Predicting Astrophysical Properties of Stars

Contents

1 Introduction

Since the dawn of humanity, humankind has speculated and wondered at what lies beyond our earth in the outer space. Only in recent times, has it become possible to understand these speculations with proof. A recent mission developed by the European Space Agency (ESA) (Perryman et al. 2001), named the GAIA space mission has been launched, with the goal of developing a significant understanding of the origin, structure, and evolution of our galaxy. To do so, the largest and most precise catalogue of our galaxy has been built by gathering photometric data of approximately 1 billion astronomical objects within it.

One of the key pieces of information to be obtained from this mission is the accurate knowledge of the recorded stars' astrophysical properties. This is to be done by recording photometric measurements of these stars, which can be used in order to characterise their properties of effective temperature, gravity, and elemental composition.

1.1 Dataset

In order to successfully obtain these properties from the obtained photometric data, accurate predictive models must be developed. To do so, synthetic datasets containing values of the responses along with their photometric data have been created in order to train predictive models. The particular dataset comes from a standard library called the BaSeL2.2 library (Lejeune, Cuisinier, and Buser 1998), which was created to provide a realistic distribution of parameter count values, featuring stars of a high range of effective temperatures. This library contains colour calibrated flux distributions for each of these stars, which were then used to simulate the photon count readings for a specific (medium) waveband filter using the GAIA photsim simulator (Bailer-Jones 2002). This results in the dataset which featured 8,286 independent observations featuring normalised photon counts in different the 16 wavebands in the filter as well as the continuous response variables:

- Effective Temperature, T_{eff}
  - Units: Kelvin, K
  - Range: {2,000 - 50,000}
- Gravity, *logg*
  - Units: logg (logm/s^2)
  - Range: {-1.02 - 5.5}
- Metallicity, *M/H*
  - Units: M/H (proportion of metallic elements to hydrogen elements)
  - Range: {-5.00 - 1}

1.2 Aims

The primary aim of this project was to find the most the most accurate predictions of these three response variables given the information from the predictor variables. To do so, the best types of predictive models were found, the results of which were compared to select the one giving the best performance. Since the goal of training models for this mission was purely to obtain the most accurate predictions, inference of these predictors was not deemed important so was not explored in this project.

The secondary objective of the project was to quantify the uncertainty of these predictions, which was done by constructing predictive intervals for this dataset.

2 Methodology

The methodology behind attaining the best predictions of our model is to first analyse the dataset in order to better understand its underlying structures. Models would then be chosen with the goal of finding optimal predictive performance for this data. Four different models known for their accurate predictive ability, and capable of performing it on a non-linear dataset were selected for evaluation. A model selection methodology was employed in order to first tune the hyperparameters of each model, and then to evaluate the best performing model for these predictions.

2.1 Variance-Bias Tradeoff

The main issue to be tackled when working on the this type of predictive modelling is finding the balance between the model's bias and variance. When the model is too complex, it may fit to the training data too well including the errors of it, corresponding to high variance of the model, and a problem known as *overfitting*. When the fit of the model does not fit the training data well enough it would not capture it's full structure, which leads to a model with high bias and poor predictions. The main methodology of this project is to find this balance between variance and bias, which leads to the best overall predictions.

2.2 Model Selection

The methodology behind getting the most accurate predictions ultimately involved finding the best model to fit onto the data. There are two phases to model selection, the first is the tuning of a model's hyperparameters. A predictive model has hyperparameters which drastically affect its performance, making it crucial to select the most optimal set. Once this has been found for each model, their predictive performances were compared against eachother to select the best one used for the final predictions. To do this, a train/test split is performed onto the data. With this the test data is then held out in order to validate the final optimal models and perform model selection. The training data is used to tune the optimal hyperparameters of each model with cross validation, in order to prevent overfitting. A diagram of this data splitting procedure can be seen in Figure 1 which will be explained in more detail throughout this section.

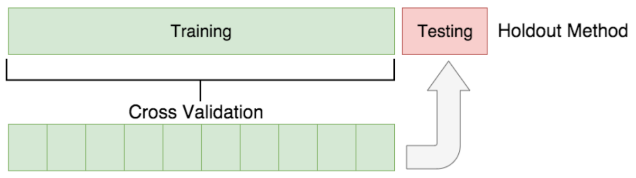


Figure 1: Diagram of Data Splitting Procedure (Raschka 2016)

The quantity that will be used as a measure of performance of a model is the Root Mean Squared Error (RMSE) of the predictions,

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where n is the total number of observations, and for observation i, y_i is its true value and $\hat{y}_i$ is its predicted value. This is a standard statistic used in predictive modelling which measures the average of the squared error, the square root of this is taken in order to have the same units as the measured values. With this, the RMSE can then be compared to the standard deviation of the response variable in order to get an indication of the overall prediction error.

To start the model selection process, a train/test split is performed onto the data, also known as a holdout set in order to validate the final model performance. The reason for this split is to prevent any possibility of optimistic bias in the model tuning phase. To do this, a stratified 20% split of the data is used as the test set. This ensured that a proportional representation of response values are used in the model validation process. Since the dataset is large, the size of the test set data is low enough to reduce pessimistic bias (Kohavi 1995) of the estimation whilst still having enough data to keep the variance low.

This training set was then used to perform k-fold cross validation on a carefully selected grid of possible hyperparameters for each model in order to tune each model. This entailed splitting the training set into k equally sized sets, and training a model onto all but one subset. This one held out subset is then used to test the model and calculate its performance. The training/testing process is repeated for all subsets, and the performance score is averaged between them. The value of k chosen in this study was 10, as this was computationally efficient, and recommended for a large dataset to have low variance and low bias (Kohavi 1995). After this, the tuned model was then used to calculate RMSE scores with the initially held out test set in order to perform model selection and assess the unbiased model accuracy.

2.2.1 Bootstrapping for Prediction Intervals

An equally important part to predictive modelling is to be able to quantify the uncertainty of them. To do so, intervals of these predictions were calculated using bootstrap resampling following the work by (Kumar and Srivistava 2012). Bootstrapping is a sampling technique which takes an equal size random sample of the data with replacement. This method of gaining prediction intervals was chosen as it is a nonparameteric method of finding intervals, which is useful for when error distributions are unlikely to be Gaussian. This method has also been shown to be very effective due to its accuracy and simplicity in design (Sullivan 2015). In this project, a less biased estimate of these intervals was used known as the ".632+" bootstrapping estimate which will be described.

2.2.2 Prediciton Intervals

Prediction intervals from bootstrapping can be derived from the overall error from a model. The error of a new observation x_0 fit onto a model can be split into three components. These are the *irreducible error* coming from the noise of the sample, as well as the model *bias* and *variance*. Each of these components must be calculated and used in order to find the estimated prediction interval at each new observation.

The model variance is estimated for each observation by bootstrapping a sample **B** times, fitting a model and generating predictions of the new observation for every bootstrapped sample, given by $\bar{y}_b(x_0)$. This is then used to estimate the mean $\hat{\mu}(x_0)$ of the real sample distribution $\hat{y}(x_0)$:

$$\hat{\mu}(x_0) = \frac{1}{B}\sum_{b=1}^{B} \bar{y}_b(x_0) \tag{1}$$

The estimate of the distribution of the model variance for this new point is then proved to be the centred value of these predictions given as $m_b = \hat{\mu}(x_0) - \bar{y}_b(x_0)$.

Then, to estimate the bias and the sample noise, the error between the actual value and prediction value is found, known as the *validation error*, $\widehat{\text{Err}}$. This can be found with a process similar to cross-validation, the procedure of validating the model with bootstrap resampling is called the *leave-one-out bootstrap.* This involves calculating the prediction performance for all observations which do not appear in the bootstrap sample. This can be written by,

$$\widehat{\text{Err}}^{(1)} = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{\mid C^{-i} \mid} \sum_{b \in C^{-i}} L(y_i, \hat{y}_i^b) \tag{2}$$

where $\mid C^{-i} \mid$ is the set of indices of the bootstrap samples that do not contain observation i, $\hat{y}_i^b$ is the prediction of the i'th term from the bootstrapped sample, b, and the loss function is the absolute difference between the two values.

The issue with this is that since bootstrapping takes samples with replacement, some of the datapoints appear more than once in the sample, whereas some do not appear at all. In fact, it has been shown that the average percentage of distinct datapoints in a sample is 63.2% which leads to a upward bias in the error measurement. The problem of the upward bias was addressed by using the average proportional knowledge to derive an estimator called the

".632" estimator (Efron 1983),

$$\widehat{\text{Err}}^{(.632)} = .368 * \overline{err} + .632 * \widehat{\text{Err}}^{(1)} \tag{3}$$

where $\overline{err}$ is the average training error. However, this was shown to break down in the case of overfit models, when the training error is too low (Efron and Tibshirani 1997). A solution was derived for this which takes into account of the amount of overfitting, the *relative overfitting rate*:

$$\hat{R} = \frac{\widehat{\text{Err}}^{(1)} - \overline{err}}{\hat{\gamma} - \overline{err}} \tag{4}$$

where $\hat{\gamma}$ is the *no-information error rate*, which comes from calculating the average error if the predictors were independent from the responses in the data. This is done by evaluating the error from taking the error from all combinations of predictors and responses

$$\hat{\gamma} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} L(y_i, \hat{y}_{i'}) \tag{5}$$

where i' is the observation that is not in i. With this, the final estimator was derived as:

$$\widehat{\text{Err}}^{(.632+)} = (1 - \hat{w}) * \overline{err} + \hat{w} * \widehat{\text{Err}}^{(1)} \tag{6}$$

where

$$\hat{w} = \frac{.632}{1 - .368\hat{R}} \tag{7}$$

This is known as the ".632+" bootstrap estimator of the overall error rate, which is what is used to estimate the bias and the noise of our predictions. This added with the estimate of the variance calculated previously gives us our estimated distribution of our predictions. This study will use this distribution to find estimate the 95% prediction intervals of the selected models.

2.3 Models

This section will describe the theory behind the models that were chosen to be used in the project, as well as their implementation details. The models used are given as Support Vector Machines, Neural Networks, Random Forests, and Stochastic Gradient Boosting. These models were chosen firstly due to their ability to predict non-linear relationships in the dataset, the necessity of which will be described in the next chapter. Support Vector Machines and Neural Networks are well known powerful prediction methods and have been specifically used in the previous GAIA studies of astrophysical property predictions (Willemsen, Bailer-Jones, and Kaempf 2004; Kaempf et al. 2005; Bailer-Jones 2007; Tsalmantza et al. 2009), so these were selected to be tested first. The remaining two tree-based ensemble models were chosen due to their exceptional predictive performance in various pieces of literature, such as Kaggle machine learning competitions (Taieb and Hyndman 2014) as well as in the model comparison studies written in (Hastie, Tibshirani, and Friedman 2009) and (Kuhn and Johnson 2013). For each of these chosen models, a grid of hyperparameters was chosen carefully from preliminary testing and research on the respected methods.

2.3.1 Principal Component Analysis

Principal Component Analysis (PCA) (Jolliffe and Cadima 2016) is a widespread technique used for to reducing dimensionality in an interpretable way. It does so by transforming variables into linear functions of themselves with the goal of maximising their variance. Each transformed variable is called a *component*, and is orthogonal to the one prior to it. For this reason, it can also be used as a means of tackling multicollinearity in a dataset. This section will describe this process in further detail, as well as describe the use of it in this project.

For a matrix of data, $\mathbf{X}$ containing n observations and p numerical variables, a linear combination of the matrix' columns is found in order to maximise its variance. This linear combination is given by $\sum_{j=1}^{p} a_j \mathbf{x_j} = \mathbf{Xa}$, with variance given by $var(\mathbf{Xa}) = \mathbf{a'Sa}$, where $\mathbf{a}$ is a vector of constants, $\mathbf{S}$ is the sample covariance matrix, and $'$ denotes the transpose. A boundary condition is imposed in order for the problem to have a well defined solution, which is usually requiring $\mathbf{a'a} = 1$. This problem then becomes equivalent to maximising $\mathbf{a'Sa} - \lambda(\mathbf{a'a} - 1)$, where λ is a Lagrange multiplier.

Differentiating with respect to vector $\mathbf{a}$, and equating to zero, produces the equation

$$\mathbf{Sa} - \lambda\mathbf{a} = \mathbf{0} \Longleftrightarrow \mathbf{Sa} = \lambda\mathbf{a} \tag{8}$$

This then becomes an eigenvector problem, where $\mathbf{a}$ is the (unit-norm) eigenvector, and λ is the eigenvalue of the covariance matrix $\mathbf{S}$. These eigenvalues are therefor the variances of the linear combinations defined by their corresponding eigenvector $\mathbf{a}$, with proof: $var(\mathbf{Xa}) = \mathbf{a'Sa} = \lambda\mathbf{a'a} = \lambda$. These eigenvalues are given in order from largest to smallest, corresponding to the largest variances. These variances are often measured by their *proportion of total variance*, π_j which they account for,

$$\pi_j = \frac{\lambda_j}{\sum_{j=1}^{p} \lambda_j} = \frac{\lambda_j}{tr(\mathbf{S})} \tag{9}$$

where tr($\mathbf{S}$) denotes the trace of $\mathbf{S}$.

It can also be noted that still remains valid if the eigenvectors are multiplied by -1, therefor the signs and scores of all loadings are arbitrary and only their relative magnitudes and relative directions are meaningful.

Since covariance matrix $\mathbf{S}$ is a $p * p$ real symmetric matrix, it has exactly p real eigenvalues, and their corresponding eigenvectors can be defined to form an orthonormal set of vectors. It is shown, (Jolliffe and Cadima 2016) that with this added orthogonality constraint, the full set of eigenvectors of $\mathbf{S}$ are the solutions to the problem of obtaining p new linear combinations $\mathbf{Xa} = \sum_{j=1}^{p} a_j \mathbf{x_j}$, each of which successively maximise the variance whilst remaining uncorrelated with the previous linear combinations. This is very important as PCA can be a direct solution to problems with multicollinearity.

These linear combinations $\mathbf{Xa}$ are what are called the *Principle Components*, whilst the eigenvectors $\mathbf{a}$ are called the *PC loadings*. These are often plotted in a *PCA Biplot* by the first two principle components as a means of visualising the data into a lower dimension.

Since the covariance matrix of the data depends on the scale of the measurement, those variables on a larger scale will have a larger effect on the transformations. So, as a preprocessing step before performing PCA, each variable's scale is standardized to have a standard deviation of one and a mean of zero.

In our analysis, PCA will be used to first reduce the dimensionality in order to visualise and describe the dataset, and later used to transform the dataset when a predictive model is hindered by multicollinearity.

2.3.2 Support Vector Machines

Support Vector Machines (SVM) are a robust, and powerful set of models which can be kernelised to extend to non-linear problems (Kuhn and Johnson 2013). They are simple in design, and for this reason will be studied first. They were originally developed as models for classification problems, but have then been adapted for the use in regression with a quantitative response known as Support Vector Regression (SVR).

As opposed to the loss function of simple linear regression, the loss function of SVR only accounts for some of the datapoints. These datapoints are those which are furthest away from the prediction plane, and contribute towards the loss linearly as opposed to a quadratic. There are two primary forms of SVR, $\epsilon - \textit{insensitive regression}$ and $\nu - \textit{SV regression}$, both differing in the methods of selecting the datapoints. However this only differs for the implementation procedure, and does not influence the optimal performance. So, in this project the former method will be used which simply works by defining a hyperparameter ϵ that sets the minimum threshold for the contributing residuals.

Model parameters for SVR are then estimated using this loss function along with an added penalty for large residuals given by the cost parameter C, which effectively regularises the model. This results in the minimisation of

$$Cost \sum_{i=1}^{n} L(y_i - \hat{y}_i) + \sum_{j=1}^{P} \beta_j^2 \tag{10}$$

where L is the loss function described previously.

The prediction function of a linear support vector machine is very similar to a simple linear regression model. For a new sample u, the prediction equation can be written as functions of a set of unknown parameters α_i and the corresponding training set data points x_{ij}.

$$\begin{aligned}
\hat{y} &= \beta_0 + \beta_1 u_1 + ... + \beta_P u_P \\
&= \beta_0 + \sum_{j=1}^{P} \beta_j u_j \\
&= \beta_0 + \sum_{j=1}^{P} \sum_{i=1}^{n} \alpha_i x_{ij} u_j
\end{aligned}$$

$$= \beta_0 + \sum_{i=1}^{n} \alpha_i (\sum_{j=1}^{P} x_{ij} u_j) \tag{11}$$

Here it can be seen that there are as many α parameters as there are datapoints in the training set, which results in a large prediction equation with many parameters to tune. However, this is not as problematic since the samples which are within the ϵ threshold of the regression line are exactly zero. Only the samples corresponding to predictions outside this threshold are used for the equation, which are why the are known as the *support vectors*.

Equation (11) also shows that the sample values enter in the equation as a sum of cross products with the training set values, ie. the dot product (**x'u**). This is very important, as it allows SVR to be extended to non-linear problems through kernelising the sample space. This equation can then be rewritten as

$$f(\mathbf{u}) = \beta_0 + \sum_{i=1}^{n} \alpha_i K(x_i, \mathbf{u}) \tag{12}$$

where K() is the kernel function. A variety of kernel functions can be used, two of which will be used in this study:

$$\text{polynomial} = (\phi(\mathbf{x}'\mathbf{u} + 1)^{degree})$$

$$\text{radial basis} = exp(-\sigma \|x - \mathbf{u}\|^2)$$

where ϕ and σ are the scaling hyperparameters. It has been shown that the radial basis kernel reliably gives the best performance, and has a lower computational cost (Cristianini and Shawe-Taylor 2000). Using the polynomial kernel results in an extra parateter to tune which is the *degree* of the polynomial.

In SVM, the predictors are taken in as a sum of their cross products, therefore their scales affect the model. For this reason, all of the predictors were centred and scaled as a preprocessing step before using the method.

Since there are many parameters, an appropriate methodology must be created to tune these models. To start with, the two different non-linear kernels will be kept separate. There is a relationship between the two hyperparameters C and ϵ since both effect the residuals in the loss function. This is problematic however it has been shown (Cristianini and Shawe-Taylor 2000) that the former is much more flexible than the latter. A lower value of ϵ also results in a greater number of support vectors to train, drastically increasing computational time. So following the tuning methodology set by (Kuhn and Johnson 2013), the two hyperparameters C and the *scales* were tuned first with a set value of ϵ. The tuning of ϵ was then performed onto the best model found in the first tuning phase.

2.3.3 Neural Networks

Neural networks are powerful non-linear regression algorithms loosely based on the functioning of the brain (Kuhn and Johnson 2013). They obtain predictions using layers of interconnected variables, called *hidden units*, each of which are linear combinations of the original predictors. There are many variations of of neural networks which differ in their number of hidden layers of nodes, the interaction between layers, and the methods of formulation. To keep things simple, and computational time reasonable simple single layer neural network will be studied in this project.

This project will use only a single layer of hidden units to manage the high computational time of these methods. It has also been shown that the addition of multiple layers does not provide any benefit to most problems (Heaton 2008). So this model works by having a single layer of hidden units, h, each being a linear combination of all or some of the predictors, p. It is common that the linear combination is transformed by a sigmoid function giving:

$$h_k(\mathbf{x}) = g(\beta_{0k} + \sum_{i=1}^{P} x_j \beta_{jk})$$

$$\text{where } g(u) = \frac{1}{1 + \exp^{-u}}$$

where P is total number of predictors and H is the total number of hidden units. Each of these hidden units is then related to the outcome with a linear function:

$$f(\mathbf{x}) = \gamma_0 + \sum_{k=1}^{H} \gamma_k h_k \tag{13}$$

where H is the total number of hidden units. This results in the total number of parameters in the model equalling $H(P+1)+H+1$, which get very large as the number of hidden units increases.

The problem of finding the best parameters is an optimisation problem, which minimises the loss function usually defined as the residual sum of squares. Since there are no constraints, the optimisation problem is purely numerical, and usually solved by a gradient descent algorithm. The most common method of solving this problem is with a highly efficient algorithm called *back-propagation* (Smith 1993), which is highly efficient and is what will be used in this project.

Due to these algorithms using gradients to find solutions to a set of added parameters, they can be adversely effected by collinearity in the variables. This is problematic in a similar manor to collinearity in multiple linear regression, which causes highly unstable models as the change in one parameter can result in a significant chance in another. This can be very problematic, especially in the case of trying to find a consistent gradient moving to a global optimum solution. To solve this issue, the model was performed on the components of the Principle Component Analysis which are all orthogonal to one another.

The large number of coefficients of a neural network also makes it prone to overfitting. In order to moderate this, a regularisation method using a weight parameter, λ is used. This feature is similar to ridge regression, where coefficients are penalized so that large values for them must have a significant effect on the model errors in order to be tolerated. Putting this formally, the weighted loss function is given as:

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{k=1}^{H} \sum_{j=0}^{P} \beta_{jk}^2 + \lambda \sum_{k=0}^{H} \gamma_k^2 \tag{14}$$

2.3.4 Tree Based Ensembles

Decision trees are models which partition the data with a series of if-then statements for the predictors. These statements split the data into subspaces, ultimately leading to the terminal nodes, which are known as the *leaves* of the tree.

One of the most utilised and best performing methods for constructing regression trees is with the *classification and regression tree* (CART) methodology (Breiman et al. 1984). This works by searching the entire dataset S for every possible value of each predictor to find the best position to partition the data into two groups S_1 and S_2, according to the measure of the split sum of squares error:

$$\mathrm{SSE}^S = \sum_{i \in S_1} (y_i - \bar{y}_1)^2 + \sum_{i \in S_2} (y_i - \bar{y}_2)^2 \tag{15}$$

where $\bar{y}_1$ and $\bar{y}_2$ are the averages of the training set outcomes within the groups. This process is repeated within the newly formed sets until the number of splits falls below a predefined threshold.

These methods can have many advantages. Due to their construction, these methods are able to be fit to any type of data, ie. with skewed data and data with unequally scaled predictors. So for these algorithms, there is no need to preprocess them. This construction also results in an implicit variable selection scheme, where only the most impactful predictor is chosen to be included at each step. It has also been shown that these algorithms, when allowed to grow sufficiently deep, can capture complex structures in data very well, and have very low bias (Kuhn and Johnson 2013). These models are also capable of performing inference of the predictors by keeping track of each variable that is chosen for the split along with their reduction in the loss function, and calculating a *variable importance score* for each.

The construction of trees do have very evident disadvantages as well. Since a small change in the input data can drastically effect the outcome of the results, they are often unstable with high variance. Another problem is their predictive accuracy since most datasets cannot be simply split in rectangular regions containing homogeneous outcomes.

However, these problems can be advantageous when used in ensemble methods. The idea of ensembles is to use bootstrapping to take many samples of the data, and create a model for each of these samples. These models are then aggregated together by averaging the result form every model to create the final ensemble model, which reduces their variance very effectively. Two different tree based ensemble methods known for their highly effective predictive performance will be used in this study, *Random Forests* and *Stochastic Gradient Boosting.*

Random Forests

Random Forests is a tree-based ensemble method which works by building a large selection of independent trees. The main idea behind this is that these trees are de-correlated, allowing for the prediction variance to be reduced optimally. This is done by adding a random component to the tree construction which works by randomly selecting a subset of predictors, *m_try* to choose from when looking for the best split. This is then the tuning hyperparameter of the algorithm.

The number of trees used for the forest must also be specified. Since random forests are protected from overfitting (Breiman 2001) the predictive performance of the model would not be hindered by additional trees, however for computational efficiency the number of trees should be kept to a minimum.

Stochastic Gradient Boosting

Boosting is another ensemble method usually used with tree based models which work by sequentially building trees where each tree learns and improves from the previous. Very shallow and weak trees are used which can be constructed quickly and learn slowly, avoiding overfitting and allowing for minor adjustments in each iteration. Since the origination of these models, they have been shown to be very powerful predictive models, often outperforming any other.

This can be generalised as a stagewise additive model which minimises the given loss function (Friedman, Hastie, and Tibshirani 2000). Therefore the algorithm can be described to work by iteratively adding a tree component with maximum tree depth max_{depth} at each given iteration, for a total of n_{iter} iterations. Since the algorithm's goal is to choose the optimal learner at each iteration, it is susceptible to overfitting. To resolve this, a regularisation term is added to it called the *learning rate*, given by the symbol λ. This hyperparameter decides the fraction of the predicted value which is added to the previous iterations value.

To develop this algorithm further, a randomisation component was added to the algorithm where for each iteration, a random selection of a fraction of the training data is used for prediction, controlled by the *bagging fraction* hyperparameter. This final version of the algorithm is called *stochastic gradient boosting*.

3 Analysis

This section will show the results of the data analysis. To begin this section, an exploration of the dataset will be provided along with the details on the preprocessing of the data prior. The results of the model tuning process will then be showed, and then followed by the results of the final model selection. An analysis on these predictions with the final model will then be given, detailing their estimated prediction intervals along with a diagnostic of their residuals.

3.1 Exploratory Analysis

To begin the exploration of the data, a scatterplot matrix of all the variables in the dataset was created. From this plot, a number of details can be seen. To start with, the predictors all have various different structures between themselves, along with some having a great degree of collinearity. As described in the methods section, this collinearity can cause problems and detriments in performance in some models. The structures between each predictor and temperature can be seen quite clearly, those for gravity become very complex, and less clear, whereas those for metallicity can hardly be seen. This could be very problematic in modelling, as it seems from this that the predictors individually do not give much information about the metallicity response, which could indicate a need for variable selection. Since temperature can be seen as the most simple to predict, it was chosen to be studied first.

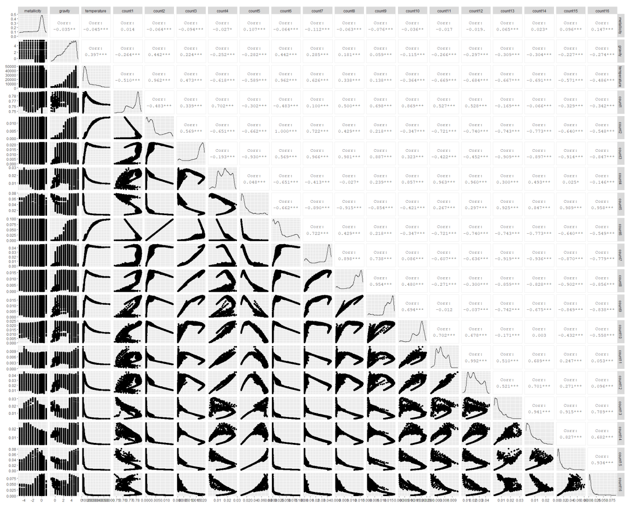
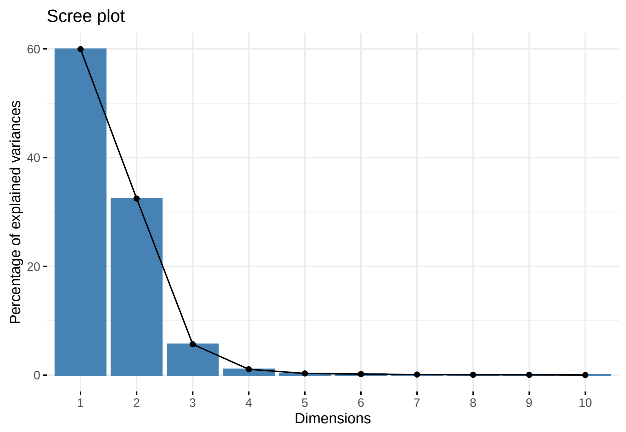
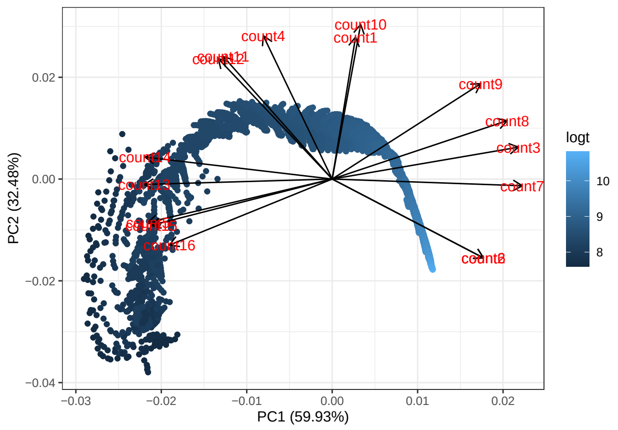


Figure 2: Scatterplot Matrix of all Variables in Dataset

To solve the issue of collinearity, a Principle Component Analysis was performed onto the dataset. As seen in the scree plots, PCA of this data managed to summarise over 90% of the total variance in the predictors in just two components. This being the case, these two components could be used to visualise the data very well, by plotting them in a PCA Biplot (Figure 3) which is used to infer numerous details about our data. To begin with, it can be seen that there is a clear structure in the biplot, known as a arch curve. This arch curve is a result of complex higher order structures of the data remaining in the components (Hastie and Stuetzle 1989), indicating the need for non-linear models. It can also be seen that many of the loadings are directly beside or opposite to one another, again showing the large degree of multicollinearity in the predictors. Here for temperature, there is a clear gradient going along the curve, which can be modelled effectively with relatively simple models. The temperature could be modelled with these two components alone, however since the goal of the project is to attain predictions as accurate as possible, two decisions were made to not use these two components exclusively. Firstly, it can be seen that the structure is lost at the lower temperatures, and secondly, it was decided that it would be best to not lose any information of the data. So when PCA were used as the inputs in the model, all 16 components would be used. This being the case, polynomial terms on a multiple linear regression model could be implemented in an attempt to fit this curve. However with so many predictor variables, it would be too difficult to decide which terms would require a higher order, and to which degree these should be.



(a) Screeplot



(b) Biplot Showing Temperature Gradient

Figure 3: PCA Plots

3.2 Data Preprocessing

The histogram matrix of all the variables are now shown below. The temperature is very heavily right skewed, which can be quantified using the skewness statistic given as

$$\text{skewness} = \frac{\sum(x_i - \bar{x})^3}{(n-1)v^{3/2}}$$

$$\text{where } v = \frac{\sum(x_i - \bar{x})^2}{(n-1)}$$

where for the variable of interest, x, $\bar{x}$ is its mean and n its total number of observations. As a rule of thumb, an absolute skewness value greater than 1 is an indication of high skewness.

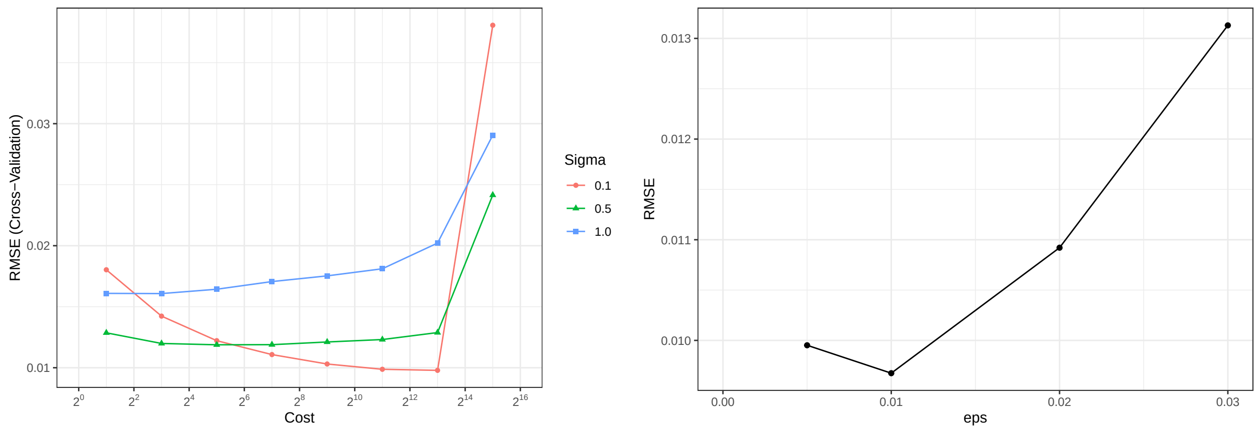
Skewness in the response variables causes problems in predictive models. This is especially the case with the squared error loss function used in all the models, since these high values of residuals in these outliers of high temperatures would cause the model to heavily account for these outlier values, negatively effecting the predictions of all other values. It can be seen that temperature is heavily right skewed, with a skewness value of 1.6. A log transformation was performed to solve this issue, resulting in a transformed value of 0.3. The remaining two responses both have moderate skewness values of 0.6, so were not transformed.

As specified in Section 2.3, PCA and SVM models require scaling and centring of the predictor variables which were performed before modelling.

3.3 Model Tuning

This section will show the results from the hyperparameter tuning phase for each model. Details are shown for this process done first on the temperature response. The same process was repeated to for the other two response variables, but will not be shown for brevity of the report.

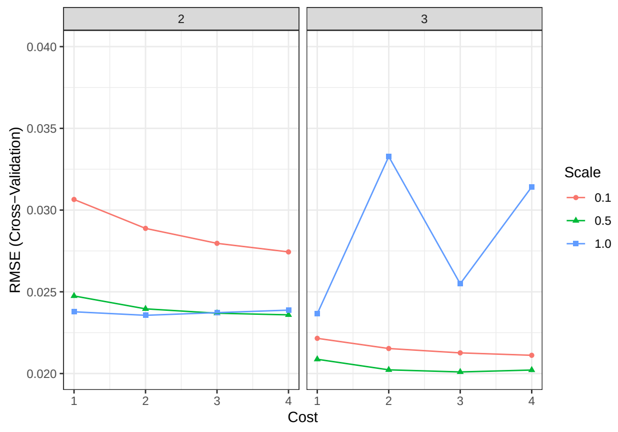
The tuning results of the radial basis kernel for SVR is shown in Figure 4. For the first two parameters, it can clearly be seen that a lower value of the kernel scaling value S is more optimal, where a S value of 0.5 produces good results for a relatively low value of C, which then levels off with an increase in C from overfitting. The best results were seen with the lowest value of S of 0.1, to which the performance continues to increase with the cost C. This shows that the data needs to be kernelised just a minimal amount to fit a linear relationship well, as increasing the cost of the residuals C to such a high value continues to increase the performance without overfitting, resulting in an optimal value of 2^{13} is reached.



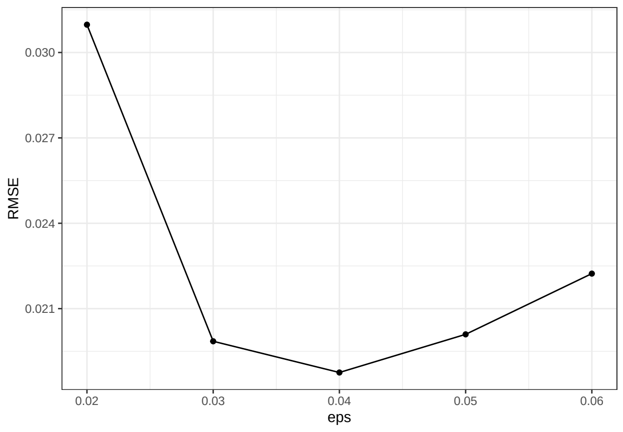
(a) Tuning of Cost and Scaling (threshold set to 0.01) (b) Tuning of Residual Threshold

Figure 4: Tuning Plots of Radial SVM

The results of the polynomial SVM kernel are then shown in Figure 5. It should first be noted that the fourth degree polynomial was highly unstable, easily overfitting to the data. This was seen during preliminary testing, where higher values of C and S led to extremely high RMSE scores, therefor only polynomials of degree 2 and 3 were formally tested. With this it can be seen that the optimal values of the hyperparameters for the first tuning phase are a polynomial of *degree* 3, with a C of 3 and a S of 0.5. The best tuned epsilon value is then given as 0.4, giving a final cross-validation RMSE of 0.0205. Here the radial basis kernel produced significantly better cross validation RMSE of 0.009, therefore it was chosen as the final model for comparison for the temperature response.



(a) Tuning of Cost and Scaling (columns separating scaling values and threshold set to 0.05)



(b) Tuning of Residual Threshold

Figure 5: Tuning Plots of Polynomial SVM

Tuning plots of Neural Networks are shown next in Figure 6. The optimal value for the weight parameter can be seen to be 0, showing that the algorithm for temperature does not need to be regularised. The results for neural networks clearly show that the performance increases with the number of hidden nodes, where at this point the performance increase has levelled off, indicating that there is no need to increase this further. This would be beneficial as the number of hidden nodes significantly increases the computational time required to train the model, as described prior. The optimal value of hidden units for this weight decay can be seen to be 13.

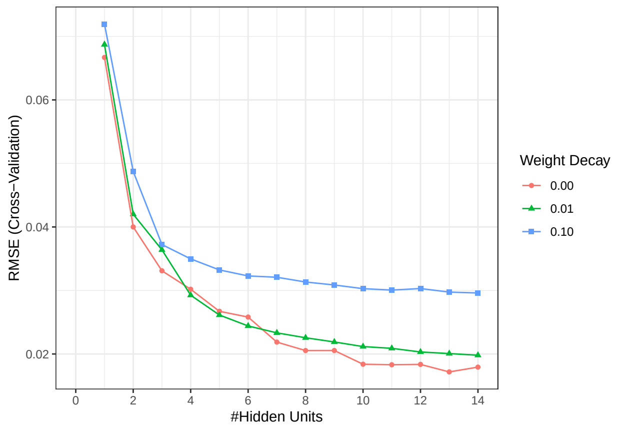
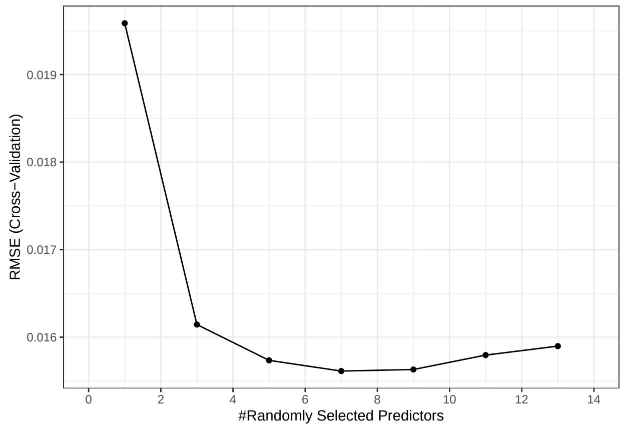
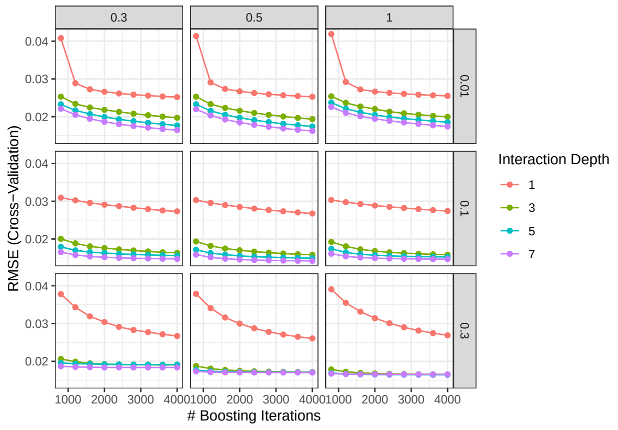


Figure 6: Tuning Plots of Neural Networks

The hyperparameter tuning of random forests (Figure 7a) was simple as there is only one to tune. The optimal value for the value $m_t ry$ can be seen as 7. This result is the value which sufficiently randomises each tree whilst maintaining its ability to fit to the data accurately.



(a) Random Forests



(b) SGB (Bagging Fraction separated in columns and Learning Rate in rows)

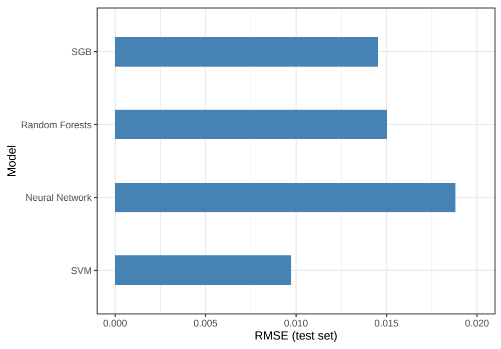
Figure 7: Tuning Plots of Ensemble Methods

Stochastic gradient boosting had many parameters to tune, however the best values for these can clearly be seen (Figure 7b). To start with, having a greater depth of tree for our data clearly produced better results. This is likely due to a requirement of more sophisticated models in order to appropriately fit the complex underlying structures of the data. The optimal value for the learning rate can be seen to be 0.1, providing the best results consistently across the grid. This value is what regulates the rate to prevent a sub optimal convergence towards a local optima, whilst converging quick enough in order to reach a prediction in the given number of iterations. The known issue of overfitting cannot be seen in these results, as the optimal values for the number of iterations was always the maximum allowed of 4,000, even with an unpenalized learning rate of 1. Finally, the stochastic component which controls the proportion of the data to be considered at each iteration can be seen to have a minor effect, of which a value of 0.5 is optimal.

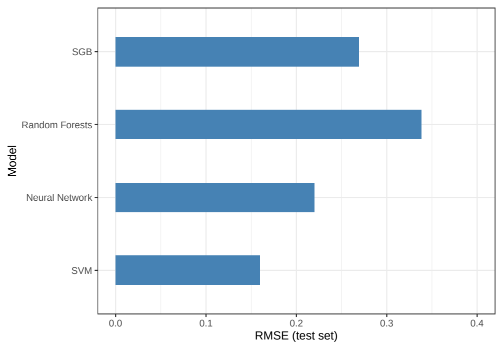
3.4 Model Comparison

The found best tuned models were then validated using the initially held out test set. These validations scores were used to compare the models to choose for the final predictions.

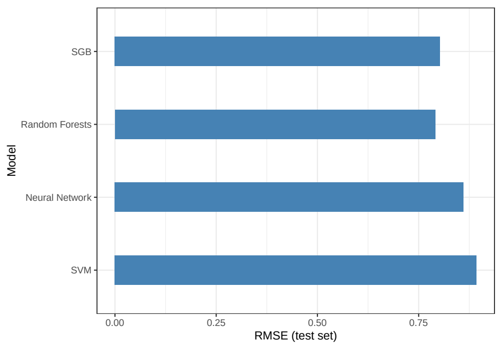
The comparisons for the temperature result clearly show that the best model is the SVM radial model, with a validation RMSE of 0.009. The runners up can be seen to be the two tree based models with SGB slightly outperforming Random Forests, followed by neural networks. All of these values produced very good validation scores, which can be compared to the standard deviation of the response variable of 0.008.



(a) Temperature



(b) Gravity



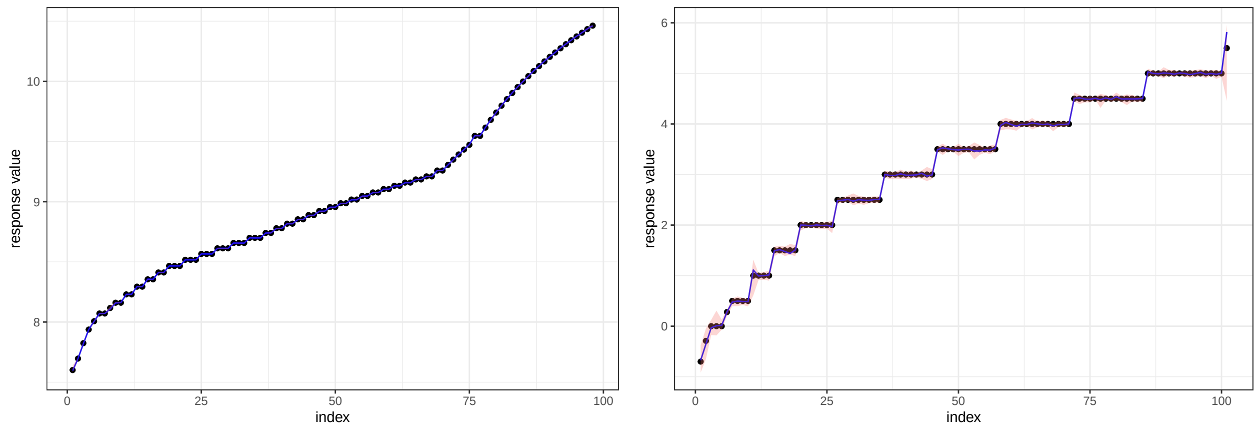
(c) Metallicity

Figure 8: Test set validation scores of optimally tuned models for all response variables

The model comparisons for the other two variables yielded interesting results. For predicting gravity, the best model was again clearly shown to be the SVM with radial basis kernel. However the ordering of the runners up changes completely, where Neural Networks clearly gives the second best performance, followed by SGB which all clearly outperform Random Forests. This ordering is completely changed again in the metallicity parameter, where Random Forests performs the best, whilst SVM performs the worst. However it should be noted that these performances are all similar to eachother and no model performs well, where the best validation RMSE is given as 0.72, which is a great deal larger than the variable's standard deviation of 0.02.

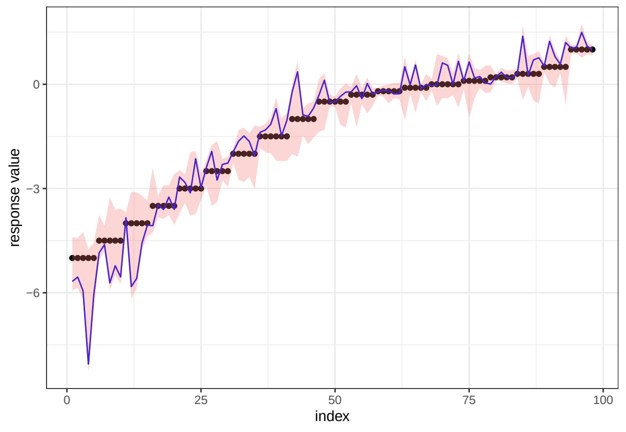
3.5 Prediction Intervals

This section will show the computed prediction intervals using the .632+ bootstrap method described in models. To allow the results to be shown clearly, a stratified sample of 100 datapoints was used to plot the prediction intervals. These plots show the true values of the points, the estimated prediction model using the median of the bootstrap values shown as a blue line, along with an area defining their 95% confidence intervals shown in red.



(a) Temperature

(b) Gravity



(c) Metallicity

Figure 9: Prediction Intervals

As can be seen, the temperature response is model fits the data near perfectly, with a very small prediction interval. The gravity response also shows accurate predictions, with the model fitting nearly directly to the true values, and a small confidence interval interval along with it. As for metallicity, the estimated model is hardly seen to align with the true values. The estimated model often lies in the extremes of the interval which get larger when predicting lower values of the metallicity. This non-normal distribution with heteroscedasity present clearly shows the need for a non-parameteric prediction interval for at all ranges of the response. For this variable, these predictive intervals show the extent and result of the poor predictive performance of our best model.

4 Discussion and Recommendations

From this study, it was shown that the radial basis kernel of SVM excels at remapping this data successfully which allows for a linear model to be fitted well onto it. The relatively poor performance of the polynomial SVM kernel proves this fact, showing that the radial basis kernel is responsible for the high performance of the model. This was able to predict the responses temperature and gravity with exceptional accuracy, showing significantly better results compared to the other models. Therefor it is recommended that these models are used for future GAIA work on this task.

In this study, no model was able to attain accurate predictions for the metallicity response given this dataset. This is interesting as this was not the case for previous GAIA star parameter studies (Willemsen, Bailer-Jones, and Kaempf 2004, @bailer–jones2007), which managed to find accurate results for all parameters, the latter of which was done using radial basis SVM models. The differences in these studies come from the specific datasets that were used. Firstly, an additional library of results was used in these datasets (Tsalmantza et al. 2009), featuring an additional specialised set of 4000 observations which was formulated specifically to improve the accuracies of this GAIA parametrisation task. Another key difference is that different wavelength band filters have been used for this study, which could very well contain wavebands that contain more information about the response variables. Lastly, the datasets contained additional Gaussian noise, which is known to improve performance in non-linear data due to a phenomena called stochastic resonance (Gammaitoni et al. 1998). So, for future study with the objective of finding the best predictive performance of these parameters, it is recommended that a dataset from a different wavelength filter is studied.

The difference in ordering of the best models for each response shows the need for multiple models to be tested, as the performance of models depends highly on the specific structure of the data. For the methods used in this study, with exception of SVM, the loss function used has been the sum of squared errors, however it might be beneficial to explore other loss function options. Other loss functions such as an absolute error loss function could be used instead, which do not penalise large residuals nearly as much. This could be useful especially for the library in this study as it contains a broad range of stars, including ‘cold stars’ which cannot be modelled accurately without adjustments to the colour calibration. The library does feature a correction to adjust for this, however all errors could not be accounted for (Lejeune, Cuisinier, and Buser 1998). A sum of squared error loss function would cause the model to heavily account for these values with high error , possibly hindering the predictive performance of other models.

This project did not take into consideration any variable inference, therefore no remarks were made on the specific effects of the predictor variables. This was the case since the selected wavebands are arbitrary, they are selected according to the waveband filter chosen in the simulation. However if inference is of interest, some suggestions can be made in order to gain information about the predictor variables. As mentioned in the model description, ensemble methods have an ability to perform inference on the predictor variables. Another method of providing more interpretable results is to perform clustering of the data (Hastie, Tibshirani, and Friedman 2009). This is an unsupervised learning task which organises the data observations which are similar to eachother. The PCA biplots shown previously give indication that clustering would work well, as datapoints which are close together are similar in their responses, and clear clusters can be seen which differ in their relationship to the rest of the data at the low temperature. With this, areas of the dataspace can be organised into specific groups, where inferences can be made.

Using a grid search strategy of tuning the hyperparameters is effective to allow for simple tuning of models over a large space of possible values, especially when there are many local minima in this space. However, to get more precise measurements of the optimal hyperparameters and to avoid the need to arbitrarily and discretely predefine their search space, other methods can be used. One such method is a gradient-based optimisation strategy, which optimises the hyperparameters using gradient descent.

5 Conclusion

In this project, a detailed investigation was performed with the objective of acquiring predictive performance of stars. A succesful account was made by first giving an appropriate analysis of the particular dataset, highlighting the presence of complex non-linear structures within it. A model selection strategy of tuning the hyperparameters and validating each model was employed in order to attain the model with the best predictive performance. This performance was assessed by attaining their nonparametric prediciton intervals. Using these, it was shown that accurate predictions of the two physical parameters temperature and gravity were achieved using the radial basis kernel of SVM, which was found to successfully kernelise the dataspace to allow for a support vector linear model to be fit. The same success could not be found when predicting the final parameter metallicity, although reasons for this were made clear, and recommendations for future work have been given.

Bibliography

Bailer-Jones, C. A. L. 2002. "Overview of the 'Photsim' GAIA Photometry Simulator." ICAPCBJ.

Bailer-Jones, Coryn AL. 2007. "Simulation of Stellar and Galaxy Spectra in the Nominal PS1 Photometric System and First Attempts at Discrimination and Parametrization."

Breiman, Leo. 2001. "Random Forests." *Machine Learning* 45 (1): 5–32.

Breiman, Leo, Jerome Friedman, Charles J. Stone, and Richard A. Olshen. 1984. *Classification and Regression Trees.* CRC press.

Cristianini, Nello, and John Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods.* Cambridge university press.

Efron, Bradley. 1983. "Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation." *Journal of the American Statistical Association* 78 (382): 316–31.

Efron, Bradley, and Robert Tibshirani. 1997. "Improvements on Cross-Validation: The 632+ Bootstrap Method." *Journal of the American Statistical Association* 92 (438): 548–60.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. 2000. "Additive Logistic Regression: A Statistical View of Boosting (with Discussion and a Rejoinder by the Authors)." *The Annals of Statistics* 28 (2): 337–407.

Gammaitoni, Luca, Peter Hänggi, Peter Jung, and Fabio Marchesoni. 1998. "Stochastic Resonance." *Reviews of Modern Physics* 70 (1): 223.

Hastie, Trevor, and Werner Stuetzle. 1989. "Principal Curves." *Journal of the American Statistical Association* 84 (406): 502–16.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Science & Business Media.

Heaton, Jeff. 2008. *Introduction to Neural Networks with Java.* Heaton Research, Inc.

Jolliffe, Ian T., and Jorge Cadima. 2016. "Principal Component Analysis: A Review and Recent Developments." *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374 (2065): 20150202.

Kaempf, T. A., P. G. Willemsen, C. A. L. Bailer-Jones, and K. S. de Boer. 2005. "Parameterisation of Rvs Spectra with Artificial Neural Networks First Steps." In *10th RVS Workshop, Cambridge (September 2005).*

Kohavi, Ron. 1995. "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection." In *Ijcai*, 14:1137–45. Montreal, Canada.

Kuhn, Max, and Kjell Johnson. 2013. *Applied Predictive Modeling.* Vol. 26. Springer.

Kumar, Sricharan, and Ashok N. Srivistava. 2012. "Bootstrap Prediction Intervals in Non-Parametric Regression with Applications to Anomaly Detection."

Lejeune, Th, F. Cuisinier, and R. Buser. 1998. "A Standard Stellar Library for Evolutionary Synthesis-II. The M Dwarf Extension." *Astronomy and Astrophysics Supplement Series* 130 (1): 65–75.

Perryman, M. A. C., Klaas S. de Boer, G. Gilmore, E. Høg, M. G. Lattanzi, Lennart Lindegren, X. Luri, F. Mignard, O. Pace, and P. T. De Zeeuw. 2001. "GAIA: Composition, Formation and Evolution of the Galaxy." *Astronomy & Astrophysics* 369 (1): 339–63.

Raschka, Sebastian. 2016. "Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning." https://sebastianraschka.com/blog/2016/model-evaluation-selection-part3.html.

Smith, Murray. 1993. *Neural Networks for Statistical Modeling.* Thomson Learning.

Sullivan, Timothy John. 2015. *Introduction to Uncertainty Quantification.* Vol. 63. Springer.

Taieb, Souhaib Ben, and Rob J. Hyndman. 2014. "A Gradient Boosting Approach to the Kaggle Load Forecasting Competition." *International Journal of Forecasting* 30 (2): 382–94.

Tsalmantza, P., M. Kontizas, B. Rocca-Volmerange, C. A. L. Bailer-Jones, E. Kontizas, I. Bellas-Velidis, E. Livanou, R. Korakitis, A. Dapergolas, and A. Vallenari. 2009. "Towards a Library of Synthetic Galaxy Spectra and Preliminary Results of Classification and Parametrization of Unresolved Galaxies for Gaia. II." *Astronomy & Astrophysics* 504 (3): 1071–84.

Willemsen, P. G., C. A. L. Bailer-Jones, and T. A. Kaempf. 2004. "Analysis of Stellar Parameter Uncertainty Estimates from Bootstrapping Neural Networks." Tech. rep., Gaia-ICAP, ICAP-PW-004.