Prediction of oestrus intervals for guide dogs

Callum Ilkiw[a], **Satoshi Komuro**[a], **and Yiming Ma**[a]

[a]Mathematics for Real-World Systems II, University of Warwick, CV4 7AL Coventry, United Kingdom

This manuscript was compiled on June 16, 2022

It has long been known that the domesticated bitch comes into season approximately once every 7 months. Whilst previous research has looked at which features of a bitch might cause variation from this mean, results have often be inconclusive or contradictory. This study uses several machine learning techniques to produce predictive models which estimate the time between each bitch's oestrus periods, based on her unique features. Additionally, the paper comments upon which features influence this interval time the most, based on automated relevance detection methods. All data provided for this study comes from the Guide Dogs UK breeding programme with the interest of improving colony management and helping their production of assistance dogs. The data analysed consisted of 4693 observations of oestrus, between 877 unique bitches, over the years 2002 to 2019. Features analysed included age, breed, diet and 19 more. The best interval prediction model managed to limit the error to a mean of 26.45 days. This was a significant improvement over the mean 41.52 error produced by the current method. The best performing models were random forest regression, linear regression and a neural network built for this problem, with the random forest regression scoring the smallest mean error. On feature importance, the automated models found that the average of a bitch's previous seasons, whether a bitch had attempted mating or been pregnant last season and the bitch's breed all had the most significant impact on the length of her interval. Despite previous studies support for the concept, we did not find any evidence of seasonality in the oestrus intervals of these bitches.

guide dogs|dog breeding|oestrus intervals|machine learning|data science

1. Introduction

Guide Dogs UK currently has around 1300 puppies soon to go into training and help disabled people's lives throughout the UK. To ensure guide dogs are available to all those who need them, it has become important for Guide Dogs UK to keep producing a consistent number of new puppies, ready to be trained for future work. It is heuristically known that the bitches come into season every 7 months on average. Whilst previous biological work (1) has confirmed this average, they have also found a large amount of variation and been having a long running debate about which features of a bitch affect this interval time. Guide Dogs UK contacted the authors with the demand of a mathematically/statistically sophisticated model to predict these interval times. The objective of this project is to determine whether a prediction model can be built using historical breeding and population data that is more accurate than the current constant prediction and to validate this model for use by planners in the breeding programme. A secondary goal has also emerged, to analyse feature importance and report which features of a bitch seem to impact their oestrus interval.

A. Background.

A.1. Biological. Biological studies looking at the oestrus interval times of domesticated dog breeds have been as numerous as they are contradictory. Previous studies have mostly focused on some single feature of a bitch and its relation to the oestrus interval. **Seasonality** in bitches has been the most regular source of debate, seeming to stem from the observed seasonality in the dog's wild relatives (2). Whilst studies have supported this in both free-roaming and laboratory bitches (3–5), other studies found no link between the oestrus cycles of domesticated dogs and the season of the year (6, 7) . This includes our partners' study (1) that looked at exactly the same population, over the years 2005-2014 (Note: this paper has an extensive literature review on this one feature). The biological cause for this seems to be linked to day length, and is therefore, also dependant upon how controlled the bitch's environment is. Although our partners' previous study found no significant link, we will still be looking at seasonality in our models, with the hope of confirming their findings. The underlying concepts of day length, environment and even country will be less applicable to this study, as it solely looks at a controlled UK population. **Pregnancy** is one of the only agreed upon factors that affect a bitch's oestrus interval, with pregnancy always seeming to cause an increase on interval length, of 40-50 days (6). **Breed** has also been widely accepted as an important feature in predicting interval length. Linde-Forsberg and Wallen's paper (6) found that this link was more complex, where the effect that pregnancy had on a bitch's season, was determined by its breed. These findings imply that any model capable of predicting any bitch's interval, must have some level of complexity that can deal with these type of interactions. **Split seasons** are an additional biological concept that add complexity to this problem (8). Split seasons occur when a bitch shows signs of oestrus but is not actually able to breed. Following this, the bitch enters a

Significance Statement

For large breeding centres, such as Guide Dogs UK's breeding centre, colony management becomes an important thing to consider. This work is building a system that can create accurate interval estimates based off each bitch's individual profile, increasing colony management efficiency, saving manpower and improving life planning for the assistance dogs produced. This work also contributes to the long-standing question of what features of a bitch and her environment effect interval times. The project has taken a mathematical/data-driven approach where others have been more based in the biology. The hope is to answer these questions, or at least contribute to the discussion with a fresh angle.

"true" season after roughly 3-4 weeks. In the data supplied to us, this resulted in 2 data entries for each split, a 1st and 2nd half.

A.2. Data Science. The core of this project is based upon the Data Science for Social Good (DSSG) pipeline.(9) The construction of a pipeline allows for easy automation of the process and makes finding and resolving problems within the code a lot simpler.

The pipeline of this project is shown in Fig.1.

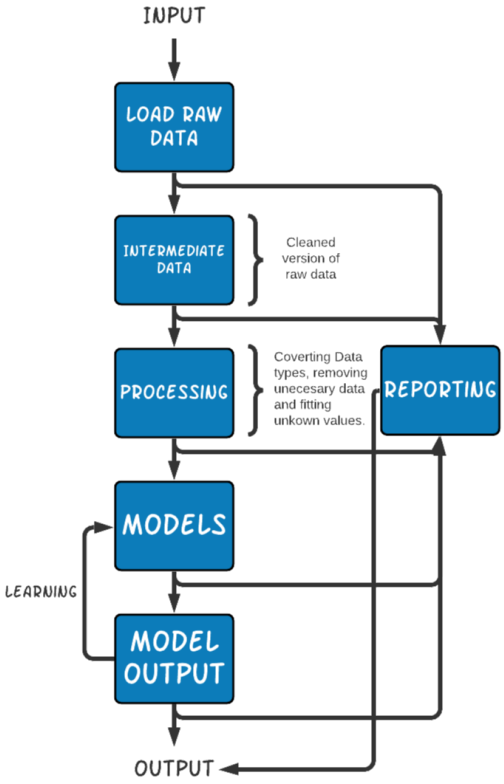

Fig 1. Pipeline of this project

Firstly, raw data was inputted. Then intermediate data was created to ensure the raw data was not accidentally changed or deleted over the course of this study. In pre-processing, the main problem was filling in missing values, methods used here included one-hot encoding of categorical data and feature selection. Following pre-processing, models were selected and trained before being evaluated and compared.

Most of the mathematical/data science problems faced in this project were solved imperially, by repeated computation (e.g. optimal models and hyper-parameters). A few of the more complex parts of building a predictive program are discussed here. **Model updating** after production of the model could self-produce bias to the future data we collect and predict. Detailed in the paper by Liley et al. (10) is a mathematical proof that updating an already-published model can result in a feedback loop, where future data points have been affected by the model's influence on the real world. Although some strategies are suggested in this paper, it seems the problem is unlikely to have serious impact on a biological model, such as the one in this paper. **Conditional confidence intervals** are a highly complex problem, that seems to have no universal solution (11). The problem becomes even more complex within the scope of this paper, looking at real-world data with an unknown distribution and many dimensions.

B. Data Description and Initial Analysis. The data analysed in this paper originated as one main file and two supplementary files. All of them were excel sheets collected by Guide Dogs UK staff, the initial purpose of which was to track the dogs, not to analyse. The data was recorded between January 2002 and February 2019. There are 4693 data points in total, with target variable `Time_from_previous_season` and many other variables. Note that the raw data contains some missing values and N/A. Fig.2 shows an example data point.

First to be analysed was `Time_from_previous_season`. Fig.3 shows the box plot for `Time_from_previous_season` and Table 1 summarises the important points.

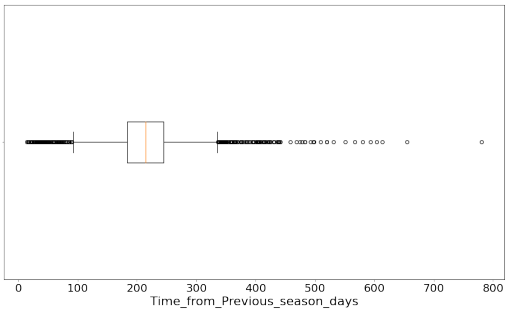

Fig 3. Box plot for `Time_from_previous_season_days`

Size	4064
Mean	216.68
Median(Q2)	215
Third quartile(Q3)	245
First quartile(Q1)	184
Outliers*	259

Table 1. Box plot description

The mean of the data is 216.68 days, which matches the currently accepted 7 months between oestrus periods. The median is 215 days, which coincides with the mean. Also, the

*Above than Q3 + 1.5 *(Q3 - Q1) or below than Q1 - 1.5 * (Q3 - Q1)

Data ID	Dog ID	Dog Name	Date of Birth	Breed Name	Colour	Sex	Number of previous pregnancies	Pregnant last season	Caesarean last season	Sire: Pedigree Name	Dam: Pedigree Name	Health Code	Season start date	Age at season	Time from previous season days	HR Notes	Diet when entered season	BCS when entered season	Weight when entered season	Optimum weight	Diet type	ON tx for disease?	What?
1	350634	VECTRA	24/1/2002	Golden Re	Yellow	Bitch	5	1	0	BRETT	URSUI	Season	4/2/2010	8.04	223.00	non-matin	Euk Mainte	Unknown	30.85	28	Adult		

Fig 2. Example data point

difference between the third quantile and median coincides with that between the first quantile and median, approximately a month. Last but not least, there are 259 outliers in this data. Given the limited size of data, this is not trivial.

Further analysis is done in Fig.4 and Table 2. Normal distribution and log-normal distribution were taken as examples, and fitted to the histogram.

To no surprise, both distributions gave p values 0.00 by χ^2 test.

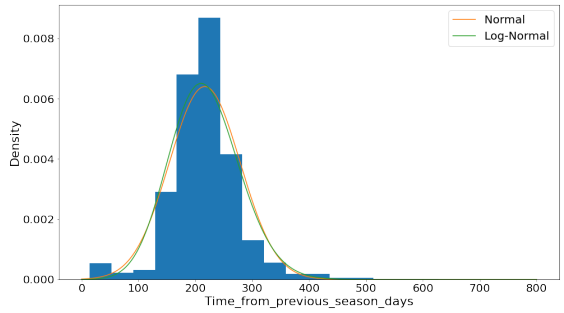

Fig 4. Histogram for `Time_from_previous_season_days`

Distribution	χ^2	p value
Normal	8.77×10^{11}	0.00
Log-Normal	2.73×10^{12}	0.00

Table 2. Fitting description

As well as the analysis of the target variable, analysis was performed on the rest of the data. Several of these results acted as "sanity checks" for future model development. Here are the main takeaways:

- Unique values:
  - 877 bitches
  - 191 Sires (Male parents)
  - 461 Dams (Female parents)
- At 2744 data points, pure Labradors take up a majority of the data.
- Conditional means:
  - `Cesarean_last_season`: `True` - 225.8 , `False` - 214.5.
  - `Breed`: Lowest is German Shepherds - 173, Highest is Golden Retrievers - 246.
  - `Pregnant_last_season`: `True` - 215, `False` - 217. After pre-processing and outlier removal, this changed significantly to: `True` - 241.2, `False` - 202.2.
- `"1st on breeding programme"` entered for 628 data points. Since this refers to the fist time a bitch has entered oestrus, these data points provided no useful information for model training.
- Data points per season: spring - 1208, summer - 1135, autumn - 1088 and winter - 1262.

These results support the validity and necessity of pursing a sophisticated, many dimensional, model.

2. Methodology

A. Pre-Processing.

A.1. Data Cleaning. The original data set was delivered in a messy state: there were numerous missing values and nonnumerical entries, so the first stage was data cleaning. Several techniques were used to cleanse the raw data, such as one-hot encoding categorical features [†]. Further details are shown in Appendix A.

A.2. Feature Extraction. Some features in the original data set are about current oestrus seasons, such as `Season_start_date`, but to enable the model to predict future `Time_from_previous_season_days`, it had to be trained based on data from the past. Hence, it is necessary to add new features like `Last_season_start_date`, `Age_at_last_season`, `Mating _last_season` and `Maiden_last_season`. Whilst the first 4 should be self-explanatory, it is necessary to clarify that `Maiden_last_season` is a binary feature that is true iff the bitch had never had sex directly following her last season.

Taking genetics into consideration, lengths of oestrus cycles of dams may influence those of their daughters, so the mean oestrus interval length of the dam of each dog was also extracted (if they were in the same data set), which is named as `Dam_season_interval_mean`. Also, as discussed in Wigham et al. (1), seasonality (spring, summer, autumn and winter) may have an impact on dogs' oestrus cycles, so this data was extracted as (`Season`) from the feature `Last_season_start_date`. Another new feature acquired is the mean of oestrus intervals of each dog (`Mean_previous_intervals`), with the assumption that interval lengths might oscillate around their means. Last but not least, the difference from the optimum weight may be more useful than the optimum weight itself, and as a result, it was also extracted, `Diff_from_opt_weight` from `Weight_when_entered_season` and `Optimum_weight`.

Table 3 summarises the features that have been obtained so far.

A.3. Data Splitting. It was decided that 60% of the data would make up the training set, 20% for validation and the other 20% for testing. Since the feature `Mean_previous_intervals` calculates the arithmetic average of previous oestrus interval lengths, the data had to be split in a temporal way to avoid data leakage [‡], which means the test set was composed of the newest 20% data, the validation set consisted of the next 20%, and the training set contained the oldest 60%.

On the training set, each models' parameters were tuned so that the training error is minimised. On the validation set, the hyper-parameters were tuned using exhaustive grid search [§]. As for the test set, this was used to estimate the generalised error of each model.

[†] One-hot encoding transforms a categorical variable into a group of bits (a vector), among which only a single bit is high (0) and the others are low (0). For example, suppose the feature `Colour` takes 3 values `"brown"`, `"golden"` and `"white"`. A valid one-hot transformation may encode `"brown"` as `[1, 0, 0]`, `"golden"` as `[0, 1, 0]` and `"white"` as `[0, 0, 1]`.

[‡] If a training set contains information of the test set, machine learning models can exploit this and "cheat" during evaluation on the test set. As a result, the estimated generalisation error will become inaccurate.

[§] Given a model of a certain type (e.g. the random forest) and a set of values of hyper-parameters, exhaustive grid search trains all models with different combinations of hyper-parameter values first, and then it evaluates their performances on the validation set and returns the model (or values of its corresponding hyper-parameters) with the lowest validation error.

Feature Name	Feature Type	Missing
Data_ID	int	✗
Dog_ID	int	✗
Dog_Name	int	✗
Date_of_Birth	int	✗
Sire_Breed	np.ndarray	✗
Dam_Breed	np.ndarray	✗
Colour	np.ndarray	✗
Sex	int	✗
Number_of_previous_pregnancies	int	✗
Pregnant_last_season	int	✗
Caesarean_last_season	int	✗
Sire_Pedigree_Name	int	✓
Dam_Pedigree_Name	int	✓
Health_Code	int	✗
Season_start_date	int	✗
Last_season_start_date	int	✗
Age_at_season	float	✗
Age_at_last_season	float	✗
Time_from_previous_season_days	int	✗
Mating	int	✓
Mating_last_season	int	✓
Maiden	int	✓
Maiden_last_season	int	✓
Diet_when_entered_season	np.ndarray	✓
BCS_when_entered_season	float	✓
Weight_when_entered_season	float	✓
Optimum_weight	float	✓
Diff_from_opt_weight	float	✗
Dam_season_interval_mean	float	✓
Mean_previous_intervals	float	✓
Season	np.ndarray	✓

Table 3. Summary of features after data cleaning and feature extraction.

A.4. Feature Selection: Stage 1. Some features, such as `Data_ID` are related to IDs of data. Other features like `Sex` are constant variables, which do not provide us with any information. As a result, they were excluded from the models. Since the co-existence of new features and those which have been used to extract them may lead to multicollinearity, which can bring about problems like over-fitting and non-convergence, these original features were removed manually. Features like `BCS_when_entered_season` have more than 50% data missing, so even it were possible to impute these missing values, the estimation would still be very inaccurate. Thus, these variables were also dropped

On the other hand, vector features were broken into scalars so that one datum of all input features can be transformed into a long vector. For example, `Season`, which has been one-hot encoded into a 4 bits, was decomposed into `Season_1`, `Season_2`, `Season_3` and `Season_4`, and each of them were scalars.

In summary, `Time_from_previous_season_days` is the output variable, which the machine learning models predicted, and the following features were manually selected as input variables.

A.5. Missing Value Imputation. `KNNImputer` (12, 13) was used with `n_neighbors = 5` to impute missing values.

- `Sire_Breed_1 - Sire_Breed_8`;
- `Dam_Breed_1 - Sire_Breed_8`;
- `Colour_1 - Colour_11`;
- `Number_of_previous_pregnancies`;
- `Pregnant_last_season`;
- `Caesarean_last_season_1 - Caesarean_last_season_3`;
- `Mating_last_season`;
- `Maiden_last_season`;
- `Diet_when_entered_season_1 - Diet_when_entered_season_13`;
- `Weight_when_entered_season`;
- `Diff_from_opt_weight`;
- `Age_at_last_season`;
- `Dam_season_interval_mean`;
- `Mean_previous_intervals`;
- `Season_1 - Season_4`.

List 1. Features left after manual selection

A.6. Outlier Detection. Since some algorithms are sensitive to outliers, the removal of outliers should lead to better predictions (14, 15). Note that "outlier" in this context refers to the explanatory variables sense, not the target variable `Time_from_previous_season_days`. So they were declared outliers based on their input variables, not their output.

Outlier detection algorithms are usually unsupervised learning, because the number of outliers is much smaller than that of the normal data-points (16).

Here, four types of outlier detection algorithms were adopted, and if three of them judged a data point as an outlier, it was then considered as an outlier and was deleted from the training data-set.

The four algorithms were:

- `EllipticEnvelope` (12, 17): assuming that the underlying distribution is Gaussian;
- `GaussianMixture` (12): assuming that the underlying distribution is Gaussian mixture.;
- `IsolationForest` (12, 18, 19): "isolating" observations by selecting a feature randomly and also a split value between the maximum and minimum values of the selected feature randomly as well;
- `OneClassSVM` (12): unsupervised outlier detection based on Support Vector Machine (SVM).

A.7. Feature Selection: Stage 2. All features were standardised to have zero mean and unit standard error. This technique is known to accelerate convergence.

B. Oestrus Interval Prediction. All implementations for these models, excluding the Neural Network (which was built by (20) and tuned by (21)), are provided by SciKit (12).

Let $\boldsymbol{x}$ be the vector of input variables after feature selection and y be the output variable `Time_from_previous_`

- `Sire_Breed_1 - Sire_Breed_8`;
- `Dam_Breed_1 - Sire_Breed_8`;
- `Colour_1 - Colour_11`;
- `Number_of_previous_pregnancies`;
- `Pregnant_last_season`;
- `Caesarean_last_season_1 - Caesarean_last_season_3`;
- `Mating_last_season`;
- `Maiden_last_season`;
- `Diet_when_entered_season_1 - Diet_when_entered_season_13`;
- `Weight_when_entered_season`;
- `Diff_from_opt_weight`;
- `Age_at_last_season`;
- `Dam_season_interval_mean`;
- `Mean_previous_intervals`;
- `Season_1 - Season_4`.

List 2. Features left after variance selection and random-forest selection

`season_days`. Denote the training set as $(\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}})$, the validation set as $(\mathbf{X}_{\text{val}}, \mathbf{Y}_{\text{val}})$ and the test set as $(\mathbf{X}_{\text{test}}, \mathbf{Y}_{\text{test}})$ [¶]. The aim of each machine learning model is to fit a function $f(\mathbf{X};\, \mathbf{W}_{\text{p}},\, \mathbf{W}_{\text{h}})$, where $\mathbf{W}_{\text{p}}$ and $\mathbf{W}_{\text{h}}$ represents the parameters and hyper-parameters of the model, such that the error measured under a certain metric $\|\, \mathbf{Y}_{\text{val}} - f(\mathbf{X}_{\text{val}};\, \mathbf{W}_{\text{p}},\, \mathbf{W}_{\text{h}})\|$ is minimised.

On the training set $(\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}})$, models were evaluated by the mean squared error, since it contributes to convergence by allowing optimisation algorithms to utilise the gradient. This means for a given combination of values of hyper-parameters $\mathbf{W}_{\text{h}} \in \mathbf{\Omega}_{\text{h}}$, the equation to be solved is:

$$\min_{\mathbf{W}_{\text{p}} \in \mathbf{\Omega}_{\text{p}}} \texttt{MSE}\left(\mathbf{Y}_{\text{train}}, f(\mathbf{X}_{\text{train}};\, \mathbf{W}_{\text{p}},\, \mathbf{W}_{\text{h}})\right). \qquad [1]$$

After solving Eq. (1) and denoting its solution as $\mathbf{W}^*_{\text{p}}$, models were evaluated by the mean absolute error [‖] on the validation set $(\mathbf{X}_{\text{val}}, \mathbf{Y}_{\text{val}})$ to find the best combination of hyper-parameters. Thus, in this step, the equation to solve is:

$$\min_{\mathbf{W}_{\text{h}} \in \mathbf{\Omega}_{\text{h}}} \texttt{MAE}\left(\mathbf{Y}_{\text{val}}, f(\mathbf{X}_{\text{val}};\, \mathbf{W}^*_{\text{p}},\, \mathbf{W}_{\text{h}})\right), \qquad [2]$$

where $\mathbf{W}^*_{\text{p}}$ is dependent on $\mathbf{W}_{\text{h}}$ and can be solved from Eq. (1). As for the test set $(\mathbf{X}_{\text{test}}, \mathbf{Y}_{\text{test}})$, it is used to estimate generalisation errors.

B.1. Baseline. The baseline model uses the arithmetic average of y to make predictions. Since there are no hyper-parameters in this model, the training set $(\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}})$ and the validation set $(\mathbf{X}_{\text{val}}, \mathbf{Y}_{\text{val}})$ can be concatenated to form a large test set $(\mathbf{X}_{\text{train\_val}}, \mathbf{Y}_{\text{train\_val}})$. For any input vector $\boldsymbol{x}$, the baseline predicts

$$f(\boldsymbol{x}) = \bar{\mathbf{Y}}_{\text{train\_val}}. \qquad [3]$$

B.2. Linear Regression. In addition to the standard linear regression (creates a linear equation of all features with a weight w_i then minimises the squared difference of its estimations to the true values), a regularisation term is added to prevent overfitting. (23) This type of regression is called "ridge regression". (24)

For any input vector $\boldsymbol{x}$, linear regression predicts

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x},$$

where

$$\boldsymbol{w} = \arg\min(\|\, \mathbf{Y}_{\text{train}} - \boldsymbol{w}^T\, \mathbf{X}_{\text{train}}\, \|_2^2 + \alpha \|\boldsymbol{w}\|_2^2)$$

and α is a hyper-parameter estimated by validation data-set.

B.3. SVR. Using kernel functions, SVR transforms the data to be almost linearly separable. It then uses decision boundaries to make predictions. For any input vector $\boldsymbol{x}$, SVR predicts

$$f(\boldsymbol{x}) = \sum_{k=1}^{n} (\alpha_i - \hat{\alpha_i}) K(\boldsymbol{x}, \boldsymbol{x_i}) + b$$

where α_i, $\hat{\alpha_i}$, and kernel function K are hyper-parameters estimated by validation data-set.

B.4. Gaussian Process Regression. Assume $\{y;\, \boldsymbol{x}\}$ is a Gaussian process with $\boldsymbol{x}$ parameterizing the mean function and the variance function, then it is known that $[\mathbf{Y}_{\text{train}}, \mathbf{Y}_{\text{test}}]$ follows a normal distribution. The training set is used to estimate a prior distribution $f(\mathbf{Y}_{\text{train}};\, \mathbf{X}_{\text{train}})$, from which the posterior distribution $f(\mathbf{Y}_{\text{test}};\, \mathbf{X}_{\text{test}} \mid \mathbf{Y}_{\text{train}};\, \mathbf{X}_{\text{train}})$ can be derived by using the Bayes' theorem or the result from (25): suppose a random vector $\boldsymbol{y} = [\boldsymbol{y}_1^\intercal,\, \boldsymbol{y}_2^\intercal]^\intercal$ follows the multivariate Gaussian distribution with the mean $[\boldsymbol{\mu}_1^\intercal,\, \boldsymbol{\mu}_2^\intercal]^\intercal$ and the co-variance matrix

$$\begin{bmatrix} \mathbf{\Sigma}_{11} & \mathbf{\Sigma}_{12} \\ \mathbf{\Sigma}_{21} & \mathbf{\Sigma}_{22} \end{bmatrix}$$

then the conditional distribution $\boldsymbol{y}_1 \mid \boldsymbol{y}_2 = \boldsymbol{a}$ also follows the Gaussian distribution with the mean

$$\boldsymbol{\mu} = \boldsymbol{\mu}_1 + \mathbf{\Sigma}_{12} \mathbf{\Sigma}_{22}^{-1} (\boldsymbol{a} - \boldsymbol{\mu}_2)$$

and the co-variance

$$\mathbf{\Sigma} = \mathbf{\Sigma}_{11} - \mathbf{\Sigma}_{12} \mathbf{\Sigma}_{22}^{-1} \mathbf{\Sigma}_{21}.$$

After obtaining the posterior distribution, its mean, $\boldsymbol{\mu}$, was used to make point estimation.

B.5. Bagging K-Nearest Neighbour Regression. Given a feature vector $\boldsymbol{x}$, the k-nearest neighbours algorithm (26) finds the k nearest data points in $\mathbf{X}_{\text{train}}$ first and then averages their corresponding target values, either uniformly or weighted by the distances [**]. To be specific, let $\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(k)}$ be the k nearest neighbours of $\boldsymbol{x}$ in the training set, and let $y^{(1)}, \cdots, y^{(k)}$ be their corresponding labels. If `"uniform"` is the selected strategy, then the prediction is given by:

$$f(\boldsymbol{x};\, k) = \frac{1}{k} \sum_{i=1}^{k} y^{(k)}.$$

[¶] Since the shallow layers in a deep neural network will perform feature selection themselves, there was no need for any other feature selection techniques for it. Thus, the training, validation and test sets without feature selection of the second stage were used for this model.

[‖] Models evaluated under the mean absolute error are more robust against outliers (22).

[**] Not having an explicit training phase is an interesting property of the k-NN algorithm.

As for the "distance" strategy, the prediction is weighted by inverses of l_2 distances:

$$f(\boldsymbol{x};\, k) = \sum_{i=1}^{k} \frac{y^{(k)}}{\|\boldsymbol{x} - \boldsymbol{x}^{(k)}\|_2}.$$

To reduce the variance of a single k-NN estimator, randomness can be introduced by selecting m random subsets from the original training set first and then building m k-NN models on them independently. Then predictions are made on $\boldsymbol{x}$ by averaging the outputs from them:

$$f(\boldsymbol{x};\, m,\, k) = \frac{1}{m} \sum_{j=1}^{m} f_j(\boldsymbol{x};\, k),$$

where $f_j(\boldsymbol{x};\, k)$ is the k-NN model trained on the j-th subsets.

B.6. Random Forest Regression. Firstly, decision tree regression is a supervised machine learning method that make a prediction based on simple decision rules learnt through the training data-set.

Random forest regression consists of a certain number of decision trees and makes predicts as an average of these trees' predictions. This type of learning method is called ensemble learning. (27)

Hyper-parameters include the number of maximum tree depth, the number of minimum sample split size, the number of minimum sample leaf size, and the number of estimators.

B.7. Adaptive Boosting Regression. Suppose there is a weak regressor defined already, such as linear regression and decision tree regression model. An AdaBoost (28, 29) model fits the the regressor on $(\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}})$ several times and adjusts the weight of each sample during each training. This empowers later regressors to handle difficult cases better.

Initially, the weak learner $f_0(\cdot;\, \mathbf{W}_{\text{p}},\, \mathbf{W}_{\text{h}})$ is trained on the $(\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}})$, with the loss from each example $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}$ being equally weighted. Then a new learner is initialised $f_1(\cdot;\, \mathbf{W}_{\text{p}},\, \mathbf{W}_{\text{h}})$ of the same structure and trained on the same data set, with the loss from each example re-weighted by the loss of the previous model on this example. For example, if $\|y^{(i)} - f_0(\boldsymbol{x}^{(i)};\, \mathbf{W}_{\text{p}},\, \mathbf{W}_{\text{h}})\|$ is very large, then the model increases the weight of f_1's loss on this example $(\boldsymbol{x}^{(i)}, y^{(i)})$ in its cost function. Suppose this process is repeated for K times, so at the end, K regressors of the same type are obtained. To make a prediction on the unseen data, the AdaBoost model outputs a weighted average of the predictions of those weak learners.

B.8. Gradient Tree Boosting Regression. Just like AdaBoost mentioned above, gradient tree boosting (30, 31) also ensembles several weak learners (which are decision trees in this case) to form a much more powerful model.

Following Cheng's notation in (32), suppose successive iterations are performed K times. At stage k, let F_k be the current imperfect model. The goal is to improve this by adding an amendment tree f_k such that,

$$y \approx F_k(\boldsymbol{x}) + f_k(\boldsymbol{x}).$$

Which means $f_k(\boldsymbol{x})$ is used to fit the residual $y - F_k(\boldsymbol{x})$. $f_k(\boldsymbol{x})$ is selected to be the decision tree that minimises the mean squared error $\| \mathbf{Y}_{\text{train}} - F_k(\mathbf{X}_{\text{train}})\|_2^2$, and in the next iteration, let $F_{k+1} = F_k + f_k$. After these K iterations, F_K is used to generate predictions.

B.9. Neural Network. A neural network is also an iterative process. Suppose there is an L-layer neural network. For each layer l, it takes the output $\boldsymbol{a}^{[l-1]}$ from the previous layer $l-1$ as input, linearly transforms it by

$$\boldsymbol{z}^{[l]} = \mathbf{W}^{[l]} \boldsymbol{a}^{[l-1]} + \boldsymbol{b}^{[l]},$$

and then feeds it to a nonlinear activation function $f^{[l]}$ to generate the output of the current layer, which means

$$\boldsymbol{a}^{[l]} = f^{[l]}(\boldsymbol{z}^{[l]}).$$

Although deeper neural network models can usually be used to fit more complex functions, except potential overfitting, they also suffer from some optimisation problems, such as exploding or vanishing gradients (33). Thus, the number of hidden layers in this problem is set to be 6, and layers maximum sizes were also within certain limitations so that the network can generalise well. Unlike the other models, this neural network was built by Keras (20) and its hyperparameters were tuned by random search in Keras Tuner (21).

C. Feature Importance. This paper looked at 2 separate methods of feature importance detection offered by the SciKit-learn python package (12). The supervised learning methods used were random forest feature importance detection and ARD(Automatic Relevance Determination). It is important to note that, due to the different techniques each method uses, the importance values produced are not directly comparable.

C.1. Random Forest. Random forest regression provides an importance measurement which based on Gini Impurity. Gini Impurity is defined as follows:

For a node i in decision tree t, assume that there are c classes in this node. Let $p(i \mid t)$ be the probability that sample belongs to class i.

Then, Gini Impurity $I_G(t)$ is

$$I_G(t) = 1 - \sum_{i=1}^{c} p(i \mid t)^2.$$

Random forest algorithm measures feature importance through how much does dividing a feature contribute to reduce Gini Impurity. (34)

C.2. ARD. ARD regression is a Bayesian method. Beginning with an elliptical Gaussian distribution for the weights of every feature, it updates to maximise the log-likelihood of the data points observed. With the additional costs associated with higher weights, this leads to a sparse model with many weights near zero.

3. Results

A. Interval Prediction. The performance of each model on the test set was evaluated by 6 different metrics, and corresponding errors have been summarised in Table 4. As it shows, the best three models under the metric of the mean absolute deviation were random forest regression, neural network, and linear regression, with errors of 26.45, 26.54 and 27.67 days, respectively.

Fig.5 illustrates the comparison of true values and the prediction of these three models.

Metrics / Models	Mean AE	Median AE	Max AE	RMSE	R^2 Score	Explained Variance Score
Baseline	41.517590	29.400500	350.400500	59.602455	-0.012004	0.000000
Linear Regression	**27.666271**	18.391864	304.292438	43.263306	0.466796	0.468531
Support Vector Regression	30.706676	20.025322	345.603623	48.107523	0.340705	0.374717
Gaussian Process Regression	27.939698	**17.446348**	311.712937	44.641516	0.432283	0.436579
Bagging K-NN Regression	32.828029	21.709373	332.046778	50.697218	0.267813	0.277047
Random Forest Regression	**26.452921**	**17.948695**	**282.529606**	**40.515314**	**0.532381**	**0.536497**
AdaBoost + Linear Regression	35.831447	28.038653	290.827322	49.533579	0.301038	0.320889
AdaBoost + Decision Tree Regression	28.560564	19.391304	**289.521739**	43.195489	0.468466	0.469669
Gradient Boosting Regression	28.001512	20.156420	295.187005	**41.264630**	**0.514924**	**0.517879**
Neural Network	**26.541662**	**17.421860**	**290.635681**	**41.589225**	**0.507262**	**0.507433**

Table 4. Model comparison

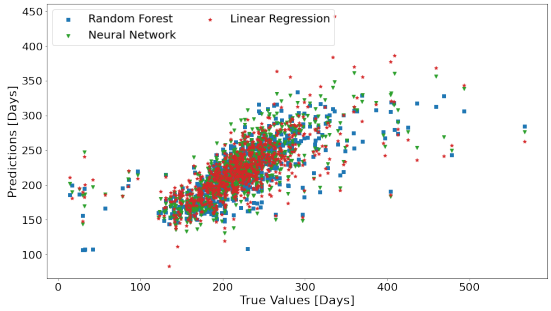

Fig 5. Comparison of true values and predictions of the best three models

B. Feature Importance. Results on feature importance come both from the feature importance models, and the coefficients of the linear regression prediction model.

The random forest feature importance model gives the results seen in the bar chart 6. The full results of the model can be found in Appendix C. According to this model, the most important features were `Mean_previous_intervals`, `Age_at_last_season`, `Dam_season_interval_mean`, `Mating_last_season` and `Pregnant_last_season`, with respective weights 0.3769, 0.1259, 0.1009, 0.06886 and 0.05361.

The **ARD results** can be seen plotted in Fig.7, and the full table of results is in Appendix B. The model has found the most important features to be, in order, `Mean_previous_intervals`, `Pregnant_last_season`, `Maiden_last_season`, `Mating_last_season` and `Dam_Breed_2`, with coefficients 29.46, 29.02, -15.98, 14.83 and -11.36, respectively. Where `Dam_Breed_2` refers to the binary option of whether the bitch's dam was a German Shepherd. `Mean_previous_intervals`, `Maiden_last_season` and `Mating_last_season` are all defined in A.2 of Section 2, whilst `Pregnant_last_season` is defined in Appendix A. Note that for ARD, importance is decided by the absolute value of its coefficients. Note: The weights between the 2 models are not directly comparable.

As well as the models directly looking at importance, the prediction models can also carry information about feature importance. The coefficients for linear regression are in Appendix D. The highest absolute coefficients are given for `Pregnant_last_season`, `Mean_previous_intervals`, `Caesarean_last_season_1`, `Mating_last_season` and `Dam_Breed_2` with respective coefficients 24.33, 24.32, 18.69, 15.02 and -13.64. Where `Caesarean_last_season_1` means the bitch had a caesarean last season.

4. Discussion

A. Interval Prediction. The first observation that can be made from these error results, is the **large reduction in error**. In the category of mean absolute error, the baseline model scored 41.5 days. Compare this to the random forest and neural network models which both scored around 26.5 days, and there is a difference of 15 days, showing how inadequate the current model is performing. There is still a large amount of error in all models. A lot of this error will simply be the result of working with limited real-world data for a biological problem. Specifically, the **maximum absolute error** is over 280 days in all models. Whilst this does show inaccuracy in the models, this is not a major disadvantage in terms of real-world estimation, since any interval this unusual would probably be caused by specific medical issues within the bitch. Since the end users of this will be professionals in dog breeding, they understand when to spot these anomalies.

Out of all available models, the best results were given by: linear regression, random forest regression, and the neural network. **Linear regression** performed far better than expected, as it is the simplest model excluding the baseline. Whilst it did not perform the best by any metric, it is worth noting that through the use of the linear regression coefficients in Appendix D, any reader could calculate a prediction for a bitch's oestrus interval time, and come away with a fairly accurate answer. The **random forest** and **neural network** compete at very similar levels. When all error metrics are taken into account, random forest is giving the smallest error.

A.1. Error Analysis. The box plots of absolute errors of the best three models, as well as the baseline, are shown in Fig.8. As it shows, for the top performing models, medians of errors were close to 17 days, much smaller than that of the baseline, which is approximately 29. Combined with the fact that the interquartile ranges of the best models were also narrower than that of the baseline, it demonstrates that significant improvement has been made in reducing prediction error. Additionally Fig.8 shows some large errors for all four models, implying the original data set was extremely noisy, in a way no model may ever be able to compensate for.

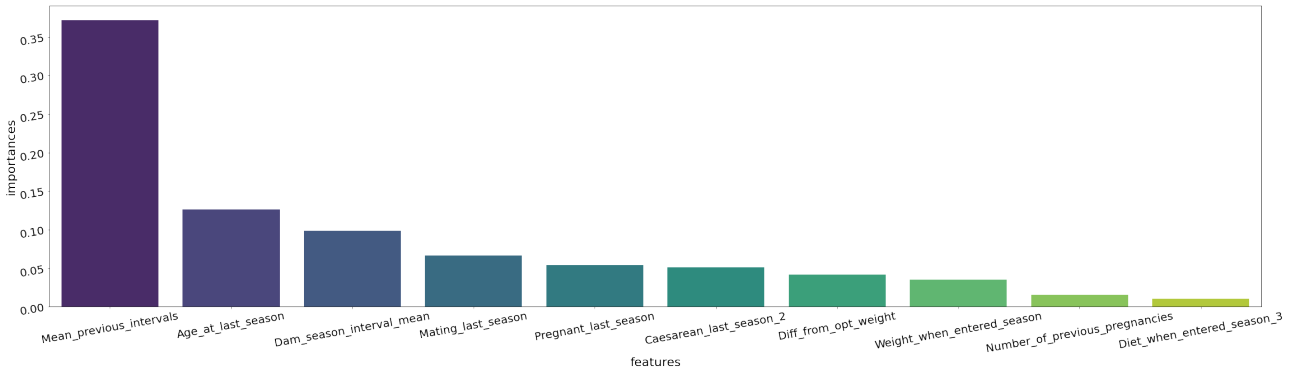

Fig 6. Bar graph showing the importance of features according to the random forest model

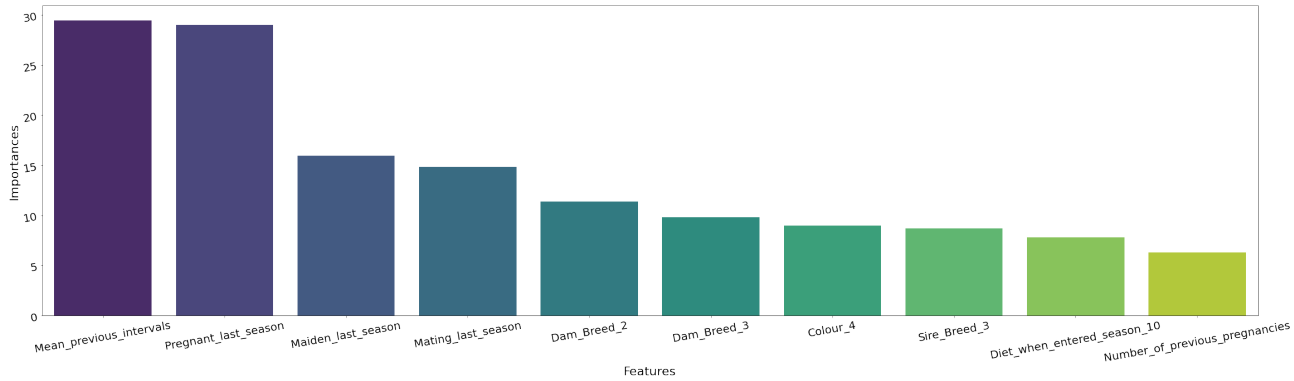

Fig 7. Bar graph showing the importance of features according to the ARD model

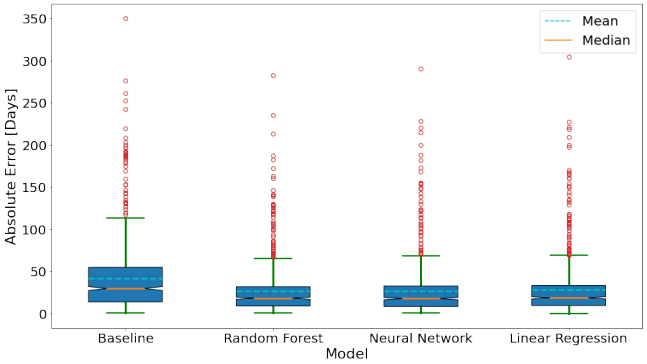

Fig 8. Box plots of absolute errors on the test set

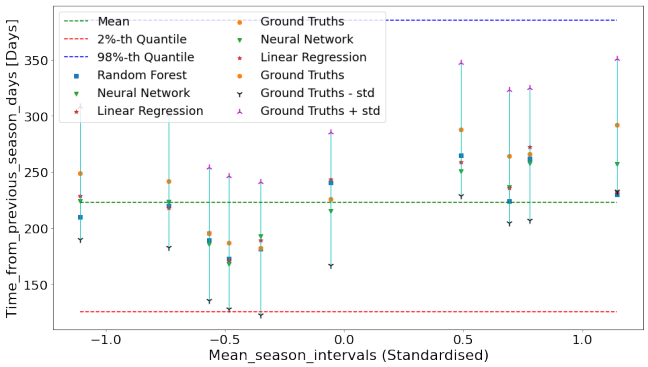

Fig 10. Distributions of 10 randomly selected data points

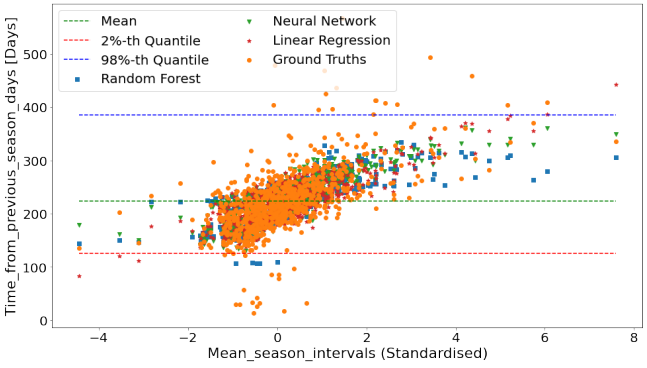

Fig 9. Distributions of predictions and ground truths

In section 3, all of the three feature importance models indicate that the most important feature is `Mean_season_intervals`, so a 2-D scatter plot is included to show distributions of predictions and ground truths against it. Fig.9 shows that most data points lie within the 2nd percentile (125.16) and 98th percentile (385.36) of the target variable, predictions of the best three models are much more accurate predictions [††]. As for data whose `Time_from_ previous_season_days` are beyond this range, which might be classified as outliers, the better models can have larger prediction errors. These 4% data need to be examined further, due to their irregularly small or large values.

[††] This can be seen much more clearly in Fig.10.

A.2. Interval Estimation Using Quantile Regression. Quantile regression is an extension of linear regression which predicts the quantile instead of the mean. It is known that quantile regression is robust to outliers (35).

Given that linear regression performs unexpectedly well, the possible solution to improve the prediction precision of outliers is interval estimate using quantile regression.

Firstly, 95% and 5% quantile regression were carried out. The result is shown in Fig.11.

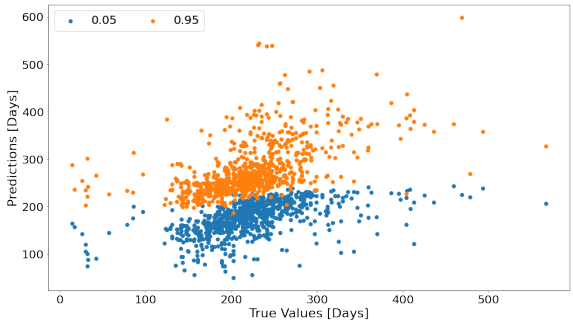

Fig 11. Quantile regression for 0.05 and 0.95 quantile

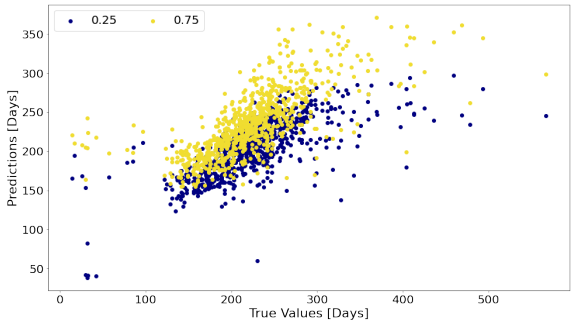

Fig 12. Quantile regression for 0.25 and 0.75 quantile

It appears that 81.6% of the data points are within this interval. Which is fairly impressive. However, the average confidence interval length is 100.87 days, which is too long for practical purpose considering that oestrus intervals are more or less 210 days.

Next, a 75% and 25% quantile regression were tested. The result is shown in Fig.12.

In this case, the average confidence interval is 33.3 days, which is not very bad as before. However, this confidence interval contains 42.1% of data and is not great.

To be summarised, as of now quantile regression is not an insightful method, although it is potentially an interesting direction.

A.3. Outlier Detection. Another alternative solution for outliers is applying outlier detection algorithms for the testing data sets. Namely, they try to detect the abnormality in the target value `Time_from_previous_season_days` though the explanatory variables. If it is possible, then it may advantageous to build other special models for the data points judged automatically as outliers.

Here, 4 algorithms were tried, all of which have been explained earlier in A.6 of Section 2. Fig.13 - 15 show how many times each data point was detected as a outlier.

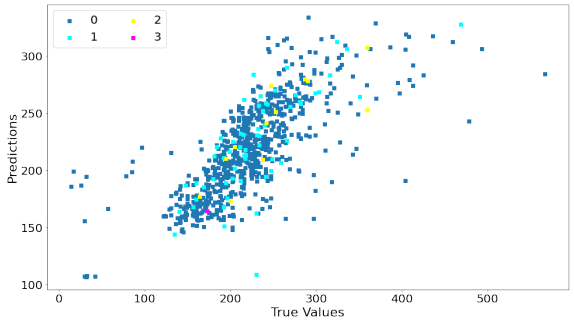

Fig 13. Outlier detection for the target variable in random forest

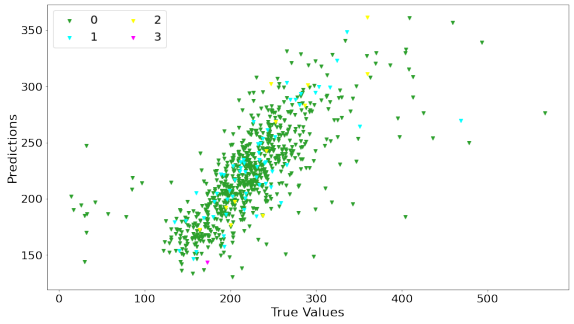

Fig 14. Outlier detection for the target variable in neural network

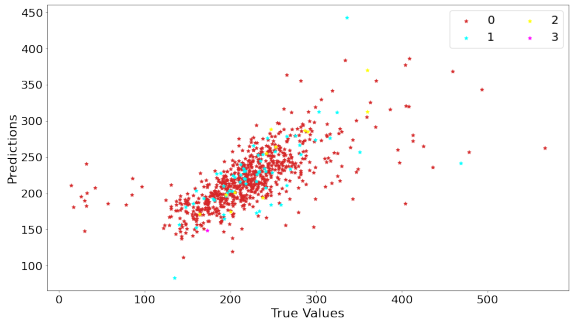

Fig 15. Outliers detection for the target variable in linear regression

These graphs help to show that outliers detected by outlier detection methods are not particularly abnormal in the target variable in any method. Hence, outlier detection algorithms are not very helpful here.

B. Feature Importance. The results for feature importance vary significantly between the 3 models developed for this project. This is explained by the wildly different implementations of the 3 models. In all 3 evaluations, 3 features were positioned in the top 5 most important: `Mean_previous_intervals`, `Mating _last_season` and `Pregnant_last_season`. `Mean_previous_intervals` is a feature created for this project based off the mean values of any intervals from the same bitch already in the data. With this being one of the top performing features, it gives evidence to each bitch having a level of internal consistency. This seems to go slightly against the findings of Bouchard et al. 1991 (36) that found inter-bitch correlation with interval times, was very low when compared to intra-bitch correlation. Although this paper finds mean previous intervals are inaccurate predictors, which this study's findings would agree with, these findings show that it can contribute well to an estimation when used in the presence of other features. `Pregnant_last_season` is a binary feature that is `True` if and only if the bitch in question was pregnant last season. A large, positive, importance was always expected for this feature from a biological sense, as the time spent pregnant normally adds time to a bitch's entire breeding cycle. Our initial analysis, showed conditional averages of 215 for bitches with `Pregnant_last_season` being `True` and 217 for those without. This significantly changed after pre-processing was performed and outliers were omitted, giving an average of 241.2 with pregnancy and 202.2 without. This implies that a significant number of outliers were pulling down the average for those bitches that had been pregnant and pulling up the average for those that were not. For the set of pregnant bitches, this can be most likely explained by the "split season" data that was removed for model fitting. Several of the data points which held the 2nd part of a split would still be referred to as `Pregnant_last_season`, giving some incredibly short interval times (less than 100 days). These were removed in the outlier detection stage. `Mating_last_season` also appears to have significant importance. Considering this from a biological angle, this is almost definitely due to its correlation with `Pregnant_last_season` which had a greater importance in all models except for random forest. There could, however, be other effects caused by mating, even if a bitch was not successfully bred, but this is beyond the authors' dog breeding knowledge.

With regard to breed, 2 breeds dominated the importance results. These are breeds 2 (German Shepherd) and 3 (Golden Retriever). In ARD, `Dam_Breed_2` and `Dam_Breed_3` took positions 5 and 6, respectively, whilst `Sire_Breed_3` took 8th place. In the linear regression model, `Dam_Breed_2` and `Sire_Breed_3` are in 5th and 6th, and dam breed 3 took 8th place. The random forest importance did not rank breed very highly, but when it does appear, `Sire_Breed_3` and `Dam_Breed_3` are the first amongst them. The most noticeable oddity of these results is the fact `Sire_Breed_2` is never given much importance by these models, implying that the effect being a German Shepherd has on oestrus may be inherited from the dam. These results on breed line up incredibly well with the initial analysis. Where it was discovered German Shepherd bitches have an average interval time of 173 days (the lowest of all breed averages) and Golden Retrievers have an average interval time of 246 (the highest). LR and ARD also provided a direction of impact these features had and, once again, it lines up with our analysis. `Dam_Breed_2` has a large negative coefficient in both models, but `Dam_Breed_3` and `Sire_Breed_3` have large positive coefficients in both models. Where a negative coefficient would imply a smaller season, and the opposite is true of a positive one. Note that the `Sire_Breed_2` coefficient is always positive, but since it is significantly smaller, it will have a lesser affect.

Seasonality in bitches was looked at for feature importance, but to mixed results. Whilst the random forest model rated the seasons quite highly in terms of importance, other models seem to give varying results. They have little impact, but `Season_2` (autumn) had the most, with a negative coefficient of roughly 5 days in LR and ARD. Compared with other features, this paper concludes that season does not seem to significantly impact oestrus interval time of the bitches in this study, confirming the results of Wigham et al. (1).

5. Conclusion

The initial analysis of this paper confirmed the currently accepted 7 month average for oestrus intervals in domesticated bitches. It also found that any model capable of capturing the full scope of variation from this average would need to be complex, and many-dimensional, in order to give accurate predictions for future interval times. Over the course of this study, machine learning models were successfully built to predict bitches' oestrus intervals. Techniques such as missing value imputation, outlier detection and standardisation were applied in pre-processing. All point estimation models developed were shown capable of reducing error significantly from the baseline, in spite of the existence of large internal noises in the data. Out of tested models, the random forest and the neural network developed for this project reduced mean absolute error the most (from a mean of 41.5 days to a mean of 26.5 days) , whilst linear regression was shown to be a suitable method for those looking for a simpler implementation (mean error of 27.7 days). Additionally, data-driven evidence showed that the mean of a bitch's previous seasons, their state of pregnancy last season and their breed, can have significant impact on her oestrus interval times. It found little evidence that seasonality, weight or diet have a noticeable effect on oestrus intervals.

6. Future Work

As was mentioned in Section 4, these models do not perform well when `Time_from_previous_season_days` takes extreme values. This probably means the original data set was greatly affected by noises. These data points need to be looker at further from both a data analysis and biological point of view. Once outliers are dealt with, this could either lead to a universal, improved, model or several additional "outlier models" that deal with specific cases, e.g. split seasons and illness. Additionally, this noise made the production of a good confidence interval model very difficult. So, in finally dealing with outliers better, a new pathway may be opened, allowing the creation of accurate confidence interval models. The Gaussian process regression model was ideal, because it can return the posterior distribution. However, due to the very existence of noises, the interval-estimate model shown in this paper was not useful in practice – either σ is appropriately small but with the ground truth unfortunately falling beyond

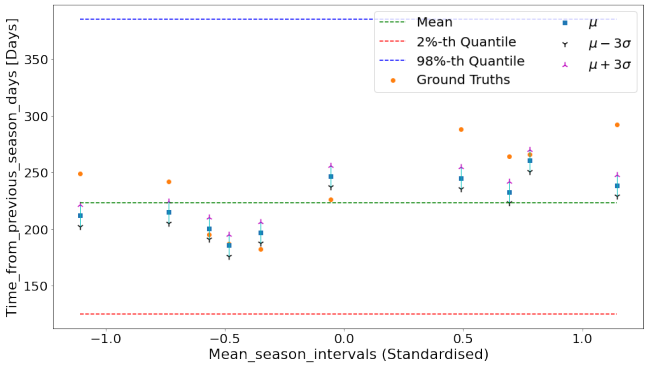

Fig 16. This estimate has short intervals but fewer ground truths fall into the them.

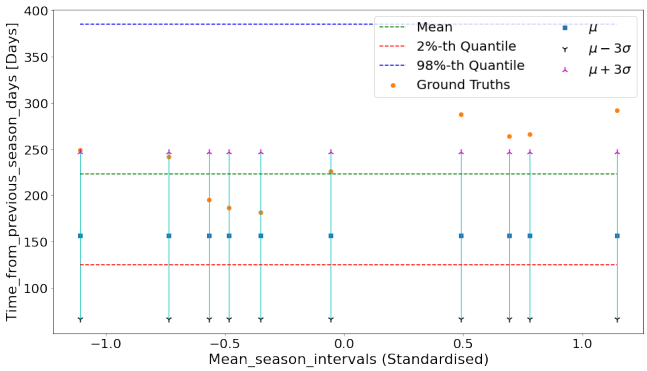

Fig 17. More ground truths fall into the predicted intervals, but the lengths are too large.

the 99.7% confidence interval $[\mu - 3\sigma, \mu + 3\sigma]$ (as shown in Fig.16), or σ is too large to make interval estimation reliable [‡‡] (as shown in Fig.17).

On the other hand, several features were not included in the data received, due to various constraints. One feature discussed in Linde 1992 (6) is that of litter size. This was a feature unavailable in the data, that could have had a serious effect on its results.

ACKNOWLEDGMENTS. This project could not have happened without the support of our supervisor Prof. Colm Connaughton. We would also like the acknowledge the help of Rachel S. Moxon, our contact with Guide Dogs UK. Thank you to EPSRC for funding this work.

1. Wigham EE, Moxon RS, England GC, Wood JL, Morters MK (2017) Seasonality in oestrus and litter size in an assistance dog breeding colony in the united kingdom. *Veterinary Record* 181(14):371–371.
2. Mech LD (1974) Canis lupus. *Mammalian species* N/A(37):1–6.
3. Christie D, Bell E (1971) Some observations on the seasonal incidence and frequency of oestrus in breeding bitches in britain. *Journal of Small Animal Practice* 12(3):159–167.
4. Chawla S, Reece J (2002) Timing of oestrus and reproductive behaviour in indian street dogs. *The Veterinary Record* 150(14):450.
5. Mutembei H, Mutiga E, Tsuma V (2000) A retrospective study on some reproductive parameters of german shepherd bitches in kenya: research communication. *Journal of the South African Veterinary Association* 71(2):115–117.
6. Linde-Forsberg C, Wallén A (1992) Effects of whelping and season of the year on the interoestrous intervals in dogs. *Journal of Small Animal Practice* 33(2):67–70.
7. Sokolowski J, Stover D, VanRavenswaay F (1977) Seasonal incidence of estrus and interestrous interval for bitches of seven breeds. *Journal of the American Veterinary Medical Association* 171(3):271–273.
8. Risvanli A, Ocal H, Kalkan C (2016) Abnormalities in the sexual cycle of bitches in *Canine Medicine*, ed. Kaoud HAE. (IntechOpen, Rijeka).
9. Donnelly B, et al. (2020) dssg/hitchhikers-guide.
10. Liley J, et al. (2020) Model updating after interventions paradoxically introduces bias. *arXiv preprint arXiv:2010.11530*.
11. Beutner E, Heinemann A, Smeekes S (2021) A justification of conditional confidence intervals. *Electronic Journal of Statistics* 15(1):2517–2565.
12. Pedregosa F, et al. (2011) Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.
13. Troyanskaya O, et al. (2001) Missing value estimation methods for dna microarrays. *Bioinformatics* 17(6):520–525.
14. Osborne JW, Overbay A (2004) The power of outliers (and why researchers should always check for them). *Practical Assessment, Research, and Evaluation* 9(1):6.
15. Stevens JP (1984) Outliers and influential data points in regression analysis. *Psychological bulletin* 95(2):334.
16. Pang G, Shen C, Cao L, Hengel Avd (2020) Deep learning for anomaly detection: A review. *arXiv preprint arXiv:2007.02500*.
17. Rousseeuw PJ, Driessen KV (1999) A fast algorithm for the minimum covariance determinant estimator. *Technometrics* 41(3):212–223.
18. Liu FT, Ting KM, Zhou ZH (2008) Isolation forest in *2008 eighth ieee international conference on data mining*. (IEEE), pp. 413–422.
19. Liu FT, Ting KM, Zhou ZH (2012) Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6(1):1–39.
20. Chollet F, , et al. (2015) Keras (https://keras.io).
21. O'Malley T, et al. (2019) Keras Tuner (https://github.com/keras-team/keras-tuner).
22. Ke Q, Kanade T (2003) Robust subspace computation using l1 norm in *CMU-CS-03-172*.
23. Bickel PJ, et al. (2006) Regularization in statistics. *Test* 15(2):271–344.
24. Hoerl AE, Kennard RW (1970) Ridge regression: applications to nonorthogonal problems. *Technometrics* 12(1):69–82.
25. Eaton ML (1983) Multivariate statistics: a vector space approach. *JOHN WILEY & SONS, INC., 605 THIRD AVE., NEW YORK, NY 10158, USA, 1983, 512*.
26. Altman NS (1992) An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician* 46(3):175–185.
27. Sagi O, Rokach L (2018) Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8(4):e1249.
28. Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* 55(1):119–139.
29. Drucker H (1997) Improving regressors using boosting techniques in *ICML*. (Citeseer), Vol. 97, pp. 107–115.
30. Friedman JH (2001) Greedy function approximation: a gradient boosting machine. *Annals of statistics* pp. 1189–1232.
31. Friedman JH (2002) Stochastic gradient boosting. *Computational statistics & data analysis* 38(4):367–378.
32. Li C (2016) A gentle introduction to gradient boosting (https://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf).
33. Hochreiter S (1998) The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6(02):107–116.
34. Wipf DP, Nagarajan SS, Platt J, Koller D, Singer Y (2007) A new view of automatic relevance determination. in *NIPS*. pp. 1625–1632.
35. John OO (2015) Robustness of quantile regression to outliers. *American Journal of Applied Mathematics and Statistics* 3(2):86–88.
36. Bouchard G, et al. (1991) Seasonality and variability of the interestrous interval in the bitch. *Theriogenology* 36(1):41–50.

7. Appendices

A. Data Cleaning. At this stage, the data was cleansed by converting nonnumerical values into numbers or arrays of numbers. Since feature extraction was added later, missing value imputation was performed after this.

A.1. Data Type Conversion. Features whose representations need to be changed are shown below.

- `Dog_Name`: The value type of this feature is `str`, so one way to deal with it is hashing.
- `Date_of_Birth`: After loading the spreadsheet, Python automatically convert strings like `2016/1/24` into `datetime` instances like `Timestamp('2002-01-24 00:00:00')`. Thus, what we need to do here is extracting information about the year (2002), month (1) and day (24), and then return an integer 20020124.
- `Breed_Name`: The data type of this feature is `str`, and this variable is categorical and takes 13 different values, such as `"Curly Coated Retriever"` [§§] and `"Golden Retriever x`

[‡‡] An ideal interval estimate should satisfy that the interval length is comparatively small and at the same time as many points as possible should lie within the interval. For example, a desirable estimate would predict $[145, 155]$ as the 99.7% confidence interval, when the ground truth is 150.

[§§] This means both the sire and the dam are curly coated retrievers.

`Flat Coated Retriever"` [¶¶]. Each name is going to be divided into two parts: `Sire_Breed` and `Dam_Breed`. Then we use one-hot encoding to process these two generated features.

- `Colour`: The data type of this categorical variable is also `str`, so we one-hot encode it.
- `Sex`: This feature takes only one value, which is `"Bitch"`, so just use `1` to replace this string.
- `Number_of_previous_pregnancies`: This variable takes 7 values `0`, `1`, ... , `6`. Although the range is really small, we still consider it as a continuous feature, because there is a cumulative effect in its value.
- `Pregnant_last_season`: This variable has 3 different values, which are `0`, `1` and `"unknown"`. We substitute `"unknown"` with `np.nan` first, and then the data entries of this type will be removed in cleaning `Time_from_previous_season_days` [***].
- `Caesarean_last_season`: This feature takes 4 values: `0`, `1`, `"N/A"` [†††] and `"unknown"`. Similar to the previous case, the data entries of with `Caesarean_last_season` being `"unknown"` will be dropped.
- `Sire_Pedigree_Name` & `Dam_Pedigree_Name`: The data of this variable is inconsistent: some have the format of `"BRETT (44182) Guidewell Beau 515392 (Dog)"`, and some will be loaded as `np.nan` by pandas (their original value is the empty string `""`). However, the only valuable information from this feature is the ID `44182` of the dog, so we just extract the ID number and abandon other information.
- `Health_Code`: This feature takes only one value, which is `"Season Start"`, so we use 1 (which is chosen arbitrarily) to replace it.
- `Season_start_date`: Do the same as what we have done in cleaning `Date_of_Birth`.
- `Age_at_season`: The data type of this variable is `float`, so we do not have to transform the data. Although there are some missing values, they can be inferred from `Season_start_date` and `Date_of_Birth`.
- `Time_from_previous_season_days`: There are two types of the data of this feature: strings and numbers. Strings are either `"unknown"` or `"1st on breeding programme"`, both of which stands for missing values. Thus, we replace them with `np.nan`, and we will drop such items later.
- `HR_Notes`: We divide it into 3 parts. The first part `split` takes three values: `[1, 0, 0]` which means the season was the first split, `[0, 1, 0]` which means the season was the second split, and `[0, 0, 1]` which means the season was not split. The second part `mating` indicates whether the dog mated (`1`) or not (`0`) during the last season. The third part `maiden` shows whether the dog was maiden (`1`) or not (`0`) during the last season.
- `Diet_when_entered_season`: This categorical variable has a lot of missing values, so we use the data from `diet 01 01 2006 to 24 08 2020.xlsx` to fit them first, and then we one-hot encode this feature. To be specific, in missing value imputation, we find the data of the same dog first, and then use the value of the closest recorded date to replace the missing value in the original data set.
- `BCS_when_entered_season`: Replace all missing values with `np.nan`.
- `Weight_when_entered_season`: Missing values are replaced with `np.nan`, and we can use the data from `bodyweight.xlsx` to fit them.
- `Optimum_weight`: Similar to what we have done in the previous case.

[¶¶] This means the sire is a golden retriever and the dam is a flat coated retriever.

[***] This is because the only data entry taking this value also has value of the output variable being missing.

[†††] `"N/A"` means not applicable, which is for those dogs which were not pregnant during their last seasons.

A.2. Removal of Data with Missing Intervals and of the First Split. Since items whose `Time_from_previous_season_days` are missing are not usable in supervised learning, we have to drop them. Also, some dogs may have split seasons, but during their first splits, they do not ovulate. Thus, there is no biological significance to predict such split, and we simply remove the data of the first splits (i.e. those whose `split = [1, 0, 0]`), and add `Time_from_previous_season_days` of the two splits together to make a whole season interval.

After data cleaning and this removal, features still having missing values are

- `Sire_Pedigree_Name`;
- `Dam_Pedigree_Name`;
- `Diet_when_entered_season`;
- `BCS_when_entered_season`;
- `Weight_when_entered_season`;
- `Optimum_weight`.

B. Full ARD Results. The full results of ARD are shown in Table 5.

C. Full Random forest Results. The full results of Random Forest are shown in Table 6.

D. Full LR importance Results. The full results of LR importance are shown in Table 7.

features	importances
Mean_previous_intervals	29.461497
Pregnant_last_season	29.020527
Maiden_last_season	-15.977564
Mating_last_season	14.831783
Dam_Breed_2	-11.355464
Dam_Breed_3	9.842916
Colour_4	8.962232
Sire_Breed_3	8.727026
Diet_when_entered_season_10	7.801767
Number_of_previous_pregnancies	-6.31461
Dam_Breed_7	-5.480089
Sire_Breed_7	-5.480089
Season_0	5.41143
Dam_Breed_4	4.423515
Season_2	-4.325435
Diet_when_entered_season_9	3.374292
Diet_when_entered_season_0	-3.280859
Diet_when_entered_season_8	-1.7692
Age_at_last_season	-1.165488
Colour_10	1.008385
Caesarean_last_season_1	0.827747
Dam_season_interval_mean	0.601252
Colour_8	-0.011213
Caesarean_last_season_0	-0.002931
Sire_Breed_2	-0.002619
Dam_Breed_1	0.002442
Dam_Breed_6	-0.002094
Colour_9	0.001991
Colour_2	-0.001442
Dam_Breed_5	-0.001256
Diet_when_entered_season_3	-0.001188
Diet_when_entered_season_6	-0.001029
Diet_when_entered_season_2	0.000817
Diet_when_entered_season_1	0.000812
Caesarean_last_season_2	-0.000775
Colour_7	-0.000701
Sire_Breed_1	0.000695
Colour_0	-0.000669
Diet_when_entered_season_5	0.00065
Season_1	0.00062
Colour_1	-0.000558
Season_3	-0.000494
Diet_when_entered_season_11	-0.000483
Diet_when_entered_season_7	-0.000309
Colour_6	-0.000278
Sire_Breed_5	-0.000256
Colour_3	-0.000253
Weight_when_entered_season	-0.000152
Sire_Breed_0	0.000076
Dam_Breed_0	0.000076
Colour_5	-0.000031
Diet_when_entered_season_4	0.000012
Diff_from_opt_weight	-0.000007
Sire_Breed_6	0.0
Diet_when_entered_season_12	0.0
Sire_Breed_4	0.0

Table 5. Automatic Relevance Regression Coefficients

features	importances
Mean_previous_intervals	0.371944
Age_at_last_season	0.126169
Dam_season_interval_mean	0.098669
Mating_last_season	0.066181
Pregnant_last_season	0.054241
Caesarean_last_season_2	0.051458
Diff_from_opt_weight	0.041766
Weight_when_entered_season	0.035529
Number_of_previous_pregnancies	0.015775
Diet_when_entered_season_3	0.010759
Season_1	0.010688
Sire_Breed_3	0.010342
Season_0	0.008079
Season_2	0.007793
Season_3	0.007671
Diet_when_entered_season_7	0.007585
Colour_10	0.006944
Caesarean_last_season_0	0.006391
Colour_0	0.006389
Diet_when_entered_season_1	0.006386
Diet_when_entered_season_8	0.005771
Dam_Breed_3	0.005723
Colour_7	0.004356
Dam_Breed_5	0.004285
Diet_when_entered_season_5	0.003519
Caesarean_last_season_1	0.003456
Sire_Breed_5	0.003438
Sire_Breed_2	0.003079
Colour_2	0.002711
Colour_5	0.002384
Colour_8	0.00182
Dam_Breed_2	0.001581
Maiden_last_season	0.001549
Colour_6	0.001331
Diet_when_entered_season_2	0.001283
Dam_Breed_1	0.000839
Sire_Breed_1	0.000553
Colour_4	0.000548
Diet_when_entered_season_0	0.000346
Colour_9	0.000303
Diet_when_entered_season_10	0.000136
Dam_Breed_4	0.00009
Colour_1	0.000072
Sire_Breed_0	0.000023
Colour_3	0.000017
Dam_Breed_0	0.000016
Dam_Breed_6	0.000013
Diet_when_entered_season_6	0.0
Dam_Breed_7	0.0
Sire_Breed_7	0.0
Diet_when_entered_season_12	0.0
Diet_when_entered_season_9	0.0
Sire_Breed_6	0.0
Diet_when_entered_season_4	0.0
Sire_Breed_4	0.0
Diet_when_entered_season_11	0.0

Table 6. Random Forest Coefficients

Features	Importance
Pregnant_last_season	24.325443
Mean_previous_intervals	24.319204
Caesarean_last_season_1	18.686875
Mating_last_season	15.017106
Dam_Breed_2	-13.636035
Sire_Breed_3	10.123664
Diet_when_entered_season_2	10.101656
Dam_Breed_3	9.031015
Diet_when_entered_season_3	8.763656
Colour_10	8.351856
Diet_when_entered_season_5	7.734572
Colour_0	6.927703
Diet_when_entered_season_1	6.898951
Diet_when_entered_season_7	6.489626
Season_2	-5.845605
Caesarean_last_season_0	5.638568
Season_0	5.046608
Number_of_previous_pregnancies	-4.723435
Dam_Breed_5	-4.091594
Dam_season_interval_mean	3.504125
Colour_2	-2.888650
Season_1	2.239755
Diet_when_entered_season_8	2.096886
Age_at_last_season	-1.896190
Sire_Breed_5	-1.828621
Season_3	-1.440758
Sire_Breed_2	1.252593
Colour_5	0.668362
Diet_when_entered_season_0	0.657098
Diff_from_opt_weight	-0.521505
Weight_when_entered_season	0.312684

Table 7. Linear Regression Coefficients