Imperial College London

Department of Mathematics

Prediction of Donkey Weights using Constrained Models and Explainable AI

Chetali Agarwal

CID: 01234567

Supervised by Professor Jonathan Rougier

7 September 2023

Submitted in partial fulfilment of the requirements for the MSc in Statistics of Imperial College London

The work contained in this thesis is my own work unless otherwise stated.

Signed: CHETALI AGARWAL Date: 7th September, 2023

Abstract

The objective of the thesis is to devise a flexible statistical tool for estimating weight of a donkey on a field without assuming any functional relationship for each predictor variable. In order to constrain the model due to monotonicity in its predictor variables, shape constrained additive models (SCAM) with monotonically increasing splines which models the smoothness in the data is used to model the data. Using the SCAM, an algorithm implementing the construction framework of parallel scale nomogram is used to build a easy-to-use visualization tool. In contrast, the XGBoost model, which is a black-box model, is assumed to provide better predictive accuracy as compared to the traditional nomogram method since it assumes interactions. Nevertheless, it has been discovered that the nomogram performs better in terms of predictive accuracy on the holdout set as compared to XGBoost. The predictions of the black-box model are further decomposed using model-agnostic Explainable AI (XAI) algorithms i.e. LIME and SHAP which offers interpretability to understand feature importance on an individual instance level as well as on a global level. The XAI algorithms allow to break down the model in terms of different features and understand interactions between the term. The visualizations provide key insights to why black-box models which focus on interaction effects do not perform as well as the SCAM based nomogram which focuses on smoothness.

Acknowledgements

I would like to thank Prof. Jonathan Rougier for his guidance and constructive feedback during the completion of this work.

Contents

Contents

List of Figures

List of Figures

List of Tables

1. Introduction

In this work we are concerned with improving the original analysis “How to weigh a donkey in the Kenyan countryside” by Milner and Rougier (2014) of calculating donkey weights in the light of technological advancements in statistical modelling and artificial intelligence explainability in the recent years.

Donkeys are really esteemed assets most in the Kenyan countryside and they provide critical services such as transporting of products around, ferrying individuals and also engage in ploughing. In the account of a donkeys illness, it is imperative to administer an appropriate treatment of drug dosage which is significantly contingent upon the animals weight. Assessing donkey weights is more complicated on the field than it sounds.

To simplify the problem of donkey weight prediction while accounting the unknown pattern in the data, we construct a parallel scale nomogram to calculate donkey weights using an extension of generalized additive models and compare its predictive accuracy with that of a black box model, XGBoost. The rationale for constructing a nomogram is to provide veterinarians with an easily accessible and reliable statistical instrument to weigh donkeys in the field which can otherwise be a difficult and cumbersome task. The working hypothesis is that there is an accuracy-interpretabilty tradeoff between the two unique approaches. Shape constrained additive model (SCAM) is assumed to have higher interpretability as it can be visualized using a nomogram but lower accuracy as the SCAM visualized into a nomogram does not assume interactions between continuous variables and weight is a smooth function of measurable features like girth and length. XGBoost is known for outperforming many machine learning algorithms and is considered to have higher predictive ability than SCAM. A comparison is made between the above-mentioned algorithms to verify the hypothesis and extract insights about donkeys. Ultimately, the predictions of the black-box model are deconstructed using explainable AI algorithms, namely LIME and SHAP, to provide interpretability of XGBoost. The relevant literature has been introduced at the appropriate point in each chapter.

1.1. Background

The paper Milner and Rougier (2014) aimed to construct a nomogram to weigh donkeys using a graphical instrument in order to provide the right quantity of drug dosage to them. It aims to improve existing linear regression which is defined in equation (1.1) :

$$\underbrace{a + b\cdot \log(\text{Girth})}_{f(\text{Girth})} + \underbrace{c\cdot \log(\text{Length})}_{g(\text{Length})} = h(\text{Weight}) \tag{1.1}$$

1. Introduction

which was used to model the weight of donkeys based on different body measurements. The analysis is based on data collected in 2010 by Kate Milner, an activity funded by "The Donkey Sanctuary", a UK registered charity. The total population of donkeys in Kenya in 2010 is estimated to be 1.8 million. The data is based on accessible measurements of 544 donkeys selected over 17 different regions spanning over Naivasha district where they were used to pull carts and in Yatta district as pack donkeys. Donkeys that were either pregnant or had a visible disease were excluded from the sample to avoid bias. It transforms the linear model into a nomogram, a diagram with multiple scales representing variables where the value of the variable of interest can be calculated by drawing a straight line between dependent variables measured on 3 parallel scales. For simplicity purposes, Type 1 parallel scale nomograms were used for ease of understanding. To measure the weight, 4 body measurements were recorded for each donkey: liveweight (kg), heart girth (cm), height (cm), and length (cm) along with other discrete factors such as age and sex. A ordered factor "Body Condition Score" was calculated – a scale running from 1 (emaciated) through 3 (healthy) to 5 (obese) as mentioned in original paper Milner and Rougier (2014).

1. Introduction

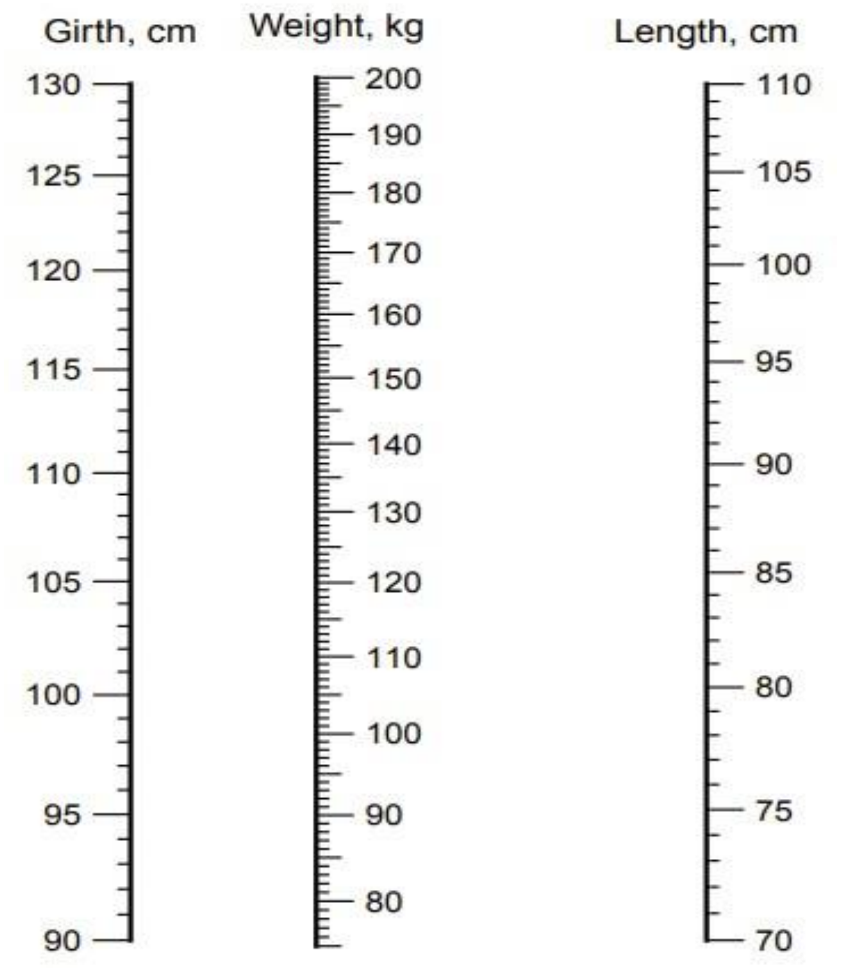


Figure 1.1.: Nomogram for Kenyan donkeys with BCS = 3 and *Age* ≥ 5. To predict weight, join the girth and the length values with a straight line - Placed here for reference, for comparison with new nomogram figure 3.2 on page 26

Factor			
BCS		Age	
1.5	-10	<2	-8
2	-6	2-5	-4
2.5	-5	>5	none
3	none		

3.5	+6		
4	+14		

Table 1.1.: Additive adjustments for factors at non-reference levels, in kilograms for old nomogram

2. Generalized Additive Models

The goal of this work is to fit a flexible statistical model such that it will allow veterinarians to estimate the weight of donkeys based on measurements that are easy to take in the field, including when the donkey is sick and lying down. Accurate estimation of donkey weights is very crucial to calculating the drug dosage which is required to treat animals on the field with pinpoint precision. For this purpose, we choose generalized additive models that model any linear or any non-linear patterns if present in the data using splines. To explain the model and its implementation in detail, the chapter is explained in the following sections:

1. Introduction to GAM - What are GAMs and why are they used? Estimation procedure for various parameters in GAM
2. Impact of Spline Interpolation on GAM - A brief introduction into B splines and monotonically constrained SCOP splines
3. Exploratory Data Analysis - Data exploration to understand correlation and the relationship between multiple variables.
4. Model Development using GAM - Training the data using shape constrained additive models and evaluating its predictive accuracy.

2.1. Introduction to GAM

Generalized Additive Models are considered an extension to generalized linear models as it allows for flexible non-linearities in several predictor variables but retains the additive nature of linear models. Unlike linear models, in generalized additive models, the model is not restricted to terms strictly linear in its parameters but also includes smooth arbitrary effects i.e. GAM is a sum of smooth functions, also referred to as splines as shown by below Equation. 2.1 as proposed in (Wood, 2017, See Equation 4.1)

$$g(\mu_i) = \mathbf{X}^*_i \theta + f_1(x_{1i}) + f_2(x_{2i}) + \cdots + f_p(x_{pi}) + \epsilon_i \tag{2.1}$$

Here, $\mu_i \equiv \mathrm{E}(Y_i)$ and Y_i is a response variable and follows $\overset{\text{an}}{\text{a}}$ exponential family distribution, $Y_i \sim EF(\mu_i, \phi)$ with mean μ_i and scale parameter ϕ with $\mathbf{X}^*_i$ is a row for observation i representing the parametric model components, θ is the corresponding parametric vector with f_j are the corresponding smooth functions for covariates x_k as explained in Wood (2017, see Section 4.1). GAMs can be represented using basis expansions, with each smooth having its associated penalty controlling the wiggliness of the spline. A spline is a function which is a sum of simpler basis functions with its respective coefficient where the complete set of basis function is called basis and the process of its expansion is referred to as basis expansion. The basis functions in GAM can either be for a single feature or a combination of features. GAM's aim is to estimate the coefficient of the basis functions for each spline as defined in equation (2.2). In equation (2.2), $b_k(x)$ is the kth such basis function and β_k is the corresponding spline coefficient.

$$f(x) = \sum_{k=1}^{K} \beta_k b_k(x) \tag{2.2}$$

of the

Polynomials are defined for the whole range of the predictor variable and on the other hand, GAMs are defined for a limited range of the covariate and hence they model the data better. This is due to a term "Runge Phenomenon" which is a by-product of polynomial interpolation using high degree over a set of equi-spaced interpolation points. As data complexity or non-linearity increases, a simultaneous increase of degree of polynomials leads to improving performance. However, the model wavers significantly at the boundary points leading to inaccurate fits. GAMs are a better model choice since it fits on a limited range of the covariate and hence adapts to the data more accurately as mentioned in Wik (2013).

To

GAMs can model ~~the noise~~ and the underlying pattern in the data which might lead to overfitting. In order to avoid it, GAMs use penalized regression splines where the excessive wiggliness ~~of the spline~~ due to a large number of knots is penalized. The loss function controls the model's smoothness by adding a "wiggliness" penalty to the least squares loss as mentioned in (Wood, 2017, See Section 4.2). It is fitted by minimizing the below equation (2.3) as defined in a Lecture series of the Heap:

$$L = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \int_0^1 [f''(x)]^2 dx \tag{2.3}$$

or maximizing equation (2.4):

$$\mathbf{L}_p(\beta) = \mathbf{L}(\beta) - \lambda(\beta)^T S(\beta) \tag{2.4}$$

where $\mathbf{L}_p(\beta)$ is the penalized log-likelihood and $\mathbf{L}(\beta)$ is the log-likelihood, the measure of how well we fit the data given our estimates of the parameters for the basis functions and λ refers to the amount of penalty that is applied for the given level of wiggliness as given in the below equation (2.5). As proposed in (Wood, 2017, See Secion 4.2.2), the smoothing parameter λ controls the trade-off between model fit and its measure of wiggliness or the model complexity $\mathbf{W}$ as mentioned in (Wood, 2017, See Secion 4.2.2). The measure of wiggliness is defined in Equation (2.5):

$$\int_0^1 [f''(x)]^2 dx = \boldsymbol{\beta}^T \mathbf{S} \boldsymbol{\beta} = \mathbf{W} \tag{2.5}$$

Here, S is a penalty matrix which is created from basis functions which when pre and post multiplied with β results in **W**. When $\lambda = 0$, we are essentially fitting an unpenalized model, on the contrary if $\lambda \to \infty$, we pay an increasing cost of wiggliness and hence the model will tend towards a linear model as mentioned in (Wood, 2017, See Secion 4.2.2). Estimation of the coefficients of parametric and non-parametric components is by fixing λ first. Linear models or Generalized Linear Models fit unpenalized models and GAM's allow to penalize the complexity of the model and choose the value of λ. The estimation of the smoothness parameter λ has 2 different approaches i.e Predictive which aims to minimize out of sample error or Bayesian which puts priors on the basis coefficients. Predictive approach mainly uses the default Generalized Cross Validation (GCV) but there also exists AIC and Mallow's C_p. Bayesian approach has Maximum Likelihood and Restricted Maximum Likelihood as the two methods. However, for better stability, predictive approaches using GCV are less preferred because it can lead to under-smooth or over-smooth because it can lead to over-flattening over a wide range of λ as mentioned in (Wood, 2017, See Section 4.5) and additional information provided in Lecture series of the Heap. On the other side, REML leads to lower variation in the estimation of λ. The vector of unknown coefficients that minimizes equation (2.3) is given as follows in equation (2.6).

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{S})^{-1}\mathbf{X}^T\mathbf{y} \tag{2.6}$$

Estimating the basis complexity or the number of basis functions **k** which is the maximum wiggliness that is allowed for each smooth i.e., number of small functions that compose of a curve. Post the smoothing, the penalty shrinks the size of k, leading to effective degrees of freedom i.e., $EDF < k$

i.e.,

2.2. Impact of Spline Interpolation on GAM

Under numerical analysis, a *spline* is a continuous function which is a smooth piece-wise polynomial function. For a large dataset, interpolating values was primarily done by high order interpolation i.e fitting n order polynomial to n+1 data points as mentioned in (Childress, 2023). For large datasets, high degree global polynomial interpolation is not preferred due to **Runge Phenomenon** that is oscillatory behaviour towards the boundary points and computational complexity. Runge phenomenon occurs because of the twin requirements of interpolation and continuity i.e. non-zero derivatives at all data points. In such cases, spline interpolation is preferred and is also more suitable than linear interpolation because splines require the function to be continuous for all data points, but linear interpolation leads to non-smooth functions i.e. discontinuous derivatives at certain data points as mentioned in (McClarren, 2017, See Section 2.3). Cubic splines is a smooth curve that joins multiple cubic piece-wise polynomials between adjacent points such that the spline connects and is continuous at each data point i.e. there are no sharp changes in

are

the slope between adjacent data points. In the sections below, we consider a penalized cubic-B spline for fitting the increasingly monotonically constrained splines as a cubic B spline is easily differentiable.

2.2.1. B Splines

Cubic splines is a smooth curve that joins multiple cubic piece-wise polynomials between adjacent points such that the spline connects and is continuous at each data point i.e. there are no sharp changes in the slope between adjacent data points. The points at which the sections of cubic polynomials are joined together are referred to as *knots* of the spline. B Splines is an alternative way used for curve fitting and for data compression and is preferred over natural cubic splines because the basis functions are strictly local since each basis function takes non-zero values only between m + 3 adjacent knots where m + 1 is the order of the basis as mentioned in (Wood, 2017, See Section 5.3.3). Here k are the number of basis functions, with $\{P_1, P_2, \cdots, P_k\}$ representing the control points and number of knots is p which needs to satisfy $p = k + m + 2$ and be supplied such that the knot vector is defined as $\{u_1 < u_2 < u_2 < \cdots < u_p\}$, the spline is evaluated over $[x_{m+2}, x_{k+1}]$ as mentioned in (Wood, 2017, See Section 5.3.3). The below Figure 2.1 is taken from Rougier and Duncan (2023, See Figure 1)

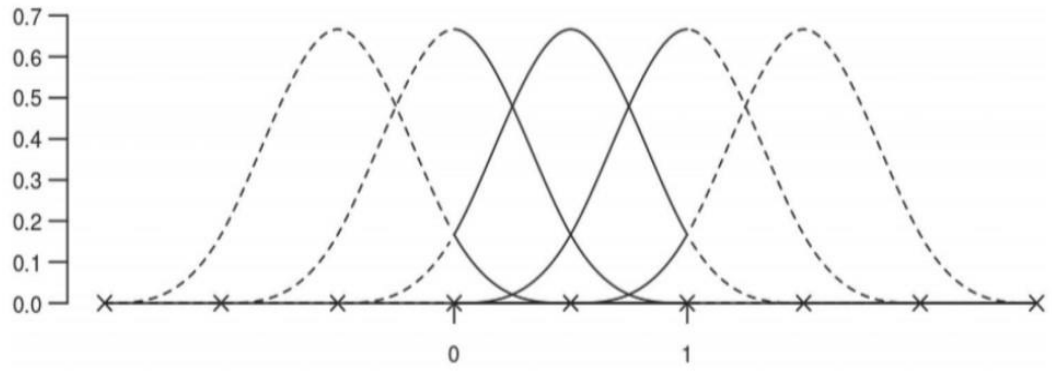


Figure 2.1.: Cubic B splines with number of basis dimensions is 5 and equally spaced 9 knots. The 5 basis functions are shown by the sold line in [0,1]. For most points where $x \in [0,1]$, there are 4 non-zero basis functions but at the knot locations, there are 3.

For a cubic B spline with k = 5, number of knots **p** = 9. B spline curve of degree (m+1) is defined as the following:

$$\mathbf{f}(u) = \sum_{i=1}^{k} B_{i,m}(u)\mathbf{P}_i$$

Here, $B_i^m(u)$ are the B splines basis functions of degree m and is defined recursively using the below Cox-De Boor equations in (Wood, 2017, See Section 5.3.3).

$$B_i^m(u) = \frac{u-u_i}{u_{i+m+1}-u_i}B_i^{m-1}(u) + \frac{u_{i+m+2}-u}{u_{i+m+2}-x_{i+1}}B_{i+1}^{m-1}(u)$$

and

$$B_i^{-1}(u) = \begin{cases} 1 & , u_i \le u < u_{i+1} \\ 0 & , \text{otherwise} \end{cases}$$

2.2.2. SCOP Splines and SCAM

The key feature of a *shape constrained P spline* i.e. SCOP as compared to unconstrained GAM's are the unknown shape constrained smooth functions for predictor variables where the model imposes monotonicity by coefficient constraints. Typically, **P splines** are penalized B splines where the model coefficients are determined using the likelihood and an additional penalty term used to constrain the wiggliness achieved using the model fit. A monotonically increasing smooth function is proposed in Pya and Wood (2015, see Section 2.2) as:

$$m(x) = \sum_{j=1}^{q} \gamma_j B_j(x) \tag{2.7}$$

Here, q refers to the basis dimension, B_j are at-least second order B spline basis over [a,b] based on uniform knots with corresponding γ_j spline coefficients. As defined in Rougier and Duncan (2023, see Section 3.2 Equation 17):

$$m'(x,\gamma) = 3\sum_{j=1}^{q+1} \frac{\gamma_j - \gamma_{j-1}}{x^{(j+3)} - x^{(j)}} B_j(x) \quad ; \quad \gamma_0 = \gamma_{q+1} = 0 \tag{2.8}$$

add space

Sufficient condition for $m'(x,\gamma) \ge 0$ is $\gamma_j \ge \gamma_{j-1}$ because $B_j(x) \ge 0$ and the knots are in increasing order. As in Pya and Wood (2015, see Section 4.2.2), let γ_0 be arbitrary. We let $\beta_j = \gamma_j - \gamma_{j-1}$ and constraint β_j to be strictly positive by exponentiating it i.e. $\beta_j = exp(c_j)$ where c_j is just a constant. In Pya and Wood (2015),we set β_1 to be an arbitrary value. Using re-parameterizing as mentioned in Pya and Wood (2015, see Section 4.2.2) such that $\gamma = \mathbf{\Sigma}\beta_e$ where $\beta = \{c_1,c_2,c_3,\cdots,c_k\}$ and $\beta_e = \{\gamma_1,e^{c_2},e^{c_3},\cdots,e^{c_k}\}^T$ with

$$\Sigma_{ij} = \begin{cases} 0 & , \text{if } i<j \\ 1, & \text{if } i \ge j. \end{cases}$$

which in matrix representation is

$$\underbrace{\begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_q \end{bmatrix}}_{\gamma} = \underbrace{\begin{bmatrix} 1 & 0 & \cdots & & & \\ 1 & 1 & 0 & \cdots & & \\ 1 & 1 & 1 & 0 & \cdots & \\ \vdots & \vdots & \vdots & \ddots & & \\ 1 & 1 & \cdots & & & \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_q \end{bmatrix}}_{\beta_e} \tag{2.9}$$

and $\mathbf{m} = [m(x_1), m(x_2), \ldots, m(x_n)]^T$ is a vector of values at x_i observed points and $\mathbf{X}$ is a matrix where $X_{ij} = B_j(x_i)$ and $\mathbf{m} = \mathbf{X\Sigma}\beta_e$. These derivations have been mentioned in Pya and Wood (2015).

In this work, we construct a parallel scale nomogram, and focus on one-dimensional cases and exclude multidimensional SCOP splines. Hence, the aim is to minimize equation (2.10) where λ is the measure of wiggliness.

$$\left\|\mathbf{y} - \mathbf{X\Sigma}\widetilde{\beta}\right\| + \lambda \left\|\mathbf{D}\beta\right\|^2 \tag{2.10}$$

In monotonic splines, we penalize m(x) to control wiggliness of the function. For $j > 2$, β_j is the log difference in γ_j, and as proposed in the paper "Shape Constrained Additive Models" Pya and Wood (2015), it penalizes the squared differences between adjacent β_j starting from β_2. It uses the penalty, $\|\mathbf{D}\beta\|^2$ where D is a $(q-2) * q$ matrix with the following values: $D_{i,i+1}, -D_{i,i+2} = 1$ and 0 otherwise where $i = 1, 2, \cdots q-2$.

$$D\beta = \underbrace{\begin{bmatrix} 0 & 1 & -1 & 0 & 0 & \cdots & \cdots \\ \vdots & 0 & 1 & -1 & 0 & \cdots & \cdots \\ \vdots & \vdots & \ddots & 1 & -1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots \\ \cdots & \cdots & \cdots & \cdots & 1 & -1 & \end{bmatrix}}_{D} \underbrace{\begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_q \end{bmatrix}}_{\beta} \tag{2.11}$$

As mentioned in Pya and Wood (2015), "The penalty is zero when all the β_j after β_1 are equal, so that the γ_j form a uniformly increasing sequence and m(x) is an increasing straight line. As a result our penalty shares with a second order P-spline penalty, the basic feature of 'smoothing towards a straight line', but in manner that is computationally convenient for constrained smoothing. ".

Additionally, we apply centering constraints on m(x) such that $\sum_{i=1}^{n} m(x_i) = 1$ as mentioned in Pya and Wood (2015). Due to this reason, if number of basis dimension supplied is 15, there will be 14 basis functions because of the identifiability constraint applied on the basis. For the outer loop optimization to calculate smoothing parameter values λ_1 and λ_2, BFGS i.e. "Broyden Fletcher Goldfarb Shanno" algorithm is used and "Newton" numerical optimization method is used for calculating model coefficients. Approximation of the number of basis dimensions is chosen by minimizing GCV over a range of q values.

2.3. Exploratory Data Analysis

The dataset used to predict the donkey weight has body measurements of 544 donkeys, involving measurements that include Height, Girth, Length, Age, Gender, BCS and Weight. The original dataset has Age which is factored in 6 different levels : $y < 2$, $2 \leq y < 5$, $5 \leq y < 10$, $10 \leq y < 15$, $15 \leq y < 20$ and $20 \leq y$. Gender is factored into three distinct levels: female, gelding and stallion. In Figure 2.2, we plot the Pearson correlation heat map between different continuous predictor and response variables and notice that there is high degree of correlation between girth and weight 0.903, length and weight at 0.789 and height and girth at 0.704.

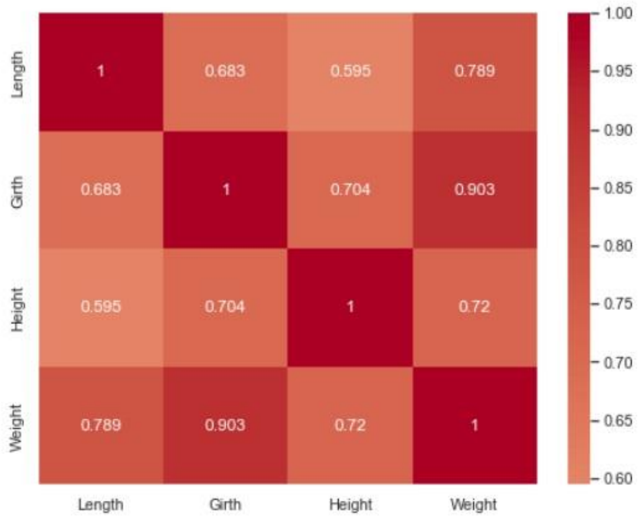


Figure 2.2.: Correlation between continuous variables

Since BCS is an ordinal factor, we consider Kendall Tau's correlation coefficient with continuous variables. There is very little correlation between BCS with the predictor variables and the response variable and hence chose to not include BCS in the data modelling and also because BCS is the only subjective variable which was incorporated into the original dataset.

Basis Position	**Girth**	Height	Length	**Weight**
Kendall's Tau	0.423	0.217	0.317	0.484

In the data, we remove outliers to avoid extreme effects on model predictions. The final count of donkeys used in model fitting is 541. Based on age, highest frequency of observations i.e. 237 in the age group $10 \leq y < 15$ and 97 in $2 \leq y < 5$. In terms of gender, female donkeys account for 46% of the population, followed by stallion at 39.4% and gelding representing 16.6% of total count. The Figure 2.3, shows the almost positive linear relationship between Girth and Height, and Girth and Length with respect to Weight. Higher values of girth with high values of either height or length show a monotonically increasing relationship with the response variable weight.

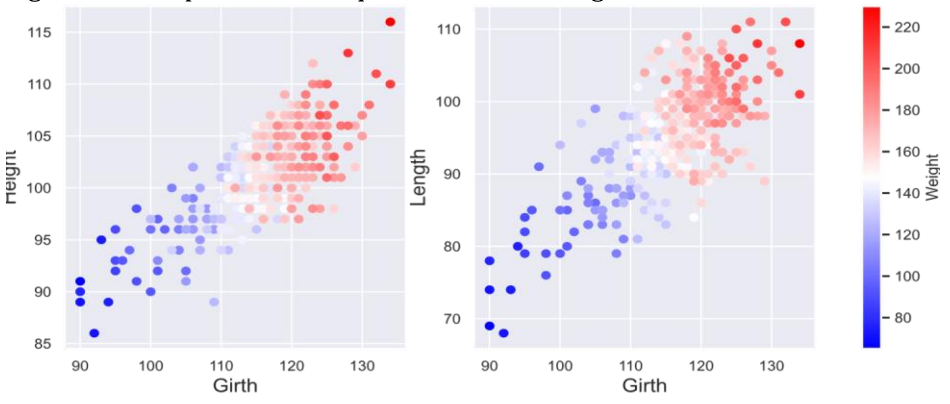


Figure 2.3.: Comparison between 3 main predictors and response variable weight

The boxplots show the distribution of weights of donkeys belonging to different ages and genders. In Figure 2.4, we can notice that there is a significant difference of 55 kgs in the average weight of donkeys belonging to the age group $y < 2$ as compared to $15 \leq y < 20$. Also, it is observed that older donkeys in age group $y \geq 20$ have a decline in weight and are usually weighed between 120 to 180 kilograms and don't vary much. Weight of animals in the adolescent years i.e $10 \leq y < 15$ are shown to more with maximum of nearly 220 which is way higher as compared donkeys in other groups.

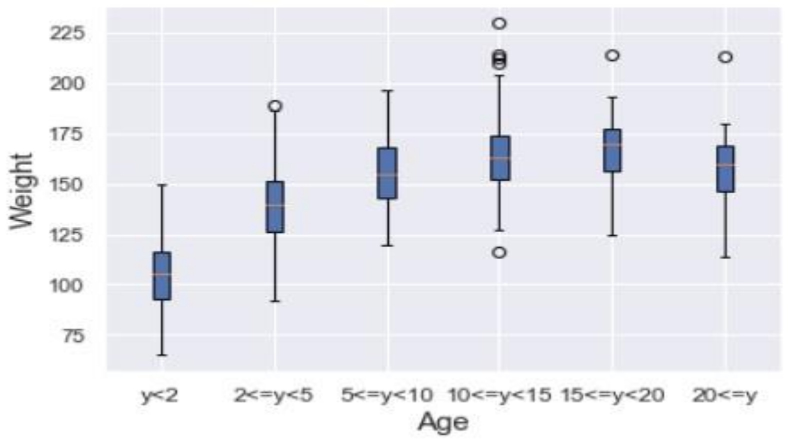


Figure 2.4.: Weight comparison for different age donkeys

In Figure 2.5, there is not much difference between the weight distribution of stallions and females as compared to gelding. The minimum weight for stallion and female is recorded to be around 90 kgs whereas for geldings, it is approximately 125 kgs which implies a stark difference. This might be attributed to lower number of geldings present in the population.

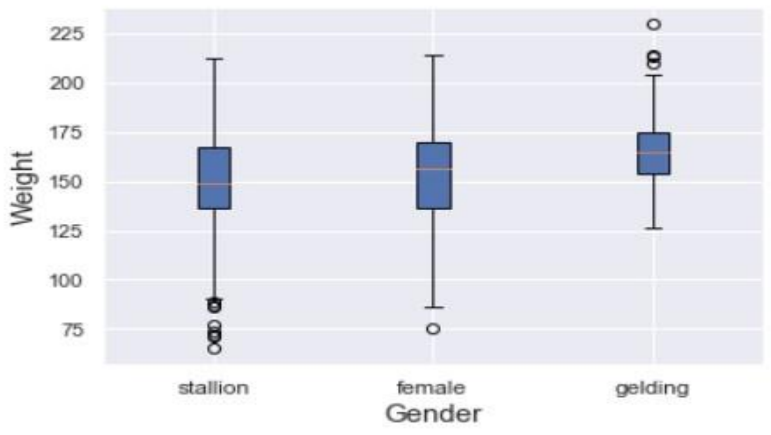


Figure 2.5.: Weight comparison for different gender donkeys

The graph below 2.6, explains the relationships between different continuous variables and age using which we can make key observations such as i) $y < 5$ donkeys are mostly on the lower end of the weight scale and also have low length and low girth and $5 \leq y < 20$ is dominating the higher end. ii) As donkeys get older usually after the age of 20, weight starts decreasing. iii) Donkeys in the age range $5 \leq y < 20$ are characterized by higher girth, length and height. The variables have been standardized to ensure the data is on the same scale.

Parallel Coordinate Plot For Multiple Age Levels

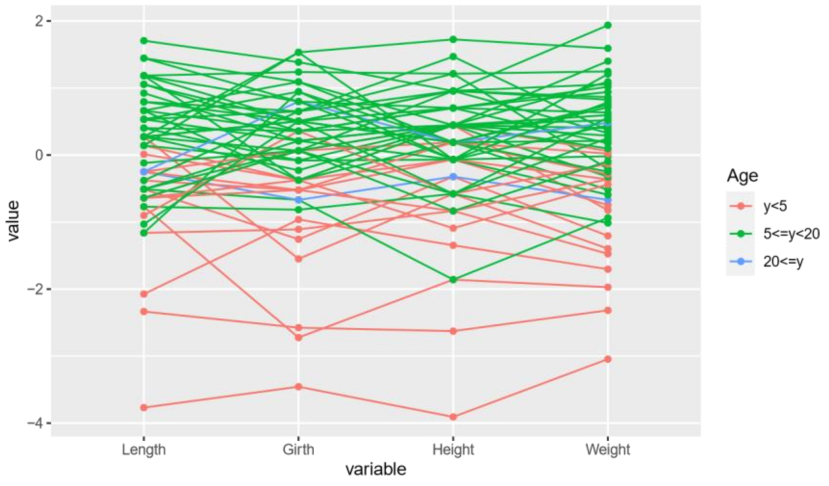


Figure 2.6.: Parallel Coordinates Graph between continuous variables for Gender levels

In Figure 2.7, it is observed that gender does not impact weight strongly. However, on average, geldings are mostly weighted more than the average weight of females and stallions as also verified by Figure 2.5.

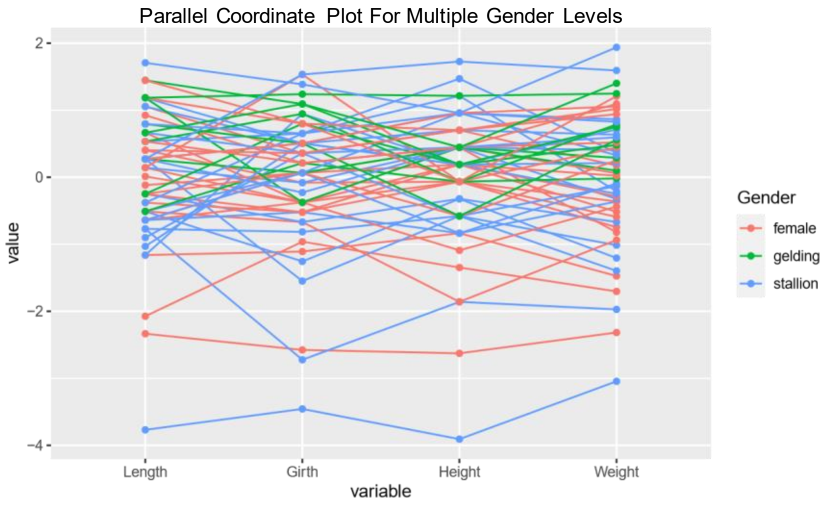


Figure 2.7.: Parallel Coordinates Graph between continuous variables for Age levels

2.4. Model Development

Model development encompasses three stages :

1. Functional Form of Weight - Defining the functional relationship between weight and girth using box-cox transformation

2. Sensitivity Analysis - Checking stability of the model by varying the number of basis dimensions to check for model fitting using the model defined in previous stage based on the 20% validation set

3. Model Refinement - Refactoring of categorical factors to ensure higher interpretability of the model

donkeys have

an

2.4.1. Functional Form of Weight and Model Choice

The model aims to calculate weights ~~of donkeys~~ based on ~~different categorical~~ and continuous features. As we know, donkey ~~has a~~ shape of a elliptical cylinder with appendages. Hence, the accurate choice of continuous predictors will be length and girth. Since weight is equivalent of volume, where girth is similar to radius, and height is equivalent to length. Hence, the variable of interest is calculated as follows:

$$Weight \propto (Girth)^2 Length \tag{2.12}$$

The equation is dimensionally consistent since donkeys can be viewed as an elliptical cylinder of constant volume with small appendages. However, these assumptions are not biologically relevant, and therefore we do not expect a simple relationship between Girth, Length and Weight. For model fitting, we use GAM's as it allows the model to capture the non-linearity in th data independently given the maximum number of basis dimensions. It is noted that there is a monotonically increasing relationship between length and weight as well as girth and weight. For this purpose, we fit shape constrained generalized additive models which puts a monotonically increasing constraint on two continuous predictor variables, length and girth. The shape constrained splines represent the non-parametric components of the model. In the model we add the categorical features as linear predictors which represent the strictly parametric components of the model and is defined in equation (2.13):

$$g(\mu_i = \beta_0 + \sum_{k=1}^{K} m_k(x_{1k}) + \sum_{j=1}^{J} m_j(x_{2j}) + \beta_1 * x_3 + \beta_2 * x_4 \tag{2.13}$$

Here, g is the function based on the box cox transformation of the response variable weight, x_1, x_2,x_3, x_4 represent length, girth, age and gender respectively and m_k, m_j represents the unknown non parametric smooth functions for length and girth respectively. The model is semi parametric because it consists of: i) Extension of linear additive components generalized to non-parametric estimates functions i.e. unknown smooth functions. ii) Parametric additional linear components which reflects the addition of categorical features such as Gender and Age where the latter is an ordered factor.

In order to train the model, we stratify the data based on weight in an ascending order so that the training, validation and test data are representative of the full range of weights. The data is split in a 70:20:10 ratio. The validation data is used for model choice. The testing data is used at the final step to calculated the predictive accuracy of the final model. We perform box-cox transformation Osborne (2010) on the response variable values in order to report the predictive accuracy in terms of standard deviation which is preferred by converting the underlying response variable distribution to normal as it attaches a random quantity to it. The following transformation was performed on weight where $\gamma \in [-2,2]$

Closer

$$y(\gamma) = \begin{cases} \frac{y^{\gamma}-1}{\lambda}, & if \gamma \neq 0 \\ log(y), & if \gamma = 0 \end{cases}$$

From the below graph 2.8, we can see that for $\gamma = 1$, we have the least root mean square on the validation set of 8.373. Although $\gamma = 1$, might not be the absolute minimum, it is significantly close and hence on that basis, it can be chosen for its convenience. The fit was to check on the transformed weight using the box-cox transformation. The root mean square of the prediction on the holdout set was calculated by transforming the transformed weight back to its original scale.

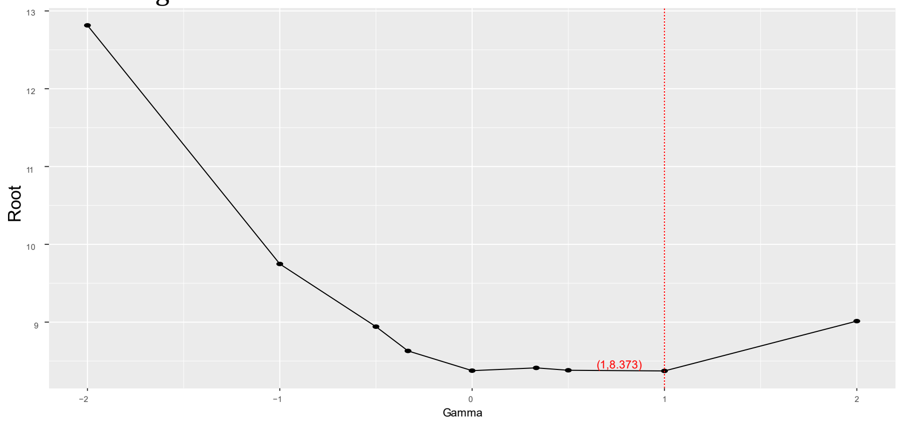


Figure 2.8.: Root MSE Comparison for Gamma values

2.4.2. Sensitivity Analysis

In penalized regression spline, the number of basis dimension is q which needs be significantly large to avoid under-fitting or over-smoothing of the data. The value of q is the maximum complexity that the model can have and is crucial to capture the non-linearity in the data. After verifying the known smooth monotonic link function G using box cox transformation on the response variable where $\gamma = 1$, it is an identity link function. To perform robustness of the model, we perform sensitivity analysis using the model in the previous section on 70% training set where the number of basis dimensions is varied between 5 to 10. It is noted that when q = 7, least root mean square error was calculated of 8.32 on the 20% validation set as displayed in Figure 2.9

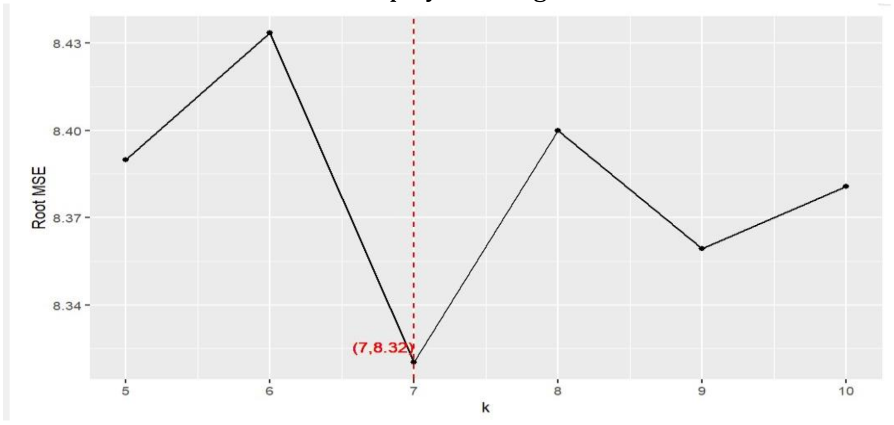


Figure 2.9.: Root MSE Comparison for different basis dimension

Based on the model chosen above, we model the data using monotonically increasing splines with identity link functions using 7 basis dimensions.

2.4.3. Model Refinement

In Table 2.1, the coefficient estimates of the parametric components are given with corresponding p values.

Parametric Coefficients	Estimate	Standard Error	t Value	P Value
(Intercept)	16.7827	3.7768	4.444	1.17e-05 ***
Age : 2<=y<5	0.5193	2.1502	0.241	0.8093
Age : 5<=y<10	4.6371	2.4287	1.909	0.0570 .
Age : 10<=y<15	5.2490	2.3203	2.262	0.0243 *
Age : 15<=y<20	5.7792	2.7898	2.072	0.0390 *
Age : 20<=y	1.8053	2.8316	0.638	0.5242
Gender : Gelding	0.9307	1.4690	0.634	0.5268
Gender : Stallion	2.3193	1.0305	2.251	0.0250 *

Table 2.1.: Summary Table of Parametric Components of SCAM 1

Analyzing the coefficients in 2.1, it is seen that the reference category has been chosen as $y < 2$ for age and female for gender. The estimate values of $2 \le y < 5$ is significantly different from $5 \le y < 10$, $10 \le y < 15$ and $15 \le y < 20$ where the approximate coefficient values are around 5.2 for each parameter. For age $y \ge 20$, the estimate is 1.8053 which is very different and therefore we can perform model refinement as the values are not significant as well at 95% confidence level. It is advantageous to re-level age with six levels into one with fewer levels as it is easier for a vet to determine the age either by assessing the physical condition of the donkey on the field or from gathering information about the animal from the donkey.

The new categories for age are defined as: $y < 5$, $5 \leq y < 20$ and $y \geq 20$. Since the nomogram to calculate the weight of donkeys is constructed only for the reference category, choosing the accurate combination of age and gender is of profound importance. The most populous combination of donkeys in the training set were female donkeys in the age group $5 \leq y < 20$ constituting nearly 30% which led to being the base-level category for modelling GAMs and constructing nomograms. Based on the refined age categorization, we model the data using monotonically increasing B splines and computed the following parametric coefficient estimates in Table 2.2 and non-parametric coefficient estimates in Table 2.3

Parametric Coefficients	Estimate	Standard Error	t Value	P Value
(Intercept)	33.1831	6.8448	4.848	1.84e-06 ***
Age : y<5	-4.928	1.3566	-3.673	0.000275 ***
Age : 20<=y	-3.182	1.9235	-1.655	0.098860 .
Gender : Gelding	0.8928	1.4389	0.620	0.535329
Gender : Stallion	2.3034	1.0164	2.266	0.024016 *

Table 2.2.: Summary Table of Parametric Components of SCAM 2

In the table 2.2, the age coefficient value of non-reference categories is now significant at 90% confidence level and also holds a reasonable explanation that for younger and extremely old donkeys, the weight would be lower than an average donkey in the age range between 5 to 20. In the Table 2.3, it is observed that effective degrees of freedom for Girth and Length are 1.112 and 1.094 which are close to 1 which applies that there is almost a linear relationship between length and weight as well as girth and weight which is shown in Figure 2.10 and Figure 2.11. The linear relationship is valid because there is low variability in length and measurements of donkeys with respect to changes in weights.

In shape constrained additive models, the model is penalized for the wiggliness with the smoothing parameter for length and girth be 2129976.2162 and 311.0732. The smoothing parameter is estimated using outer optimization of either AIC or GCV criterion. Keeping the smoothing parameters fixed, spline coefficients of the basis functions are estimated by maximum penalized likelihood by using iterative re-weighted least squares type algorithm in the inner loop as mentioned in Section 3.2 in (Pya and Wood, 2015) .

Non Parametric	EDF	Residual DF	F Value	P Value
s(Length)	1.000	1.000	142.0	<2e-16 ***
s(Girth)	2.305	2.805	198.6	<2e-16 ***

Table 2.3.: Summary Table of Non Parametric Components of SCAM 2

increas

In the below Table 2.4, it is seen that the values of the spline estimates is very constant around 2.50 for Length and 3.27 for Girth which further validates the linear relationship between the predictor variables and response variable weight. Using the derivative of a cubic penalized B-spline in equation (2.8), the coefficients for length is increasing very negligibly i.e. 0.000007 and hence Figure 2.11 imitates a straight line implying length is roughly linear. However, the coefficients for girth are increasing by a large margin of 0.15 and then the increment value declines which results in girth spline reflecting a convex curve.

Basis Dimension	Length Estimates	Girth Estimates
1	2.502807	3.012214
2	2.502808	3.023151
3	2.502815	3.173688
4	2.502833	3.347851
5	2.502846	3.308977
6	2.502845	3.329291

Table 2.4.: Spline Coefficient Estimates of SCAM 2

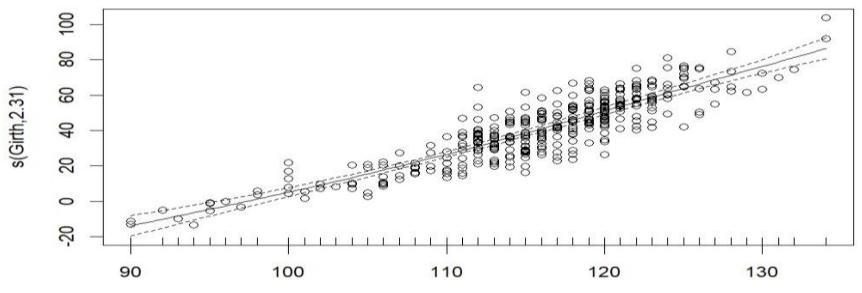


Figure 2.10.: Girth Monotonically Constrained Spline

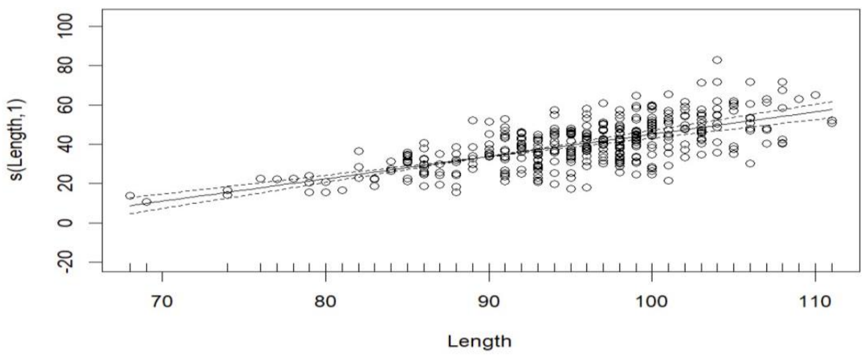


Figure 2.11.: Length Monotonically Constrained Spline

From Figure 2.10 it is observed that girth does exhibit non-linearity since the effective degree of freedom (edf) is 2.305 and is also significant at 95% level of confidence. Figure 2.11 indicates a very linear relationship which is reflected by its effective degrees of freedom which is equal to 1. If a function is evidently monotonically increasing or decreasing and the relationship with the response variable is assumed to be complex, using SCAMs is highly preferred as it allows constraints on the splines along with identifying non-linear patterns. The analysis performed in this chapter used two R libraries i.e. MGCV Wood and Wood (2015) and Pya and Pya (2023)

3. Nomograms

In the following chapter, we aim to build an understanding of nomograms and how it can be used by veterinarians on the field to predict donkey weights using shape constrained additive models. The chapter is explained in the following sections:

1. An Introduction to Nomograms - Description of nomograms and its various types
2. Parallel Scale Nomograms - Introduction to PSN along with mathematical proof of why it works and an algorithm detailing its construction
3. Using Shape Constrained Models in Nomograms - Amalgamating shape constrained additive models and visualizing it using a parallel scale nomogram
4. Predictive Accuracy of Nomogram - Evaluating the accuracy of the nomogram in terms of relative error bands on the 10% holdout set

3.1. An Introduction to Nomograms

A nomogram is a mathematical graphical instrument which represents the relationship between two or more related quantitative variables as mentioned in nom (2021). A nomogram usually consists of 3 variables, one response variable and two predictor variables and gives functional relations between two variables that are represented on parallel scales against each other. A nomogram shows a change in a variable with respect to a change in other variables. It aims to depict linear or non-linear equation representing relationships between sets of measurements.

equations

Nomograms are a two-dimensional graphical calculating device which can look oddly old fashioned, but it has been designed using great ingenuity to decipher very complicated mathematical equations. In contrast to a slide rule, which is used to solve a wide range of equations, using multiple steps, nomograms aim to solve a equation in a single step which makes it really simple and hence leads to precise results. Nomograms are designed for specific uses in different industries. For example, in the medical world, nomograms are used to get accurate predictions of blood pressure, oxygen saturation levels, temperature, etc for determining the appropriate medication dosage as provided in nom (2021).

types

Typically, nomograms are of different types to cater to different functional relationships between response variable and predictor variables. Some of the main type of nomograms are Parallel Scale Nomograms, N or Z charts, Proportional Nomograms, Concurrent Scale Nomograms and Compound Nomograms as mentioned in Doerfler (2009, see Section 2). For the purpose of this study, we focus on constructing a parallel scale nomogram due to its straightforward application on the field by the veterinarians. It is useful for the problem of interest as it allows the veterinarians on field to calculate the weight of a donkey in a single step providing fast graphical solutions with practical precision to veterinarians.

3.2. Parallel Scale Nomograms

? cases

Parallel Scale nomograms are considered as Type 1 nomograms and are known as the simplest form of nomograms to exist. It is used in case where there are 2 predictor variables and the function of the response variable is represented as a summation of a function of each predictor variable as in equation (3.1).

$$f(u) + g(v) = h(w) \tag{3.1}$$

The two independent variables are plotted on outer scales which are drawn parallel to each other. The values of the predictor variables are drawn on the outer scales and are connected by drawing a diagonal line or *isopleth* which intersects the middle scale, and represents the predicted value of the response variable. The scale is a straight line on which values of the function of predictor variable is plotted as strokes corresponding to the value of the variable. In actuarial sciences, indicating successive values of a function by strokes is usually referred to as “calibration” as mentioned in COOKSEY (1950, see Page 232-233). The “scale factor” or “scale modulus” used in construction of the scales refers to the “scale” in engineering drawing, eg : 1” to the foot. By changing the scale factor, different scales can be generated for the same function. When the calibrations are equally placed, the scale is referred to as “uniform” or “natural”. A non-uniform scale is referred popularly as “functional” with the most import functional scale being the logarithmic scale.

3.2.1. Construction of Parallel Scale Nomograms

Nomograms are constructed based on similar triangles as illustrated in the Figure 3.1 below which is based on Doerfler (2009, see Figure 2)

What are similar triangles?

Similar Triangles are triangles with corresponding angles equal to each other and corresponding sides are in proportion as well.

Why are similar triangles used?

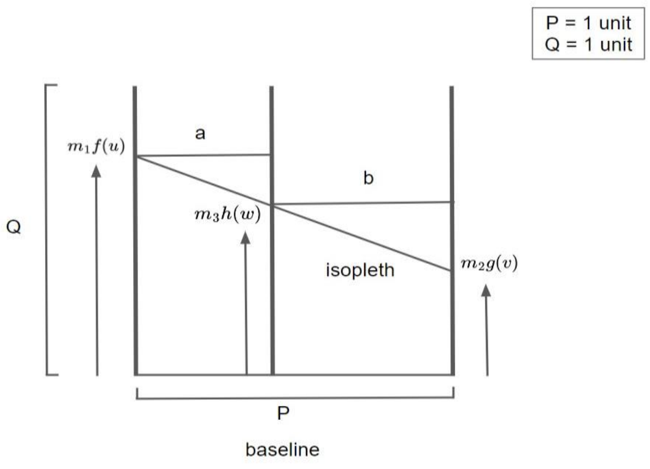


Figure 3.1.: Parallel Scale Nomogram

Two triangles are formed when the isopleth transverses the 3 parallel scales, which due to alternate exterior and interior angles leads to the formation of similar triangles invariably. Using proportionality property of similar triangles, a parallel scale nomogram for 2 continuous variables can be constructed.

Proof:

represents

Here, m_1,m_2 and m_3 represent the scaling factors of two outer scales u and v and the middle scale w respectively. Also, **a** represent the distance between scale u and scale w and **b** represents the distance between scale w and scale v. In Parallel Scale Nomograms, **a** and **b** greatly impacts the prediction on scale w.

1. **Placement of target scale W :** If there exists a relationship $f(u) + g(v) = h(w)$ and f(), g(), and h() are all strictly monotonically increasing, then the w axis must lie between the u and v axes. If scale w is placed on extreme right, varying u values and keeping w fixed will result in lower w for higher u which is incorrect.

2. **Calculate a, b m_1,m_2 and m_3** : Isopleth transversing u and v scale results in 2 similar triangles and using the proportionality of corresponding sides property as mentioned in Doerfler (2009, See Section 2.1), we get equation (3.2). Refer to Figure 3.1 for a complete graphical analysis.

$$\frac{m_1 f(u) - m_3 h(w)}{m_3 h(w) - m_2 g(v)} = \frac{a}{b} \tag{3.2}$$

$$m_1 f(u) b + m_2 g(v) a = m_3 h(w) a + m_3 h(w) b \tag{3.3}$$

Dividing by b,

$$m_1 f(u) + m_2 g(v) \frac{a}{b} = m_3 h(w)\left(1 + \frac{a}{b}\right) \quad (3.4)$$

To arrive at equation (3.1), we consider the following substitution:

$$m_1 = m_2 \frac{a}{b} = m_3 \left(1 + \frac{a}{b}\right) \quad (3.5)$$

$$\frac{m_1}{m_2} = \frac{a}{b}\,;\; m_3 = \frac{m_1 m_2}{m_1 + m_2} \quad (3.6)$$

where a + b = **W** (distance between G-scale and L-scale). Using the unique property of proportionality of similar triangles allows to calculate unknown constants for easy construction of parallel scale nomograms. Algorithm 1 based on Doerfler (2009, See Section 2.1) uses the proof and constructs a parallel scale nomogram and the code is provided in Appendix A.

Algorithm 1 Framework of Parallel Scale Nomogram Construction

1: Input: Function representing the relation between 2 predictor variables $f(u)$ & $g(v)$ and response variable $h(w)$, distance between two outer scales say **P**, length of the scale **Q** and range limit of the predictor variables $ulim = [u_{lower}, u_{upper}]$ and $vlim = [v_{lower}, v_{upper}]$.

2: Output: A parallel scale nomogram drawn for the equation to predict w.

3: Calculate the scaling factor m_1, m_2 and m_3 using $f(u)$ and $g(v)$ in the below equation:

$$m_1 = \frac{Q}{f(u_{upper}) - f(u_{lower})} \tag{3.7}$$

$$m_2 = \frac{Q}{g(v_{upper}) - g(v_{lower})} \tag{3.8}$$

$$m_3 = \frac{m_1 m_2}{m_1 + m_2} \tag{3.9}$$

4: To calculate the unknown constants **a** and **b**, we use similar triangles as explain in section

$$b = \frac{1}{\left(\left(\frac{m_1}{m_2}\right) + 1\right)} \tag{3.10}$$

where $a + b = P$

5: Calculate the limit range of the response variable w, $wlim = [w_{lower}, w_{upper}]$ by using the below equation:

$$w_{lower} = f(u_{lower}) + g(v_{lower}) \tag{3.11}$$

$$w_{upper} = f(u_{upper}) + g(v_{upper}) \tag{3.12}$$

6: Generate a sequence of values for u,v and w i.e. u_{values}, v_{values} and w_{values} between the specific limit ranges of ulim, vlim and wlim.

7: Construct the nomogram using the code snippet provided in section and calculate the calibrated values for the three scales i.e

- Draw u-scale on the left with a baseline value of u_{lower} with tickmarks spaced out as: $m_1\, [f(u_{values}) - f(u_{lower})]$

- Draw v-scale on the right at a distance of **W** with a baseline value of $v_{\mathbf{lower}}$ with tickmarks spaced out as: $m_2\,[g(v_{\mathbf{values}}) - g(v_{\mathbf{lower}})]$
- Draw w-scale on the right of u-scale at a distance of **a** with a baseline value of w_{lower} with tickmarks spaced out as: $m_3\,[h(w_{\mathbf{values}}) - h(w_{lower})]$.
- **Note :** There is an error in the original paper Doerfler (2009, See Page 461) while calculating the location of tick marks on scale w.

3.3. Using Shape Constrained Models in Nomograms

In this section, we combine the generalized additive model structure discussed in Section 2.4.3 along with the parallel nomogram construction in Section 3.2.1. The process to amalgamate these concepts is defined in Algorithm 2. The 2 key constraints on the shape constrained additive models are:

1. $f(\cdot)$, $g(\cdot)$ and $h(\cdot)$ are all monotonically increasing functions.
2. There exists two categorical predictor variables in the dataset i.e. **A** and **S**.

Parallel Scale Nomograms using SCAMs as shown in 3.2 can be constructed for any dataset with 3 continuous variables and 2 predictor variables following the aforementioned constraints. It can be constructed in a simple manner without using the fitted coefficients of the model. Using 3 APIs of the scam library Pya and Pya (2023) i.e. plot, predict and approx, we can build the two splines f(u) and g(v) using a fitted scam object in R.

To verify the prediction accuracy of the nomogram with respect to the scam predictions, we choose a donkey from the holdout set which has the following predictor variable values : Female, Age between 5 to 20, Girth = 110 and Length = 88. In Figure 3.2, we can place a ruler on Length = 88 and Girth = 110, it is seen that it crosses the Weight axis at approximately 128.9, which is also the SCAM prediction and using the predict() method in scam library Pya and Pya (2023).

Algorithm 2 Framework of using SCAM to Construct Parallel Scale Nomogram

```
1: Input: Fitted SCAM object $model_{scam}$
2: Output: A parallel scale nomogram drawn based on shape constrained additive models
   $model_{scam}$ with 2 continuous variables **u** and **v** and 2 categorical features **A** and **S** for
   predicting a third continuous variable **w**
3: Calculate the spline basis functions **f(u)** and **g(v)** for variable **u** and **v** respectively using
   the following steps :
     • Extract the dataset **D** using the model method of $model_{scam}$ and intercept value
       **cbar** for $model_{scam}$ which is the value on the first index in list of model coefficients
     • Using inbuilt **plot** API of scam library in R on $model_{scam}$ to produce an object **lists**
       which consists of two functions of girth and length used in fitting the model and
       its corresponding predicted weight values. These functions are offset and hence
       the first spline function f(u) is re-defined in Step 4.
     • Creating splines **f(u)** and **g(v)** using the **approx** API used for linear interpolation.
       Values used for interpolation are $u_i$ and $v_i$ in the range of **u** and **v** components of
       **lists** which result in $w_u$ and $w_v$ respectively
                         $f(u) = w_u + \text{cbar}$                                   (3.13)
                         $g(v) = w_v$                                                 (3.14)
4: To define $f_{\text{offset}}(u)$ using **f(u)** because of a default offset value in plot API
     • Create a reference dataset with instances belonging only to the combination of
       reference categories of **A** and **S** which was used to fit the model.
     • Use **predict** API to make **predictions** on the reference dataset.
     • Calculate offset value by computing the mean of the difference between the
       prediction value and the fitted spline value for variable u and v for each data
       point in reference dataset. Here n is the number of points in reference dataset
       and we extract the value for variable **u** and **v** for datapoint j i.e. $u_j$ and $v_j$ and use
       the following equation:
         $\textbf{offset} = \frac{1}{n}\sum_{j=1}^{n}[\text{predictions}[j] - (f(u_j) + g(v_j))]$   (3.15)
     • Define $f_{\text{offset}}(u)$ using the below equation:
                         $f_{\text{offset}}(u) = w_u + \text{cbar} + \text{offset}$               (3.16)
       and **h(W)** being an identity function i.e. $h(w) = w$
5: Define the range limits **ulim** and **vlim** for variables **u** and **v** respectively using
   minimum and maximum variable values in the dataset.
6: Using the above-mentioned functions and Algorithm 1, a parallel scale nomogram is
   constructed based on the following fitted shape constrained additive model:
                         $h(w) = f_{\text{offset}}(u) + g(v)$                              (3.17)
```

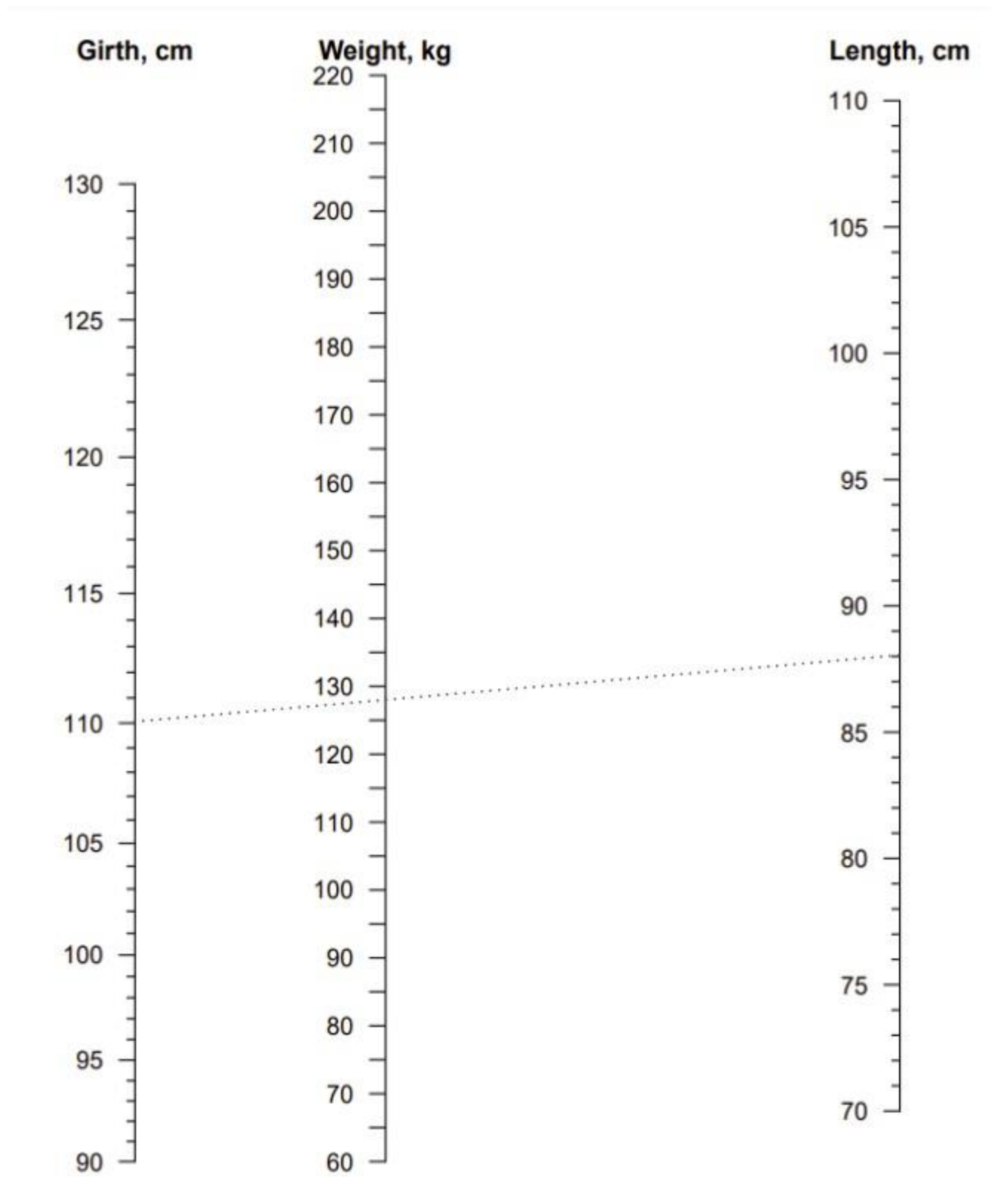


Figure 3.2.: Nomogram for Kenyan donkeys with Gender as Female and 5 ≤ *Age* < 20.To predict weight, join the girth and the length values with a straight line

Factor			
Gender		Age	
Stallion	+2	y <5	-5
Gelding	+1	y >= 20	-3

Table 3.1.: Additive adjustments for factors at non-reference levels, in kilograms

3.4. Predictive Accuracy of Nomogram

In this section, we check the nomograms accuracy using the holdout dataset which consists of one-tenth of the donkey population. The donkeys have been stratified so the model development and model assessment are representative of the varying weights present in the dataset. In Figure 3.3, we analyze the predictions with respect to 10% and 20% relative error bands which have been explained in Table 3.2. Almost 86% of the observations are between the 10% error bands and very few observations lie between the 10% and 20% relative error band.

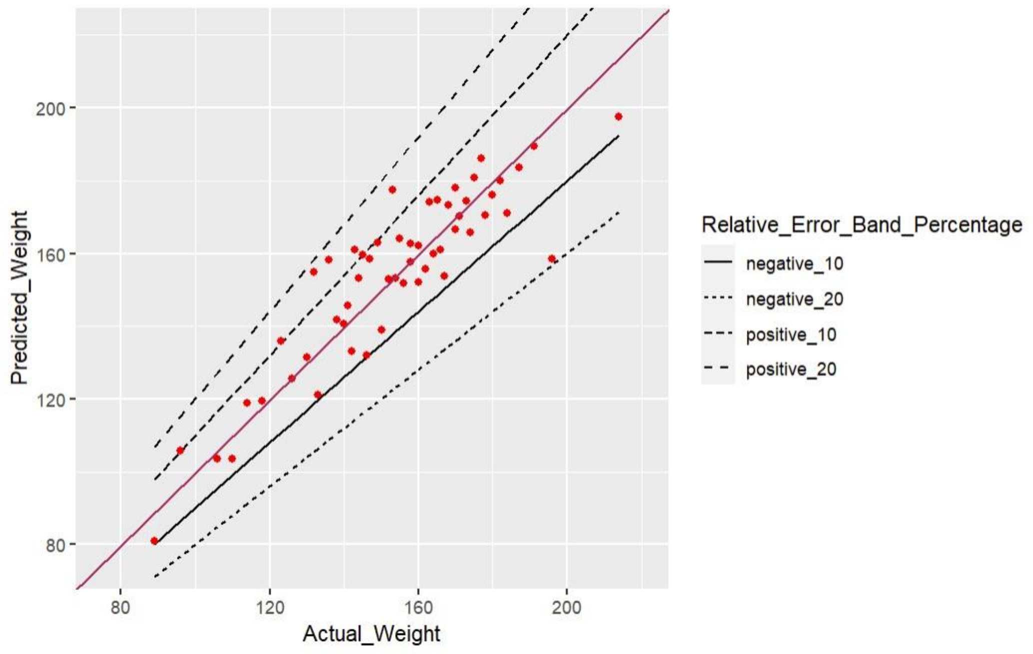


Figure 3.3.: Hold-out sample of 54 donkeys. Actual weight versus predicted weight with relative error bands

*Relative Error = {(Actual-Predicted)}/{Actual} * 100*				
Relative Error	**<-10%**	**-10 to 0%**	**0 to 10%**	**>+ 10%**
Proportion	**12.96**	**37.04**	**48.15**	**1.85**

Table 3.2.: Distribution of relative errors of our tool in the holdout sample of 54 donkeys

On the holdout set, we also calculate the root mean square error on the predictions and the actual weight which is 10.89661. On average using MSE and relative error as perfomance metrics, we conclude that the tools accuracy is ±10% and is consistent over the range of predicted weights between 80 and 200 kgs.

4. Use of Ensemble Techniques

In this chapter, we take an alternative approach to predicting donkey weights using black box models. These models are very computationally heavy and hence the model predictions are difficult to deconstruct into simpler components as compared to model discussed in Chapter 2. To explain it in further detail, the chapter is explained in the following sections:

1. Use of Extreme Gradient Boosting in Machine Learning - The main principle behind XGBoost regressor and its advantages over other decision tree ensemble learning methods along with underlying proof of model metrics
2. Model Development - Modelling the data using XGBoost on the 20% validation dataset and checking the predictive accuracy on the 10% holdout dataset
3. Impossibility of a Black Box based Nomogram - Justification of why a black-box model cannot be visualized as a parallel scale nomogram

4.1. Use of Extreme Gradient Boosting in Machine Learning

Ensemble techniques are widely used machine learning algorithms which are known to provide really accurate model predictions. It combines several weak base learners in parallel or sequentially to build one strong learner. Ensemble method discussed below is based on decision tree which is a sequential tree based algorithm. Extreme Gradient Boosting popularly known as XGBoost is one of the most commonly used decision tree algorithms which belongs to supervised learning algorithm. XGBoost is a sequential and a scalable distributed gradient boosted decision tree model used extensively for both classification and regression.

widely used

4.1.1. Introduction of XGBoost in Regression

In this section, we explore the use of XGBoost to predict donkey weights using a range of predictor variables. Black box models unlike parallel scale nomograms do not restrict with the use of two predictor variables. XGBoost Trees are the recent addition to the ensemble learning techniques, developed in the year 2016 and is mentioned in the original paper Chen and Guestrin (2016). XGBoost combines several week learners i.e. decision trees in a sequential manner and builds a single strong learner. Similar to unextreme gradient boosted trees, decision trees are built on the residuals but use unique XGBoost trees. The model aims to cluster similar residuals together based on similarity scores and choose the split that leads to maximum information gain. It is an improved version of other ensemble tree based algorithms in the following ways:

1. XGBoost Trees uses advanced regularization i.e L1 and L2 norm in comparison to Gradient Boosted Trees and hence it avoids overfitting of the model.

2. It leads to faster computation as the process can be parallelized and distributed across clusters in comparison to GBT trees, hence reducing the training time and improving memory and space utilization. The biggest advantage of the algorithm is its scalability and runs 10 times faster than GBT

3. The model is known to perform well because the target variable of the successive weak learners are the residuals from the previous learners. Unlike random forest, the model does not allow a random selection of data for the subsequent weak learners, but chooses the residuals that are significantly high from preceding learners.

4. The algorithm focuses on reducing overfitting by using column sub-sampling (when number of features are high) and also scales newly added weights by factor after each step of tree boosting to reduce the influence of each tree on the model.

Mathematics of XGBoost in Regression

The below explanation on XGBoost for Regression has primarily been derived from Chen and Guestrin (2016) and also takes additional insights from Starmer (a) and Starmer (b).

In XGBoost, suppose using a single continuous predictor variable x and a response variable y, the first prediction for a data point is given an arbitrary value i.e. $p^0_i = 0.5$ where $i = 1,2,\cdots,n$ where n is the number of data points in the dataset. Based on the constant p^0_i value, residuals are calculated i.e $(y_i - p^0_i)$. The residuals for all data points belong to the root node 'r'. XGBoost is used to predict output values of leaf nodes which is represented by O_{value}. The loss function which needs to be minimized by optimizing O_{value} is given as equation (4.1).

$$\sum_{i=1}^{n} L(y_i, p_i) + \frac{1}{2}\lambda O_{\text{value}}^2 \tag{4.1}$$

where:

$$L(y_i, p_i) = \frac{1}{2}(y_i - p_i)^2 \tag{4.2}$$

and λ is the regularization parameter which reduces the predictions sensitivity to individual datapoints and prevents overfitting. Based on the original XGBoost paper, the equation to be minimized is equation (4.3) where γ is a user defined tree complexity penalty parameter used to encourage pruning and T is the number of terminal nodes.

$$\sum_{i=1}^{n} L(y_i, p_i) + \gamma T + \frac{1}{2}\lambda O_{value}^2 \tag{4.3}$$

[illegible] Gamma G

However, since pruning takes place only after the complete tree is constructed, γT is omitted. Since, we are optimizing the output from the first tree, the p_i is replaced by $p_i + O_{value}$. Hence, the loss function used is

$$\sum_{i=1}^{n} L(y_i, p_i^0 + O_{value}) + \frac{1}{2}\lambda O_{value}^2 \tag{4.4}$$

In equation (4.4), if $\lambda > 0$, then the O_{value} gets reduced. When $\lambda = 0$, the optimal O_{value} is the minimum point of the loss function for different O_{values} i.e. it is the point where its first derivative is equal to 0. As λ increases, $O_{value} \rightarrow 0$. XGBoost uses a second order Taylor expansion to approximate $L(y_i, p_i^0 + O_{value})$ which is defined below in Equation

$$L(y_i, p_i^0 + O_{value}) \approx L(y_i, p_i) + \left[\frac{d}{dp_i} L(y_i, p_i)\right] O_{value} + \frac{1}{2}\left[\frac{d^2}{dp_i^2} L(y_i, p_i)\right] O_{value}^2 \tag{4.5}$$

Since $\left[\frac{d}{dp_i} L(y_i, p_i)\right]$ is the gradient and is referred to as g and $\left[\frac{d^2}{dp_i^2} L(y_i, p_i)\right]$ is the hessian and is referred to as h. For simplicity of notation, equation (4.5) is defined as:

$$L(y_i, p_i^0 + O_{value}) \approx L(y_i, p_i) + gO_{value} + \frac{1}{2}hO_{value}^2 \tag{4.6}$$

Using equation (4.4) and 4.6, we get:

$$\sum_{i=1}^{n} L(y_i, p_i^0 + O_{value}) + \frac{1}{2}\lambda O_{value}^2 = L(y_1, p_1^0) + g_1 O_{value} + \frac{1}{2}h_1 O_{value}^2 + L(y_2, p_2^0) + g_2 O_{value} + \frac{1}{2}h_2 O_{value}^2 + \cdots + L(y_n, p_n^0) + g_n O_{value} + \frac{1}{2}h_n O_{value}^2 \tag{4.7}$$

Since, we want to find the optimal O_{value}, we can eliminate terms not depending on

O_{value} in equation (4.7),

$$\sum_{i=1}^{n} L(y_i, p_i^0 + O_{\text{value}}) + \frac{1}{2}\lambda O_{\text{value}}^2 \approx (g_1+g_2+\cdots+g_n)O_{\text{value}} + \frac{1}{2}(h_1+h_2+\cdots+h_n+\lambda)O_{\text{value}}^2 \tag{4.8}$$

Taking first derivative of equation (4.8) with respect to O_{value} and equating it to 0, we get:

$$O_{\text{value}} = \frac{-(g_1+g_2+\cdots+g_n)}{(h_1+h_2+\cdots+h_n+\lambda)} \tag{4.9}$$

We know $g_i = -(y_i - p_i)$ which is negative residual and $h_i = 1$, hence we simplify equation (4.9) as :

$$O_{\text{value}} = \frac{\text{Sum of Residuals}}{n+\lambda} \tag{4.10}$$

Since we are Taylor's approximation to solve for the optimal output value of feature node, we use the following process to calculate similarity score which is based on the above computation of output value. Multiplying the RHS of equation (4.8) with -1 and substituting it with equation (4.9)

$$\text{Similarity Score} = \frac{1}{2}\frac{(g_1+g_2+\cdots+g_n)^2}{(h_1+h_2+\cdots+h_n+\lambda)} \tag{4.11}$$

Eliminating $\frac{1}{2}$ since similarity score is a relative comparison and substituting g_i and h_i values, we get similarity score as:

$$\text{Similarity Score} = \frac{\text{Sum of Residuals}^2}{n+\lambda} \tag{4.12}$$

When residuals are similar, the similarity score is high and when they are different in terms of positive and negative, the similarity score is low. To decide the threshold value of a feature based on which the root node is split, we calculate the average of 2 consecutive values of x. The decision is based on the value of threshold that results in the highest gain. To calculate how much better the terminal node clusters similar residuals in comparison to root node, we compute gain for different averages. Suppose there are 2 leaf nodes 'a' and 'b', gain of the split for threshold x_j where j = $\frac{(x_i+x_{i+1})}{2}$ is given as follows:

$$Gain_{x_j} = Similarity_a + Similarity_b - Similarity_r \tag{4.13}$$

The decision to split a node further is based on the minimum number of samples in a node and maximum depth of the tree. The default value for minimum number of samples is 1 and maximum depth of the tree is 6. If the model has not reached the stopping criterion, the gain is calculated for each node. Once a tree has been built to its complete depth as per the stopping criterion, gain of the lowest branch of the tree is used to perform pruning by choosing an arbitrary value of γ. If Gain − $\gamma < 0$, we prune the branch and continue upward. Also, when $\lambda > 0$, it results in more pruning because the similarity score and gains are smaller and results in smaller output values for leaves. The percentage change in the similarity score provided there is an increase in λ is inversely proportional to the number of residuals in a node. Based on the output value, new predictions are made using the first tree i.e.

$$p_{1i} = p_{0i} + \eta O_{\text{value}} \tag{4.14}$$

which generates a new set of residuals that are lower than the original residuals. The new residuals are used to construct the next XGBoost tree which will result in lower residuals and is an iterative process till the residuals are very small or maximum number of trees limit has reached.

4.2. Model Development

This section entails the implementation of XGBoost Regression to the donkey dataset which is usually known to outperform multiple machine learning algorithms. The approach in Chapter 2 i.e. using SCAMs focuses on the additive nature between the unknown smooth functions which implies no interaction but focuses on the smoothness. However, XGBoost is complex as it incorporates interactions but does not allow for much smoothness which can lead to low accuracy if the relationships being considered are smooth. Both the approaches have their advantages and disadvantages and hence it is difficult to make an apriori assumption on which model will be more suitable. To initiate the model development for XGBoost, the following methodological steps are undertaken:

1. Feature Engineering - The stage involves data transformation using one-hot encoding technique
2. Hyperparameter Tuning for Model Choice - Performing hyperparameter tuning using k-fold cross validation to improve model choice on the 20% validation dataset
3. Predictive Accuracy of XGBoost - Evaluating the model based on best set of hyperparameters on the 10% holdout set.

4.2.1. Feature Engineering

In the dataset, we consider three continuous predictor variables i.e. Height, Length and Girth and two categorical features Gender and Age which have been factored into 3 levels each based on the model refinement performed in Section 2.4.3 and the target variable is weight which is continuous. Age is a categorical variable which has an ordering since it is observed that as age increases there is an increase in weight. However, a further increase i.e. when it belongs to Age ≥ 20, the weight starts to decline. Due to developments in Chapter 5, it is necessary to fit 2 XGBoost models, Model A and Model B. This is to facilitate comparisons predictions of XGBoost using Explainable AI methods in Chapter 5. Dataset for Model A is not one-hot encoded as the encoding takes place during the model training process. For Model B, gender is one-hot encoded since it is an unordered factor and hence one hot encoding is the most suitable transformation technique. The model performs one-hot encoding for Age regardless of the ordinal nature as the model is expected to capture the underlying patterns in the data.

4.2.2. Hyperparameter Tuning for Model Choice

The model is split into a 70:20:10 ratio for training, validation and holdout respectively. The data has been stratified in a identical way as in Section 2.4. In XGBoost Regressor, there are 5 important parameters that can lead to better model fit using hyperparameter tuning and k- fold cross validation. The 5 parameters are

1. **n estimators** : Maximum number of XGBoost trees that the model can build. It is selected to be [100,200,300].

2. **max depth** : Maximum depth of an individual XGBoost tree. It is selected to be [2,3].

3. **reg lambda** : Regularization parameter to prevent overfitting. It is selected to be [0,1.0,10.0].

4. **gamma** : Tree complexity measure to influence pruning. It is selected to be [0,0.25,1.0].

5. **learning rate** : Learning Rate to calculate predictions based on output value of nodes. It is selected to be [0.1,0.01,0.05].

The number of folds is 10, and to improve computational efficiency, the model is trained on 80% of the number of observations in the training dataset for each XGBoost Tree. The scoring metric to choose the best model fit is chosen as root mean square error to facilitate comparisons with model choice results generated in Section 2.4. For Model A, in Figure 4.1, it is observed that at Depth = 2, the model has the highest accuracy in terms of negative MSE for different number of XGBoost trees. The best combination is for depth of tree = 2 and the number of XGBoost Trees = 300. In Figure 4.2, it is observed that as the learning rate increases, reaches the maximum and declines in terms of negative MSE. The best hyperparameters that the model chose in a 10-fold cross validation are as follows : Gamma = 1, Learning Rate = 0.05, Maximum Tree Depth = 2, Number of Trees = 300 and Regularizer Lambda = 0 which recorded the highest negative mean square error of -179.177947 for the training set.

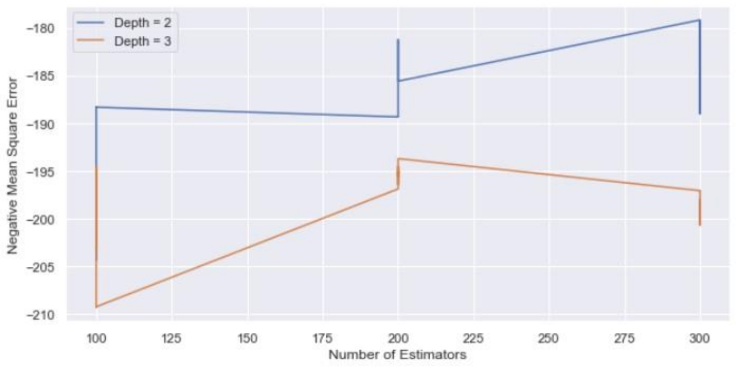


Figure 4.1.: Performance of Model A for Number of Trees and Depth in terms of Negative MSE

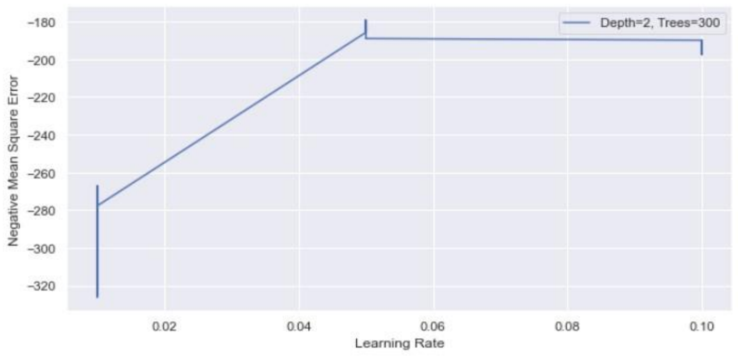


Figure 4.2.: Performance of Model A for different Learning Rates in terms of Negative MSE

For Model B, it is observed that at Depth = 2, the model has the highest accuracy in terms of negative MSE for different number of XGBoost trees. The best combination is for depth of tree = 2 and the number of XGBoost Trees = 100. The best hyperparameters that the model chose in a 10-fold cross validation are as follows : Gamma = 0, Learning Rate = 0.1, Maximum Tree Depth = 2, Number of Trees = 100 and Regularizer Lambda = 0 which recorded the highest negative mean square error of -178.490222 for the training set.

4.2.3. Predictive Accuracy of XGBoost

On the holdout set of 54 donkeys, using Model A we get the RMSE of 12.476 and using Model B we get 12.084 which is very similar. This implies that one-hot encoding does not have a significant impact on the model performance. It is also observed that the RMSE for the black box model is higher than the RMSE of nomogram which is contradicts the working hypothesis.

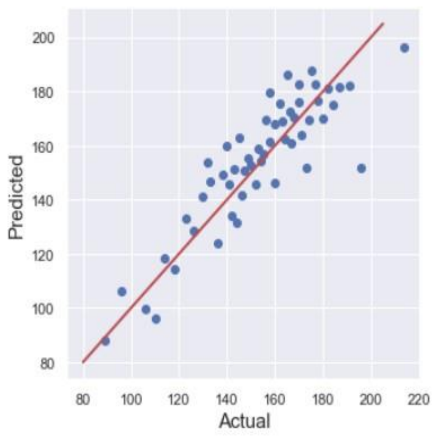


Figure 4.3.: Actual Values of Weight versus Predicted values using Model A

4.3. Impossibility of a Black Box based Nomogram

The biggest challenge facing black box models is lack of explainability. The internal mechanisms of black box models are complex and not inherently explainable to humans. XGBoost, an ensemble technique that exploits interactions between multiple features is known to provide higher predictive accuracy. However, there is a trade-off between predictive accuracy and interpretability of these models.

In this section and the following chapter, we use interpretability and explainability of black box models interchangeably and try to draw a comparison between feasibility of a parallel scale nomogram using ensemble techniques. In decision tree based ensemble algorithms that uses bagging and bootstrap aggregation, the model is affected not only by individual features but also by its interactions with other features. A parallel scale nomogram is constructed with 3 continuous features which represent the 3 parallel scales in the nomogram. The models used to build these nomograms assume no interactions between the features and contribute to the model individually. For instance, the parallel scale nomogram in Figure 3.2,is based on shape constrained additive models which assume no interactions. The SCAM is used as it exploits smoothness whereas the XGBoost model uses interactions.

In this section, we aim to explain that multiple machine learning models like XGBoost that are known for producing higher predictive accuracy models which might be more useful but due to its increasing complexity, its predictions cannot be easily understood by veterinarians on a field. It is assumed that the easiest method for veterinarians to calculate donkey weights is using a piece of paper and pen. However, to use XGBoost or any other black box model, a vet would require a smartphone where the algorithm can run on the application. Both technologies are ubiquitous now i.e. a clipboard with a nomogram on it, and a smartphone. The two approaches are used differently with the latter offering lesser interpretability since it is based on a black box model. There is a higher probability of a veterinarian to trust the nomograms predictions as it provides more of a visual understanding as compared ~~to a black box prediction~~ on an app. For this reason, parallel scale nomograms are the most commonly used instruments for measuring weights of animals on donkeys as it is ~~extremely simple~~ to understand how to use it and also provides visual interpretations. To calculate the weights of donkeys that do not belong to reference categories, the vet needs to perform additive adjustments which are provided below the nomogram. This is not the case with ensemble techniques. A second drawback of using XGBoost algorithm is the lack of interpretability using visualizations since it cannot be converted to a parallel scale nomogram. When a vet has access to a parallel scale nomogram, it is extremely simple to understand what relationship is shared between the girth and length of the donkey with the weight. This breakdown of individual feature importance will be impossible in XGBoost.

5. Explainable Artificial Intelligence

In this chapter, we understand the use of explainable AI (XAI) trying to disintegrate black box models to offer more interpretability using 2 algorithms, LIME and SHAP. XAI is still a fast-moving, developing field and in this chapter, we have presented a snapshot of the algorithms including its current drawbacks. To explain it in further detail, the chapter is explained in the following sections:

1. An Introduction to Explainable AI - The booming industry of XAI and the developing need to deconstruct black-box models
2. Local Interpretable Model Agnostic Explanations - LIME is a XAI algorithm that focuses on building local linear models to explain individual instances
3. Shapley Additive Explanations - SHAP is a more stable XAI algorithm which is famous due to its shapley value properties and additive nature. It is used extensively for feature selection as well

5.1. An Introduction to Explainable AI

Explainable Artificial Intelligence (XAI) is a rapidly expanding research area aiming to address the intricacies of various machine learning algorithms that are considered to be black box models. It aims to explain the opaque nature of the model by understanding important features that impact predictions, model learning process during training. XAI plays a pivotal role in users *trusting individual predictions* and *trusting the model* as a whole Ribeiro et al. (2016, See Page 1), consequently substantiating the rationale underpinning the model predictions. With key advancements in deep learning and different velocities of data types, complexities of black box models have become increasingly incomprehensible. When models are used for decision making, trusting individual model predictions is very important such as in cancer diagnosis, credit fraud, etc. Interpretable Machine Learning allows to break down the algorithm to understand the functioning of the models which allows to provide explanations to end users justifying the model predictions. There is often a trade-off between interpretability and complexity of ML models.

XAI algorithms are categorized based on the following factors: **Agnosticity, Scope, Data Type and Explanation Type**. There exists model-agnostic and model specific approaches which provide either local or global explanations or both working for different data types and result in different kinds of explanations. A model specific approach can work only a limited range of black box models such as random forests, neural networks, etc and provides for better understanding since it has access to the internal workings of the model. On the other hand, model agnostic approaches can be applied to any blackbox model and

can be used to compare the explanations produced by applying the same framework to different complex machine learning algorithms. Explainability is usually in the form of visualizations, feature summaries, local instances or surrogate/explainer models. DeepFindr (a) In this work, we emphasize on two model agnostic approaches :
LIME and **SHAP**.

5.2. Local Interpretable Model Agnostic Explanations

5.2.1. Understanding LIME for Local Explanations

LIME is model agnostic and hence can be used for any machine learning algorithm. Many times, the decision boundary is highly non-linear which implies there is no simple solution to explain model predictions for a new data point. LIME aims to focus on the neighbouring area of the instance by fitting a linear interpretable model in the local area which is referred to as a surrogate model. Ribeiro et al. (2016, See Section 1) It is a local approximation of th black-box model for a specific input. It is based only on the response function i.e. the relationship between predictors and response variable since the model internals are hidden. The explanations are required to be locally faithful and need not be globally faithfully. It is based on the following equation (5.1) which explains the fidelity-interpretability trade off as proposed in Ribeiro et al. (2016, See Section 3.2)

$$\xi(x) = \underset{g \in G}{\text{argmin}}\ L(f, g, \pi_x) + \Omega(g) \qquad (5.1)$$

Here, L represents the locality aware loss function, x represents the input features of the instance of interest usually. The complex model is denoted by f, the explainer is denoted by g which is a part of a family of simple interpretable models G. G is a collection of linear models such as linear regression and all its variants. In LIME, G is set to sparse linear models i.e. $g(z') = w_g \cdot z'$. π_x is a proximity measure which represents the neighbourhood of the instance x. $\Omega(g)$ is a regularization term that controls the complexity of the simple interpretable model g Ribeiro et al. (2016, See Section 3.2). In summary, the optimization problem aims to minimize the loss and the complexity. However, computation of equation (5.1) is intractable and we can approximate it using K LASSO where we select K important features out of d (total number of features) using LASSO which is followed by learning weights i.e. w using least squares.

If the original representation of instance $x \in R^d$, its interpretable representation using binary vectors is $x' \in \{0,1\}^{d'}$. To calculate, $L(f,g,\pi_x)$, various data points are sampled around x' which are weighted according to the proximity measure π_x referred to as "sampling for data approximation". The points are sampled using perturbation by sampling from a normal distribution using the mean and the standard deviation of each feature and drawing nonzero elements of x. A perturbed data point is $z' \in \{0,1\}^{d'}$ where the original representation is $z \in \mathbb{R}^d$. Compute the predictions for the new data points using the complex model f and surrogate model g. For example, for linear regression, the loss function is sum of residual squares weighted using π_x as based in the original paper Ribeiro et al. (2016, See Equation 2, Section 3.4) defined below in equation (5.2):

$$\mathcal{L}(f, g, \pi_x) = \sum_{z, z' \in \mathcal{Z}} \pi_x(z) \left(f(z) - g(z')\right)^2 \tag{5.2}$$

In the original paper Ribeiro et al. (2016, See Section 3.4), tabular data $\pi_x(z)$ is calculated based on an exponential smoothing kernel i.e.

$$\pi_x(z) = exp(\frac{-D(x, z)^2}{\sigma^2}) \tag{5.3}$$

where D is a distance function (usually Euclidean distance) and σ is the kernel width. This implies that the points z close to x are weighted more, hence implying local faithfulness of the model. The need for a metric to define a neighbourhood will often be a complex issue when the covariates in the blackbox predictor have different units and different effects. Also, for g, a sparse linear model is used leading to higher feature interpretability due to zero weights on many variables. However, for a higher dimensional feature space such as for images, proximity measures differ significantly as compared to those of tabular data.

combine them

To illustrate, consider applying LIME to the XGBoost model for predicting donkey Weight as a function of Age, Sex, Girth, Height and Length. The 3 different levels under age have been re-coded as $y < 5 = 0$, $5 \leq y < 20 = 1$ and $y \geq 20 = 2$. The 3 different levels under gender i.e. gelding, stallion, female have been re-coded as 0,1,2 respectively. We can consider selecting a data point **x** that needs explained which can be: Age = $y < 5$, Height = 95, Girth = 100, Gender = female and Length = 105. Then x' is given as [0,95,100,2,105]. The next step is to sample data points using Gaussian sampling i.e **x'** around the instance which can lead to a new data point i.e. Age = $y < 5$, Height = 97, Girth = 100, Gender = stallion and Length = 102 to be around an instance. Then **z'** is [0,97,100,1,102]. Similarly, we sample multiple new datapoints and compute the predictions of z using the blackbox model f and z' by using the surrogate model g which is a weighted local linear regression model. Since number of parameters is less than 6, the method used for feature selection is forward selection. Based on the distance metric,√

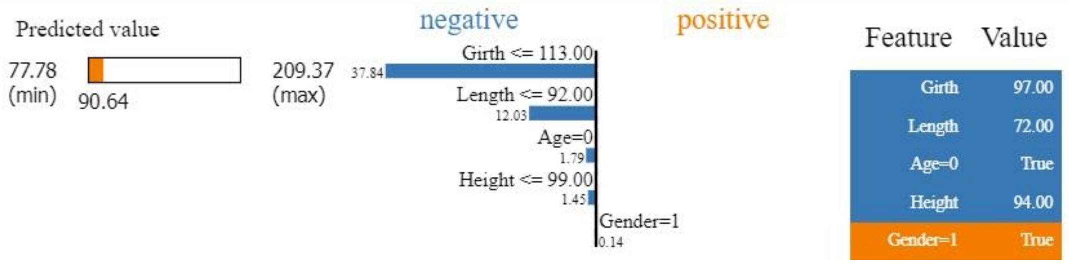
weightage based on exponential kernel with a kernel width of $0.75 * \overline{d}$ where d = 5, is assigned to the difference between the predictions for a single data using f and g models to maintain local faithfulness. The kernel width determines how large the neighbouring area is, with a smaller σ implying that a data point needs to be very close to have significant impact on the local model whereas a larger σ implies that similar weightage is given to the neighbouring data points and is not very dependent on the distance between the instance and the samples. Based on the new samples, a local weighted linear model is fitted for the samples data points.

Implementation of LIME on Black-Box Models for Local Explanation

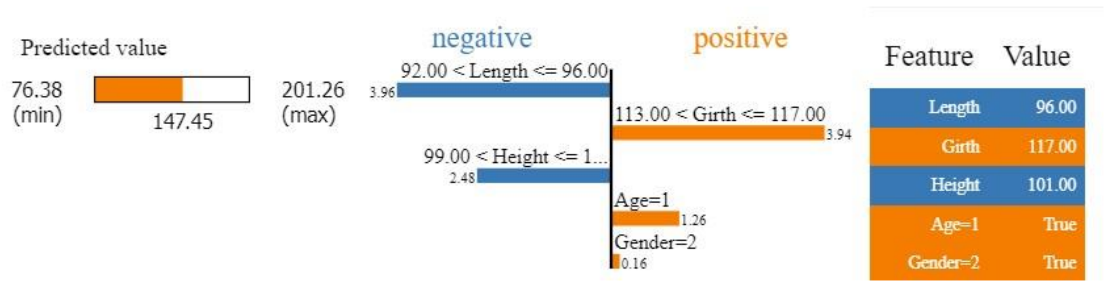
For the implementation of LIME, it is known to be very sensitive to one-hot encoded data as it generates samples through perturbations of existing instance x. If one-hot encoded data is provided, there is a possibility that a sample might be generated that is completely impossible and inaccurate. For instance, if gender has 3 levels and gender gelding is represented as [1,0,0], gender stallion is represented as [0,1,0] and gender female is represented as [0,0,1]. LIME can sample data in a way such that the new instance has gender taking values [1,1,0] which implies that it is a stallion and female and hence meaningless. But for LIME, it can only take integer values as its predictor variables and hence the various levels have been encoded as integers with no ordinal relationship. The XGBoost model built for implementing LIME has Age and Gender as categorical features where the column values are the encoded integer values. However, the dataset used in LIME is not one-hot encoded and the encoding takes place internally when the model is given as the input.

Since the model builds a new linear model for each prediction, it is highly unstable since it can be significantly impacted by marginal changes in predictor values. Secondly, computing samples by perturbations for each data instance is very computationally expensive for a large dataset.

In the below example, we have plotted explanations for three individual instances which aims to deconstructs the model predictions based on feature contributions. It is observed that when girth has lower values, it usually leads to a negative impact on the weight of the donkey. However, when the value of girth is above a threshold, it contributes positively. Girth above 105 usually leads to above average weight prediction. Apart from girth, length is the second most important feature that contributes significantly to weight with a threshold of approximately 97 and has a similar effect on weight as girth. Height approximately has a threshold of 103 above which it leads to increasing weight predictions. Local explanations give reasonable understanding of the prediction which can be used to monitor the performance.

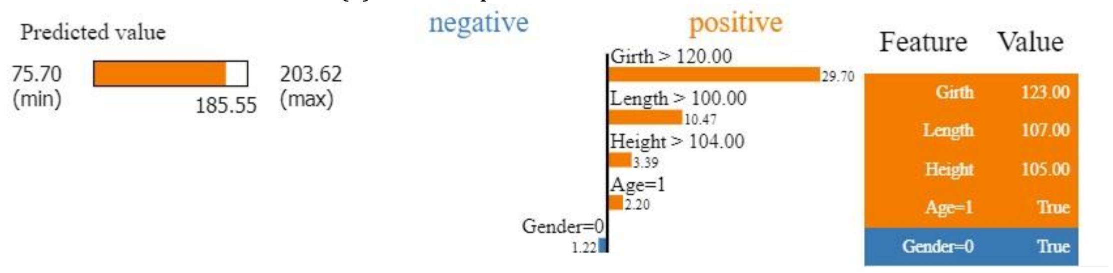


(a) Local Explanation for Instance 1



Feature	Value
Length	96.00
Girth	117.00
Height	101.00
Age=1	True
Gender=2	True

(b) Local Explanation for Instance 24



Feature	Value
Girth	123.00
Length	107.00
Height	105.00
Age=1	True
Gender=0	True

(c) Local Explanation for Instance 51

Figure 5.1.: Local explanation of individual instances reflecting feature importance for each data point in Model A using LIME

5.2.2. Understanding LIME for Global Explanations

LIME is widely used for understanding the model predictions on a global level to improve trustworthiness of the model. It aims to explain representative instances to explain the model as a whole. Choosing the instances holds significant value as it is a blueprint of the model performance and global importance values of features. Hence, we use submodular picks for explaining black box model predictions as proposed in Ribeiro et al. (2016).

Each user is assigned a quantity **B** representing the number of instances that need to be analyzed to gather global model understanding. **X** represents the instances where there are **n** instances. The model aims to choose instances that result in non-redundant explanations to reflect diversity resulting from different features. Each instance x_i has an explanation i.e. $g_i = \xi(x_i)$. We define an explanation matrix $(n * d')$ **W** where d represents the number of observations and it represents local importance of features for each instance. The global importance of feature j in the explanation space is represented by I_j and is defined as follows in Ribeiro et al. (2016, See Section 4):

$$I_j = \sqrt{\sum_{i=1}^{n} W_{ij}} \tag{5.4}$$

where $W_{ij} = |w_{gij}|$ in $g(z') = w_g \cdot z'$. Here, w_{gij} are the weight values calculated for explanation g of instance i for variable j as explained in Section 5.2.1. Features that contribute to a diverse range of explanations have higher importance scores. If a feature j is present in more instances as compared to feature k, then $I_j > I_k$. Selecting instances that maximize non-redundant coverage using significant number of features is represented by a function c and is defined as follows in Ribeiro et al. (2016, See Section 4):

$$c(V,W,I) = \sum_{j=1}^{d'} 1_{[\exists i \in V : W_{ij} > 0]} I_j \tag{5.5}$$

Here V represents the subset of X where $|V| \leq B$. Coverage function c is calculated for a set of instances V and computes the total importance of the features that appear atleast in one instance of V. The main challenge is finding V such that we can maximise c and is referred to as the pick problem, defined in the below equation in Ribeiro et al. (2016, See Equation 4):

$$Pick(W,I) = \underset{V,|V| \leq B}{\operatorname{argmax}}\, c(V,W,I) \tag{5.6}$$

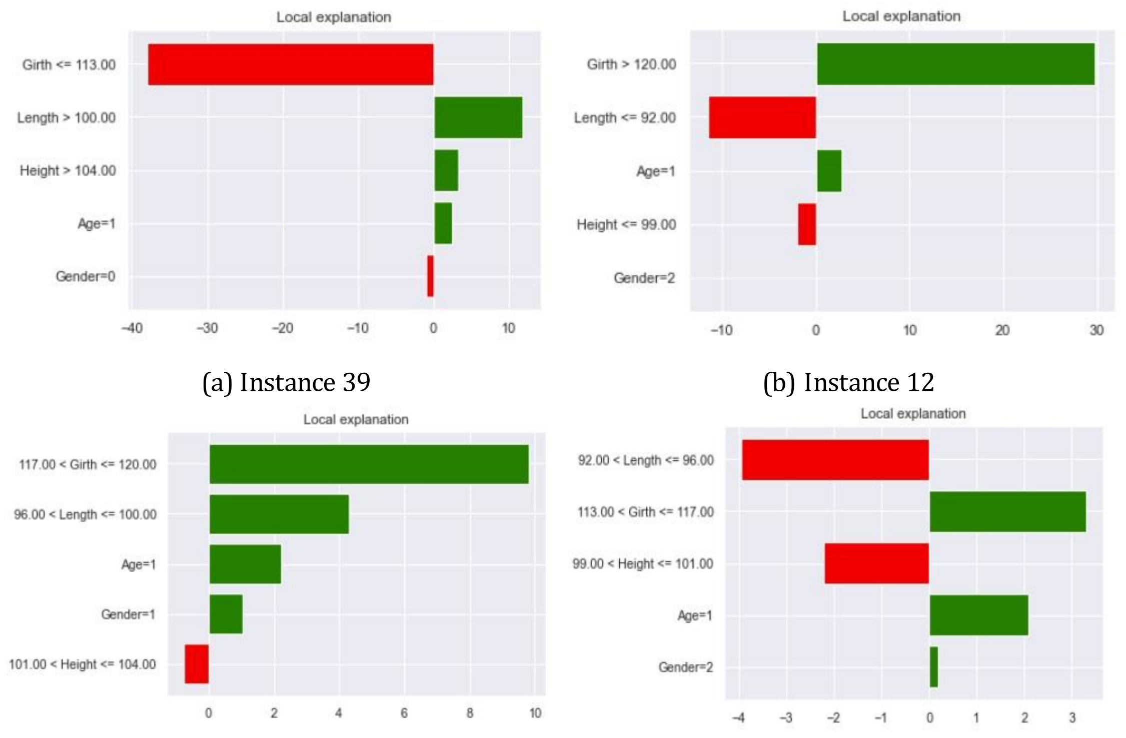
As mentioned in Ribeiro et al. (2016, See Section 4), let $c(V \cup \{i\},W,I) - c(V,W,I)$ be the marginal gain of adding an instance i to a set V. Due to sub-modularity, a greedy algorithm that adds an instance with highest marginal coverage gain iteratively to the solution using:

$$V \leftarrow V \cup \operatorname{argmax}_i c(V \cup \{i\}, W, I) \tag{5.7}$$

to the solution offers a constant factor approximation guarantee of $1 - \frac{1}{e}$ to the optimum. The approach described above is referred to as the sub-modular pick. This approach is proposed ~~in~~ based on the LIME paper Ribeiro et al. (2016, See Section 4).

Implementation of LIME on Black-Box Models for Global Explanation

In Figure 5.2, we plot four instances that have been automatically selected by the submodular pick lime algorithm which maximises the coverage. The instances have been selected to provide non-redundant explanations of the models through the different features using multiple instances. The observations that maximise the coverage in equation (5.2.2) are 39, 12, 27 and 38. This serves as a blueprint of different aspects of the model with respect to feature importance and aims to provide a representative explanation of the model ~~as a whole~~.



(a) Instance 39 (b) Instance 12

(c) Instance 27 (d) Instance 38

Figure 5.2.: Global Explanation of Model A by selecting 4 instances that maximise coverage function using SP-LIME

It is observed that Girth, Length and Height are the most important features that contributes to the model predictions. The global explanation of the model is in accordance with the local explanations and seem to have similar effect.

5.3. Shapley Additive Explanations

SHAP is an XAI algorithm which draws its basis from cooperative game theory and is explained comprehensively in Lundberg and Lee (2017). Its foundation is the calculation of Shapley values which is similar to calculating feature importance values. It is a model agnostic approach, however, it also has model specific explainer versions for deep learning and random forests as mentioned in DeepFindr (b). In game theory context, each feature can be treated as a player and the difference between the average prediction of the model and the prediction of a single instance can be considered as the payoff. Shapley values help determine the contribution of each feature/player to the payoff. It is usually a local approximation technique i.e. to understand individual predictions of black box models but is also used for global model explanation by aggregating individual predictions. Shapley values have same unit of measurements as the predictions and hence shapley values for different features of an instance can be added to explain a prediction. To explain it using a simple example, for a multiple linear regression model, $\hat{f}(x) = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p$, effect of feature j on a data point is the difference between the feature effect and the average effect : $\beta_j x_j - \beta_j E(X_j)$.

The shapley values are defined in the below equation (5.8) as proposed in Lundberg and Lee (2017, See Equation 4):

$$\phi_i(f,x) = \sum_{z' \subseteq x'} \frac{|z'|!(M - |z'| - 1)!}{M!} \left[f_x(z') - f_x(z' \backslash i)\right] \tag{5.8}$$

where z' is a binary mask of 1's and 0's representing the presence or absence of a feature. The shapley value calculated for a individual feature i, eg : Age = 70 where f refers to the black box model and x is the row of input and x' refers to an interpretable version of input x using a mapping function. We consider a collection of $z' \subseteq x'$ subsets which implies all combinations of features that are iterated over to account for interaction effects between features Lundberg and Lee (2017, See Section 2.4). The difference $[f_x(z') - f_x(z' \backslash i)]$ explains the contribution of feature i to the prediction for the subset z' which is referred to as the marginal value. $\frac{|z'|!(M-|z'|-1)!}{M!}$ is the weighting factor which represents the proportion of features present in the subset where M represents the total number of features present in the dataset. The SHAP algorithm is used to approximate each prediction f(x) by g(z') using a linear transformation i.e. a binary mask as proposed in
Lundberg and Lee (2017, See Equation 1) is defined below in equation (5.9)

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z_i' \tag{5.9}$$

where g is the explanation model, $z' \in \{0,1\}^M$, ϕ_0 is the base value or the average prediction of the model and $\phi_j \in \mathbb{R}$ refers to the shapley value for each feature.

There exists 2^M subsets and is very computationally expensive to calculate the shapley values for a single feature. Hence we can sample few subsets using Monte Carlo estimation. In the subset, the excluded feature was imputed by the average value of the feature in the dataset or a random value of the feature from any other instance in the dataset. If this procedure is repeated for all the subsets, the relevance of the features which need to be eliminated are usually sampled out due to randomization. Random features have no predictive power and therefore have insignificant impact on the model prediction as shown in DeepFindr (b). However, this is still considered an ineffective use of computational power for classic shapley value monte carlo estimation due to choice of random subsets.

Additive Feature Explanation have certain properties that are satisfied only by shapley values which are defined below and have been proposed

in Lundberg and Lee (2017, See Section 3):

1. Efficiency : The feature contributions i.e. shapley values of each feature for an instance needs to sum up to the payoffs i.e. the difference between average model prediction and local prediction.

$$\sum_{j=1}^{M} \phi_j = \hat{f}(x) - E_X(\hat{f}(x)) \tag{5.10}$$

2. Symmetry : Contribution of two features values j and k needs to be the same if they have equal contributions to all possible coalitions. If : $val(S \cup \{x_j\}) = val(S \cup \{x_k\})\ \forall\ S \subseteq \{x_1, x_2, .., x_M\} \backslash \{x_j, x_k\}$ (5.11) then: $\phi_j = \phi_k$ where S refers to the coalition of features.

3. Additivity : If model prediction is sum of two model components, then shapley values for the model should be the sum of model components. If combined payoffs are $val + val^{+}$, the respective shapley values are as follows: $\phi_j + \phi^{+}_j$.

4. Dummy : If a feature j does not change the predicted value of a prediction regardless of the coalition, its shapley value should be equal to 0.

$$val(S \cup \{x_j\}) = val(S) \qquad \forall\ S \subseteq \{x_1, x_2, .., x_M\} \tag{5.12}$$

then: $\phi_j = 0$

SHAP proposes an improved way for improving sampling efficiency using Kernel SHAP, a combination of LIME and SHAP proposed in (Lundberg and Lee, 2017, See Section 4.1). The drawback of determining the accurate $\pi_z(x)$ in LIME is fixed using Kernel SHAP. For sampling approximations, Kernel SHAP uses kernel-based approximation and is considered to be similar to LIME. It takes the original data instance and generates coalitions and makes predictions on it. Kernel SHAP fits a weighted linear model and choosing weights accurately can result in parameter coefficients of linear model representing shapley values for the features. Here the weight is a function of a distance function where the kernel indicates how to weigh each coalition. In LIME, model is trained on the local samples near the original data instance where the weight is the proximity measure which measures distance between the original data point and sampled data points in the neighbourhood. In SHAP, the kernel assigns weights to different coalitions or subsets which is defined as follows :

$$\pi_x(z') = \frac{(M-1)}{\binom{M}{|z'|}\,|z'|\,(M-|z'|)} \tag{5.13}$$

where z' is a binary mask of 1's and 0's representing the presence or absence of a feature. In Kernel SHAP, larger weightage is given to higher and lower coalitions. Lower coalitions are important because it gives significant impact of a feature on a prediction because it is more salient and results in more information gain. The weight is based on the number of input features instead of the absolute values of the features. In contrast to LIME, proximity measure in SHAP does not rely much on data type to extract proximity measures.

To illustrate, consider applying SHAP to the XGBoost model for predicting donkey Weight as a function of Age, Sex, Girth, Height and Length. Here, we use normal SHAP as M = 5 which implies only 2^5 subsets for each feature value. Hence, it is not very computationally expensive. SHAP is not sensitive to one-hot encoded as compared to LIME mentioned in Section 5.2.1 since new datapoints are not generated. In SHAP, subsets of existing datapoints are taken for calculating SHAP values of a particular value of an instance. The SHAP algorithm cannot handle internal encoding that was implemented for XGBoost for LIME and hence the data has to be manually one-hot encoded.

For instance, if the data point to be explained is **x**: Age = $y < 5$, Height = 95, Girth = 100, Gender = female and Length = 105. Here Age = $y < 5$ is one-hot encoded as [1,0,0] and Gender = Female is encoded as [0,0,1]. Hence the interpretable version of instance **x** is given as **x'** = [1,0,0,95,100,0,0,1,105]. To calculate the shapley value for girth = 100, we generate subsets/samples z' from x' where the length of x' and z' is the same. Features that are not included in the sample x' are replaced by the mean value of the predictor variable in the whole dataset. For example, given there are n data points, one subset z'_1 will have all the original values of **x'** except Height and Length keeping Girth fixed. These values are replaced by average height of 98 and average length of 103 from the dataset and resulting in a subset z'_1 = [1,0,0,98,100,0,0,1,103]. Model predictions using XGBoost model is calculated for sample z'_1 and for $z'_1 \backslash i$, girth value is replaced in z'_1 with the average girth value of the dataset. The difference between the $f(z'_1)$ and $z'_1 \backslash i$
is weighted with respect to the normalizing constant value in equation (5.8). Similarly, all possible subsets are calculated and the values are summed up to provide the shapley value for Girth = 100. These values are referred to as shapley value for observation i for variable girth ϕ_i. The shapley values are summed up for all the predictor variables. The average model prediction ϕ_0 is calculated by taking the average of each feature in the dataset and

using the black box model f to make a prediction on it. The resultant value ϕ_0 remains constant. The summation of ϕ_0 with the sum of shapley value results in model prediction for the individual data instance. This is an example of how SHAP deconstructs individual prediction values and gives insight into feature importance.

5.3.1. Implementation of SHAP on Black-Box Models

SHAP can provide local explanations similar to LIME without using a surrogate model. Figure 5.3 referred to as a **waterfall plot** is used to explain the local explanation for instance 24 where **E(f[x])** is the average predicted weight across all the observations in the test dataset and **f[x]** is the predicted weight for 24th observation in the test dataset. **E(f[x])** is 151.945 which is equivalent to ϕ_0 in equation (5.9),**f[x]** is around 149.775 equivalent to g(z') and M = 9. Height, Gender Female and Age between 5 to 20 contributes negatively to the prediction by reducing the value of weight by 2.49,0.41, 0.35 kgs respectively. Since there is not much difference between the average model prediction and the individual prediction, the shap values are relatively small.

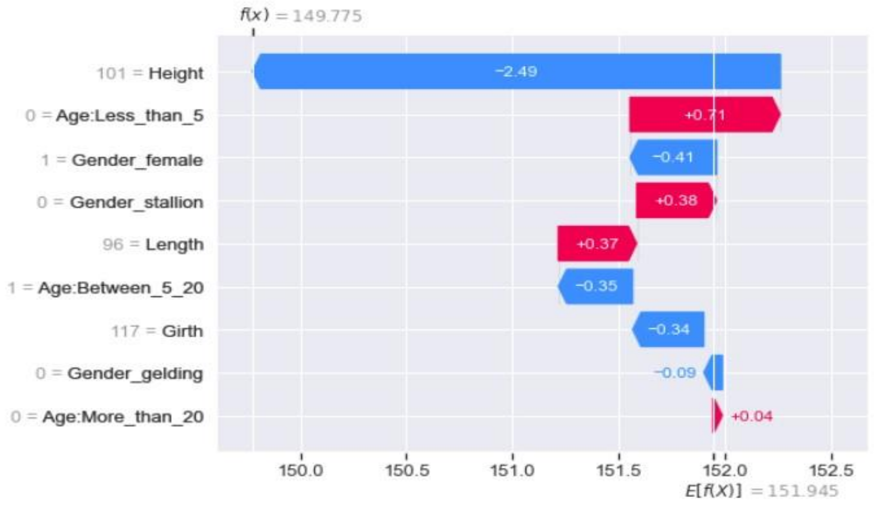


Figure 5.3.: Waterfall plot of an individual instance

In the graph below 5.4, we plot the **absolute mean shap** values for all the features and can be used to derive feature importance of variables to understand the impact that each variable has on model predictions. In accordance with the LIME global predictions in Figure 5.2, we see that girth has the highest impact followed by length, height, gender and age. We examine the absolute values of SHAP in order to preclude any mutual offsets between positive and negative values. The graph is extremely useful when number of variables are very high and we can perform feature selection based on I_j. The figure is based on absolute mean values i.e.

$$I_j = \frac{1}{n}\sum_{i=1}^{n}\left|\phi_j^{(i)}\right| \tag{5.14}$$

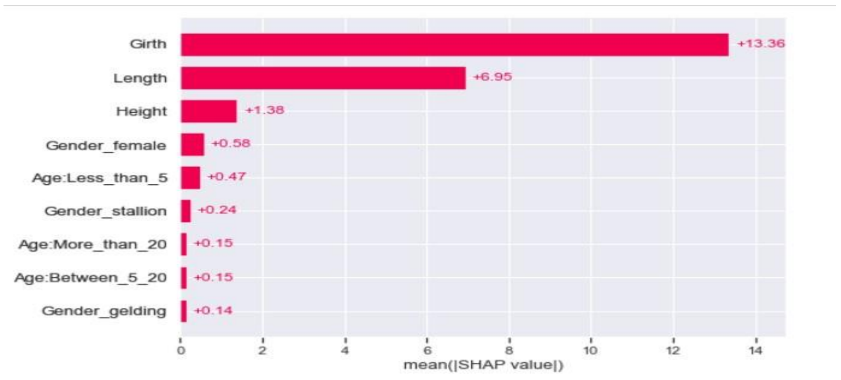
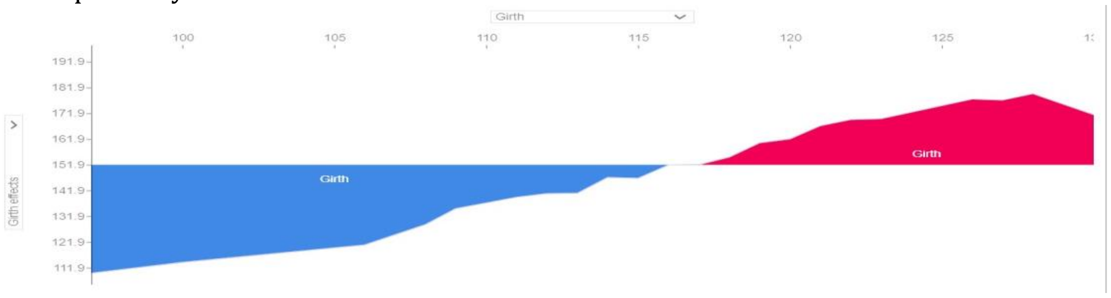
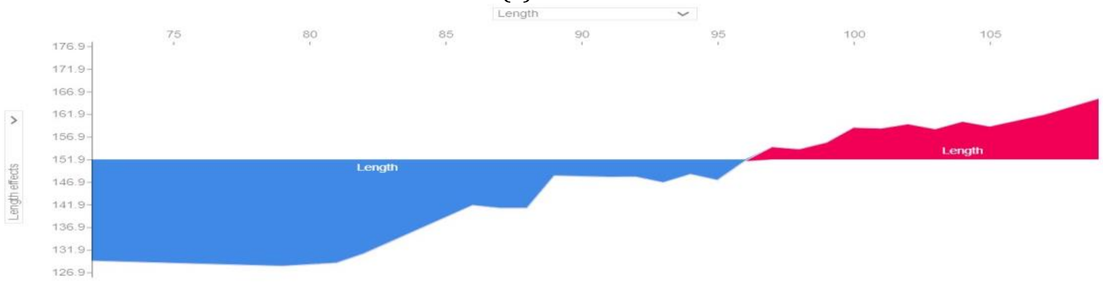


Figure 5.4.: Absolute Mean SHAP for Feature Importance for Model 2

SHAP is also widely used for model analysis where Figure 5.5 is a **stacked force plot** which represents vertically stacked force plots of individual predictions to give a understanding of the model as a whole. We can see that increasing girth and length values, leads to increasing shap values which has a positive impact on the response variable weight. The inflexion point above which girth and length starts impacting weight positively is 115 and 96 respectively.



(a) Girth



(b) Length

Figure 5.5.: Comparison of Girth and Length with its respective SHAP values

A **beeswarm plot** is an equivalent of a summary plot where we see that only for very high values of girth, the model is affected positively as SHAP values are in the range of +5 to +20. However, higher values of length has lower effect on the model in comparison to Length since the range of SHAP values is mostly between +8 and +15. For continuous variables, height is least significant in terms of affecting the model because even the highest height value can contribute approximately only by +8 kgs.

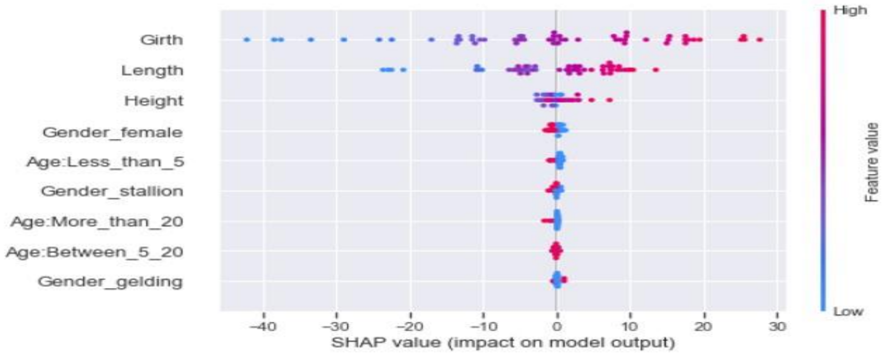
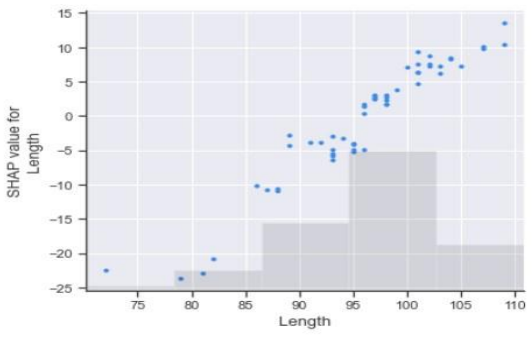
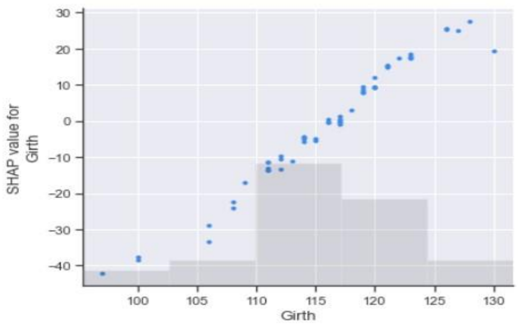


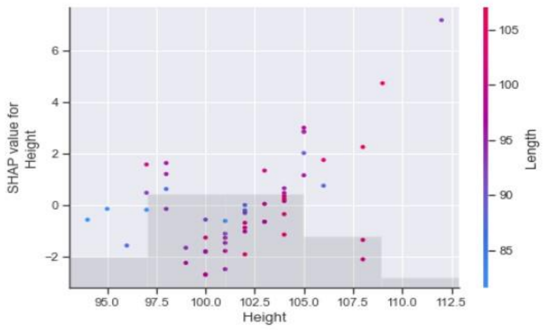
Figure 5.6.: Summary Plot of Features in Model B using SHAP



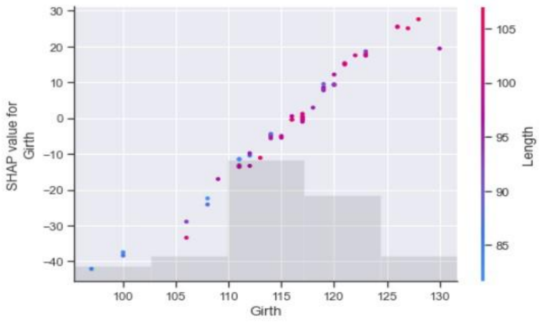
(a) Length Dependence Plot



(b) Girth Dependence Plot



(c) Length and Height Dependence Plot



(d) Length and Girth Dependence Plot

Figure 5.7.: Partial Dependence Plot of Features in Model B

In Figure 5.7 we plot **dependence plots** which is a plot of values of a feature against corresponding SHAP values and is particularly useful when a predictor variable has a non-linear relationship with the response variable. From the figure, we can understand that when girth has values below 115, it decreases the model predictions of weight. Length increases the weight of the model when its values are above 95. When its value ~~is below~~ 95, even extremely high values of girth don't impact the model positively. For relationships between length and height, it is observed that height as a non-linear relationship ~~and~~ higher values of length contribute negligibly to the weight of the model. However, for values of height between 85 and 103, higher values of length only contribute negatively. This implies that as a standalone variable, height does not impact the model hugely.

It is assumed that as a donkey gets older, there is an increase in the girth of the donkey. In order to verify the interaction effects, we draw a matrix plot which consists of the mean absolute values of the interaction effects between the 9 variables. It is shown that the one-hot encoded factor variables have very low interaction with the continuous variables. The highest interaction effect is observed between length and girth at 2 and girth and height at 1.2. Interaction between 2 variables x_i and x_j is calculated using the below equation :

$$\phi_{i,j} = \sum_{S\subseteq\setminus\{i,j\}} \frac{|S|!(M-|S|-2)!}{2(M-1)!}\delta_{ij}(S) \tag{5.15}$$

where $i/= j$ and

$$\delta_{ij}(S) = \hat{f}_x(S \cup \{i,j\}) - \hat{f}_x(S \cup \{i\}) - \hat{f}_x(S \cup \{j\}) + \hat{f}_x(S) \tag{5.16}$$

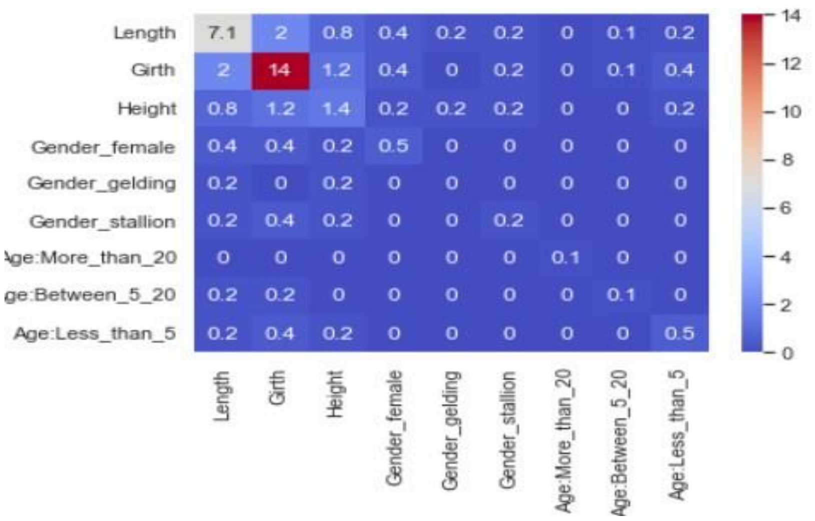


Figure 5.8.: SHAP interaction values between predictors of Model B

5.4. Comparison between LIME and SHAP approaches

There are certain similarities and dissimilarities between the post-hoc explanation methods, LIME and SHAP and it has been discussed below:

is

A key similarity between LIME and SHAP are that both can be used to understand local and global model predictions and can be used for feature selection when large number of predictor variables are present. Secondly, both can be used for different data types such as tabular data, text, images, etc and hence making it model agnostic. A key difference in LIME and SHAP is based on the condition of symmetry between two variables x_k and x_j which have shared information. In SHAP, the condition will give equal importance to two different predictors x_k and x_j if they contribute equally to a prediction. However, in LIME, if we use a LASSO Regression surrogate model, which used L^1 norm based on Manhattan distance, it will shrink the value of coefficient x_j to 0 and x_k is concluded to be the more dominant feature in the prediction.

LIME has a significant drawback that the local explanations are very sensitive to the **neighbourhood** of the local instance. Depending on the kernel width of the exponential kernel, explanations of the same instance can lead to contrasting conclusions. There is an ongoing study to determine the accurate kernel width which has a default value of $0.75 * \sqrt{d}$ where d refers to the number of features. There is no exact reason as to why the kernel width is chosen to be $0.75 * \sqrt{d}$ hence making LIME predictions very unstable. Secondly, LIME is not well equipped to handle **one-hot encoded data** since it performs perturbations of the data instance by sampling from Gaussian distribution without assuming any correlation between any features or considering the feature distribution which can lead to unlikely sampled data points. SHAP overcomes these drawbacks as it does not perturb the data instance, rather it takes its subsets from the existing data instance. The features that are missing in the instance are replaced by average values of the feature in the whole dataset. Hence, the subsets are generated remain meaningful to the dataset. Secondly, due to the additive nature of SHAP, summation of the individual SHAP values for the levels of a categorical variable will be equal to the SHAP value of the categorical variable. This offers more interpretability of grouped categorical variables as it allows users to gather importance of individual levels as well the categorical variable.

subsets
that

A drawback of SHAP is that it might create Frankenstein examples i.e. suppose when we take a subset of features to predict if an house price, and some of the factors include age and salary. Say we are calculating the shapley value of a feature k, and in a subset, salary is not part of it and age = 2. The value of salary is imputed by the expected value of salary from the complete dataset which say for example is GBP 30,000. This leads to Frankenstein example. Hence, when features are highly correlated, SHAP values should not be used.

6. Discussion

In this project we have discussed 2 distinct approaches to estimating donkey weights using a traditional statistical approach i.e. Shape Constrained Additive Models (SCAM) and a black box model involving ensemble techniques such as Extreme Gradient Boosted Trees. A shape constrained models with 3 monotonically increasing functions $f(\cdot), g(\cdot)$ and $h(\cdot)$ and 2 categorical features has been converted into a parallel scale nomogram using the Algorithm 2 which is based on the base Algorithm 1 of constructing parallel scale nomograms. Algorithm 2 can be further added as to the Pya and Pya (2023) for constructing parallel scale nomograms following the aforementioned conditions. The original hypothesis of of the project is that the XGBoost model will perform better in terms of predictive accuracy on the holdout set as compared to the SCAM and there would be a explainability - accuracy tradeoff between the two models with SCAMs having higher explainability and lower accuracy. However, the hypothesis has been proved wrong i.e. SCAM outperforms the XGBoost model. Hence, the SCAM has better predictive accuracy and interpretability since it can be visualized using a parallel scale nomogram and is preferred to calculate donkey weights as it allows the vet to understand how changes in length and girth are impacting the weight. This is because there is not much interaction between the features as evident in Figure 5.8. SCAM is more accurate because the model exploits the smoothness of the length and girth splines while not considering interactions and hence is not negatively impacted by lack of interactions in the model as compared to XGBoost model which focuses significantly on interactions that don't play a significant role in predicting donkey weights.

In the second half of the project, two model agnostic explainable AI (XAI) algorithms LIME and SHAP have been introduced to provide local as well as global explanations of the XGBoost model which is based on 4 continuous variables and 2 categorical features. In Chapter 4, Model A and Model B are two XGBoost models that are constructed due to configuration issues of XAI algorithms in Chapter 5 in handling continuous and unordered factors. The visualizations of the XAI algorithms verifies that the two important continuous variables for weight prediction are length and girth. In parallel scale nomogram, only 2 continuous predictor variables can be used for which length and girth are chosen, also verified by XAI visualizations. A further comparison to choose the suitable XAI algorithm has been discussed which allows the user to interpret a black-box model with proven accuracy. The LIME and SHAP libraries in Python an R are not model compatible with SCAMs and hence the model could not explain the predictions generated using SCAM and could not be deconstructed as compared to the local and global predictions for XGBoost in Chapter 5. A further inclusion of XAI algorithms to incorporate generalized additive models and its variants would be a desirable extension.

A. Appendix

PARALLEL SCALE NOMOGRAM CONSTRUCTION CODE

```
ylimits <- c(-1,3) scale_length <- 1
scale_width <- 1
##z = 1 + 2 * x + 3 *y

xfun <- function(x){
    1 + 2 * x
}

yfun <- function(y){
    3 * y
}

zfun <- function(z){ z
}

parallel_scale_nomogram_construction  <- function(xfun,yfun, zfun, xlim, ylim,scale_length,scale_width){

        ## create a new window with suitable shape dev.new(width = 5, height = 6)
        # inches ## create a plot outline to set the coordinates par(mar = c(2, 2, 2,
        2), las = 1, cex.axis = 0.9) plot.new() plot.window(xlim = c(0, 1), ylim = c(0,
        1))

        ##calculting different components

        #calculating the scaling factors m1 <-
        scale_length/diff(xfun(xlim)) m2 <-
        scale_length/diff(yfun(ylim)) m3 <-
        (m1*m2)/(m1+m2)

        #calculating distance between scales b <- 1/((m1/m2) +
        1) a <- 1 - b

        #calculating z limits
        zlim <- c(xfun(xlim[1]) + yfun(ylim[1]), xfun(xlim[2]) + yfun(ylim
       [2]))
```

```
        #generating a sequence of numbers for x,y and z respect xvals <- c(xlim,
        pretty(xlim,10)) xvals <- sort(unique(xvals[xvals %between% xlim])) yvals <- c(ylim,
        pretty(ylim,10)) yvals <- sort(unique(yvals[yvals %between% ylim])) zvals <- c(zlim,
        pretty(zlim,10)) zvals <- sort(unique(zvals[zvals %between% zlim]))

        #values of ticks on X axis x_func <- function(xval,m1){ x_val <-
        m1*(xfun(xval) - xfun(xlim[1]))
            return(x_val)
        }
        x_values <- x_func(xvals,m1)

        #values of ticks on Y axis y_func <- function(yval,m2){ y_val <-
        m2*(yfun(yval) - yfun(ylim[1]))
            return(y_val)
        }
        y_values <- y_func(yvals,m2)

        #values of ticks on Z axis z_func <- function(zval,m3){ z_val <- m3*(zval) z_val <- (z_val -
        min(z_val))/(max(z_val) - min(z_val)) #rescaling
       values between 0 to 1 return(z_val)
        }
        z_values <- z_func(zvals,m3)

        ##building the nomogram with modified axis() with custom labels
        #drawing the x axis axis(2, at = x_values, labels = xvals, pos = 0, xpd = NA) text(0,
        max(x_values), "X axis", pos = 3, xpd = NA, font = 2)
        #drawing the y axis axis(4, at = y_values, labels = yvals, pos = 1, xpd = NA) text(1,
        max(y_values), "Y axis", pos = 3, xpd = NA, font = 2)
        #drawing the z axis
        axis(2, at = z_values, labels = zvals, pos = a, xpd = NA) text(a, max(z_values), "Z axis", pos = 3, xpd =
        NA, font = 2) dev.print(pdf,"samp.pdf")

}

parallel_scale_nomogram_construction(xfun,yfun, zfun,xlimits,ylimits, scale_length,scale_width)
```

SCAM - PARALLEL SCALE NOMOGRAM CONSTRUCTION CODE

```
    nomogram_construction <- function(model_scam){

#extracting the dataset D dataset <- model_scam$model
#extracting the intercept cbar cbar <-
model_scam$coefficients[1] #extracting functions of length
and girth lists <- plot(model_scam)

#creating functions f(u), g(v) and h(w) f_u <- function(u){ f_u <- approx(lists[[2]]$x, lists[[2]]$fit,
xout = u)$y + cbar
    return(f_u)
}

g_v <- function(v){ g_v <- approx(lists[[1]]$x, lists[[1]]$fit, xout = v)$y
    return(g_v)
}

h_w <- function(w){
    h_w <- w
    return(h_w)
}



##creating a reference dataset freq_table <- table(dataset$Age,
dataset$Gender)
# Find the maximum frequency value max_frequency <-
max(freq_table)
# Get the row and column indices where the maximum frequency occurs max_indices <- which(freq_table ==
max_frequency, arr.ind = TRUE)
# Extract the corresponding Age and Gender values max_age <- rownames(freq_table)[max_indices[,
1]] max_gender <- colnames(freq_table)[max_indices[, 2]] reference_dataset = subset(dataset, Age ==
max_age & Gender == max_
   gender)

#predictions on the reference dataset
pred_new_scam <- predict(model_scam,reference_dataset)
#calculating the offset value
offset_value <- pred_new_scam[1] - (f_u(reference_dataset$Girth[1]) + g_v(reference_dataset$Length[1]))

    #redefininng f(u) using offset value f_u_offset <- function(u){ f_u_offset <- approx(lists[[2]]$x, lists[[2]]$fit,
    xout = u)$y + cbar
     + offset_value return(f_u_offset)
}
```

0
1

```
    #verigying the results reference_dataset$scam_pred <- pred_new_scam
    reference_dataset$nomogram_pred <- f_u_offset(reference_dataset$Girth)
      +                                g_v(reference_dataset$Length)


    ## create a new window with suitable shape dev.new(width = 5, height =

    6) # inches ## create a plot outline to set the coordinates

    par(mar = c(2, 2, 2, 2), las = 1, cex.axis = 0.9) plot.new() plot.window(xlim =
    c(0, 1), ylim = c(0, 1))
=======================================================================

#DEFINING THE FUNCTIONS OF u, v and w SCALE

=======================================================================

    #defining u limit ulim <- c(min(dataset$Girth),max(dataset$Girth))
    # defining u_values u_values <- pretty(ulim, n = 10) u_values <- sort(unique(u_values[between(u_values,
    min(ulim), max(ulim)
      )])) u_values_new <- seq(min(u_values),max(u_values), by = 1) u_values_new <-
    setdiff(u_values_new,u_values) u_values_new_labels <- rep(" ",length(u_values_new))

    #defining v limit vlim <- c(min(dataset$Length),max(dataset$Length))
    # defining v_values v_values <- pretty(vlim, n = 10) v_values <- sort(unique(v_values[between(v_values,
    min(vlim), max(vlim)
      )])) v_values_new <- seq(min(v_values),max(v_values), by = 1) v_values_new <-
    setdiff(v_values_new,v_values) v_values_new_labels <- rep(" ",length(v_values_new))

    #defining upper and lower limits of h(w) scale w_lower <-
    f_u_offset(ulim[1]) + g_v(vlim[1]) w_upper <- f_u_offset(ulim[2]) +
    g_v(vlim[2]) wlim <- c(w_lower,w_upper)



    #length of scale scale_length <- 1
    #scaling factors
    m1 <- scale_length/((f_u_offset(ulim[2]))-(f_u_offset(ulim[1]))) m2 <- scale_length/((g_v(vlim[2]))-
    (g_v(vlim[1])))
```

```
    m3 <- (m1*m2)/(m1+m2)

    #distance   between   scales   b   <-
    1/((m1/m2) + 1) a <- 1 - b

    #extracting a sequence of values for the weight scale w_values <- pretty(wlim, n =
    10) w_values <- seq(min(w_values),max(w_values), by = 5)

    w_function_values  =  m3  *(h_w(w_values)-  h_w(wlim[1]))  w_vec_range  <-  (w_function_values  -
    min(w_function_values))/(max(w_
       function_values) - min(w_function_values))

    sequence <- seq(min(w_values),max(w_values), by = 10) # Repeat the character "a" for
    the length of the sequence space_sequence <- rep(" ", length(sequence)) # Interleave
    the numbers and the "a" character w_labels <- c(rbind(sequence, space_sequence)) #
    Convert the sequence elements to character w_labels <- as.character(w_labels) #
    Calculate the length of the sequence seq_length <- length(w_labels)
    # Extract all elements except the last one extracted_sequence <- w_labels[1:(seq_length -
    1)]


    #constructing the nomogram axis(2, at = m1 *(f_u_offset(u_values)- f_u_offset(ulim[1])), labels =
       u_values, pos = 0)
    text(0, max(w_vec_range), "Girth, cm", pos = 3, xpd = NA, font = 2) axis(2, at = m1
    *(f_u_offset(u_values_new)- func_G_offset(ulim[1])),
       labels = u_values_new_labels, pos = 0, tcl = -0.25, lwd = 0, lwd.ticks = 1)

    axis(2, at = m2 *(g_v(v_values)- g_v(vlim[1])), labels = v_values, pos
       = 1) text(1, max(w_vec_range), "Length, cm", pos = 3, xpd = NA, font = 2) axis(2, at = m2
    *(g_v(v_values_new)- g_v(vlim[1])), labels = v_values_new_labels, pos = 1, tcl = -0.25, lwd = 0, lwd.ticks
    = 1)

    axis(2, at = w_vec_range, labels = extracted_sequence, pos = a,tcl =
       -0.5)
    text(a, max(w_vec_range), "Weight, kg", pos = 3, xpd = NA, font = 2)

}
```

Bibliography

Oct 2013. URL https://en.wikipedia.org/w/index.php?oldid=577854180.

Nomogram - academic kids, Nov 2021. URL https://academickids.com/ encyclopedia/index.php/Nomogram.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

H.R. Childress. What is a spline? (with pictures), Aug 2023. URL http://www.allthescience.org/what-is-a-spline.htm.

W. J. COOKSEY. Nomograms. *Transactions of the Faculty of Actuaries*, 20(175): 229–261, 1950. ISSN 00713686. URL http://www.jstor.org/stable/41218398.

DeepFindr. Explainable ai explained! — 4 lime, a. URL https://www.youtube.com/watch?v=d6j6bofhj2M&list=PLV8yxwGOxvvovp-j6ztxhF3QcKXT6vORU&index=5.

DeepFindr. Explainable ai explained! — 4 shap, b. URL https://www.youtube.com/watch?v=9haIOplEIGM&list=PLV8yxwGOxvvovp-j6ztxhF3QcKXT6vORU&index=5.

Ron Doerfler. On jargon-the lost art of nomography. *The UMAP Journal*, 30(4):457, 2009.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

Ryan McClarren. *Computational nuclear engineering and radiological science using python*. Academic Press, 2017.

Kate Milner and Jonathan Rougier. How to weigh a donkey in the kenyan countryside. *Significance*, 11(4):40–43, 2014.

Bottom of the Heap. Introduction to generalized additive models with r and mgcv. URL https://www.youtube.com/watch?v=sgw4cu8hrZM. A 3.5 hour lecture on Generlaized Additive Models.

Jason Osborne. Improving your data transformations: Applying the box-cox transformation. *Practical Assessment, Research, and Evaluation*, 15(1):12, 2010.

Natalya Pya and Maintainer Natalya Pya. Package 'scam'. 2023.

Bibliography

Natalya Pya and Simon N Wood. Shape constrained additive models. *Statistics and computing*, 25:543–559, 2015.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

Jonathan Rougier and Andrew Duncan. Bayesian modeling and inference for one-shot experiments. *Technometrics*, (just-accepted):1–17, 2023.

Josh Starmer. Xgboost part 1 (of 4): Regression, a. URL https://www.youtube.com/watch?v=OtD8wVaFm6E.

Josh Starmer. Xgboost part 3 (of 4): Mathematical details, b. URL https://www.youtube.com/watch?v=ZVFeW798-2I.

Simon Wood and Maintainer Simon Wood. Package ‘mgcv’. *R package version*, 1(29): 729, 2015.

Simon N Wood. *Generalized additive models: an introduction with R*. CRC press, 2017.