Evaluation of logistic regression and random forest classification based on prediction accuracy and metadata analysis
Abstract

Model selection is an important part of classification. In this thesis we study the two classification models logistic regression and random forest. They are compared and evaluated based on prediction accuracy and metadata analysis. The models were trained on 25 diverse datasets. We calculated the prediction accuracy of both models using RapidMiner. We also collected metadata for the datasets concerning number of observations, number of predictor variables and number of classes in the response variable.

There is a correlation between performance of logistic regression and random forest with significant correlation of 0.60 and confidence interval [0.29 0.79]. The models appear to perform similarly across the datasets with performance more influenced by choice of dataset rather than model selection.

Random forest with an average prediction accuracy of 81.66% performed better on these datasets than logistic regression with an average prediction accuracy of 73.07%. The difference is however not statistically significant with a p-value of 0.088 for Student’s t-test.

Multiple linear regression analysis reveals none of the analysed metadata have a significant linear relationship with logistic regression performance. The regression of logistic regression performance on metadata has a p-value of 0.66. We get similar results with random forest performance. The regression of random forest performance on metadata has a p-value of 0.89. None of the analysed metadata have a significant linear relationship with random forest performance.

We conclude that the prediction accuracies of logistic regression and random forest are correlated. Random forest performed slightly better on the studied datasets but the difference is not statistically significant. The studied metadata does not appear to have a significant effect on prediction accuracy of either model.

Keywords: classification, logistic regression, random forest, metadata
Contents

1 Introduction ................................................. 3
   1.1 Classification Overview .............................. 5
       1.1.1 The Classification Model ......................... 5
       1.1.2 Evaluation .......................................... 6
       1.1.3 Geometrical Interpretation ...................... 6
       1.1.4 Overfitting .......................................... 7
       1.1.5 Cross Validation .................................... 8

2 Methods .................................................... 10
   2.1 Logistic Regression ................................. 10
       2.1.1 Prediction ........................................... 11
       2.1.2 Decision Boundary ................................. 12
       2.1.3 Maximum Likelihood Estimation ............... 13
       2.1.4 Training Logistic Regression ............... 14
       2.1.5 Predictor Variable Limitations ........... 15
       2.1.6 Multinomial Logistic Regression .......... 16
   2.2 Random Forest ........................................ 17
       2.2.1 Classification Trees ......................... 17
       2.2.2 Decision Boundary ............................... 18
       2.2.3 Constructing Classification Trees ........ 19
       2.2.4 Random Forest Model .................... 20

2.3 Data .................................................... 22
       2.3.1 Metadata .......................................... 22

2.4 Implementation ......................................... 23

2.5 Statistical Analysis .................................. 26
       2.5.1 Correlation ........................................ 26
       2.5.2 Student's T-test ................................. 27
       2.5.3 Multiple Linear Regression ............... 27

3 Results .................................................. 29

4 Discussion ............................................... 33

5 Appendix A .............................................. 36

6 Appendix B ............................................. 39
1 Introduction

What separates a good guess from a bad guess is information. It is said we are living in the information age \cite{1}. Whether this has allowed our guesswork to improve is up for debate but compared to earlier periods in history we have access to unprecedented amounts of information. Historically it hasn’t made much sense to collect data unless there was a specific purpose to extract information from it. These days data can be collected by machines and stored cheaply in digital form. The prospect of transforming collected data into useful information has led to a surge in data collection. It is estimated that the globally stored data increases by 23\% per year \cite{2}.

A popular trend in the business world is to collect all available data and store it in a data warehouse. Storing useless data is considered a small price to pay for the prospect of discovering valuable information that may be hidden somewhere in the data. However, collecting vast amounts of unstructured data serves little purpose unless there is a method to make sense of it. While statistics has been around for a long time to analyse data it has usually been applied to well understood data collected for a specific experiment.

The emerging massive datasets that sometimes goes by the term "Big Data" has driven an evolution in the field of statistics where techniques from machine learning used in artificial intelligence have been incorporated to handle new requirements. Particularly the focus in traditional statistics on inference and drawing conclusions from the data plays less of a role. There is simply too much data to analyse so the focus lies instead on creating mathematical models from the data that can aid in decision making.

The beauty of statistics is that we can create a tailor made mathematical model based solely on the data at hand. What kind of model to choose can sometimes be more of an art than science but there are ways to compare how well different models fit the data, such as measuring the test error rate \cite{4, p.37}. Once a model has been created it can be used by a computer as a decision rule for similar data. What it means is that we can teach a computer to make predictions and automate parts of the decision process. When the mathematical models are implemented in this automated way it is commonly referred to as machine learning.

There are many different kinds of statistical methods used in machine learning. In this thesis we look at classification problems. They concern the separation of data in to different classes. The thesis has two main purposes. The first purpose is to compare prediction accuracy for the two classification models logistic regression and random forest. Prediction accuracy is a performance measurement that specifies the ratio of correctly classified observations in a dataset. The second purpose is to investigate if metadata can help in the process of model selection and predicting model performance for the two models. Metadata provides information about datasets and we investigate if there is a connection between metadata and prediction accuracy.

In section 1.1 we look at the classification model as a general concept. We describe the mathematical model and how it is evaluated. We look at the model...
geometrically and see how a decision boundary can separate the classes. We also discuss the problem with overfitting and how to solve it using cross validation. The section follows the framework found in *Introduction to Statistical Learning: with Applications in R* by James et al [3]. To keep the thesis consistent this framework will be adhered to as much as possible in the rest of the thesis as well, especially in section 2.1 and 2.2.

The first classification model we look at is logistic regression. It is covered in section 2.1. We go through how to make predictions and what the decision boundary looks like. Then we describe how to train the model with maximum likelihood estimation and the method of steepest descent. Logistic regression has some limitations concerning the data it can handle. We go through these limitations and find ways to work around them.

In section 2.2 we look at the random forest model. It is an ensemble of several simpler classification tree models. We look at how to make predictions with classification trees and how to construct them. Once we know how the classification trees work we proceed with explaining how the trees are combined to create a random forest model.

The data used in the thesis is described in section 2.3. We also describe what metadata we have collected and the motivation behind collecting this metadata. In section 2.4 we outline the computer implementation. There we cover the most important steps in the algorithm used for training the models. A complete description of the computer implementation can be found in Appendix B.

Section 2.5 concerns the statistical methods used to analyse the results. We go through correlation, Student’s t-test and multiple linear regression. The theory in this section is mostly based on material from *Introduction to Probability and Statistics: Principles and Applications for Engineering and the Computing Sciences* by Milton and Arnold [7].

The results of the analysis as well as model performance and collected metadata can be found in section 3. We found a relationship between performance of logistic regression and random forest. Random forest performed slightly better than logistic regression but the difference was not large enough to be significant. The metadata we studied did not appear to have any influence on model performance.

We discuss the results in section 4. The results appear to be reasonable and we put them in perspective. Choice of dataset seems more important than choice of model. The studied metadata may have too small of an influence on performance to be useful on general datasets. We suggest some potential future research. It would be interesting to determine if the performance difference between logistic regression and random forest is significant. It would also be interesting to see if metadata analysis can be more useful under less general circumstances.
1.1 Classification Overview

In classification we look at observations of data and try to predict which class the data belongs to. We could for instance be looking at blood samples from patients at a hospital and try to predict whether the patients have diabetes or not. Another example comes from the US postal service where computers read the addresses on letters and sort them [5]. We can also thank classification for sending spam emails directly to the trash instead of ending up in our inboxes.

The data consists of a sample of observations all belonging to the same population; see Tamhane and Dunlop for information on populations and samples [6, p.86]. Every observation in the data consists of a number of predictor variables (also called features or attributes) which are used to predict a discrete response variable. In the above diabetes example the predictor variables would be measurements taken from the blood sample and the response variable would be whether the observed patient has diabetes or not.

Classification is of course not very interesting if all observations come with a known response variable; then we would have no trouble making perfect predictions for every observation. What makes classification useful is when we have some observations that come with a known response variable and use these observations to create a model that can be used on other observations with unknown response variable. In our diabetes example we could have a study conducted on patients where we know their diabetes status and use the results to obtain a model that can be used to diagnose future patients.

1.1.1 The Classification Model

If we look at a dataset consisting of \( n \) observations with \( p \) predictor variables and one response variable we denote the predictor variables as \( X = (X_1, X_2, \ldots, X_p) \) and the response variable as \( Y \). We write the \( i \)th observation as a \( p + 1 \) tuple

\[
(x_i, y_i)
\]

where \( x_i = (x_{i1}, x_{i2}, \ldots, x_{ip}) \) consists of the observed predictor variables and \( y_i \) is the observed response variable.

We can look at classification as a function \( f \) that maps the predictor variables \( X \) to the response variable \( Y \) [4, p.16]. We need to be careful though because even if we assume there is a relationship between \( X \) and \( Y \) there will most likely be information we haven't taken into account. To adjust for this error we introduce an extra term \( \epsilon \) called the random error and write the classification model as

\[
Y = f(X) + \epsilon.
\]

The function \( f \) represents the best possible prediction we can make for the response variable given the predictor variables. We call \( f \) the prediction function. This function is generally unknown and we have to use an estimate \( \hat{f} \) instead based on the data at hand. Since the random error is not possible to account for with the available predictor variables we will have to accept some error in
the model and drop the error term. This gives the model estimate
\[ \hat{Y} = \hat{f}(X) \]
where \( \hat{Y} \) denotes the prediction for \( Y \). While this is only an estimate of the actual model we will for convenience sake refer to this as the model and the actual model as the actual model.

The goal of classification is to find the estimate \( \hat{f} \) that makes our predictions as good as possible. Theoretically we want to find a perfect estimate where \( \hat{f} = f \). Practically we don't know what \( f \) looks like and the random error \( \epsilon \) means we will most likely never get perfect predictions. The random error is based on information the actual model has not accounted for so no matter how well we manage to estimate the prediction function some error will remain. In the end we have to evaluate the model and decide if the results are good enough for the intended purpose.

1.1.2 Evaluation

We evaluate model performance by measuring the prediction accuracy of the model. The prediction accuracy is the ratio of correct predictions in the response class. Throughout the thesis we will use performance and prediction accuracy interchangeably. It is sometimes more convenient to look at the error rate instead of the prediction accuracy. The error rate is the ratio of incorrect predictions. The prediction accuracy and error rate sums to 1 so the choice to study one over the other is arbitrary and mostly determined by context. The error rate is defined as
\[ \frac{1}{n} \sum_{i=1}^{n} I(y_i \neq \hat{y}_i) \]
where \( \hat{y} \) is the model estimate for \( Y \) and \( I \) is an indicator function which is 1 if \( y_i \neq \hat{y}_i \) and 0 otherwise [4, p.37]. If we have perfect predictions we will get an error rate of \( \frac{1}{n} \sum_{i=1}^{n} 0 = 0 \) and if all observations are misclassified we get an error rate of \( \frac{1}{n} \sum_{i=1}^{n} 1 = 1 \). The error rate falls in the range \([0,1]\) and the closer the error rate is to 1 the worse our classification model is.

1.1.3 Geometrical Interpretation

There is a way to look at classification from the geometrical perspective. Each observation can be seen as a point in \( p \)-dimensional space where \( p \) is the number of predictor variables. Classification is about defining a boundary that separates the points into their respective classes. We call this boundary the decision boundary. It separates \( p \)-dimensional space into a number of regions. We predict an observation to belong to a class that corresponds to the region the point belongs to. The decision boundary is the geometrical equivalence of the estimate of the prediction function \( \hat{f} \). Below is an example where we have a simple line as a decision boundary between two classes.
Figure 1: A decision boundary for a hypothetical dataset with two predictor variables. The decision boundary, drawn as a dashed black line, separates the two-dimensional space into two regions. The observations in the region below the line are classified as one class. The observations above the decision boundary are classified as another class. The actual class of the observations are drawn in red and blue respectively.

Notice how the classification is not perfect. A red dot is below the decision boundary. Generally we will have some misclassifications in any model. As we shall see it can actually be worse if we try too hard to adjust for these misclassifications.

1.1.4 Overfitting

When creating the model we split the data into a training set and a test set as shown in Figure 2. The training set is the subset of data observations that is used for creating the model. The test set is put aside and used later for evaluation purposes. The reason we need to split the data into disjoint subsets is due to the problem of overfitting [9 p.107].

Figure 2: Splitting the data into a training set and a test set.

Consider the example in Figure 1. With the line we use as a boundary it is not possible to perfectly separate the two classes. The decision boundary could
however have any kind of shape. How it looks depends on the classification model. A more complex model might come up with a decision boundary that correctly classifies all points. In fact it would be quite easy to construct such a model. Still this might not be such a good idea. There could for instance be a model that draws a tiny circle around all red dots and defines the collection of circles as the region for that class. When new examples come in they will most likely not be in these circles and would be misclassified. This is the problem of overfitting. We construct a complex model fitting the data at hand perfectly but that will generalize poorly to other data from the same population.

By training the model on the training set and evaluating it on the test set we get an evaluation that isn’t affected by overfitting. A model that has been trained to overfit the training set will perform poorly on the test set. As described earlier in this section we evaluate the model performance by measuring the error rate. It is important to distinguish between the training error rate and the test error rate. The training error rate is the error rate evaluated on the training set and similarly the test error rate is evaluated on the test set. In light of the discussion on overfitting we would be ill-advised to use the training error rate to evaluate the model. Instead we exclusively use the test error rate to get a reliable evaluation of the model’s performance.

1.1.5 Cross Validation

When splitting the data into a training set and a test set we have a dilemma. A larger training set means a more accurate model. A larger test set means a more accurate model evaluation. It is a tradeoff we would rather not make. Cross validation is a method attempting to alleviate this problem.

There are many versions of cross validation. We will focus on k-fold cross validation. In k-fold cross validation we partition the data into k subsets, called folds. We use one subset as test set and the union of the other subsets as training set. The process can be seen in Figure 3 below.

![Figure 3: K-fold cross validation where a dataset is split into k folds. One fold is reserved as test set and the other folds make up the training set.](image)

The most common choice of k is in the range 5 to 20 [13]. This range seems to provide a good ratio between test set and training set. As previously described the model is trained on the training set (consisting of k-1 subsets) and evaluated on the test set. We will repeat this process k times, where we will use a different
subset as test set every time. Thus we will train and evaluate the model \( k \) times. We then take the average of all \( k \) error rates to get an estimate of the model's error rate. With the error rate for the \( i \)-th subset as test set called \( er_i \) we get the average error rate

\[
\frac{1}{k} \sum_{i=1}^{k} er_i.
\]

We end up basing the error rate on all available data, while at the same time keeping a large portion of the data for training the model. Remember though that we have \( k \) different training sets and therefore \( k \) slightly different models. What remains is to choose one model. Since we already have an estimate for the error rate we can actually create a new training set of the entire data set and train a model this way. While not exactly accurate since the error rate is based on the average of subsets and the model is trained on the entire data set it turns out that the method still produces reliable results [10, p.70]. To summarize, cross validation is a method where we don't have to split the data into a separate training set and test set. By working with different folds of the data we can still avoid the problem of overfitting.
2 Methods

The thesis is based on the evaluation and comparison of two classification models. The chosen models are logistic regression and the random forest model. As we shall see they take different approaches to the classification task and it will be interesting to see if there are any discernable differences in the results between the models. Before proceeding we describe the specifics of each model and how they fit into the general template for classification outlined in the introduction.

2.1 Logistic Regression

Logistic regression is one of the simpler classification models. It has been around for a long time but is still widely used. Because of its parametric nature it can to some extent be interpreted by looking at the parameters making it useful when experimenters want to look at relationships between variables.

A parametric model can be described entirely by a vector of parameters $\beta = (\beta_0, \beta_1, ..., \beta_p)'$. An example of a parametric model would be a straight line $y = kx + m$ where the parameters are $k$ and $m$. With known parameters the entire model can be recreated. Logistic regression is a parametric model where the parameters are coefficients to the predictor variables written as $\beta_0 + \beta_1 X_1 + ... \beta_p X_p$. Where $\beta_0$ is called the intercept. For convenience we instead write the above sum of the parameterized predictor variables in vector form as $\beta X$.

The name logistic regression is a bit unfortunate since a regression model is usually used to find a continuous response variable, whereas in classification the response variable is discrete. The term can be motivated by the fact that we in logistic regression find the probability of the response variable belonging to a certain class, and this probability is continuous [4, p. 28].

The logistic part of the name is more straightforward. Logistic regression is based on the logistic function depicted in Figure 4 and as a function of $t$ written as

$$F(t) = \frac{1}{1 + e^{-t}}.$$
Figure 4: The logistic function $F(t) = \frac{1}{1+e^{-t}}$, also known as the sigmoid function because of its sigmoid shape. It ranges from 0 to 1.

2.1.1 Prediction

In classification when we want to make a prediction for an observation it can be useful to get the probability that the observation belongs to a certain class. A probability always ranges from 0 to 1 and since the logistic function does this as well it can be used for assessing class probability. With two classes we may denote the first class as 1 and the second class as 0. We adjust the logistic function introducing the parameterized predictor variables $\beta X$. Then we can use the logistic function to describe the probability of the response variable belonging to the first class, given predictor variable data. The modified logistic regression function, as a function of $\beta X$, is shown in Figure 5 and written as

$$P(Y = 1|X) = \frac{1}{1 + e^{-\beta X}}.$$
Figure 5: Version of logistic function used in logistic regression written as $P(Y = 1|X) = \frac{1}{1 + e^{-\beta X}}$. The probability of $Y$ belonging to class 1 is a function of parameterized $X$.

By including parameters in the logistic function we can train the model to find parameters separating observations from the first class and the second class. If we get a probability $P(Y = 1|X)$ greater than or equal to 0.5 we predict that the observation belongs to the first class and otherwise we predict that the observation belongs to the second class. We represent the decision with a function $g$ where

$$g = \begin{cases} 1 & \text{when } P(Y = 1|X) \geq 0.5 \\ 0 & \text{when } P(Y = 1|X) < 0.5 \end{cases}$$

The function $g$ assigns the observations to either class 1 or class 0 based on the probability $P(Y = 1|X)$. The estimated prediction function is then $\hat{f}(X) = g(\frac{1}{1 + e^{-\beta X}})$ and we can write the logistic regression model estimate as

$$\hat{Y} = g(\frac{1}{1 + e^{-\beta X}}).$$

Note that with more than two classes we cannot use a single decision to separate the classes. We will later find a way to make logistic regression work for more than two classes as well.

### 2.1.2 Decision Boundary

Considering the discussion above about prediction for logistic regression we choose class 1 if $P(Y = 1|X)$ is greater than or equal to 0.5. This can be written as

$$\frac{1}{1 + e^{-\beta X}} \geq 0.5.$$ 

The term $e^{-\beta X}$ is larger than 0 so we may simplify the expression without concern for the $\geq$ sign being affected. We rewrite as

$$2 \geq 1 + e^{-\beta X}$$

which simplifies to

$$1 \geq e^{-\beta X}.$$ 

Taking the natural logarithm on both sides we get

$$0 \geq -\beta X$$

which we rewrite as

$$\beta X \geq 0.$$ 

From this we draw the conclusion we should predict $Y$ to belong to the first class if $\beta X$ is greater than or equal to zero. This gives a decision boundary for logistic regression which can be written as $0 = \beta X$. It is a linear equation with $p$ variables and thus constitutes a hyperplane in $p$-dimensional space. An example in two dimensions can be seen in Figure 6.
Figure 6: Example of a decision boundary for logistic regression. With two predictor variables we get a line as a decision boundary. It is determined by $\beta$, in this case $\beta = (5,-1,-1)$. This gives the line equation $X_2 = 5 - X_1$. The decision boundary is drawn as a dashed black line. Points in the region below the line are classified as class 1 and points in the region above the line are classified as class 0.

2.1.3 Maximum Likelihood Estimation

To find good estimates for the parameters $\beta$ we use maximum likelihood estimation. Maximum likelihood estimation uses a likelihood function $L$ that maximizes the joint probability densities of the observations [11, p.168]. In logistic regression the likelihood function is written as

$$L = \prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{1-y_i}$$

where for the $i$th observation $p_i = P(Y = 1|x_i)$ and $y_i$ is the observed response variable with value 0 or 1.

The likelihood function will always give a value in the range from 0 to 1 and the closer we are to optimal predictions the larger the likelihood function will be. By maximizing the likelihood function we are also optimizing the predictions. In extension we will find good parameters $\beta$ since $p_i$ is a factor in the likelihood function $L = \prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{1-y_i}$ and $p_i = P(Y = 1|x_i) = \frac{1}{1+e^{-\beta x_i}}$ contains the parameters $\beta$. Logistic regression is a parametric model so by estimating the parameters $\beta$ we have all the information necessary to make predictions on future observations. The predictions are made using the model estimate $\hat{Y} = g(\frac{1}{1+e^{-\beta x}})$ described earlier in section 2.1.1 on logistic regression prediction.
2.1.4 Training Logistic Regression

When it comes to logistic regression there is no closed-form solution for maximizing the likelihood function [11, p.619]. Finding estimates for $\beta$ that will maximize the likelihood function must be done numerically. Typically it is easier to maximize the natural logarithm of the likelihood function $\ln(L)$. It is done by the method of steepest descent or other similar techniques. In steepest descent we use the gradient of the log-likelihood function $\ln(L)$. The gradient written as

$$\nabla = \left( \frac{\partial \ln(L)}{\partial \beta_0}, \frac{\partial \ln(L)}{\partial \beta_1}, ..., \frac{\partial \ln(L)}{\partial \beta_p} \right)$$

is a vector of partial derivatives with respect to each parameter. It represents the direction of highest increase for the function. Intuitively in steepest descent we choose a starting point for the parameters $\beta$, calculate the gradient and move in that direction. From the new point we calculate the gradient again and move in the direction of this new gradient. We iterate until the function value $\ln(L)$ stops increasing or until the update difference is small enough. This procedure has been schematically depicted in Figure 7. The logarithmic likelihood function for logistic regression is convex so we are guaranteed to approach the global maximum by using steepest descent [12].

Mathematically we have the recurrence relation

$$\ln(L)_{i+1} = \ln(L)_i + \alpha \nabla(\ln(L)_i)$$

where $\ln(L)_i$ is the log-likelihood function for the $i$th iteration and $\nabla(\ln(L)_i)$ is the gradient of the log-likelihood function for the $i$th iteration. We also have the learning rate $\alpha$ which specifies the step size. The step size determines how fast the algorithm converges. In Figure 7 it influences the length of the arrows representing the iterations. A small learning rate is inefficient and takes long time to converge but a large learning rate may overshoot the optimal value giving a less precise result.

![Figure 7: Schematic example of steepest descent. The function maximum is approached by iteratively moving in the gradient direction. Each iteration is represented by a black arrow pointing from previous position to updated position on the function surface.](image-url)
When training the logistic regression model using steepest descent or some other similar numerical technique we separate the available data into a test set and a training set, or preferably into \( k \) folds as outlined in section 1.1.5 on cross validation. The method of finding the estimated prediction function \( \hat{f} \) is unique to logistic regression but the process of cross validation is the same as for all classification models.

### 2.1.5 Predictor Variable Limitations

Logistic regression is limited by the type of data in the predictor variables. The parameters \( \beta_0, \beta_1, \ldots, \beta_p \) can be seen as weights in front of the predictor variables. Only numerical data such as continuous data and binary data can have weights assigned in a meaningful way. Continuous data is assigned a weight based on how the size of the variable influences the response variable. Binary data, consisting of 1:s and 0:s, is assigned a weight based on how a variable with a 1 will influence the data. A variable with a 0 will nullify the weight so the weights can distinguish between the two binary categories. Logistic regression can also handle ordered discrete data represented by integers such as counts, for example the number of birds in a forest. This data can be considered a special case of continuous data and is treated as such by the logistic regression model.

Non-numerical data is not compatible with the logistic regression model estimate \( \hat{Y} = g(\frac{1}{1+e^{-\beta X}}) \), so all data has to be converted to numerical form. All non-numerical data can be seen as categorical data, where each category represents a class. For instance if we have a variable with color we make each color a separate class. Sometimes every observation might have a unique description and then we end up with as many categories as there are observations, but at least we have a way of making sense of the data. Categorical non-numerical data with only two categories can easily be converted to binary form with one class represented with 0 and the other class represented with 1.

For predictor variables with more than two classes we have to make adjustments. We could try to assign each category to a discrete number and would end up with categories such as 0, 1, 2, 3, etcetera. This data can be used by the logistic regression model but will not produce meaningful results. Unless the classes are ordered it does not make sense to group them using integers. For example if we have three colors red, green and blue we cannot say that one color is larger than another. Instead we have to split the variable into several dummy variables.

Introducing dummy variables is a way to treat predictor variables with multiple classes as binary variables. If we have \( k \) different previous classes we create \( k - 1 \) dummy variables [6, p.419]. One class will be a baseline class which do not require a separate dummy variable. Each dummy variable assigns 1 to observations belonging to a certain class and 0 to all other observations. If we have the classes red, green and blue we create one dummy variable with 1 for all red observations and 0 for all other observations. We create another dummy variable with 1 for all green observations and 0 for all other observations. We do not need to create a third dummy variable for the blue class since the only
observations with 0 in both of the other dummy variables are blue. Below we show the dummy variable creation.

<table>
<thead>
<tr>
<th>ColorVariable</th>
<th>DummyRed</th>
<th>DummyGreen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Red</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Green</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Red</td>
<td>⇒</td>
<td>1</td>
</tr>
<tr>
<td>Blue</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Green</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Green</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Blue</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1: Dummy variables. We create two binary dummy variables, DummyRed and DummyGreen, to represent the three class variable ColorVariable. DummyRed marks an instance of Red with a 1 and 0 otherwise. DummyGreen marks an instance of Green with a 1 and 0 otherwise.

2.1.6 Multinomial Logistic Regression

Logistic regression is based on a decision that splits the observations in two response classes. When there are more than two classes a single decision is not enough to separate the classes. There is however a way to extend logistic regression to work with any number of classes, we call this multinomial logistic regression.

In multinomial logistic regression we use a pivot class similar to the baseline class for dummy variables described in the previous section. Every other class will be compared to the pivot class. If we have \( k \) classes we will do \( k - 1 \) comparisons. For every comparison we have to create a separate logistic regression model. We may then train each model similarly to ordinary logistic regression since we do a binary comparison for each model.

In section 2.1.2 about decision boundaries for logistic regression we showed that the decision boundary is written as \( \beta X = 0 \). We study the parameterized predictor variables to decide which class the observation belongs to. In fact \( \beta X \) is the logarithmic ratio of the probability for the two classes

\[
\ln \frac{P(Y = 1|X)}{P(Y = 0|X)} = \beta X.
\]

To see this we first notice

\[
P(Y = 0|X) = 1 - P(Y = 1|X) = 1 - \frac{1}{1 + e^{-\beta X}}
\]

\[
\Leftrightarrow P(Y = 0|X) = \frac{1 + e^{-\beta X}}{1 + e^{-\beta X}} - \frac{1}{1 + e^{-\beta X}}
\]

\[
\Leftrightarrow P(Y = 0|X) = \frac{e^{-\beta X}}{1 + e^{-\beta X}}.
\]
If we take the logarithmic ratio of the probabilities we get

\[ \ln \frac{P(Y = 1|X)}{P(Y = 0|X)} = \ln \frac{\frac{1}{1 + e^{-\beta X}}}{\frac{e^{-\beta X}}{1 + e^{-\beta X}}} \]

\[ \Leftrightarrow \ln \frac{P(Y = 1|X)}{P(Y = 0|X)} = \ln \frac{1}{e^{-\beta X}} \]

\[ \Leftrightarrow \ln \frac{P(Y = 1|X)}{P(Y = 0|X)} = 0 - (-\beta X) \]

\[ \Leftrightarrow \ln \frac{P(Y = 1|X)}{P(Y = 0|X)} = \beta X \]

which we set out to show. Similarly when we compare each class in multinomial logistic regression to a pivot class \(k\) we have the logarithmic ratio

\[ \ln \frac{P(Y = i|X)}{P(Y = k|X)} = \beta_i X \]

where we compare the \(i\):th class to the pivot class \(k\) [8, p.119]. The parameters \(\beta_i\) are acquired from the \(i\):th logistic regression model.

The class \(i\) with highest value for \(\beta_i X\) is the class with highest probability and is the class we end up choosing [13]. If no class has a value for \(\beta_i X\) greater than zero, which is the decision boundary, no class has higher probability than the pivot class so in that case the pivot class is chosen.

### 2.2 Random Forest

We will compare the logistic regression model to the random forest model. A forest in nature is made up of many trees and that is the idea behind a random forest too. It is a tree-based model where several classification trees are trained on subsets of the data with certain limitations to distinguish the trees from one another. The trees are combined into a larger random forest model where each tree votes on the predicted class and the class is selected by majority vote. We begin the section by describing classification trees.

#### 2.2.1 Classification Trees

Unlike logistic regression, decision trees are not parametric models. We cannot represent a tree by a short set of parameters. Rather we represent trees by a string of choices and decisions. A decision tree can easily be visualized and resembles a tree in nature except it is usually drawn upside down. There are two kinds of decision trees; regression trees and classification trees. As the names suggests, regression trees have a continuous response variable and classification trees have a discrete response variable. We will primarily focus on classification trees since classification is the area of interest in this thesis.

A classification tree is made up of a set of connected nodes. Each node represents a decision. The decision is based on a split in one of the predictor
variables [3, p.205]. For each decision we move down one level in the tree to another node, until we reach a terminal node called a leaf. The leaves consists of instances of the response classes. The leaf we end up in determines the response class. We give an example of a classification tree in Figure 8. Trees are most easily understood by this kind of visual representation.

Figure 8: Example of a classification tree with two predictor variables $X_1, X_2$ and a binary response variable. Nodes in the tree are drawn as ellipses with regular nodes colored blue and leaf nodes colored green. In every regular node we have an expression based on one of the predictor variables which can be either true or false. We evaluate the node expression at the top. If the expression is false we move down one level following the left arrow. If the expression is true we move down one level following the right arrow. We repeatedly evaluate the expression in the node until we reach a leaf. The leaf value determines the predicted response class.

The tree structure makes classification trees very flexible regarding the data they can handle. As long as we can find expressions capable of splitting data for a variable into subgroups the variable is compatible with the tree model. The data does not even have to be numerical. We could for instance have a variable with different colors. Say red, green and blue. It is easy to construct an expression that splits the data based on if the color is red or not, or other similar splits. Furthermore the tree model has no trouble handling multiclass response variables. We do not run into the problem of having to compare response classes. In each node the tree model simply states the predicted response class.

2.2.2 Decision Boundary

We can view classification trees as partitions of the p-dimensional space spanned by the predictor variables. The space is divided into non-overlapping regions.
Each region consists of a p-dimensional hyperrectangle, also called boxes [1] p. 306]. The boxes correspond to the region of space limited by the decision chain required to arrive at a certain leaf in the tree. For example if in the example from Figure 8 we arrive at the third leaf from the left we have the limitations; \( X_1 < 0.6, \ X_2 > 0.3, \ X_1 > 0.2 \). These limitations imposed on the space gives us the distinct region corresponding to the chosen leaf. We will have as many regions as there are leaves in the classification tree. With only two predictor variables it is easy to visualize the regions, as shown Figure 9 below.

**Figure 9**: Disjoint regions of 2-dimensional space determined by the tree model in Figure 8. The 2-dimensional space is spanned by the two predictor variables \( X_1 \) and \( X_2 \) with values between 0 and 1. Every region consists of a rectangle corresponding to one of the leaves in the tree. The regions corresponding to a leaf with 1 as predicted class is drawn in blue and the regions with 0 as predicted class are drawn in red.

If we join all regions predicting the same class we will end up with one region per class. The boundary between this partitioning of space gives us the decision boundary explained in the introduction. In Figure 9 the decision boundary is the boundary between the blue and red region where we predict an observation to belong to class 1 if it falls in the blue region and class 0 if it falls in the red region.

### 2.2.3 Constructing Classification Trees

When constructing a classification tree we approach the problem recursively by splitting the problem into subproblems. Finding a way to make a good split for the top node in the tree means we can actually construct the entire tree. The decision in the top node will split the data into two regions, one where the expression in the top node is false and one where it is true. We can treat both of these regions as separate data sets and employ the same tactic of finding a good split for the top node in these regions. We continue down the tree recursively until all data have been separated in leaves.

There are several algorithms for choosing the decision in the root node such as gain ratio, information gain and Gini index. We will use the Gini index \( G \).
which is defined as

\[ G = \sum_{k=1}^{K} p_{mk}(1 - p_{mk}) \]

where \( p_{mk} \) is the proportion of observations belonging to class \( k \) in the \( m \)th region [4, p. 312]. It is a measurement of node purity. Purity here means splitting the observations into homogenous regions consisting of predominantly one class. Pure nodes will create regions with low Gini index. Nodes with mixed splits will create regions with high Gini index. We motivate this by studying the expression \( p_{mk}(1 - p_{mk}) \) from the definition of the Gini index. It is a second degree expression so it will have one extreme point. We do a second derivative test to find the extreme point. The derivative is

\[ \frac{d}{dp_{mk}} p_{mk}(1 - p_{mk}) = 1 - 2p_{mk}. \]

When the derivative is 0 we have the position of the extreme point so

\[ 1 - 2p_{mk} = 0 \Rightarrow p_{mk} = 0.5 \]

is the position of the extreme point. We find the second derivative

\[ \frac{d}{dp_{mk}} 1 - 2p_{mk} = -2. \]

The second derivative is negative so the point \( p_{mk} = 0.5 \) is a maximum point. The ratio must lie in the interval 0 to 1 so the expression is minimized by approaching either 0 or 1. A second degree expression is symmetric around the extreme point and the distance from 0 to 0.5 equals the distance from 0.5 to 1. Hence the expression is minimized by approaching both 0 and 1.

This shows that classes with either low or high ratios will increase the Gini index the least. A region with only low or high class ratios will have low Gini index. Notice a region can only have one class with high ratio near 1 because the ratios should sum to 1. This is what makes a region with low Gini index pure. There can only be one dominating class.

Selecting the split with lowest Gini index means we create nodes that tries to separate the observations into homogenous regions.

2.2.4 Random Forest Model

As mentioned in the introduction to random forests we combine many classification trees to create a random forest model. Some elements of randomness are introduced in the tree construction motivating the random part of the name.

Each classification tree is constructed from a bootstrapped data sample. Bootstrapping is a method from statistics where we create new datasets from existing datasets. We repeatedly pick one observation from the dataset. The observations are picked randomly with equal probability for all observations. The draw is made with replacement meaning we may end up picking the same
observation several times. We draw $n$ observations where $n$ is the number of observations in the original dataset [8, p. 249]. We create $B$ such bootstrapped data samples where $B$ is the number of trees we want to include in the model. An example is provided in Figure 10.

Figure 10: The bootstrap method. A dataset with three observations is used to create two bootstrapped data samples, each with three observations as well. The observations are drawn randomly with replacement. As can be seen the same observation can occur multiple times in the bootstrapped samples.

For every bootstrap sample we construct a tree as outlined in section 2.2.3 about constructing classification trees. There is however an extra restriction introduced when constructing trees for a random forest. Previously we considered all predictor variables when finding the best split for a node in the tree. In a random forest we choose a size $m$ subset of the $p$ predictor variables that we consider as valid candidates to split on. The $m$ predictor variables we consider are chosen randomly at every node. The reason we only consider a subset of the predictors at every node is that we want to avoid correlation between the trees. Correlated trees will output similar results and not be able to catch different qualities in the data [4, p. 320]. There will most likely be a set of strong predictor variables explaining much of the variance in the data. By forcing some trees to not use these strong predictor variables we will diversify the trees and be able to catch other aspects of the data.

When predicting a response for an observation we feed the observation to all classification trees in the random forest. Each tree makes a separate prediction as described in section 2.2.1 on classification trees. We count the number of predictions for each class. The class with the majority vote is the class we end up choosing as the random forest prediction.

The predictions are preferably evaluated using cross-validation to assess model accuracy.
2.3 Data

To train the models data is required. 25 diverse datasets were collected for this task. The entire list of datasets with source and data of retrieval can be found in Appendix A. Some datasets were taken from the RapidMiner software. It is a tool for creating data mining algorithms and was used for training the models. RapidMiner comes installed with a collection of datasets and the four datasets suitable for classification were included in our collection of datasets.

Most datasets, 21 to be precise, were collected from the UCI machine learning repository found at: http://archive.ics.uci.edu/ml/index.html (accessed: 2014-04-20). It is an archive with a wide range of donated datasets maintained by University of California, Irvine. The datasets at UCI are searchable by a range of categories. We chose to collect datasets with classification as default task and sorted by year prioritizing new datasets. There is no standardization of data format at UCI. This presented a problem since many datasets were presented in ways unsuitable for our intentions. We only collected datasets that were deemed easy to work with, preferably in a single "csv", "xlsx","txt" or "data" file. Another problem was the size of some datasets. Due to performance reasons when running the algorithm we could only use datasets with less than 5000 observations.

Even within the 5000 observation limit two datasets were too taxing on the system when running the algorithm. Stratified samples were taken from these datasets to reduce the number of observations to manageable levels. Stratified sampling is a method of creating a smaller sample from a dataset while retaining the class balance in the dataset [14]. These datasets are marked with double asterisks in Table 2.

2.3.1 Metadata

The performance of a classification model on a dataset depends on certain unknown aspects of the data. The data sample contains information about the population it is sampled from. The more information contained in the data the better predictions we can make. If there is a way to measure the important aspects of the data for a certain model these measurements can be used to provide a way to predict model performance before running the model.

Measurements on data is called metadata. It is data about data. An example of metadata is the number of observations in a dataset. Metadata provides a way to summarize data with a few key components. Perhaps metadata can also be used for predicting model performance. We do not know what measurements will influence performance but trying to measure aspects of the data affecting the amount of available information might be useful.

We consider a dataset to be a sample drawn from a population. As the sample size increases the random variation decreases. For example the law of large numbers states that the sample mean approximates the population mean when the number of observations is large enough [3, p. 41]. It is reasonable to assume that the number of observations \( n \) in a dataset should affect the quality
of available information by reducing random variation. The question is if this metadata is statistically significant when it comes to predicting performance. We have measured the number of observations for all datasets as part of a metadata analysis. The results can be found in Table 2 of section 3.

Another interesting metadata measurement is the number of predictor variables $p$. Consider data about basketball players and the task to predict whether they are good or not. We might have a predictor variable for player length and find that tall players statistically are better than short players. Then we might introduce another variable, shot percentage, and be able to make more accurate predictions. We have more information available about the players. The more useful predictor variables we introduce the better predictions we should be able to make. One can of course argue what constitutes a useful variable; if noisy variables are introduced it might make the model perform worse. Regardless, we suspect the number of predictor variables might influence the predictions. In Table 2 of section 3 we have included metadata measurements for the number of predictor variables.

We included a third metadata measurement as well by looking at the number of classes in the response variable. When we increase the number of classes the probability to randomly select the correct class goes down. It is given from 1 over the number of classes $k$, written $\frac{1}{k}$. For example with two classes we have $\frac{1}{2} = 50\%$ chance to randomly choose the correct class and with three classes the chance is $\frac{1}{3} = 33\%$. Therefore we expect a model to perform worse as the number of classes increases.

Metadata about the number of classes can also serve another purpose. As we mentioned in section 2.1.6 logistic regression is intended to use for binary classification. To use it for more than two classes several binary logistic regression models has to be combined in multinomial logistic regression. The random forest model on the other hand natively works with any number of classes. By studying how the number of classes influences the two models we can see if it makes a difference in performance that logistic regression is intended to use for binary classification. In Table 2 of section 3 we have included metadata measurements for the number of classes in the response variable.

### 2.4 Implementation

The entire classification task was implemented in the RapidMiner software platform, specifically RapidMiner Studio 6.0 Starter Edition. RapidMiner is a software suit used for creating data mining applications. Among many other things RapidMiner is useful for implementing classification tasks. Both logistic regression and random forests are part of the software package.

Creating an application in RapidMiner is based on selecting modules and connecting them. For every task to be performed there is a separate module. The process looks very much like a flow chart as can be seen in Figure 11. Modules receive information via input ports. They process the information and send it to other modules via output ports. The active ports between modules are connected by lines. The process can be followed from left to right. Many of
the modules have parameters that can be adjusted. For instance in the Random Forest module we can set the number of trees.

![Figure 11](image-url): Main process of the classification implementation. Datasets on the left are connected to a module for replacing missing values. In turn this module is connected to a loop module. The loop module contains the classification algorithms. The classification predictions are sent by the loop module to the process output and displayed by the program.

Here we will go through the important parts of the implementation. See Appendix B for the complete implementation. In the main process we have imported all 25 datasets. Some of them can be seen in Figure 11. We run the process once for every dataset. The active dataset is connected to a replace missing values module which handles missing values in the dataset. When an attribute has a missing value we replace it by the average of all values in the attribute. Another alternative is to omit observations containing missing values. We opted for inserting the average since then we can keep the observations with missing values. By inserting the attribute average we guarantee the value will be reasonable even if it isn't correct.

After replacing missing values we feed the data to the loop module. Since we have two different classification models we use a loop to perform the classification once for both models. Thus everything inside the loop will be repeated twice.

Inside the loop we perform a cross validation. We can see the inside of the cross validation module in Figure 12. We use a select subprocess module to specify which model to evaluate. The chosen model is applied and $k$-fold cross validation is performed. This gives an average performance from the $k$ validations as described in section 1.1.5 on cross validation. In that section we also wrote that $k$ is usually chosen in the range 5 to 20. We have set $k$ to be 10.
Figure 12: Cross Validation module. A 10-fold cross validation is performed on a model determined by the Select Subprocess module on the left. The model is applied by the Apply Model module and evaluated with the Performance module. The average performance across all 10 validations is sent as output.

When selecting the logistic regression model we have to perform a few extra steps as can be seen in Figure 13. In section 2.1.5 we explained that the logistic regression model is only compatible with numerical data. With the Nominal to Numerical module we convert data to numerical form and create dummy variables for polynomial predictor variables. We also remember that logistic regression can only classify binary data. By encapsulating the Logistic Regression module in a Polynomial by Binomial Classification module we combine several binary logistic regression models into a multinomial logistic regression model. The parameters for the Logistic Regression module are left at default.

Figure 13: Select Subprocess module. We select either logistic regression or random forest. Logistic regression requires numerical data only which is accomplished by the Nominal to Numerical module. The Logistic Regression module is not visible but contained in the Polynomial by Binomial module. This extends logistic regression to work for multiple classes.

When selecting the random forest model we set the number of trees to 100. For a couple of datasets this had to be adjusted to 10 trees due to performance reasons. RapidMiner Studio 6.0 Starter Edition has a RAM memory limit of one Gigabyte. When this limit is reached the program stops. In such cases we downsized the number of trees to keep the RAM usage under one Gigabyte.
We use the Gini index to construct the trees, as described previously in section 2.2.3.

The process output consists of the cross validated prediction accuracy of the logistic regression model and random forest model.

2.5 Statistical Analysis

2.5.1 Correlation

Model performance depends highly on the dataset the model is trained on. Some datasets contain more useful information than others and are easier to predict well. However, because different classification models use different training algorithms they obviously perform differently. Logistic regression may get very high prediction accuracy on a dataset that random forest performs poorly on, and vice versa. We want to get an indication of how similar the models are in their predictions. We expect the models to be able to access some mutual information regarding the dataset and consequently to some degree get similar performance. We will measure how similar the performance is using correlation.

Correlation measures the linear relationship between two datasets. It can take values in the range -1 to 1. Values close to 1 indicates strong positive relationship. If one of the variables increases we expect the other to increase as well. Values close to -1 indicates strong negative relationship. If one of the variables increases we expect the other to decrease. Values close to 0 indicates no relationship between the variables [7, p.419]. The theoretical correlation coefficient \( \rho \) between two random variables \( X \) and \( Y \) is

\[
\rho = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sqrt{\sigma_x^2 \sigma_y^2}}.
\]

To get an estimate of the correlation for a data sample we use the estimation

\[
r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}.
\]

We can also determine if the correlation is significant using confidence intervals for the correlation. If the confidence interval contains 0 we do not have significant correlation. The bounds for the correlation confidence interval is given by

\[
(1 + r) - (1 - r)e^{2z_{\alpha/2}/\sqrt{n-3}}
\]

for the lower bound and

\[
(1 + r) - (1 - r)e^{2z_{1-\alpha/2}/\sqrt{n-3}}
\]

for the upper bound where \( z \) is the standard normal distribution and \( \alpha \) is chosen significance level [7, p.423]. See Tamhane and Dunlop for an overview of the normal distribution [6, p.52].
2.5.2 Student’s T-test

Another interesting inquiry is to investigate if the prediction accuracy for logistic regression and random forest are significantly different. We use Student’s t-test for this evaluation. It provides a way to compare the means of two samples and we can use it to determine if the means are significantly different. Student’s t-test assumes the samples are normally distributed. The test is however robust enough to perform well even when the samples are not normally distributed. Student’s t-test also assumes equal variance for both distributions. Variance is a statistic on the variation within a sample. Because we assume the variances are equal when performing Student’s t-test we have to first test this assumption. The test statistic for testing the null hypothesis of equal variance of two samples $X$ and $Y$ is \[ s_x^2 \] where $s_x^2$ and $s_y^2$ are the estimated sample variances. The test statistic should follow an F distribution with degrees of freedom equal to the sample sizes. See Milton and Arnold for information on the F distribution [7, p.340].

If we have equal variances we proceed with Student’s t-test. It uses the pooled variance $s_p^2 = \frac{(n_1-1)s_x^2(n_2-1)s_y^2}{n_1+n_2-2}$. With the pooled variance for the two samples $X$ and $Y$ calculated we test the null hypothesis that the two means $\mu_1$ and $\mu_2$ are equal. The test statistic for Student’s t-test is

\[
\frac{\bar{x} - \bar{y}}{s_p \sqrt{1/n_1 + 1/n_2}}
\]

where $\bar{x}$ and $\bar{y}$ are the sample means [2] p.346. If the null hypothesis is true it follows a t-distribution with $n_1 + n_2 - 2$ degrees of freedom. See Milton and Arnold for information on the t-distribution [7] p.263. We get a p-value from the test statistic and if it is smaller than a certain significance level $\alpha$ we choose to reject the null hypothesis and conclude the means are not equal.

The significance level $\alpha$ is a measurement of how confident we are that the results are significant and not due to chance. The probability of getting a test statistic $t$ due to chance is less than or equal to $(1 - \alpha)$. The choice of significance level is very subjective. Commonly a significance level of $\alpha = 0.05$ is used. However it is to a large extent motivated by tradition rather than mathematical theory.

2.5.3 Multiple Linear Regression

To analyse the impact of metadata on model performance we use multiple linear regression. As the name implies it is a regression model. We have already mentioned regression models a few times. They are the counterpart to classification when the response variable is continuous. Instead of predicting what class an observation belongs to we predict a real valued number.

Multiple linear regression is a linear model meaning the response variable is the sum of the predictor variables multiplied by some regression coefficients $b$. We write the model estimate as

\[
\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_p x_p
\]
where \( \hat{y} \) is the predicted response variable given the observed predictor variables \( x_1, x_2, ..., x_p \) \([7, p.444]\). The regression coefficients indicates how much the response variable is expected to increase when the associated predictor variable increases by one unit while the other predictor variables are held constant.

We use multiple linear regression to set up two models. The response variable for the first model is performance of logistic regression and the response variable for the second model is performance of random forest. In both models the predictor variables are the metadata measurements; number of observations, number of predictor variables and number of classes in the response variable. Don’t confuse the response variables from the original datasets with the response variables for the multiple linear regression model.

When fitting multiple linear regression we first write the model estimate in matrix form

\[
\hat{y} = Xb
\]

where \( X \) is the model matrix with the first column of all 1’s and the other columns formed by the predictor variables. Solving for \( b \) reveals

\[
b = (X'X)^{-1}X'\hat{y}.
\]

With a set of training examples with given response variables we have a way of calculating the regression coefficients using matrix algebra. However, we chose to use the built in methods in the R programming language for fitting the multiple linear regression model.

We can test if the regression is significant by testing the null hypothesis that all coefficients are 0. We write the null hypothesis as \( H_0 : \beta_1 = \beta_2 = ... = \beta_p = 0 \). If the regression is not significant it means there is a sizeable chance the results produced by the model are coincidental. The test statistic is \( F \) distributed and written as

\[
\frac{SSR/p}{SSE/(n - p)}
\]

where \( p \) is the number of predictor variables, \( n \) is the number of observations, \( SSR = \sum^n_{i=1}(\hat{y}_i - \bar{y})^2 \) is the variability in \( Y \) explained by the regression model and \( SSE = \sum^n_{i=1}(\hat{y}_i - y_i)^2 \) is the variability in \( Y \) the regression model can’t explain \([7, p.474]\).
3 Results

In Table 2 we have summarized the metadata for all datasets. We have also included the results of the classification task.

<table>
<thead>
<tr>
<th>index</th>
<th>n</th>
<th>p</th>
<th>classes</th>
<th>logistic regression</th>
<th>random forest</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1000</td>
<td>3</td>
<td>2</td>
<td>47.30</td>
<td>98.10</td>
</tr>
<tr>
<td>2</td>
<td>150</td>
<td>4</td>
<td>3</td>
<td>94.67</td>
<td>94.67</td>
</tr>
<tr>
<td>3</td>
<td>208</td>
<td>60</td>
<td>2</td>
<td>74.55</td>
<td>79.36</td>
</tr>
<tr>
<td>4</td>
<td>500</td>
<td>6</td>
<td>2</td>
<td>96.00</td>
<td>88.20</td>
</tr>
<tr>
<td>5</td>
<td>1055</td>
<td>41</td>
<td>2</td>
<td>77.63</td>
<td>81.24</td>
</tr>
<tr>
<td>6</td>
<td>151</td>
<td>5</td>
<td>3</td>
<td>34.42</td>
<td>48.96</td>
</tr>
<tr>
<td>7</td>
<td>4839</td>
<td>5</td>
<td>2</td>
<td>96.30</td>
<td>95.27*</td>
</tr>
<tr>
<td>8</td>
<td>4521</td>
<td>16</td>
<td>2</td>
<td>80.20</td>
<td>88.52*</td>
</tr>
<tr>
<td>9</td>
<td>583</td>
<td>10</td>
<td>2</td>
<td>71.36</td>
<td>72.91</td>
</tr>
<tr>
<td>10</td>
<td>440</td>
<td>7</td>
<td>2</td>
<td>39.92</td>
<td>91.36</td>
</tr>
<tr>
<td>11</td>
<td>258</td>
<td>5</td>
<td>4</td>
<td>81.03</td>
<td>88.37</td>
</tr>
<tr>
<td>12</td>
<td>100</td>
<td>5</td>
<td>2</td>
<td>68.00</td>
<td>69.00</td>
</tr>
<tr>
<td>13</td>
<td>182</td>
<td>12</td>
<td>2</td>
<td>71.46</td>
<td>72.57</td>
</tr>
<tr>
<td>14</td>
<td>195</td>
<td>22</td>
<td>2</td>
<td>86.16</td>
<td>87.63</td>
</tr>
<tr>
<td>15</td>
<td>250</td>
<td>6</td>
<td>2</td>
<td>57.20</td>
<td>97.20</td>
</tr>
<tr>
<td>16</td>
<td>500</td>
<td>12</td>
<td>2</td>
<td>55.60</td>
<td>58.20</td>
</tr>
<tr>
<td>17</td>
<td>800**</td>
<td>10</td>
<td>2</td>
<td>76.62</td>
<td>74.75*</td>
</tr>
<tr>
<td>18</td>
<td>1400**</td>
<td>32</td>
<td>5</td>
<td>30.55</td>
<td>41.47</td>
</tr>
<tr>
<td>19</td>
<td>961</td>
<td>5</td>
<td>2</td>
<td>46.31</td>
<td>82.52</td>
</tr>
<tr>
<td>20</td>
<td>1567</td>
<td>590</td>
<td>2</td>
<td>92.09</td>
<td>93.36*</td>
</tr>
<tr>
<td>21</td>
<td>1372</td>
<td>4</td>
<td>2</td>
<td>98.61</td>
<td>95.70</td>
</tr>
<tr>
<td>22</td>
<td>100</td>
<td>9</td>
<td>2</td>
<td>88.00</td>
<td>88.00</td>
</tr>
<tr>
<td>23</td>
<td>209</td>
<td>7</td>
<td>3</td>
<td>94.21</td>
<td>89.93</td>
</tr>
<tr>
<td>24</td>
<td>748</td>
<td>4</td>
<td>2</td>
<td>78.21</td>
<td>78.21</td>
</tr>
<tr>
<td>25</td>
<td>2584</td>
<td>18</td>
<td>2</td>
<td>90.33</td>
<td>93.42</td>
</tr>
</tbody>
</table>

Table 2: Dataset summary with metadata and prediction accuracy. For every indexed dataset we have number of observations \( n \), number of predictor variables \( p \) and number of classes in response variable as metadata. We also have prediction accuracy of logistic regression as well as prediction accuracy of random forest as a percentage. Entries in the random forest column marked with * have been trained using 10 trees instead of the regular 100 trees. Entries in the column for number of observations marked with ** are stratified samples where the original dataset was larger.

We have plotted the performance of logistic regression and random forest in Figure 14. As we can see the two performance curves follow each other to some extent.
Figure 14: Performance of logistic regression and random forest for all datasets. The datasets are identified by their index in Appendix A. The dashed blue curve represents the prediction accuracy of logistic regression and the red curve represents the prediction accuracy of random forest.

The estimated correlation between logistic regression performance and random forest performance is

\[ r = 0.60 \]

with a 95% confidence interval

\[ [0.29, 0.79] \]

which means the correlation between the model performances is statistically significant.

We compare the mean performance across all datasets of both models using Student’s t-test. The mean performance of logistic regression is 73.07% and the mean performance of random forest is 81.96%. The test assumes equal variance and the test statistic for equal variance is \( F = 1.9195 \) with \( p = 0.1171 \). With this p-value we choose to not reject the null hypothesis of equal variance. We proceed with Student’s t-test. The test statistic is

\[ t = -1.75 \]

with corresponding p-value

\[ p = 0.088. \]
The difference in mean performance between the models is not statistically significant.

Having compared the models we move on to investigating the impact of metadata on model performance. First we plot the relationship between each metadata measurement and model performance. In Figure 15 we have plotted model performance against logarithmized number of observations. The number of observations have been logarithmized to adjust for some datasets being much larger than others. Taking the logarithm makes it easier to distinguish the characteristics of the examples with low number of observations. We see no clear trend for either model. In Figure 16 we have plotted model performance against logarithmized number of predictor variables. As before we see no clear trend for either model. Looking at a plot for the number of classes in the response variable will not reveal much. Most examples have two or three classes so the performances would simply be stacked on top of each other in the plot.

**Figure 15:** Performance of logistic regression and random forest plotted against the logarithmized number of observations for each dataset.
Figure 16: Performance of logistic regression and random forest plotted against the logarithmized number of predictor variables for each dataset. One example with 590 predictors was considered an outlier and left out of the plot.

When fitting a multiple linear regression model for logistic regression prediction accuracy as response variable and metadata as predictor variables we get the model

$$\hat{y} = 0.85 + 0.000031x_1 + 0.00026x_2 - 0.071x_3. $$

The response variable $x_1$ is the number of observations, $x_2$ is the number of predictors and $x_3$ is the number of classes. As we see all coefficients are very small. A regression significance test gives an F-value of 1.19 and p-value of $p = 0.66$.

The regression of logistic regression on the studied metadata is not statistically significant.

When fitting a multiple linear regression model for random forest prediction accuracy as response variable and metadata as predictor variables we get the model estimate

$$\hat{y} = 0.99 + 0.00020x_1 + 0.00010x_2 - 0.085x_3. $$

Once again all coefficients are very small. A regression significance test gives an F-value of 2.24 and p-value of $p = 0.89$.

The regression of random forest on the studied metadata is not statistically significant.
4 Discussion

We have studied performance of classification models. More specifically we have focused on the prediction accuracy of logistic regression and random forest. The datasets the models was trained on are diverse and when looking at the metadata in Table 1 we see no strange patterns. While not completely random the selection method and data source should allow us to generalize our results within reasonable doubt to other datasets.

The first important aspect of our analysis concerns the similarity of performance for logistic regression and random forest. In Figure 14 we see the prediction accuracy of both models across all datasets. The performances appear to be related given that the performance curves follow each other to some degree. A correlation analysis between the performances revealed an estimated correlation coefficient \( r = 0.60 \). It is quite far from 0 and we view it as a moderately strong correlation. The associated 95% confidence interval \([0.29 \ 0.79]\) confirms the correlation is sufficiently different from 0. We conclude that there is a similarity in prediction accuracy of logistic regression and random forest. If we know that one model performs poorly on a dataset we expect the other model to perform poorly as well. This has implications when trying to come up with a good classification on a dataset. If we do not get good results with one model we may try the other but unless we get lucky the performance will still be in the same region.

Even though the models appear to perform similarly on the same datasets there is a noticeable difference in mean model performance. The mean performance of logistic regression is 73.07% and the mean performance of random forest is 81.96%. With Student’s t-test we tested if the means are significantly different. We found a test statistic \( t = -1.75 \) with p-value \( p = 0.088 \). The p-value is larger than 0.05 which is the most common significance level. We suspect the random forest model may perform better than the logistic regression model but the difference is not significant. To confirm or deny if there is a difference in model performance further research is required. It would be interesting to test the difference in model performance using a larger collection of datasets. We should also note that there could be other interesting performance measurement to investigate. We focused on prediction accuracy but in some circumstances completion time and memory usage may be as important.

We see that the correlation between datasets is significant and the mean performance difference between the models is not. Even though we cannot rule out a difference in mean performance the correlation appears to be more important with a confidence interval \([0.29 \ 0.79]\) far from 0. It appears having a good dataset with useful predictor variables is more important than selecting the best model. An explanation for this is that even though the models use different algorithms they capture similar information in the dataset. In support of this idea we see in Table 2 that for datasets with index 2, 22 and 24 the models have identical performance. Such results would be unlikely if the models did not work with similar information.

It would be very useful if metadata could provide knowledge about the qual-
ity of the dataset. Then we could study the metadata to get an indication of expected prediction accuracy before running any experiments. Unfortunately our analysis indicates the analysed metadata does not provide sufficient information about the datasets. In Figure 15 and Figure 16 we plotted performance of models against number of observations and number of predictor variables. We observe nothing but random fluctuations. Visual inspection does not reveal any relationship between model performance and metadata.

Setting up a multiple linear regression model with logistic regression performance as response variable and metadata as predictor variables gives the model estimate

\[
\hat{y} = 0.85 + 0.000031x_1 + 0.00026x_2 - 0.071x_3.
\]

As noted previously in the results found in section 3 all predictor variable coefficients are very small. When testing if the regression is significant we test if any of the coefficient is significantly different from 0. The test gives a p-value of 0.66. It is much larger than any acceptable significance level. A multiple linear regression model with random forest performance as response variable and metadata as predictor variables gives similar results with a p-value of 0.89. The large p-values indicates it is very unlikely any of the metadata influences model performance.

We would expect the metadata to have some influence on the results. With more observations and more predictor variables the data should contain more information. An explanation could be that the effect of the studied metadata on performance is too small compared to the variations between datasets. One dataset can have very many predictor variables with mostly meaningless noise while another dataset can have a few high quality predictor variables. It may not be possible to measure the effect of metadata when the variation between datasets is high.

We should however not rule out metadata analysis. These results are based on the multiple linear regression model. It is a linear model so there may still exist nonlinear relationships between the analysed metadata and model performance. It can be of interest to attempt using nonlinear models when analysing the relationship between metadata and model performance.

There may also exist other more useful metadata. We have only looked at number of observations, number of predictor variables and number of classes in the response variable. For instance it could be interesting to look at the number of non-numerical predictor variables. Another idea is to focus on datasets from a specific field of research and consult with domain experts. Domain experts may have knowledge about field specific metadata that can have strong influence on classification performance. Limiting the metadata analysis to datasets from a specific field of research might also make it possible to reduce some of the variations between datasets. Perhaps focusing on metadata within a research field will provide more useful information instead of trying to find metadata for general datasets.
In summary, we draw the following conclusions. The prediction accuracy of logistic regression and random forest appears to be correlated. Random forest performed slightly better on these datasets but the difference is not statistically significant. The analysed metadata is not significantly related to prediction accuracy of either model.
5 Appendix A

In this appendix we list the datasets, their source and date of retrieval. The datasets are indexed and this index is used in other sections of the thesis when referring to specific datasets.

1. Deals
   Included in RapidMiner Studio 6.0 software
   retrieved: 2014-02-03

2. Iris
   Included in RapidMiner Studio 6.0 software
   retrieved: 2014-02-03

3. Sonar
   Included in RapidMiner Studio 6.0 software
   retrieved: 2014-02-03

4. Weighting
   Included in RapidMiner Studio 6.0 software
   retrieved: 2014-02-03

5. QSAR biodegradation Data Set
   http://archive.ics.uci.edu/ml/datasets/QSAR+biodegradation
   retrieved: 2014-01-30

6. Teaching Assistant Evaluation Data Set
   https://archive.ics.uci.edu/ml/datasets/Teaching+Assistant+Evaluation
   retrieved: 2014-04-20

7. Wilt Data Set
   http://archive.ics.uci.edu/ml/datasets/Wilt
   retrieved: 2014-04-20

8. Bank Marketing Data Set
   http://archive.ics.uci.edu/ml/datasets/Bank+Marketing
   retrieved: 2014-04-20

9. Indian Liver Patient Dataset
   http://archive.ics.uci.edu/ml/datasets/ILPD
   retrieved: 2014-04-20

10. Wholesale customers Data Set
    http://archive.ics.uci.edu/ml/datasets/Wholesale+customers
    retrieved: 2014-04-20

11. User Knowledge Modeling Data Set
    retrieved: 2014-04-20
12. BLOGGER Data Set
   http://archive.ics.uci.edu/ml/datasets/BLOGGER
   retrieved: 2014-04-20

13. Planning Relax Data Set
   http://archive.ics.uci.edu/ml/datasets/Planning+Relax
   retrieved: 2014-04-20

14. Parkinsons Data Set
   http://archive.ics.uci.edu/ml/datasets/Parkinsons
   retrieved: 2014-04-20

15. Qualitative Bankruptcy Data Set
   http://archive.ics.uci.edu/ml/datasets/Qualitative_Bankruptcy
   retrieved: 2014-04-20

16. Dresses_Attribute_Sales Data Set
   http://archive.ics.uci.edu/ml/datasets/Dresses_Attribute_Sales
   retrieved: 2014-04-20

17. MAGIC Gamma Telescope Data Set
   http://archive.ics.uci.edu/ml/datasets/MAGIC+Gamma+Telescope
   retrieved: 2014-04-20

18. Turkiye Student Evaluation Data Set
   http://archive.ics.uci.edu/ml/datasets/Turkiye+Student+Evaluation
   retrieved: 2014-04-20

19. Mammographic Mass Data Set
   http://archive.ics.uci.edu/ml/datasets/Mammographic+Mass
   retrieved: 2014-04-20

20. SECOM Data Set
    http://archive.ics.uci.edu/ml/datasets/SECOM
    retrieved: 2014-04-20

21. banknote authentication Data Set
    http://archive.ics.uci.edu/ml/datasets/banknote+authentication
    retrieved: 2014-04-20

22. Fertility Data Set
    http://archive.ics.uci.edu/ml/datasets/Fertility
    retrieved: 2014-04-20

23. seeds Data Set
    http://archive.ics.uci.edu/ml/datasets/seeds
    retrieved: 2014-04-20

24. Blood Transfusion Service Center Data Set
    http://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center
    retrieved: 2014-04-20
25. **seismic-bumps Data Set**

http://archive.ics.uci.edu/ml/datasets/seismic-bumps

retrieved: 2014-04-20
6 Appendix B

The complete RapidMiner implementation can be found in the figures in this section. Apart from the main process each figure represents a module as described in the figure caption.

Figure 17: Main process of the classification implementation. Contains data modules labeled Retrieve <dataset name>, a Replace Missing Values module as well as a Loop module.

Figure 18: Loop module inside main process. Contains a cross validation module called Validation.

Figure 19: Validation module inside Loop module. Contains a Select Subprocess module, a Apply Model module and a Performance module.
**Figure 20:** Select Subprocess module inside Validation module. Contains a Nominal to Numerical module connected to a Polynomial by Binominal Classification module and a Random Forest module.

**Figure 21:** Polynomial by Binominal Classification module inside Select Subprocess module. Contains a Logistic Regression module. Parameters for the Logistic Regression can be seen on the right and are left at default values.

**Figure 22:** Parameters for the Random Forest module inside the Select Subprocess module. The parameter number of trees is set to 100 and the parameter criterion is set to gini index. The other parameters are left at default.
References


