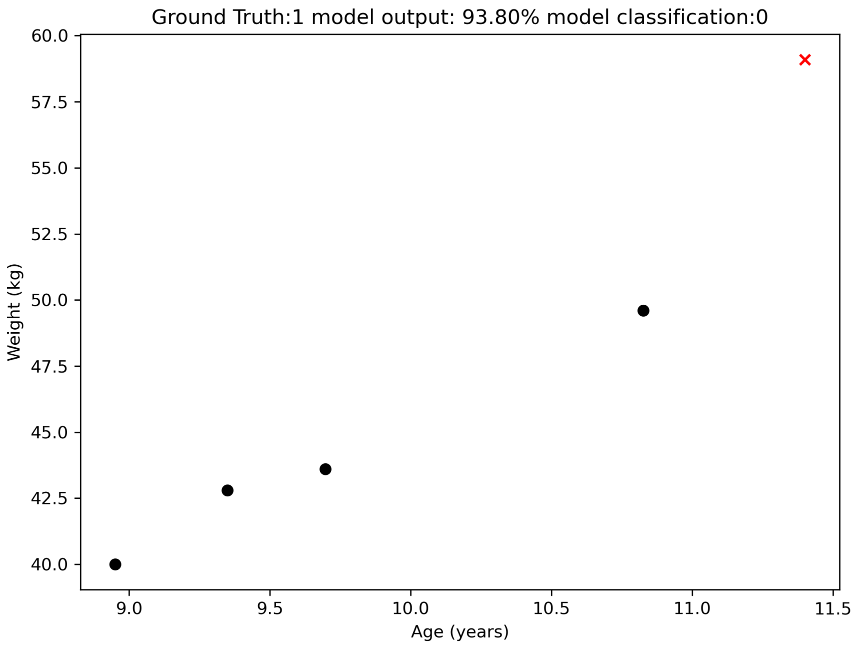


Figure A.39: False Negative Example 19

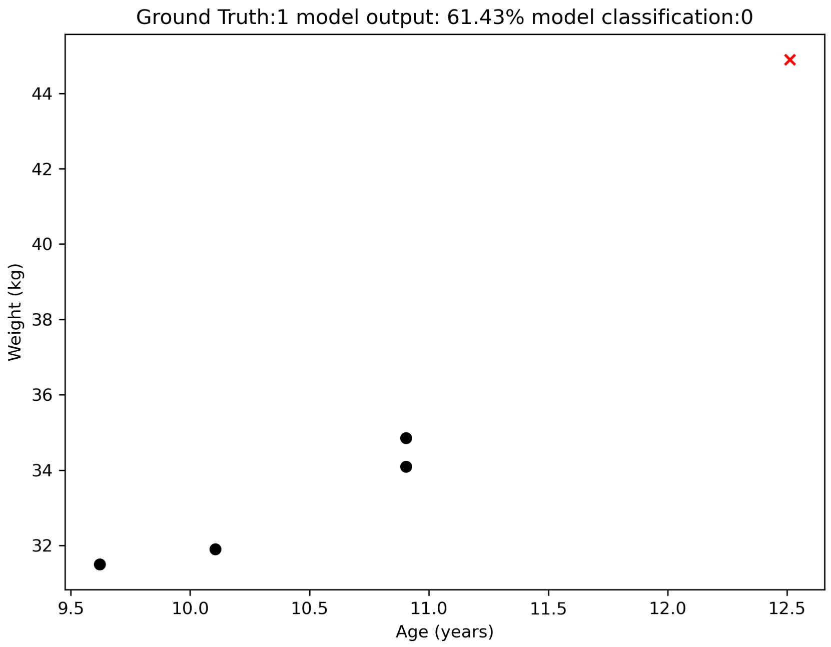


Figure A.40: False Negative Example 20

## A.3 True Positive

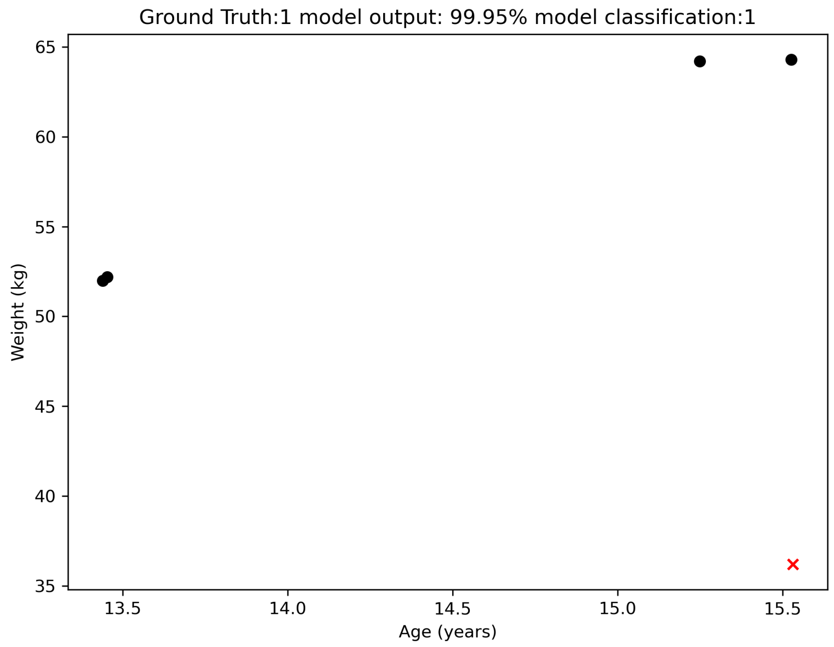


Figure A.41: True Positive Example 1

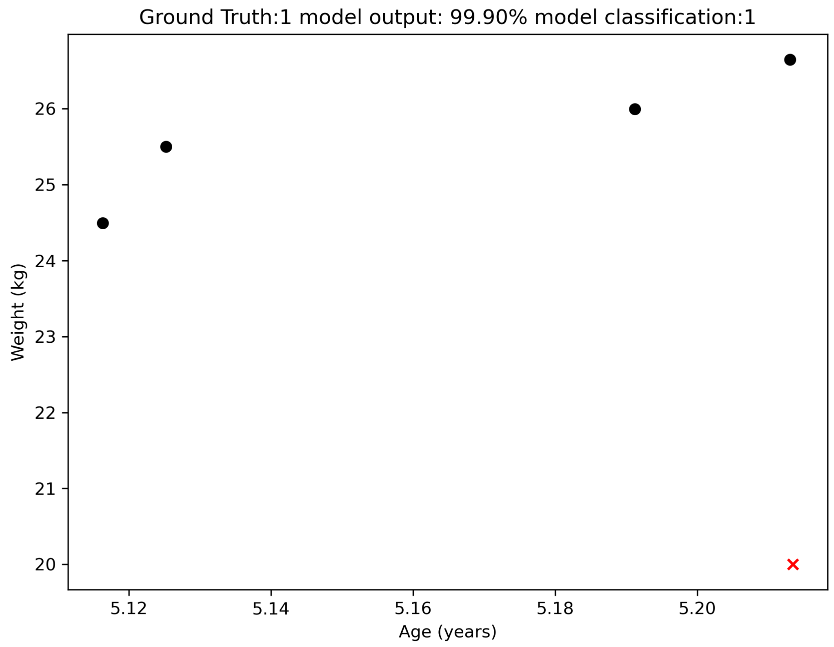


Figure A.42: True Positive Example 2

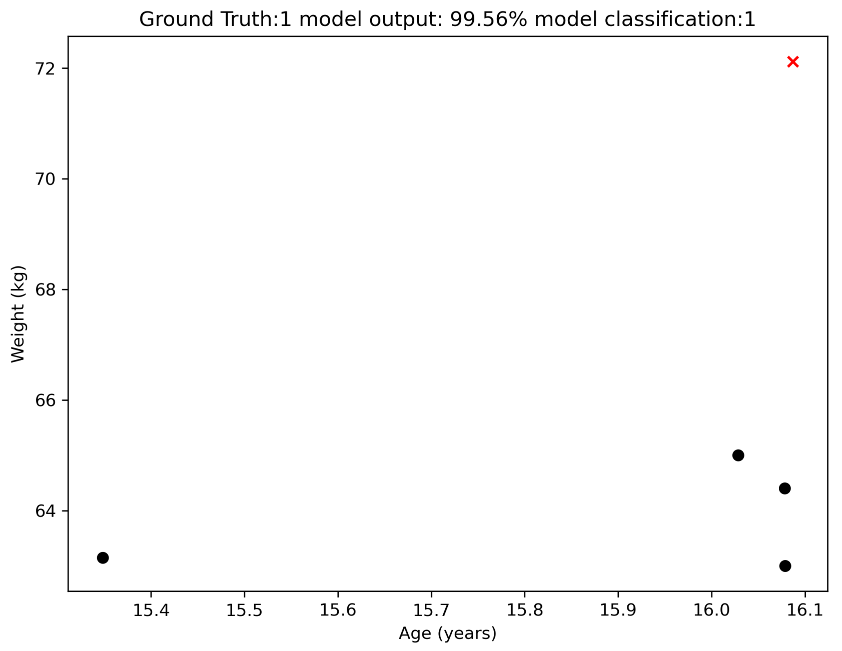


Figure A.43: True Positive Example 3

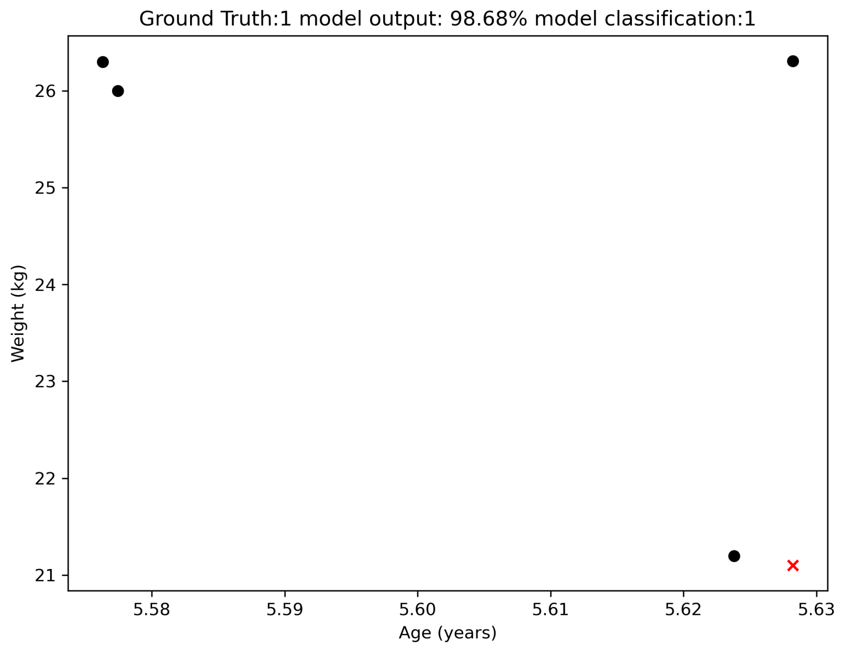


Figure A.44: True Positive Example 4

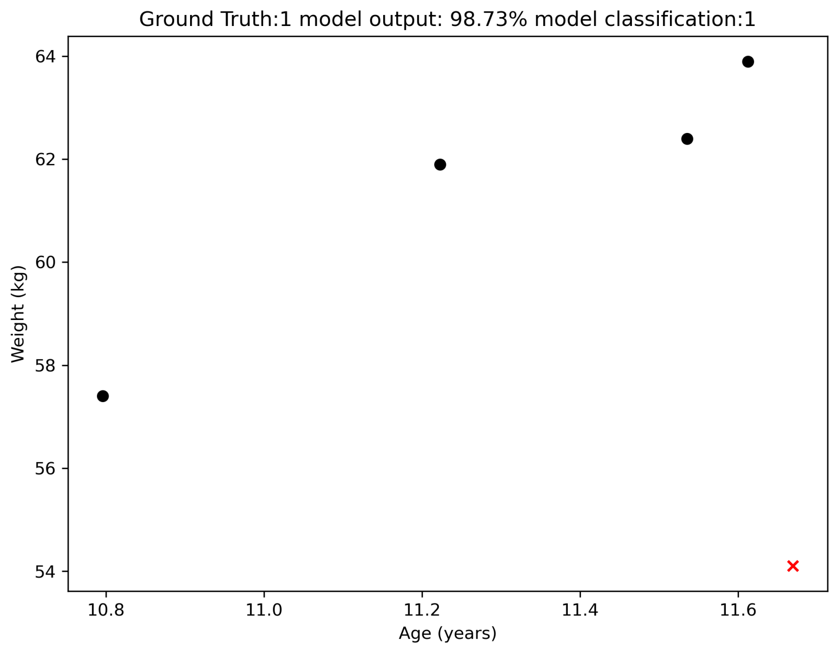


Figure A.45: True Positive Example 5

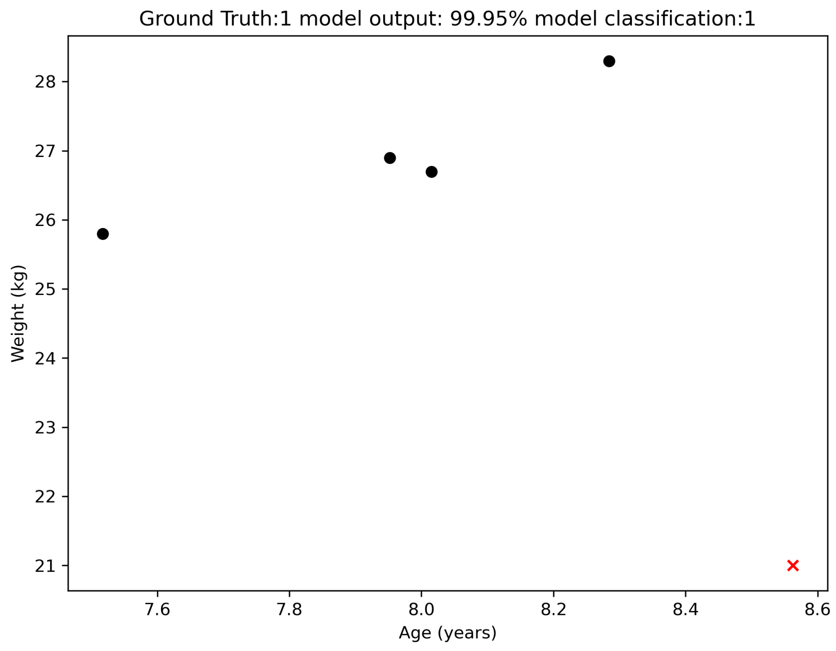


Figure A.46: True Positive Example 6

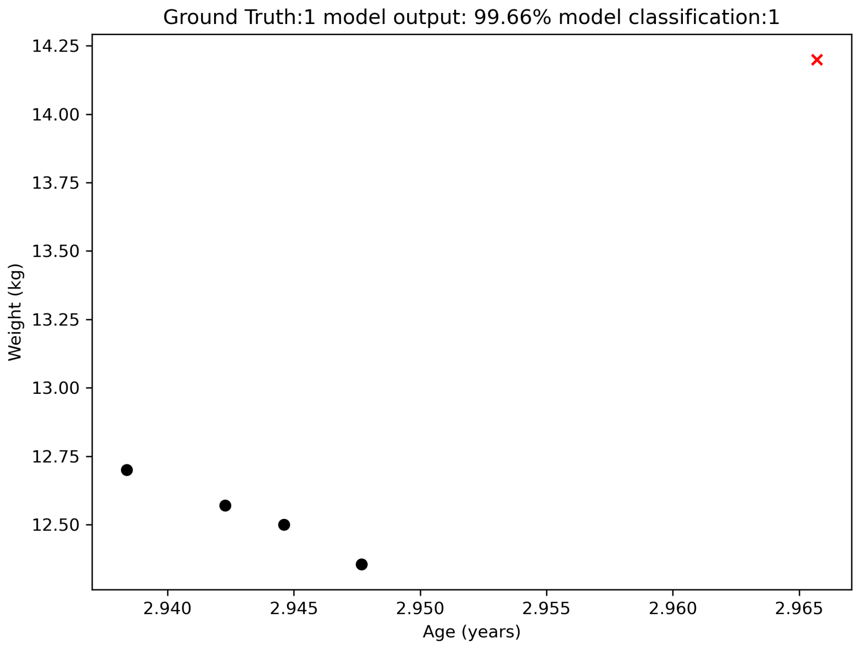


Figure A.47: True Positive Example 7

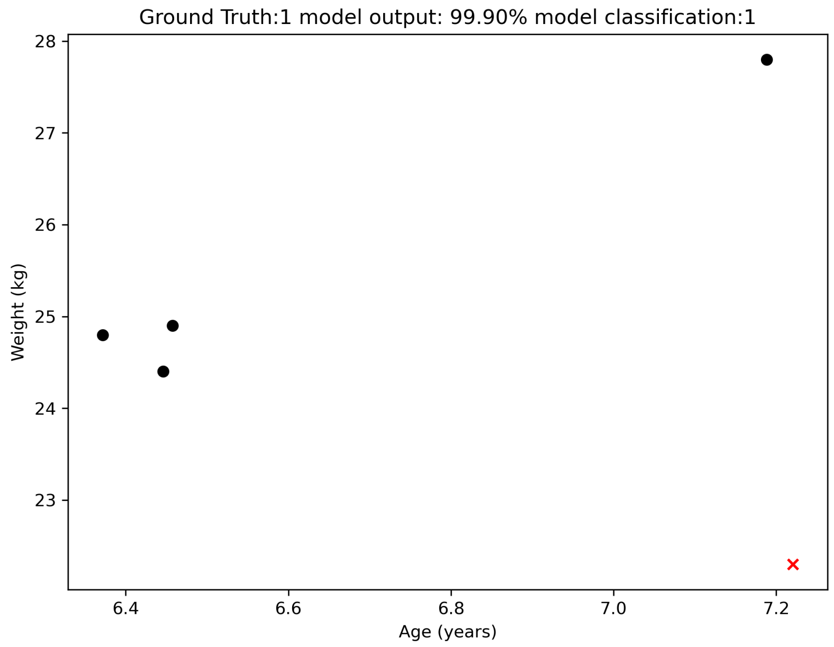


Figure A.48: True Positive Example 8

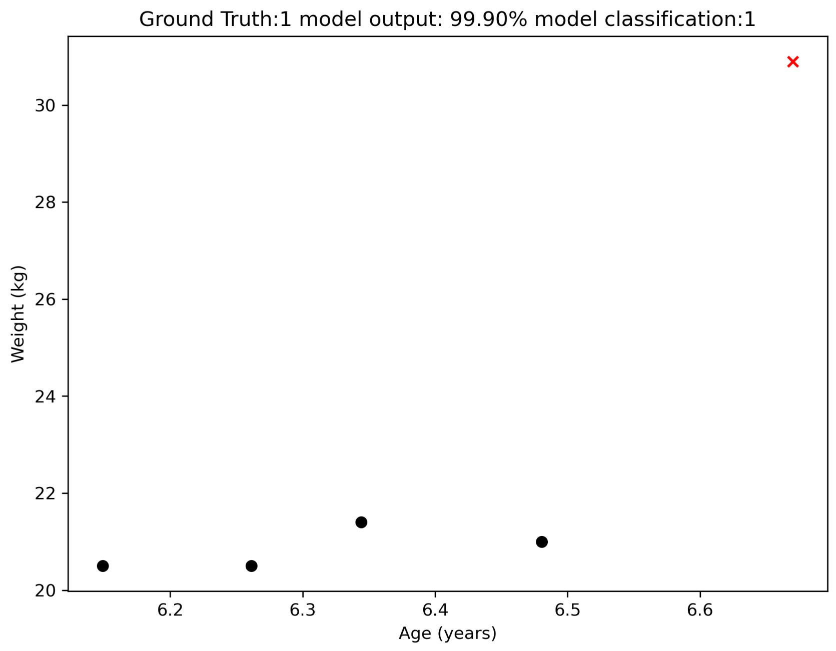


Figure A.49: True Positive Example 9

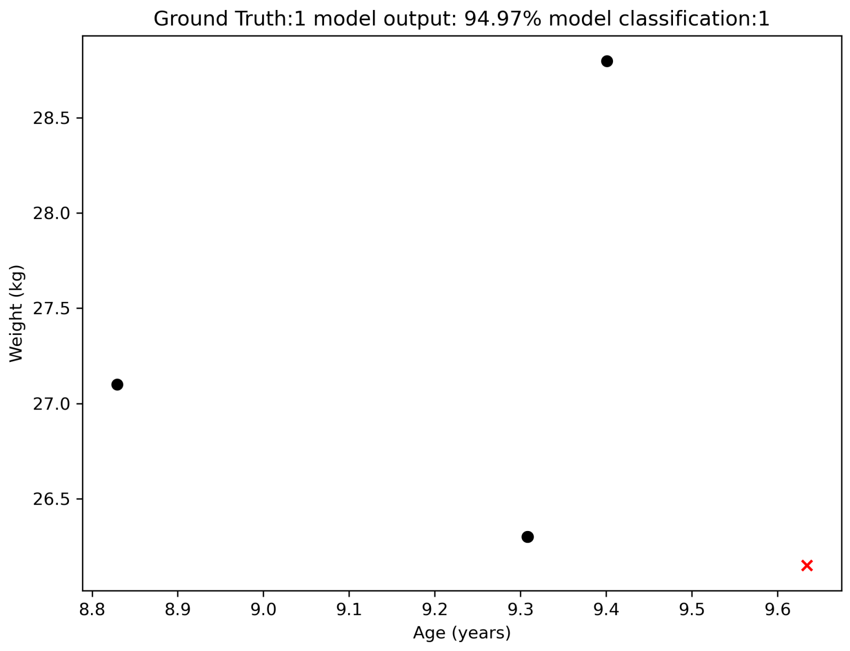


Figure A.50: True Positive Example 10

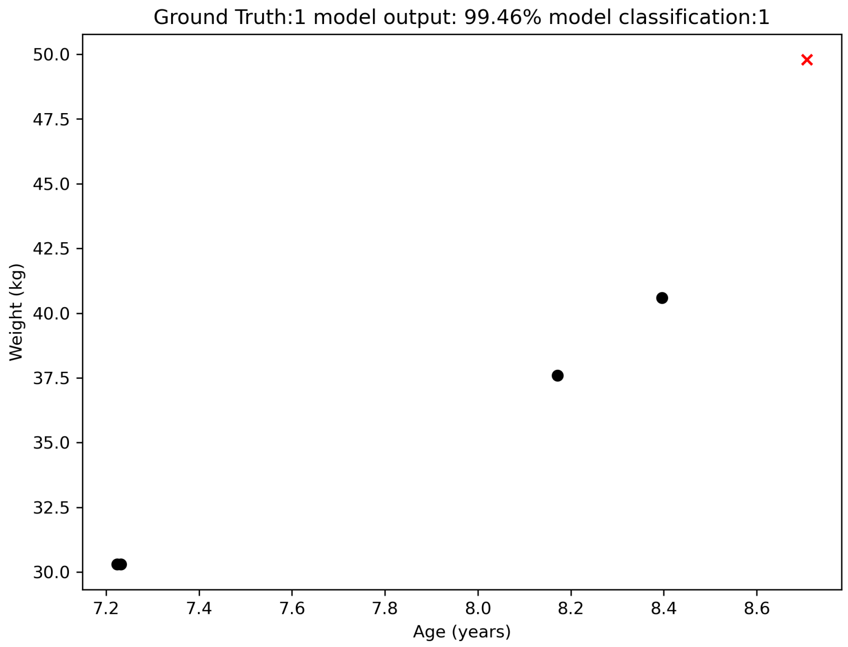


Figure A.51: True Positive Example 11

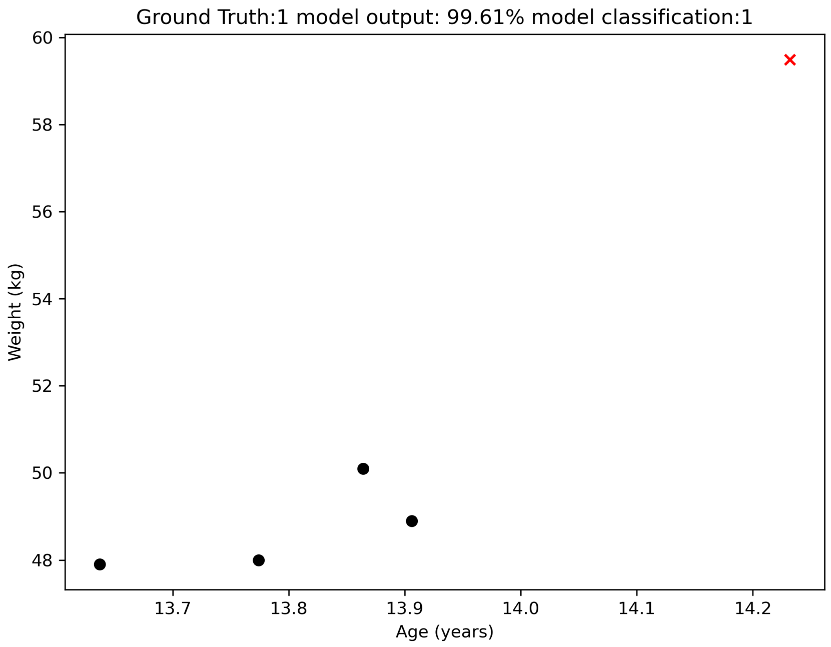


Figure A.52: True Positive Example 12

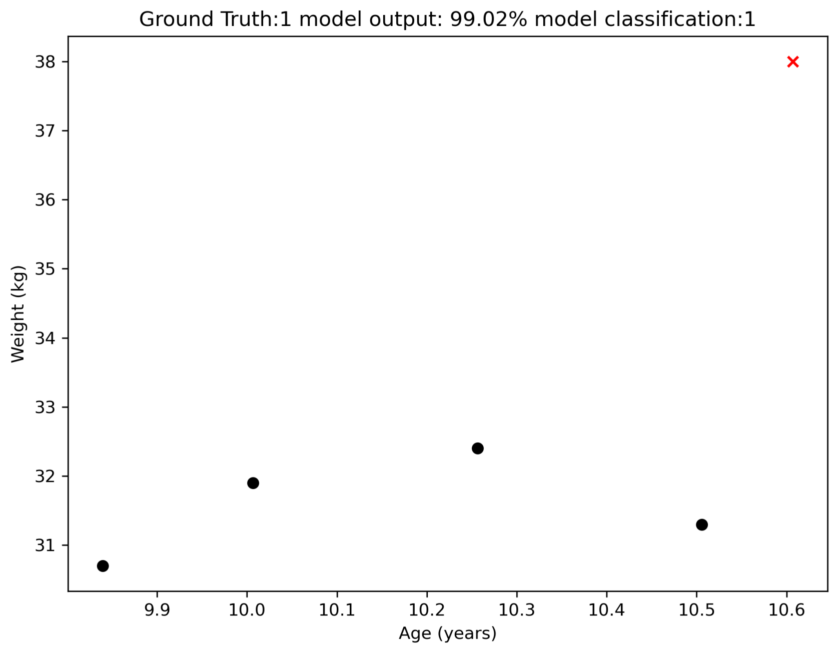


Figure A.53: True Positive Example 13

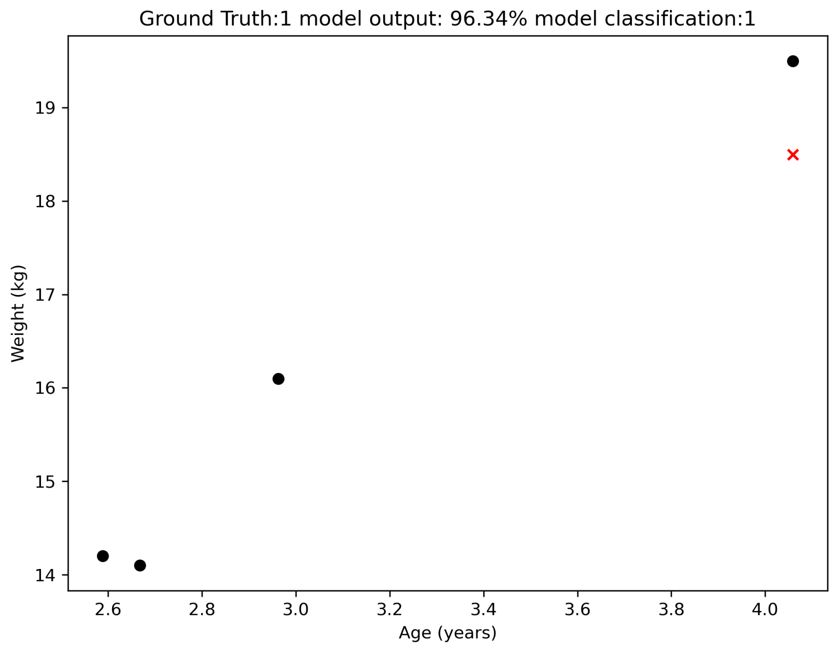


Figure A.54: True Positive Example 14

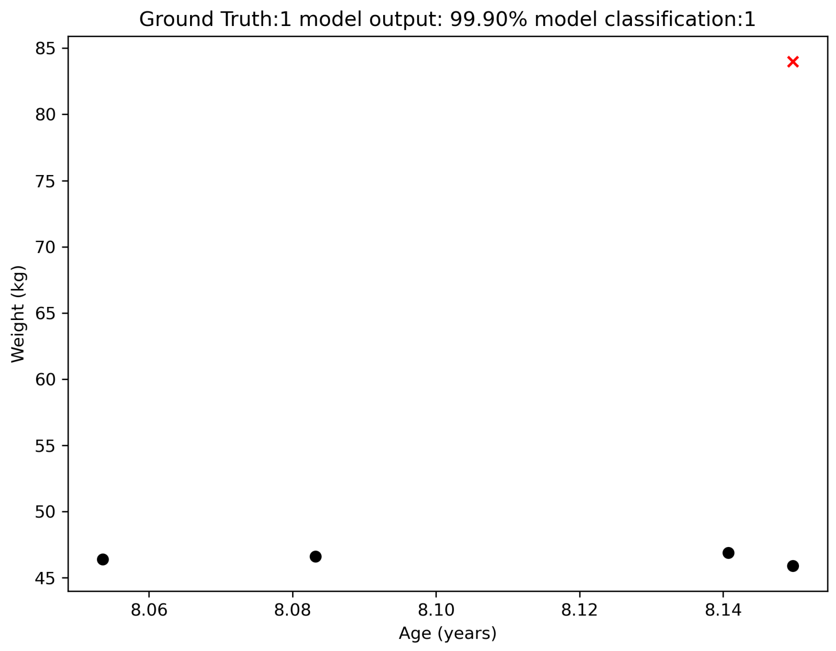


Figure A.55: True Positive Example 15

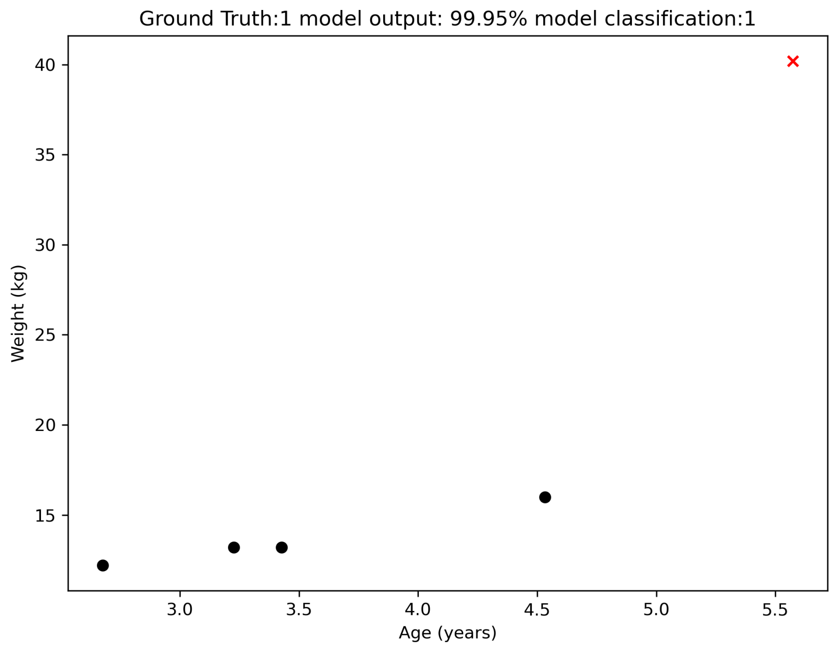


Figure A.56: True Positive Example 16

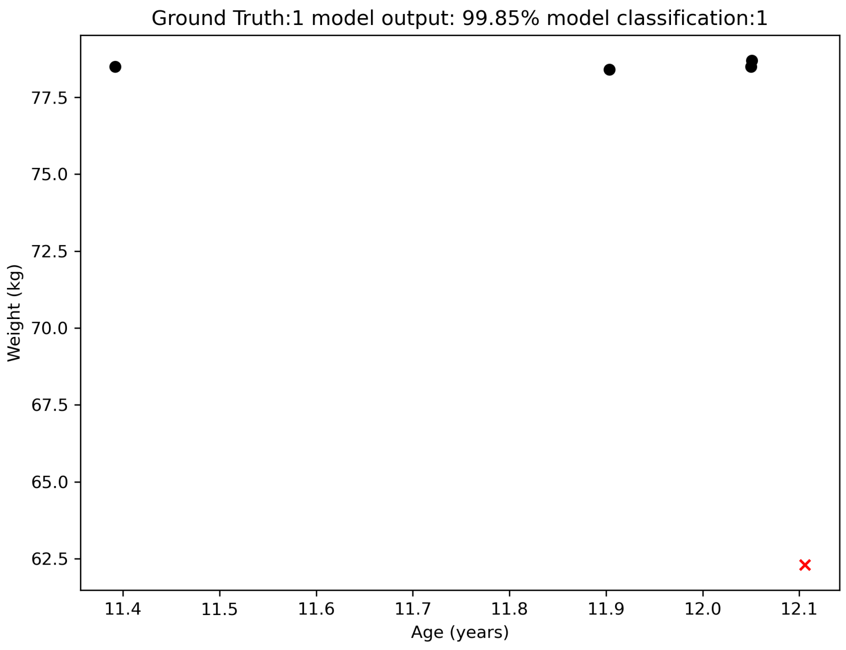


Figure A.57: True Positive Example 17

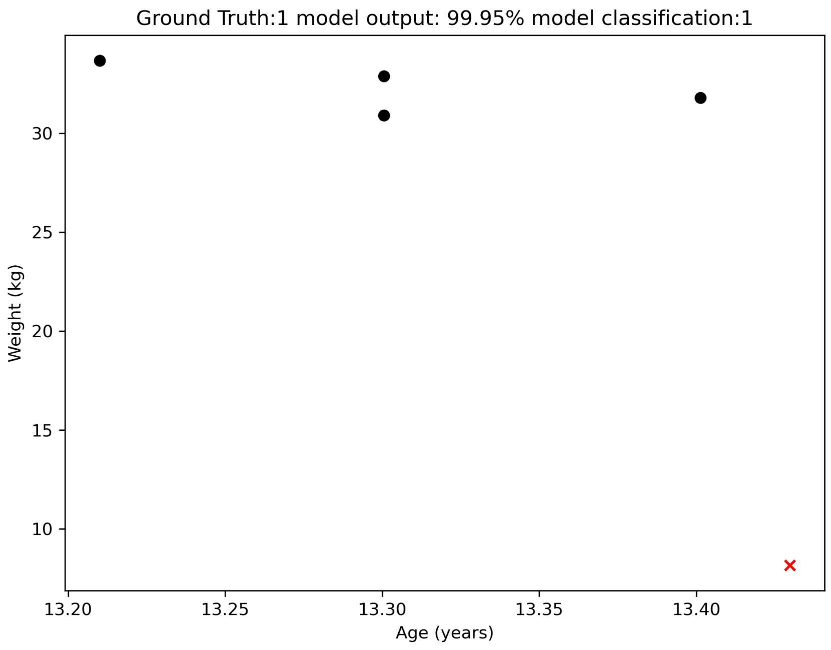


Figure A.58: True Positive Example 18

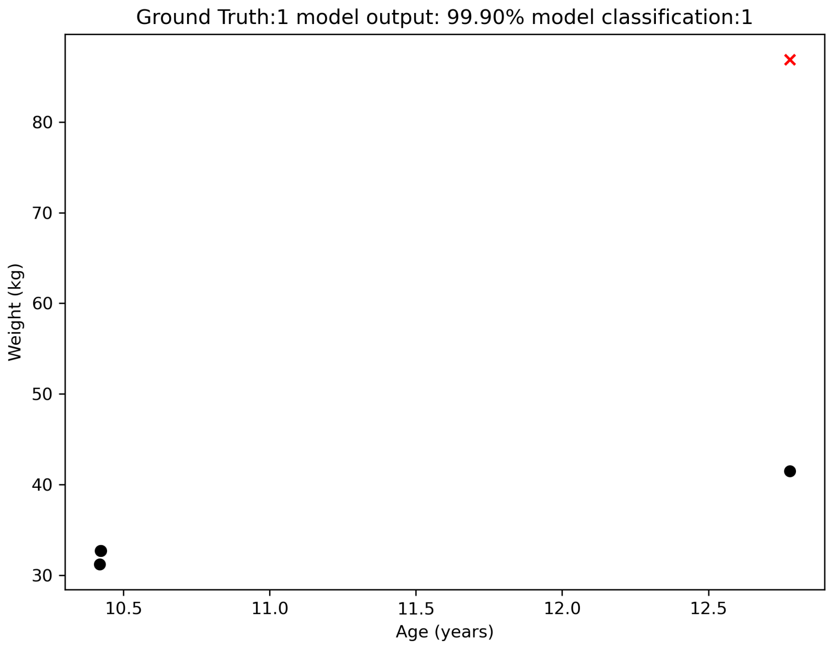


Figure A.59: True Positive Example 19

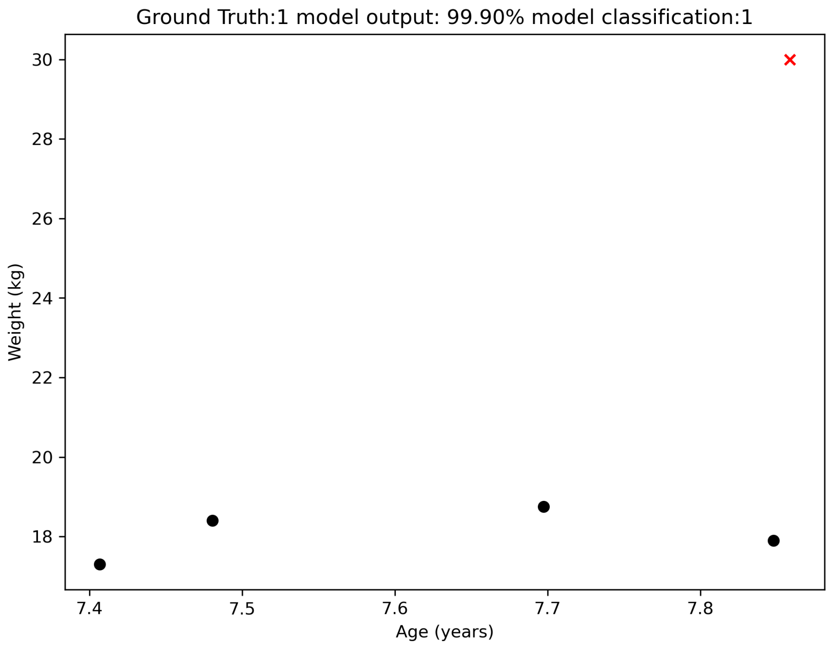


Figure A.60: True Positive Example 20

## A.4 True Negative

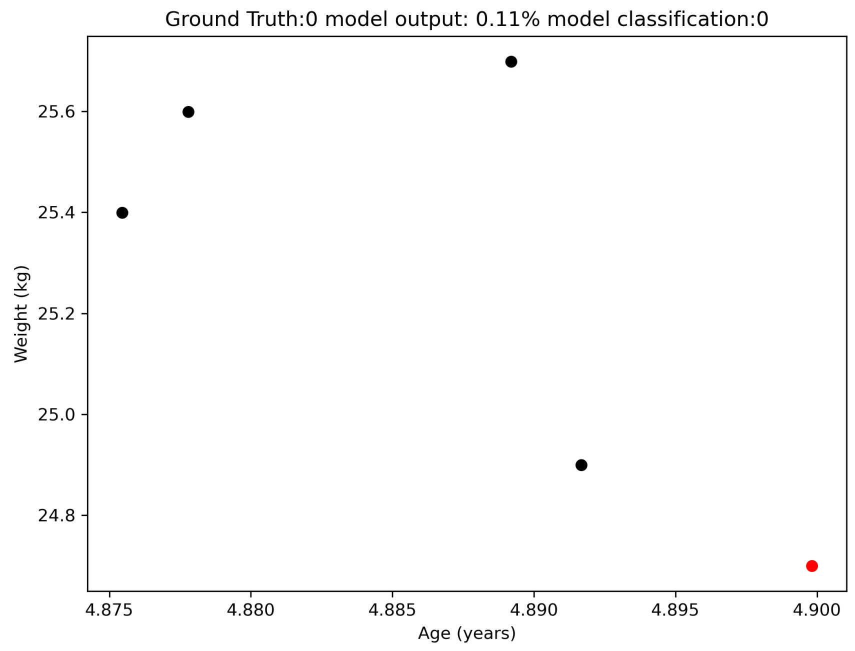


Figure A.61: True Negative Example 1

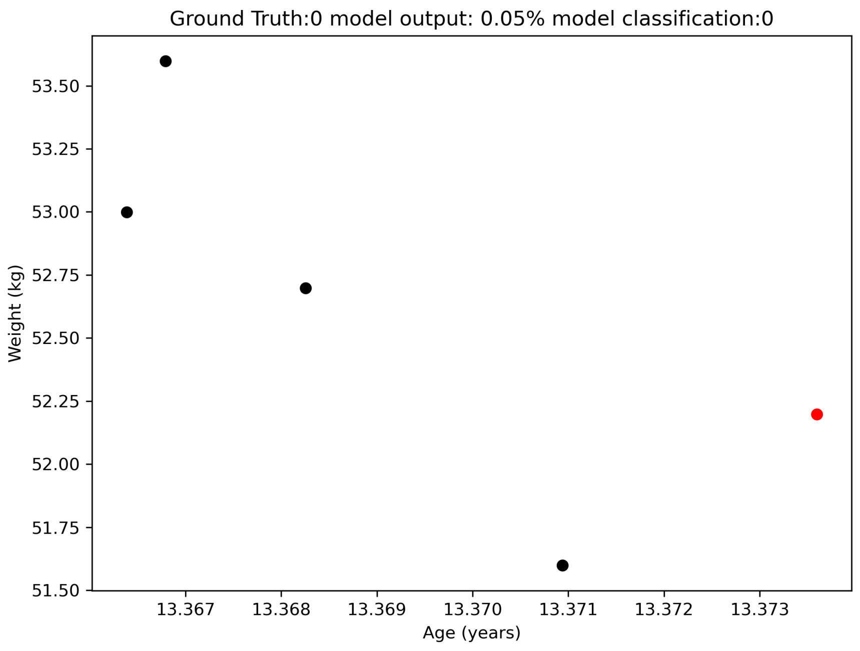


Figure A.62: True Negative Example 2

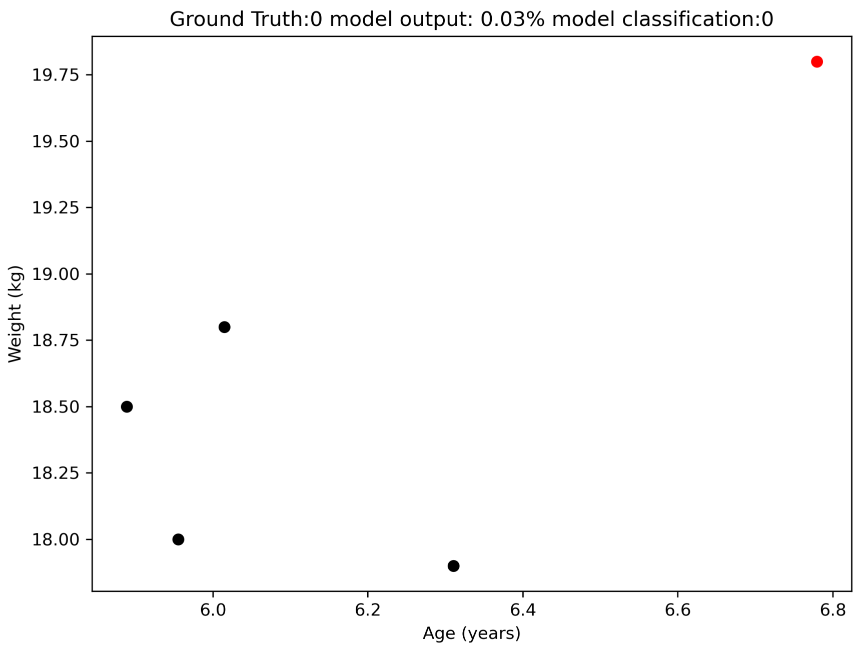


Figure A.63: True Negative Example 3

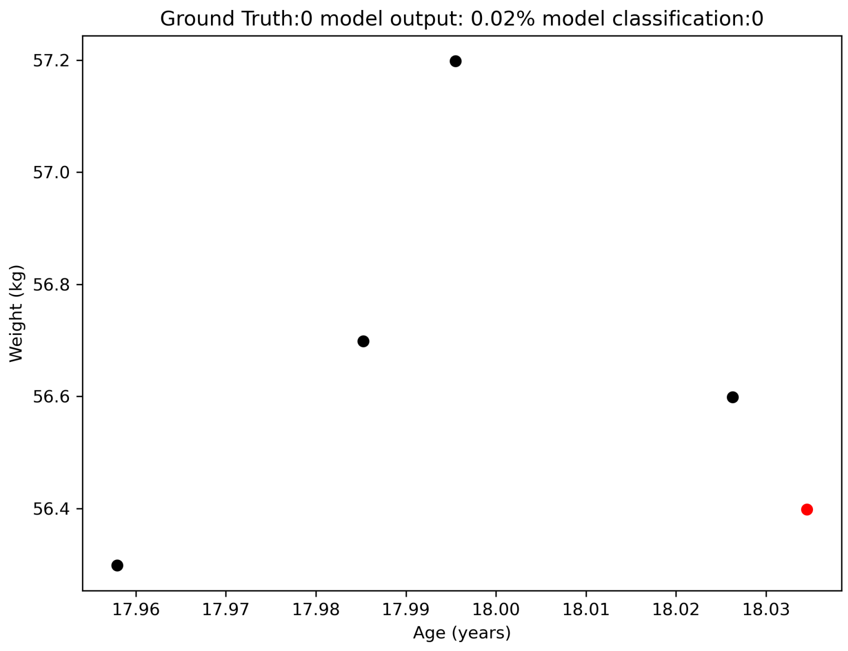


Figure A.64: True Negative Example 4

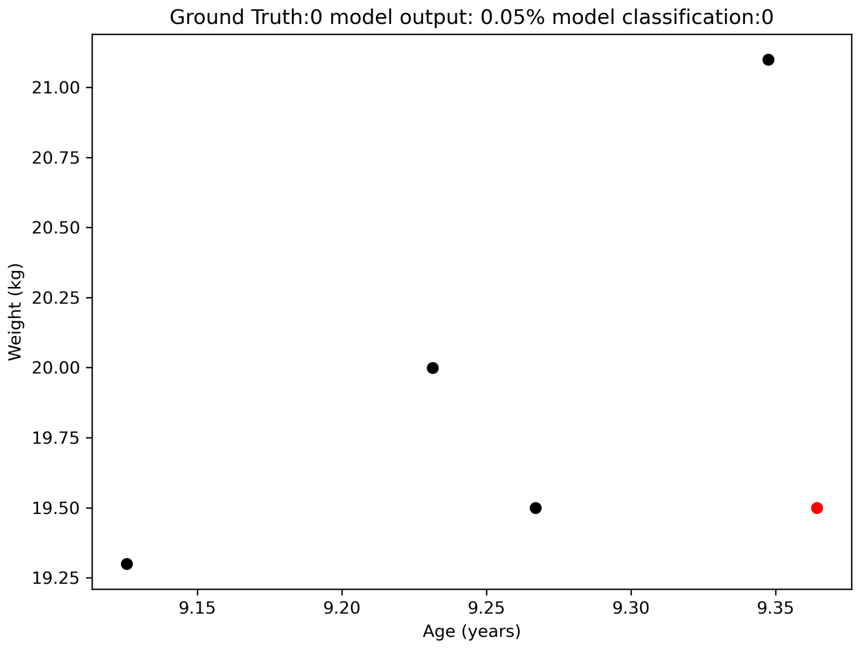


Figure A.65: True Negative Example 5

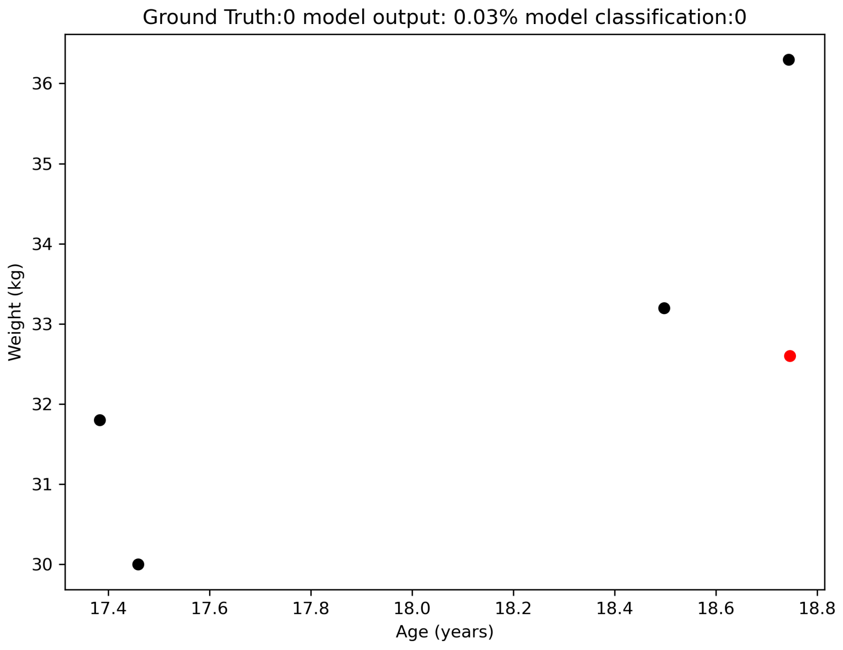


Figure A.66: True Negative Example 6

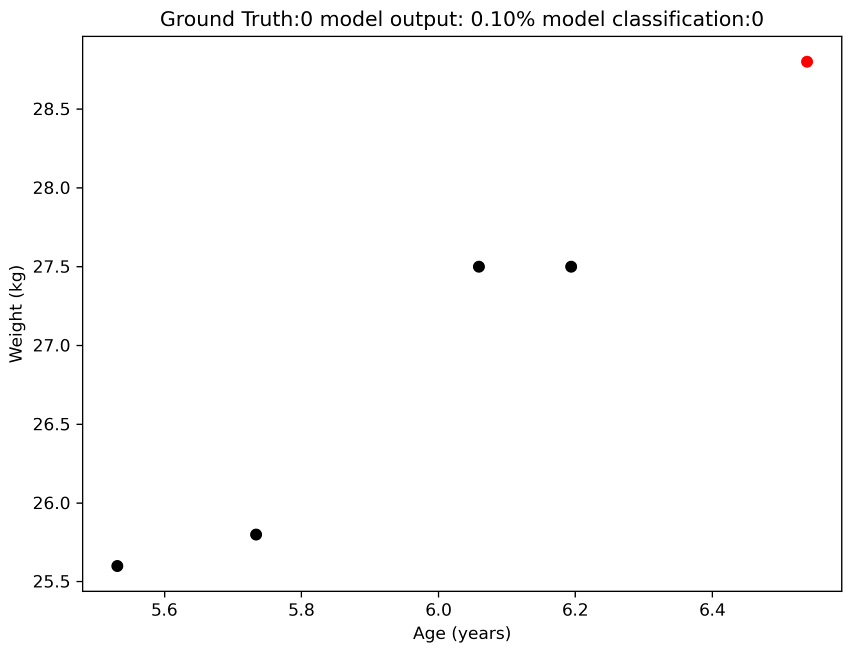


Figure A.67: True Negative Example 7

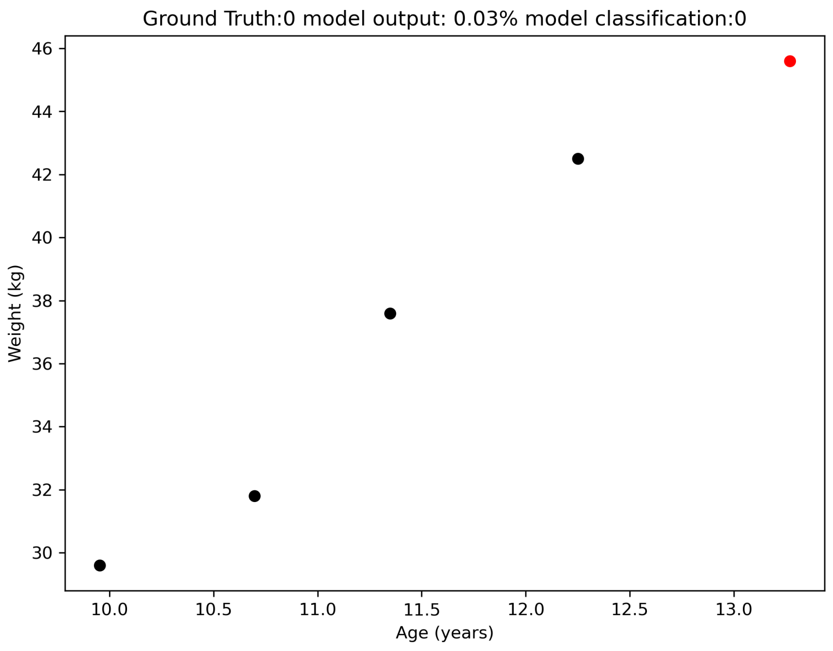


Figure A.68: True Negative Example 8

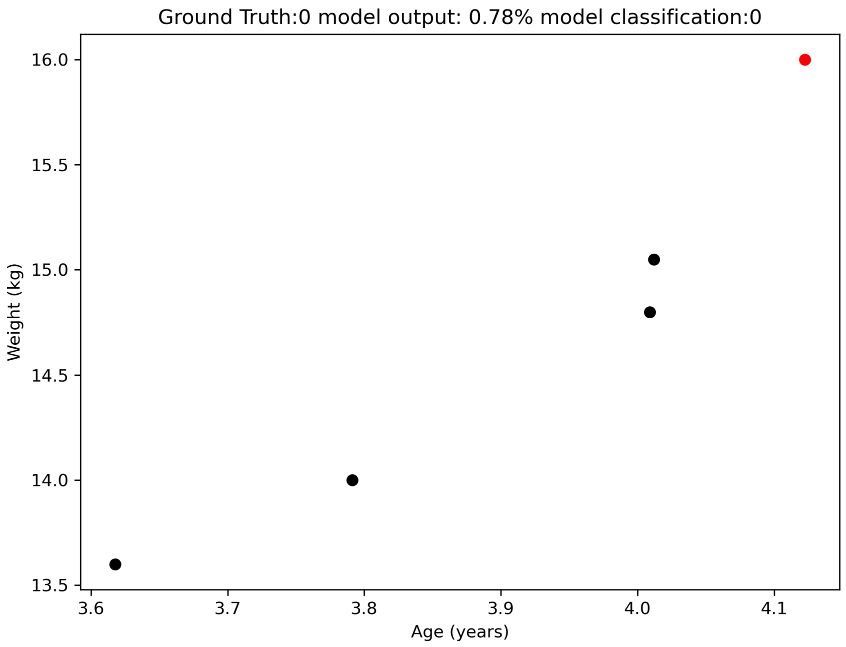


Figure A.69: True Negative Example 9

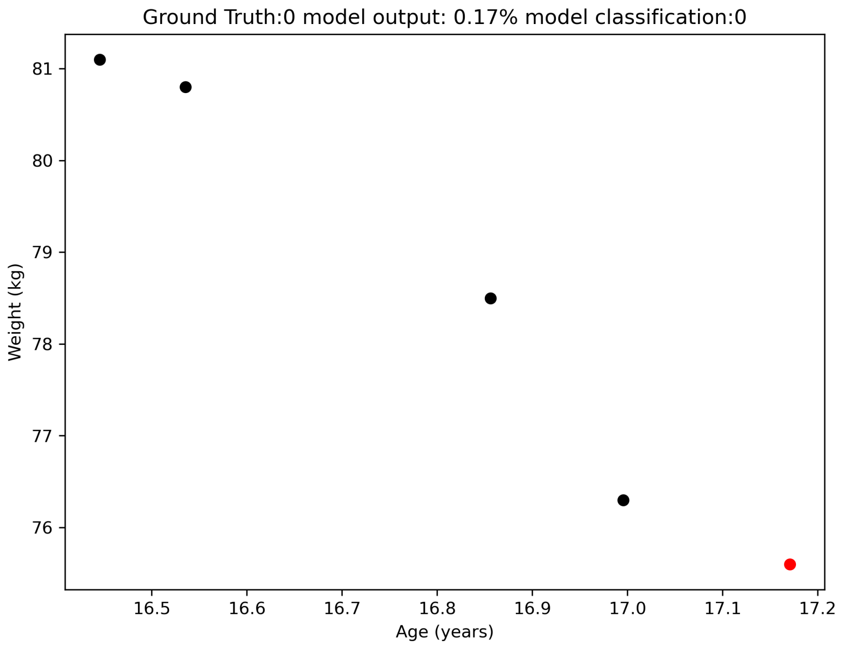


Figure A.70: True Negative Example 10

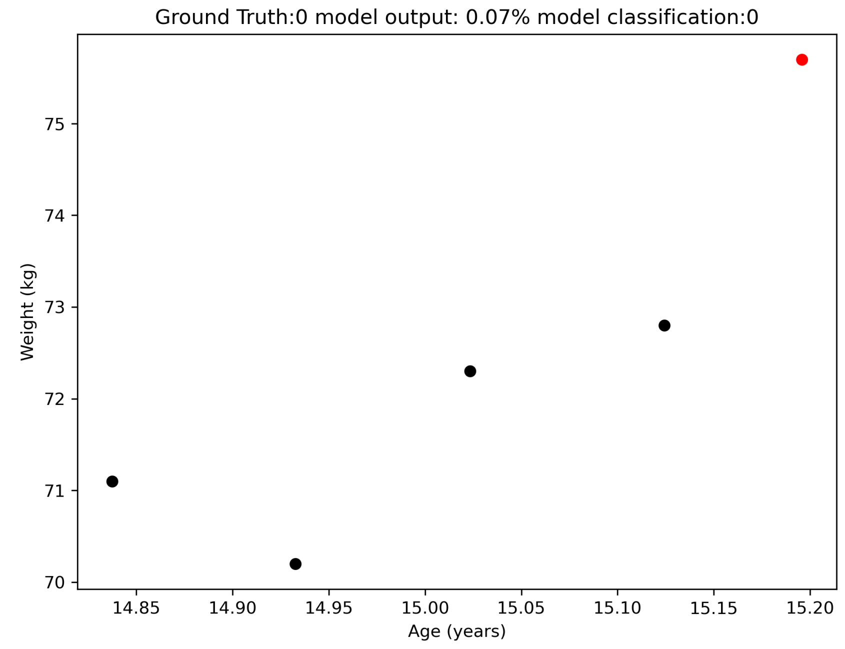


Figure A.71: True Negative Example 11

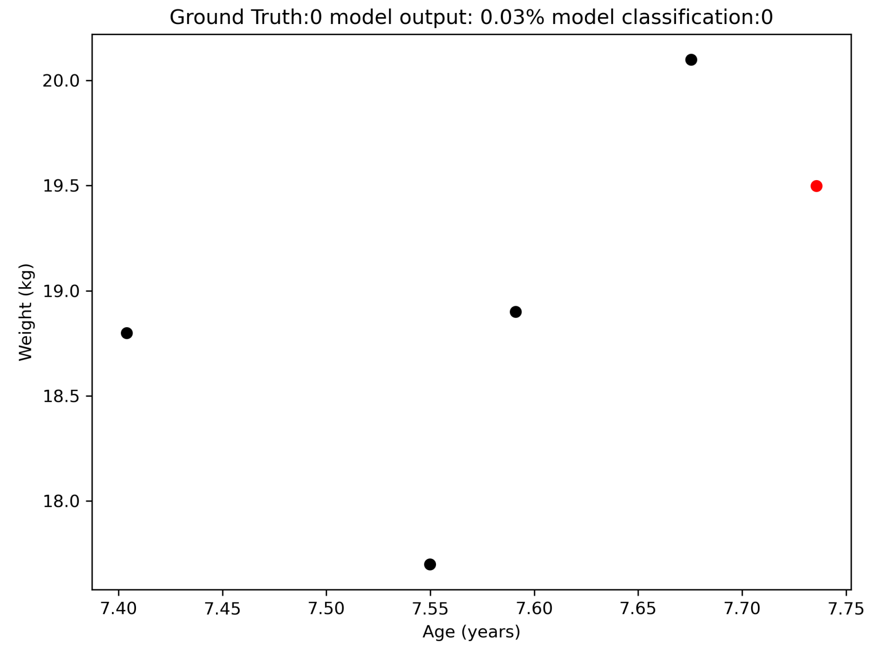


Figure A.72: True Negative Example 12

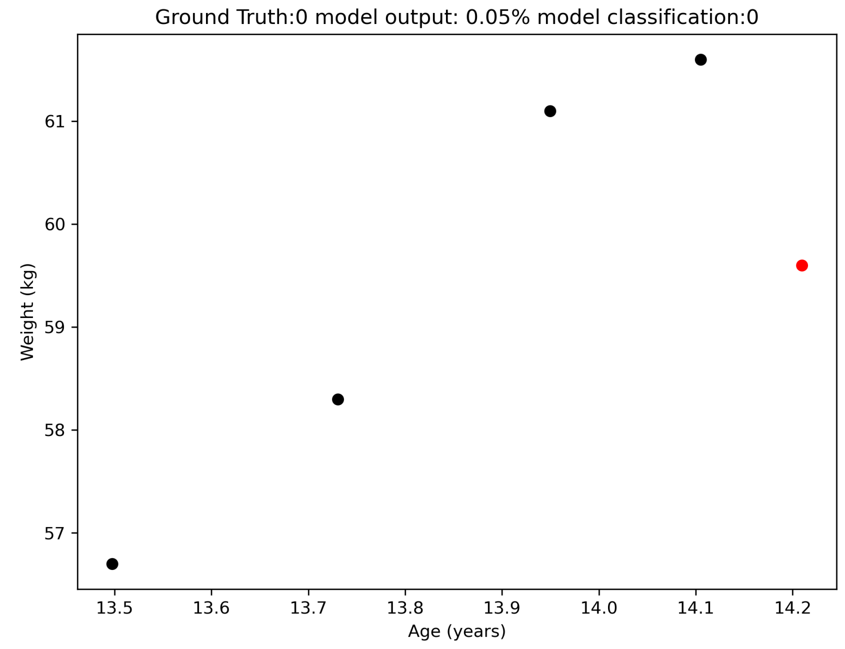


Figure A.73: True Negative Example 13

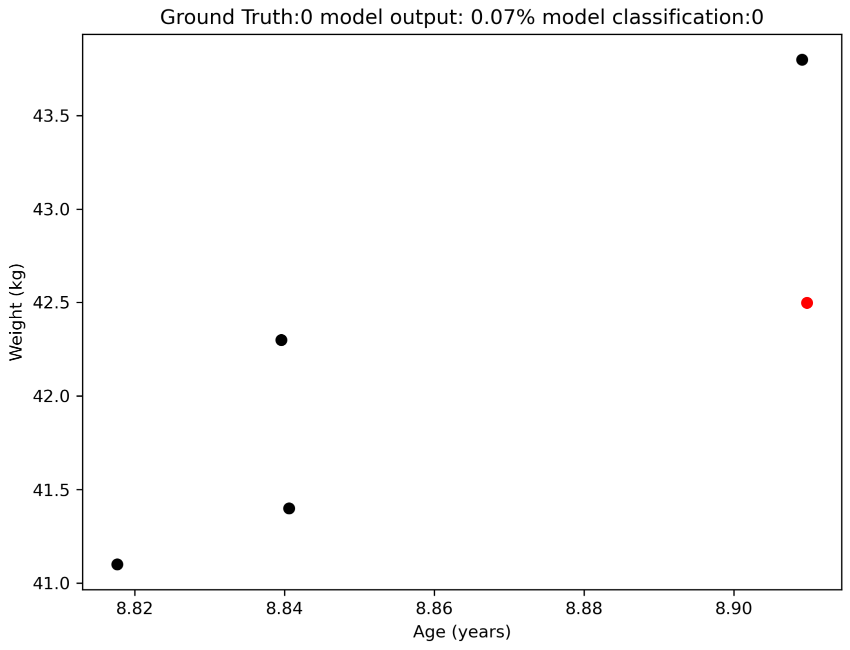


Figure A.74: True Negative Example 14

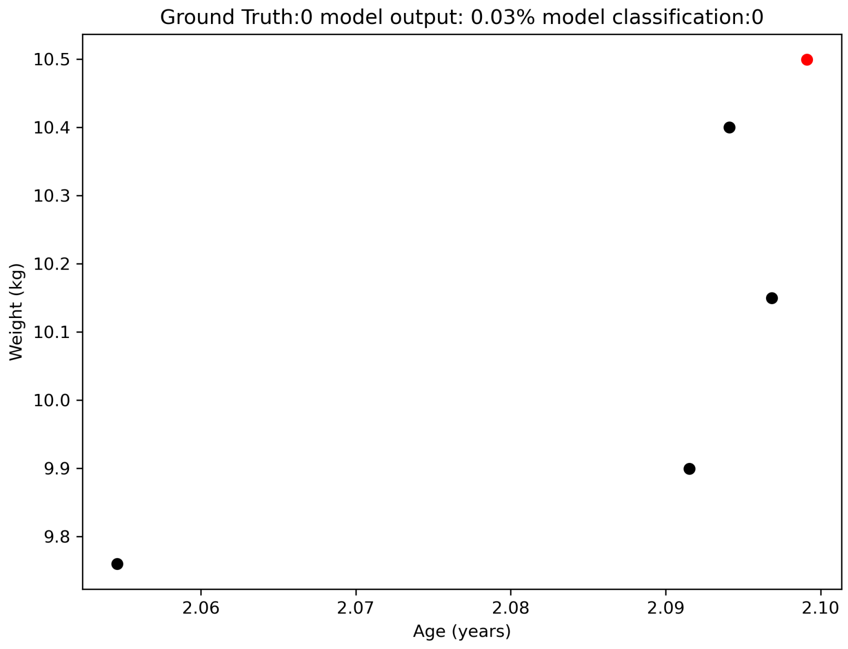


Figure A.75: True Negative Example 15

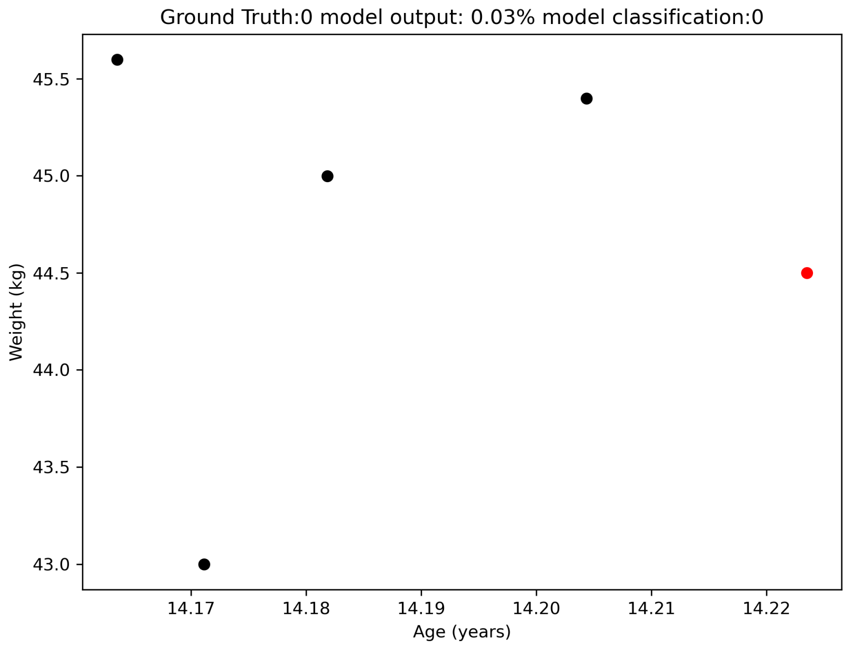


Figure A.76: True Negative Example 16

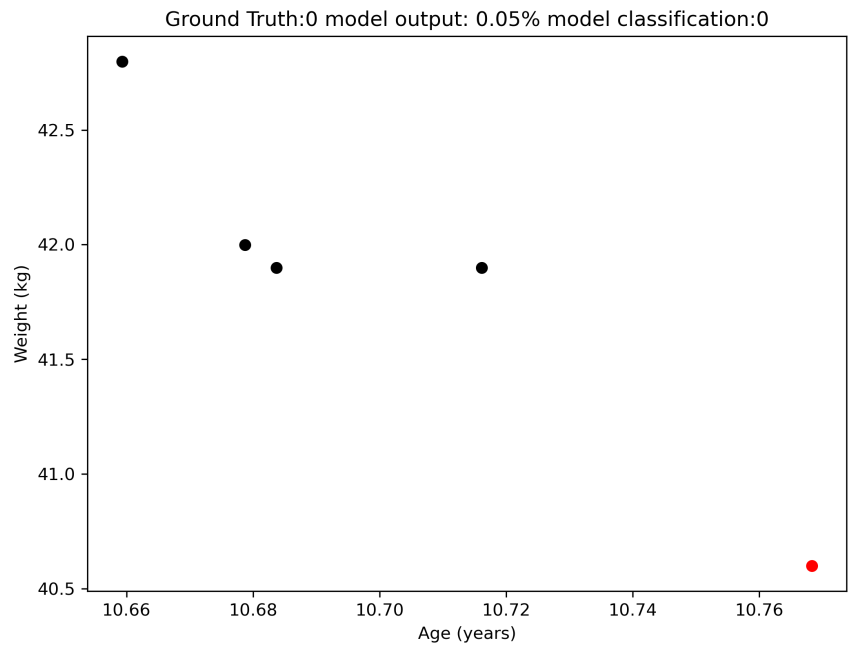


Figure A.77: True Negative Example 17

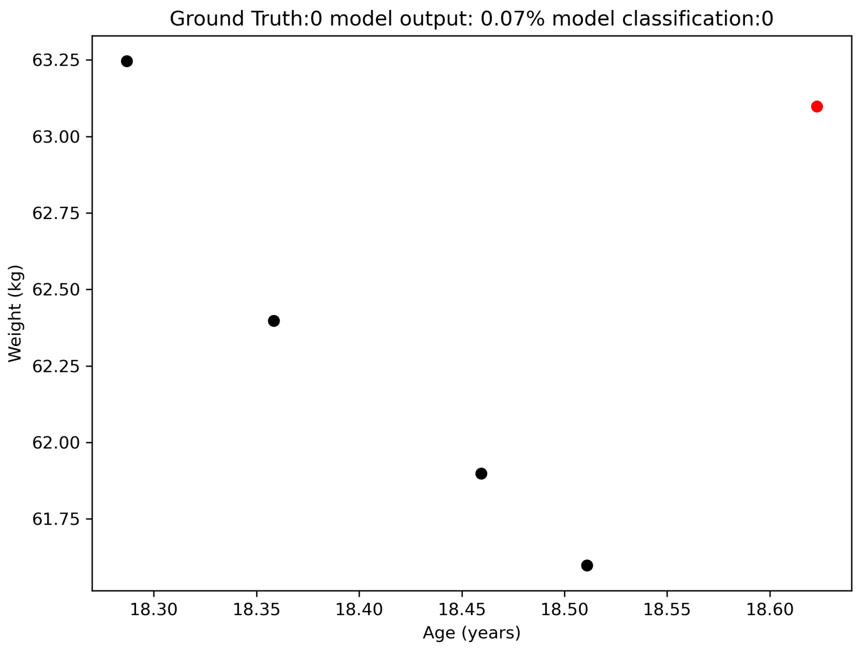


Figure A.78: True Negative Example 18

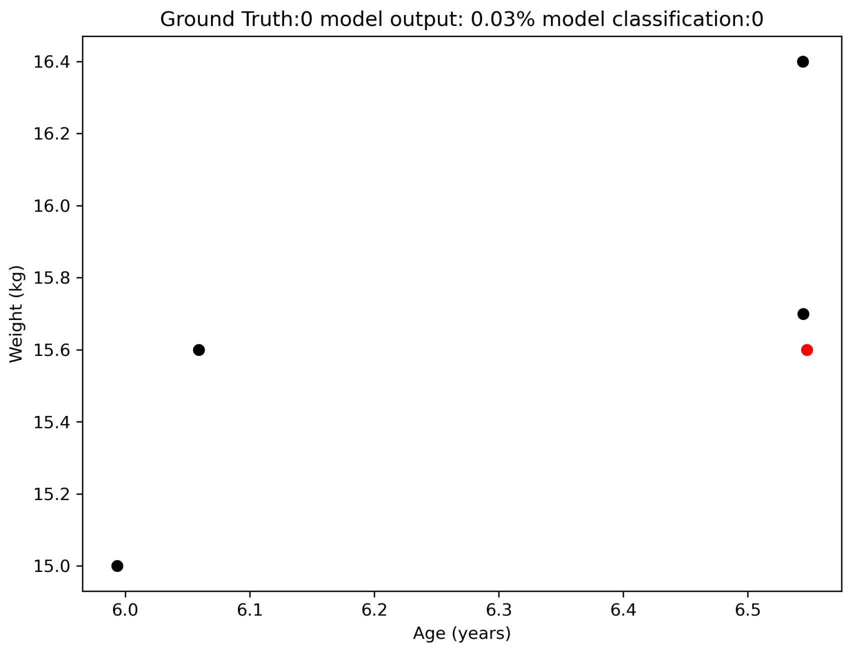


Figure A.79: True Negative Example 19

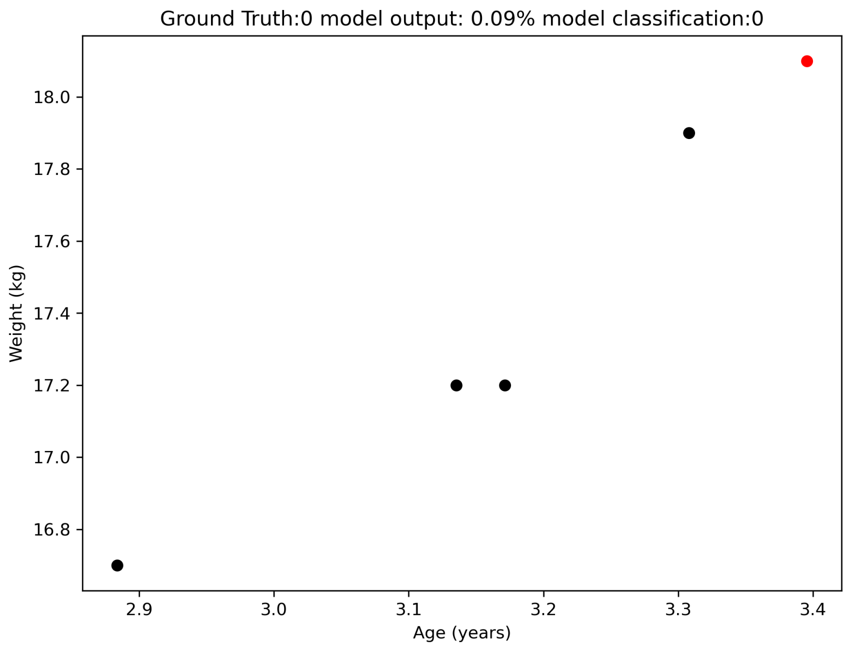


Figure A.80: True Negative Example 20