Machine learning II

Predicting the Geographical Origin of Music

Matthias Kunz, Stefan Beuchert, Adriana Sanchez, Vy Nguyen

11.01.2022

Contents

notes

```
print("scatter plot für koordinaten und box plot für varianz in long und lat") # why did we want this?
```

```
## [1] "scatter plot für koordinaten und box plot für varianz in long und lat"
```

```
print("boxplot für distances für jedes modell -> um diese vergleichen zu können")
```

```
## [1] "boxplot für distances für jedes modell -> um diese vergleichen zu können"
```

```
print("für finales ergebniss, ein karte, welche ? ist und soll koorinaten anzeigt und mit linien verbind
```

```
## [1] "für finales ergebniss, ein karte, welche ? ist und soll koorinaten anzeigt und mit linien verbin
```

librarys

```
#install.packages("installr")
#library("installr")
#install.Rtools()

#install.packages("ggmap")
#install.packages("maptools")
#install.packages("maps")
#install.packages("glmnet")
#install.packages("ISLR")
#TODO the one below necessary?
#install.packages("rgl")
#install.packages("dplyr")
#install.packages("keras")
#install.packages("tensorflow")

library("glmnet")
```

```
## Lade nötiges Paket: Matrix
```

```
## Loaded glmnet 4.1-3
```

```
library("ggplot2")
library("ggmap")
```

```
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
```

```
## Please cite ggmap if you use it! See citation("ggmap") for details.
```

```
library("maptools")
```

```
## Lade nötiges Paket: sp
```

```
## Checking rgeos availability: FALSE
## Please note that 'maptools' will be retired by the end of 2023,
## plan transition at your earliest convenience;
## some functionality will be moved to 'sp'.
##      Note: when rgeos is not available, polygon geometry     computations in maptools depend on gpcli
##      which has a restricted licence. It is disabled by default;
##      to enable gpclib, type gpclibPermit()
```

```
library("maps")
library("dplyr")
```

```
##
## Attache Paket: 'dplyr'
```

```
## Die folgenden Objekte sind maskiert von 'package:stats':
##
##     filter, lag
```

```
## Die folgenden Objekte sind maskiert von 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ISLR)
options(rgl.printRglwidget = TRUE)
library(rgl)
```

```r
library(keras)
library(tensorflow)
```

Comparison of machine learning algorithms / Introduction / Theory

This report attempts to compare two regression/classification algorithms on its behavior on a specific data set. What should be found out?

Ethnomusicology is the study of music in its social and cultural contexts. Being a highly interdisciplinary field of research, we found interesting trying to apply some machine learning algorithms: ridge regression and neural networks and compare its results.

Data set

Here should be a bit of a short summary of the data set with some key characteristics of the data set.

What is it about? Which variables are included? What type of variables? What is missing? What about missing values?

The data used in this project is provided by the authors of the paper "Predicting the Geographical Origin of Music"[Fang Zhou, Claire Q, Ross D. King (2015) how crated and used this set as basis for this paper. The key information embedded in this data set are attributes of music pieces labeled by there point of origin.

The authors collected 1,142 pieces of music from a personal CD-collection. The target values represented by longitude and latitude are the main country/area of residence of the artists that produced the music. Over all the authors collected music from 73 countries/areas not including western style music, since this category is called to have a global influence and therefor unfitted for predicting a specific country/region.

The attributes to describe the music pieces are automatically created by a software called MARSYAS(Tzanetakis 2007), a software created to extract audio features from wave files. With MARSYAS the authors could convert every music peace to a set of 116 features called 'chromatic features.' The features are numerical and the authors claim to have normalized them into a gaussian normal distribution.

Selected task is suitable for classification and regression. However we decided that regression is a better approach because in first instance we don't have enough data to be representative to each country and secondly because of the special characteristics of desired output, coordinates.

In order to deal with our spatial output data we'd created a function to scale from our predicted and real values for further model evaluation.

```r
distances <- function(predicted, actual_value) {
  dif <- predicted-actual_value
  dif <- dif * (40030/360) # scaling coordinates to km by the factor circumference (km) / 360°
  mse <- sqrt(dif[,1]^2 + dif[,2]^2)
  return(mse)
}
```

```r
data <- read.csv("Data/default_plus_chromatic_features_1059_tracks.txt", header=FALSE)
data <- as.data.frame(data)
colnames(data)[117:118] <- c("Latitude", "Longitude")
```

```r
# Maybe some more preprocessing could be done here.
anyNA(data) # testing if there is at least a single NA -> but in this dataset there isn't
```

```
## [1] FALSE
```

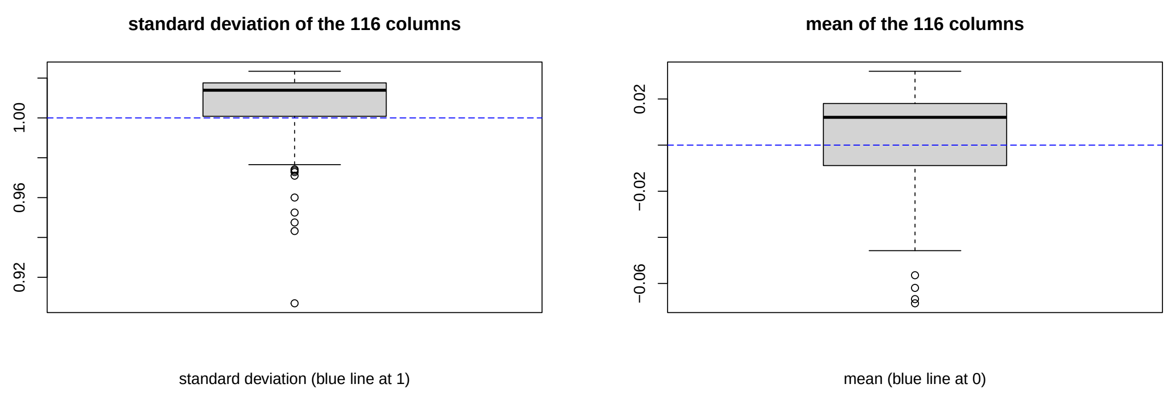
```r
# Maybe some more pre-processing could be done here.
anyDuplicated(data) # testing for duplicates -> 0 found
```

```
## [1] 0
```

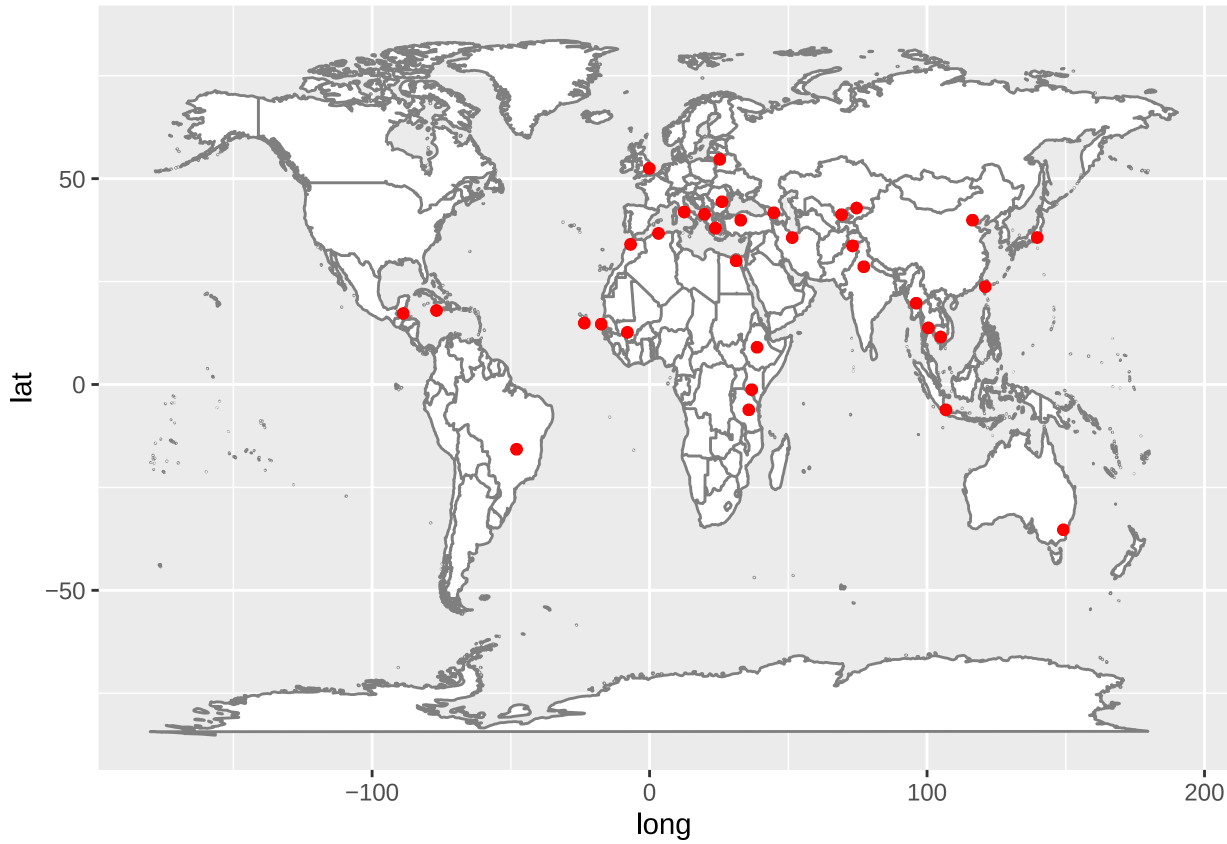
```r
# feature scaling

# separate labels from features
#label_column_names <- c("Longitude", "Latitude")
#data_labels <- data[label_column_names]
#data_features <- subset(data, select = -label_column_names)
# scale features
#data_features_scaled <- as.data.frame(scale(data_features))
# add labels to the now scaled features
#data <- cbind(data_features_scaled, data_labels)
```

some insights into the data

```r
# check if data is already standardized

# get columns without the target cols ("long" and "lat")
data_without_target_cols <- subset(data, select=-c(Latitude,Longitude))

# for each column in the data get sd and median
sd_per_col <- apply(data_without_target_cols, 2, sd) # the two stands for columns, if we would have use
sd_per_col_df <- data.frame(sd_per_col)

mean_per_col <- apply(data_without_target_cols, 2, mean)
mean_per_col_df <- data.frame(mean_per_col)

sd_and_mean_per_col_df <- merge(sd_per_col_df, mean_per_col_df, by="row.names", all=TRUE)

#par(mar = c(4, 4, .1, .1)) # to make the two plots show side by side and not above each other

boxplot(sd_per_col, data=sd_and_mean_per_col_df, xlab="standard deviation (blue line at 1)", y_lab="valu
abline(h=1, col = "blue", lty=5)

boxplot(mean_per_col, data=sd_and_mean_per_col_df, xlab="mean (blue line at 0)", y_lab="value", main="me
abline(h=0, col = "blue", lty=5)
```



```r
# basic world map with music origins
mapWorld <- borders("world", colour="gray50", fill="white")
mp <- ggplot() + mapWorld

mp + geom_point(data = data, aes(x = Longitude, y = Latitude), color = "red", alpha = 0.5)
```



Conspicuousness

When looking at the map, it seems like there are way fewer unique data points than expected. The paper states that there are 1,142 pieces from 73 countries/areas, but counting the point on the map just returns 33 data points. The following code investigates this difference.

```r
# investigate why there are fewer points on the map than regions on the map
# data is the full data set

# group data by unique combinations of long and lat and safe in data frame
# and count occurrences of each unique combination
# the unique combinations of long and lat represent regions
occurences_per_region <- data.frame(data %>% count(Longitude, Latitude, sort=TRUE))

nrow(occurences_per_region) # returns 33 -> 33 regions in the data set and not 71 like proposed in the
```

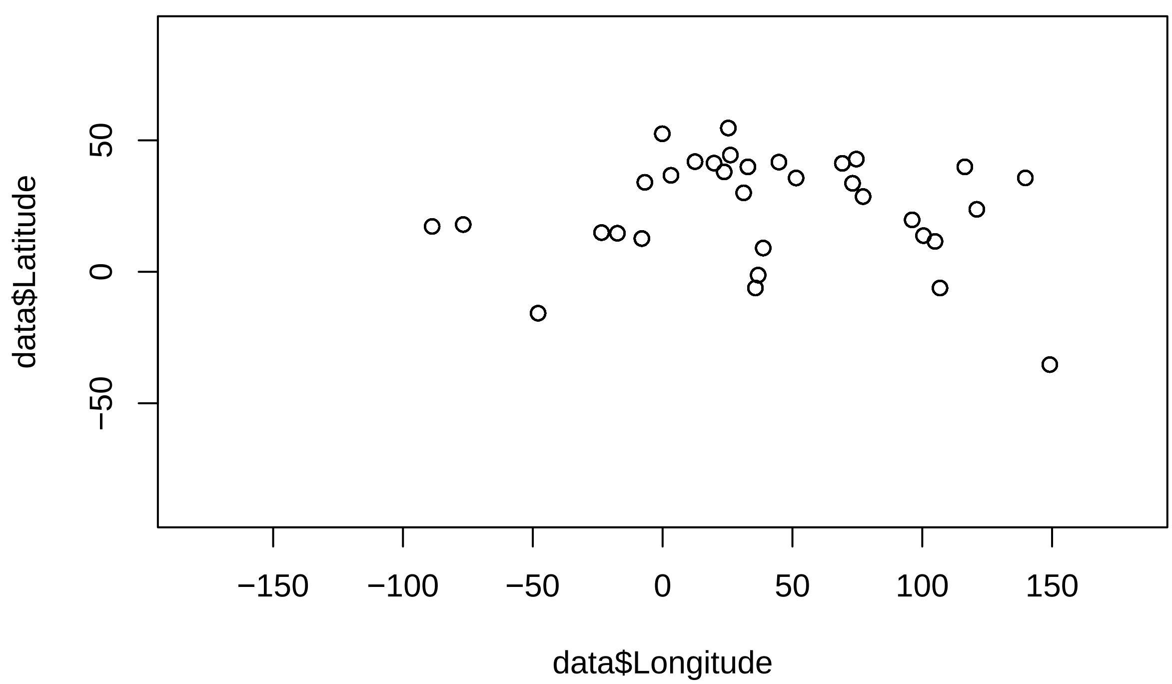
```
## [1] 33
```

```r
sum(occurences_per_region[, 'n']) # returns 1059 -> therefore there are 1095 tracks in the data set
```

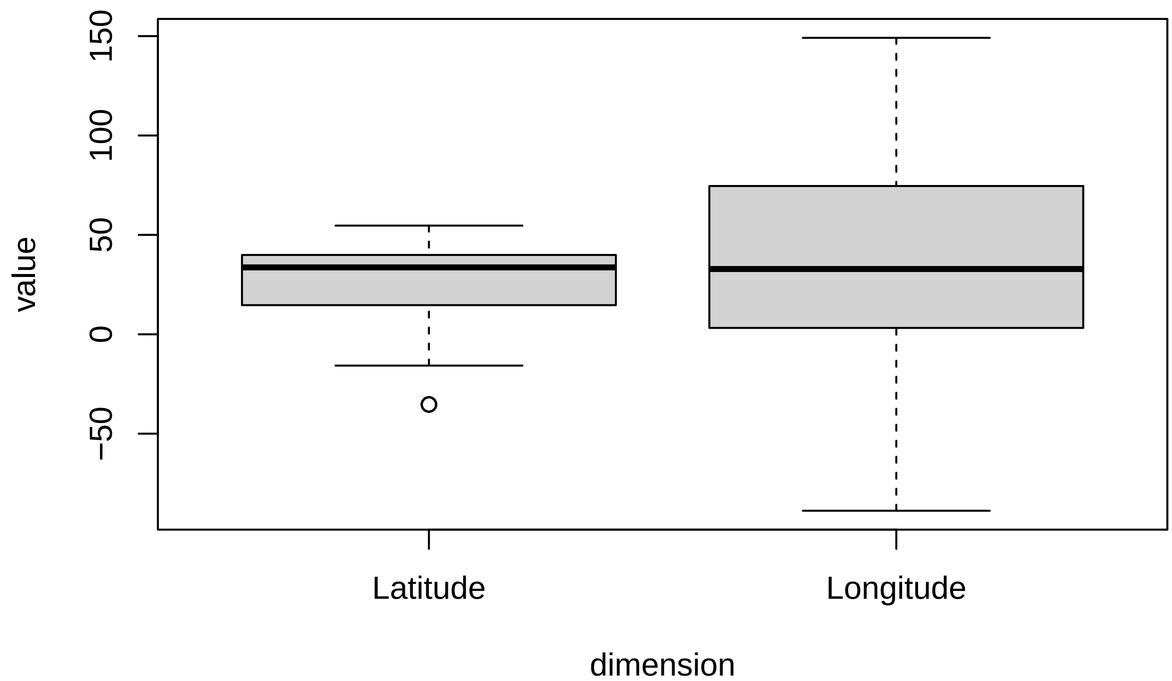
```
## [1] 1059
```

The results suggest that there are actually only 33 unique combinations of latitude and longitude in the data set. Therefore the pieces can only be categorized into 33 different categories. The reason for this discrepancy to the suggested number in the paper (73) might be that some regions have been aggregated prior to uploading the data or some row are missing. The second statement is also supported, by the fact that there are not 1,142 pieces in the data set as described in the paper, but only 1059 pieces. Whatever this (small) difference does not influence the tasked tackled/perused by this report.

```
plot(data$Longitude, data$Latitude, xlim=c(-180, 180), ylim=c(-90, 90))
```




```
df_lat <- data.frame(value = data$Latitude, dimension = "Latitude")
df_long <- data.frame(value = data$Longitude, dimension = "Longitude")
boxplot_df <- rbind(df_lat, df_long)

boxplot(value~dimension, data=boxplot_df)
```




```r
set.seed(1)
n <-dim(data)[1]
train <- sample(1:n, 0.8*n)
test <- (1:n)[-train]
```

Method

Short summary about the algorithms. Which are used? What do we do? Classification or Regression?

We treat the problem of predicting the geographical origin of music as a regression problem since we want to predict the spherical coordinates (longitude and latitude). In Zhou et al. (Fang Zhou, Claire Q, Ross D. King 2015) they list two reasons as to why it's preferably not treated as a classification problem: One, the ratio between the large number of countries/areas and the number of examples per country/area is very disproportional and would result in poor classification results. And secondly, with regression we already have a natural error metric: The geographical distance from the true position.

Baseline - Linear Regression

First, we take a baseline to get a basic understanding on how well our chosen algorithms perform. Therefore, we decided to use a linear regression. First we created a model which includes all variables. The lm() command cannot compute a model for both output variables at the same time. So it creates two separate linear models, one for each output variable:

```r
model.lm.all <- lm(cbind(Longitude, Latitude)~., data=data[train,])
pred.all <- predict(model.lm.all, newdata=data[test,])
```

To calculate a good meaningful measurement for the goodness of fit for the predictions the distance from the true location is calculated. The distances are calculated as the euclidean distances of the Longitude and Latitude between the predictions and the true location. They are measured in [km]. These will be used for all the comparisons of the algorithms between each other but also with the baseline and also with the literature (we need here another citation to the original paper)

The predictions are on average quite far from the true destination:

```
## [1] 5850.727
```

The impact of the variables was analysed. It seemed that not all of them have a significant influence on the models. So a second model was developed, just using the variable which have a significant influence on the full model, either on the Latitude or on the Longitude.

```
model.lm.sig <- lm(cbind(Longitude, Latitude)~ V4+V9+V16+V30+V32+V33+V37+V38+V61+V90+V91+V92+V95+V96
                   +V104+V5+V6+V8+V9+V11+V15+V34+V39+V63+V94+V97,
                   data=data[train,])
pred.sig <- predict(model.lm.sig, newdata=data[test,])
```

The predictions for the smaller models are on average a bit better:

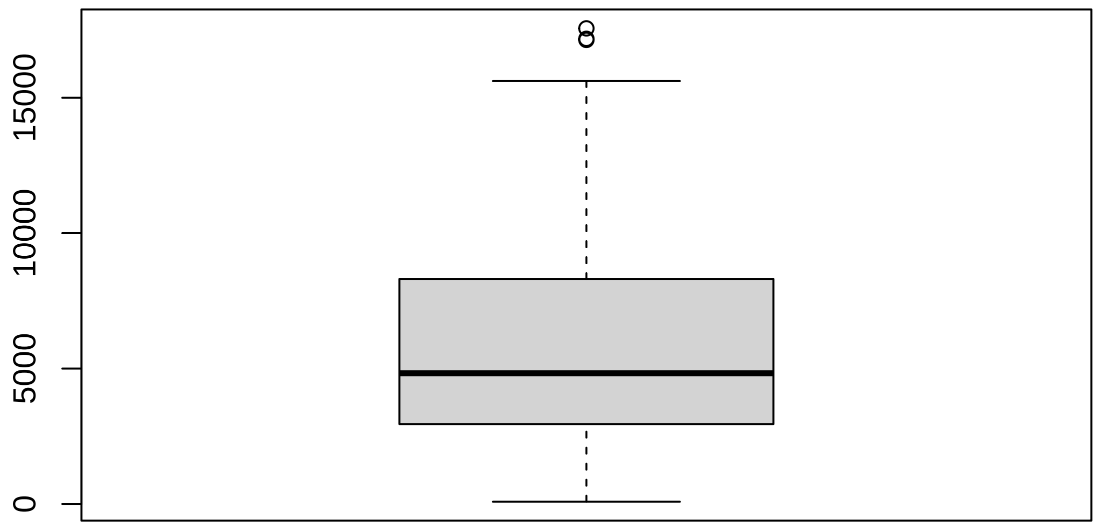
```
## [1] 5635.057
```

An ANOVA was calculated to check if there is a significant difference between the two models.

```
anova(model.lm.all, model.lm.sig)
```

```
## Analysis of Variance Table
##
## Model 1: cbind(Longitude, Latitude) ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 +
##     V8 + V9 + V10 + V11 + V12 + V13 + V14 + V15 + V16 + V17 +
##     V18 + V19 + V20 + V21 + V22 + V23 + V24 + V25 + V26 + V27 +
##     V28 + V29 + V30 + V31 + V32 + V33 + V34 + V35 + V36 + V37 +
##     V38 + V39 + V40 + V41 + V42 + V43 + V44 + V45 + V46 + V47 +
##     V48 + V49 + V50 + V51 + V52 + V53 + V54 + V55 + V56 + V57 +
##     V58 + V59 + V60 + V61 + V62 + V63 + V64 + V65 + V66 + V67 +
##     V68 + V69 + V70 + V71 + V72 + V73 + V74 + V75 + V76 + V77 +
##     V78 + V79 + V80 + V81 + V82 + V83 + V84 + V85 + V86 + V87 +
##     V88 + V89 + V90 + V91 + V92 + V93 + V94 + V95 + V96 + V97 +
##     V98 + V99 + V100 + V101 + V102 + V103 + V104 + V105 + V106 +
##     V107 + V108 + V109 + V110 + V111 + V112 + V113 + V114 + V115 +
##     V116
## Model 2: cbind(Longitude, Latitude) ~ V4 + V9 + V16 + V30 + V32 + V33 +
##     V37 + V38 + V61 + V90 + V91 + V92 + V95 + V96 + V104 + V5 +
##     V6 + V8 + V9 + V11 + V15 + V34 + V39 + V63 + V94 + V97
##   Res.Df Df Gen.var.  Pillai approx F num Df den Df    Pr(>F)
## 1    774      666.34
## 2    821 47   688.29 0.17239   1.5533     94   1548 0.0007525 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The model tells that there is a difference and, as seen before, the smaller model performs better.



Algorithm 1 - Ridge Regression

Short introduction of the first algorithm. What does it do? What are the strengths? What are weaknesses? How is it implemented, including major code snippets.

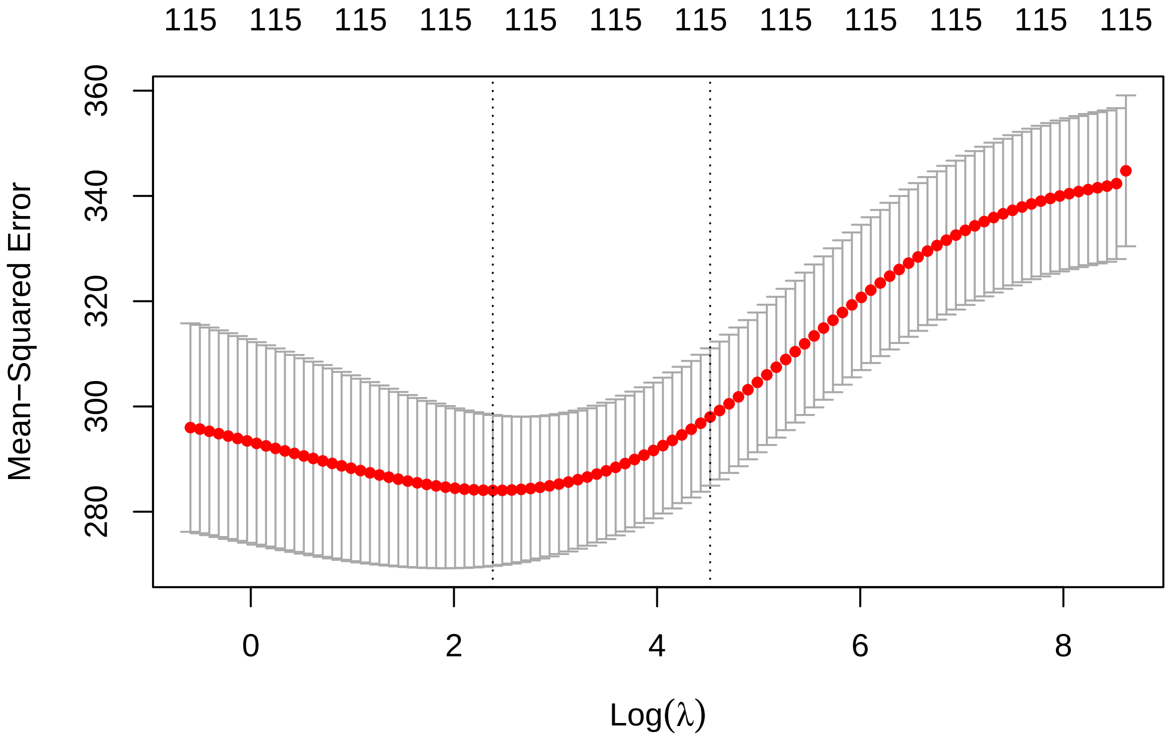
The first algorithm we tried was ridge regression. This algorithm is similar to a linear regression but while the linear regression tries to minimize the difference between the weighted input variables and the output data, the ridge regression adds a regularisation term on the input variables.

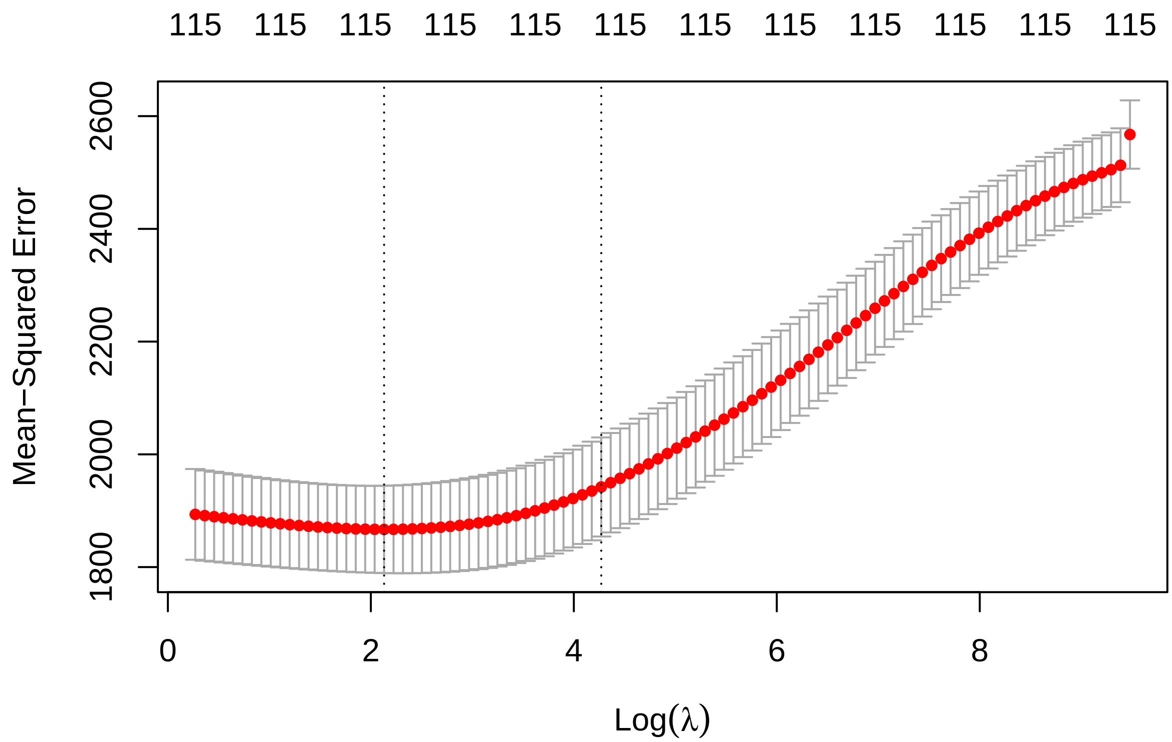
$$RSS + \lambda \sum_{j=1}^{p} \hat{\beta_j}^2 = \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \sum_{j=1}^{p} \hat{\beta}_j x_i j)^2 + \lambda \sum_{j=1}^{p} \hat{\beta}_j^2$$

In fact, this is a possibility to fight over-fitting. If lambda is big the model tends to just take the b0 into account. So the predicted value is the mean of the output variable. If lambda is small, then the model tends to be normal non-regularised model, hence the one from the linear regression.

The command glmnet is used to perform the ridge regression. Cross-validation is performed to find the optimal lambda values. In general, it would be possible with glmnet to just calculate one single model for both output variables. But unfortunately, this option is not available when doing the cross-validation. So, again two independent cross-validations are done to get two values for lambda, one for each output variable.

```
## [1] "data.frame"
```


115 115 115 115 115 115 115 115 115 115 115 115
Mean−Squared Error
360
340
320
300
280
0
2
4
6
8
Log(λ)



It can be seen that the distributions for lambda are quite similar for both models. Although the MSE for both models differ the optimal lambda is quite similar:

```
## [1] 10.83272
```

```
## [1] 8.419139
```

The model performs better with ridge regression as with the baseline. In this model two lambdas are used.

```
## [1] 4492.287
```

A second result is calculated with just one lambda. This lambda is calculated as the mean of the before calculated values of lambda. The result for the adapted version is just slightly worse than with the model with two independent lambdas:

```
## [1] 4496.877
```

Algorithm 2 - Artificial Neural Networks (NNs)

Short introduction of the second algorithm. What does it do? What are the strengths? What are weaknesses? How is it implemented, including major code snippets.

For the second algorithm we'll be using Artificial Neural Networks (NNs) as they can be used for regression problems as well. To use NNs for our predictions we will, as with all machine learning models, fit the data to the model. This is done by minimizing the loss function, which for NN-regression is the mean of squared errors MSE over the training set. In the case of fitting a neural network, it is much faster if the data are scaled/normalised first and also gives more appropriate results if the features that are used have different scales and ranges. But this is already the case here.

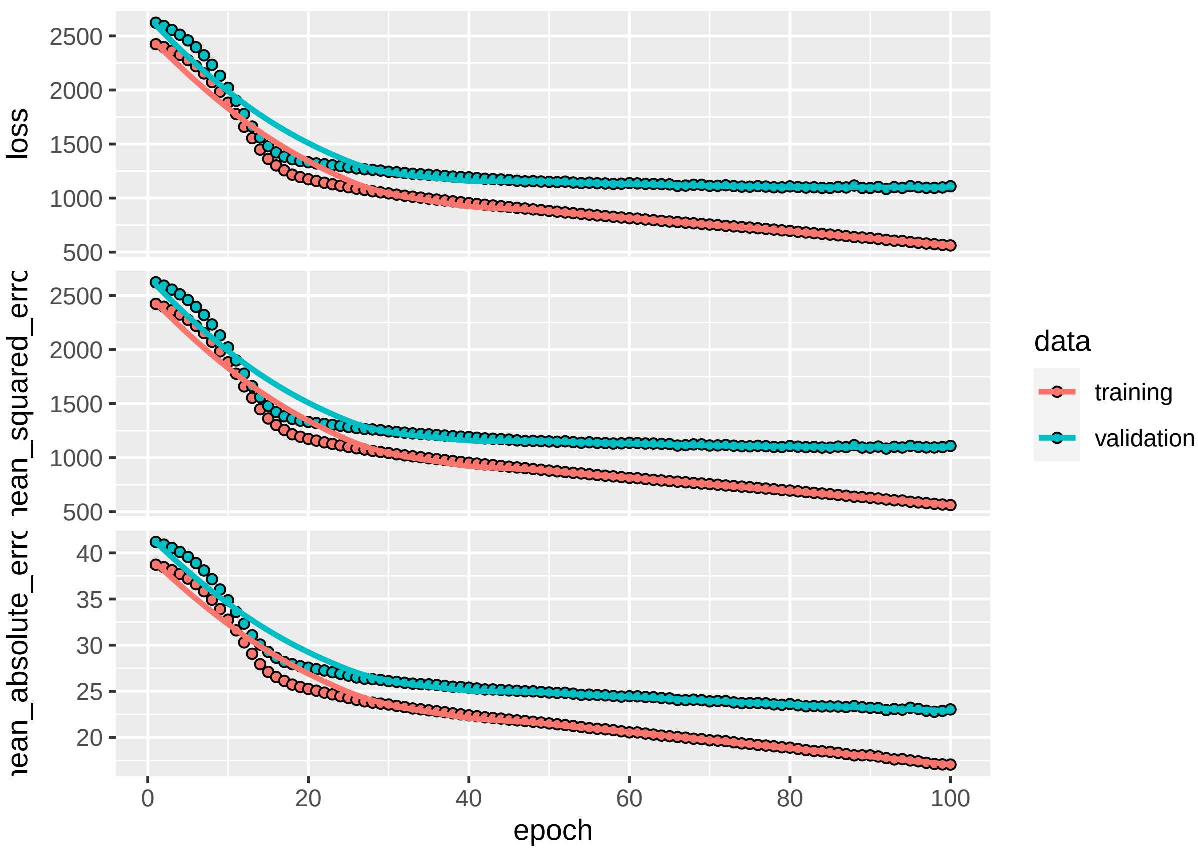
The NN takes all features from the training data and outputs two values for the Latitude and Longitude, so it is a multi-output regression. In-between Input and Output Layer are a few more layers defined which all use ReLU as activation function. For compiling and fitting the model we again use the Mean Squared Error (MSE) and Adam as the optimizer. We let the model train for 100 epochs with a batch size of 128 and validation split of 0.2.

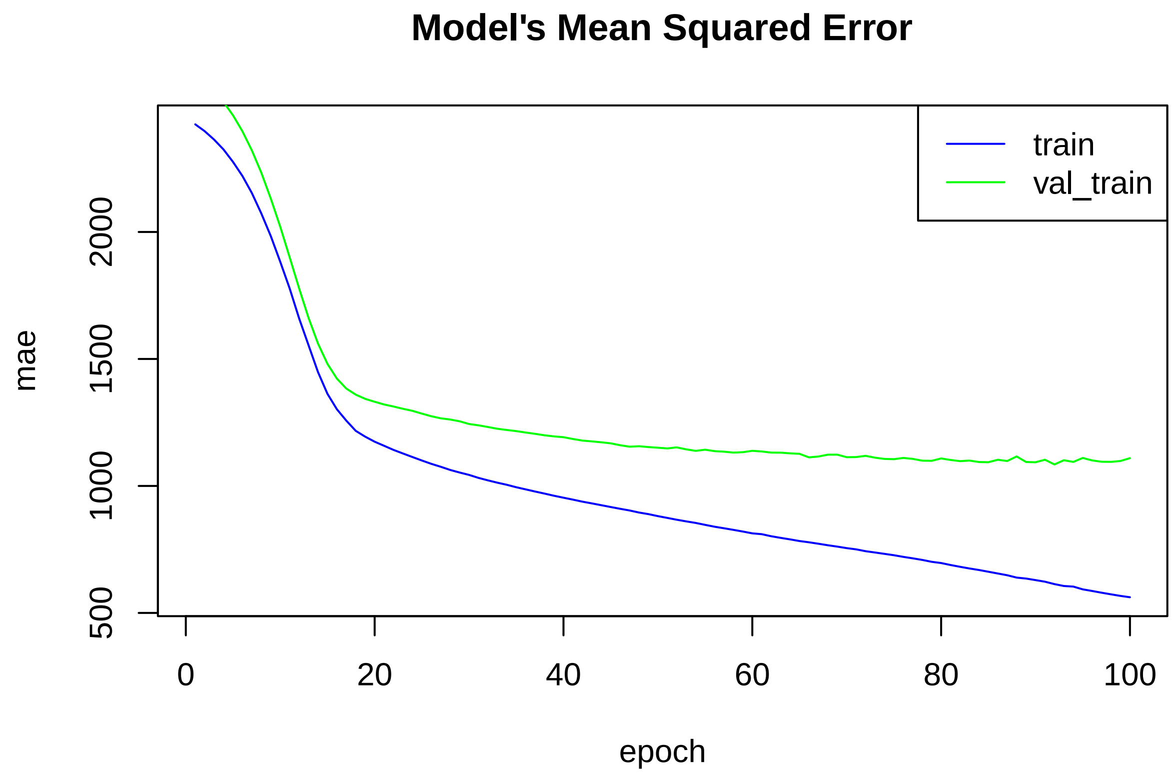
The results are visualized below.

```
## Loaded Tensorflow version 2.7.0
```

```
## Model: "sequential"
## _________________________________________________________________________________
##  Layer (type)                        Output Shape                     Param #
## =================================================================================
##  dense_3 (Dense)                     (None, 64)                       7488
##
##  dense_2 (Dense)                     (None, 16)                       1040
##
##  dense_1 (Dense)                     (None, 8)                        136
##
##  dense (Dense)                       (None, 2)                        18
##
## =================================================================================
## Total params: 8,682
## Trainable params: 8,682
## Non-trainable params: 0
## _________________________________________________________________________________
```

```
## `geom_smooth()` using formula 'y ~ x'
```






```
## [1] 665.5959
```

```
## [1] 3290.75
```

For evaluation we predict the Longitude and Latitude for the same test data as for the other regression cases and use the MSE as well as the custom distance metric. We get the following results:

```
## [1] 1185.859
```

```
## [1] 4360.799
```

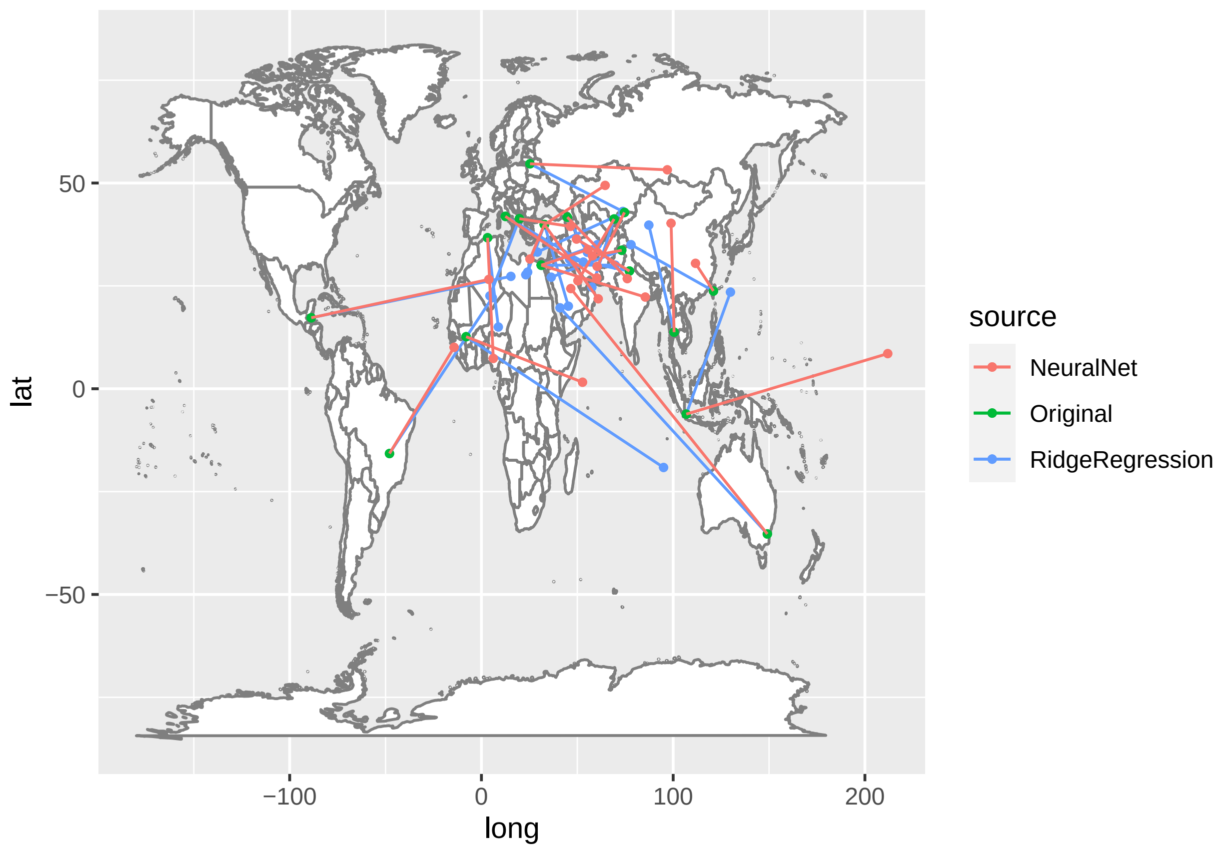
Results

```
## [1] 212   4
```

```
## [1] 212   4
```

```
## [1] 212   4
```

```
## [1] 424
```



Here some tables, summaries or especially graphs should be shown here. Maybe this section should be separated into two to show the algorithms for themselves

We decided on compare a traditional regression method against a more modern architecture like neural network assuming that the last one would perform better overall. First, in our implementation of ridge regression we are fitting two separate models in order to get the two desired output variables. Nevertheless, we know that latitude and longitude are correlated variables, fact that ridge regression doesn't take into account so we expected a bigger error than NN. In the contrary, above implementation of neural network is predicting two variables at once like examples given in paper.

Discussion

Here follows the discussion of the results. What are the major findings? How did the algorithms perform? Which one was better overall? Is it always better or were the findings which were better by the other one? Which one should be implemented? How could the algorithm be tweaked to perform even better? Where were the problems during implementation? Where are the limits for the algorithms? How precise do we predict the cities? How far is the difference in kilometers? The authors of the paper where the dataset comes from have a mean great circle error of 3113km? Are we above or below and by how much?

As we introduced, predicting data points on the Earth using a latitude/longitude representation adds complexity to our problem because of the natural characteristics of coordinates. In first place longitude is discontinuous, meaning that the longitude of two points geographically near might be significantly different and secondly because coordinates are not linear.

If we compare the two models applied by looking at MSE we know that neural network performs better than ridge regression, as we assumed.However the difference is not big _____ km. Nevertheless, the algorithms presented in In Zhou et al. (Fang Zhou, Claire Q, Ross D. King 2015) (KNN and RFR) performs significantly better than ours, achieving an average distance error from 3,100 km to 3,600 km.

Conclusion

At final some conclusions about the key findings and which algorithm should be used. What was the goal? Were and how were they achieved?

Fang Zhou, Claire Q, Ross D. King. 2015. "Predicting the Geographical Origin of Music." Paper. https://ieeexplore.ieee.org/document/7023456.

Tzanetakis, George. 2007. "Marsyas: A Case Study in Implementing Music Information Retrieval Systems." In *Intelligent Music Information Systems: Tools and Methodologies*, edited by Shepherd Shen and Liu Cui. Information Science Reference. http://marsyas.sness.net/pdfs/0000/0007/imis_bookchapter.pdf.