Universität Zürich
Institute for Computational Science
Spring Semester 2021
Final Exam ESC403 - Dave Linder

Professor Robert Feldmann (lectures)
Dr. Darren Reed (exercise classes)
Dimakopoulos Vasileios (TA)
Elia Cenci (TA)

INTRODUCTION TO DATA SCIENCE

FINAL EXAM 2021

Solutions from David Linder

June 11, 2021

This are my solutions to the final exam 2021 in introduction to data science. All the computations and the code as well as the data and plots and also this document can be found in the git-repository: https://github.com/massstab/ESC403_exam.git. I included also the .csv files asked for in the folder `report/data/`.

1 Time Series

1.1

To be considered stationary a time series should have the following properties:

- The mean $E[X_t]$ is the same for all times t.
- The variance $Var[X_t]$ is the same for all times t.
- The covariance between X_t and X_{t-1} is the same for all t, n.

That means we want

- no obvious trends.
- constant variance with time.
- constant autocorrelation structure over time.
- no periodic fluctuations (no seasonality).

I found some examples here: In figure 1 we see that most of the series are non-stationary except series (b) and (g). In series (g) there are cycles but they are not periodic.

1.2

- By looking at the data we can clearly see that this time series is not stationary. After log transformed the X-data we can see in figure 2 that although the standard deviation has a small variation the mean increases over time. Also the results of the Dickey-Fuller test is that we can not reject H_0 (TS is non-stationary) because the test statistic value is not less than any critical value. Here is the data in detail:

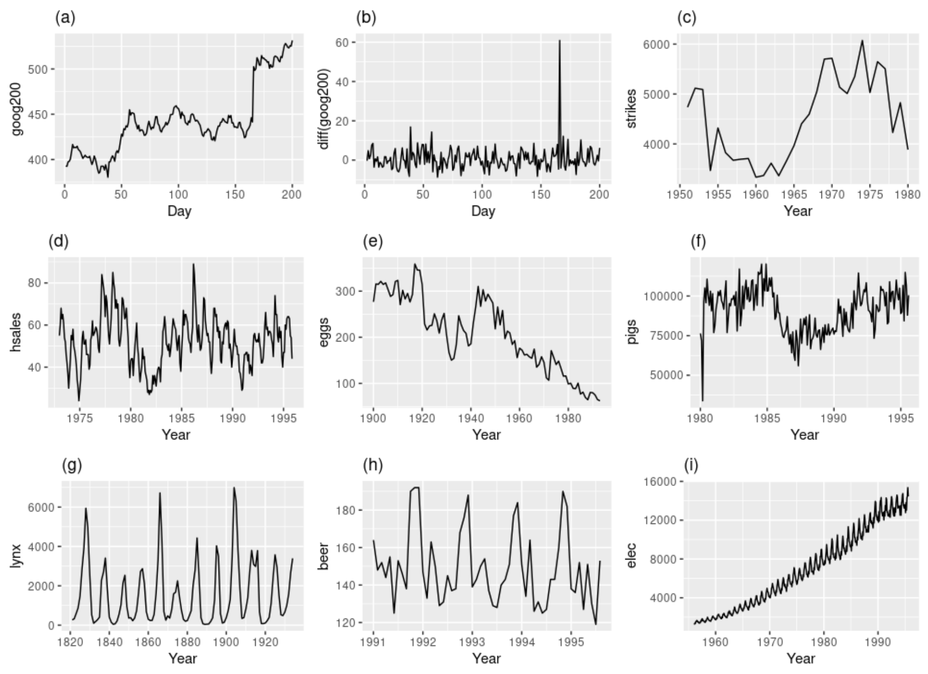

Figure 1: (a) Google stock price for 200 consecutive days; (b) Daily change in the Google stock price for 200 consecutive days; (c) Annual number of strikes in the US; (d) Monthly sales of new one-family houses sold in the US; (e) Annual price of a dozen eggs in the US (constant dollars); (f) Monthly total of pigs slaughtered in Victoria, Australia; (g) Annual total of lynx trapped in the McKenzie River district of north-west Canada; (h) Monthly Australian beer production; (i) Monthly Australian electricity production.

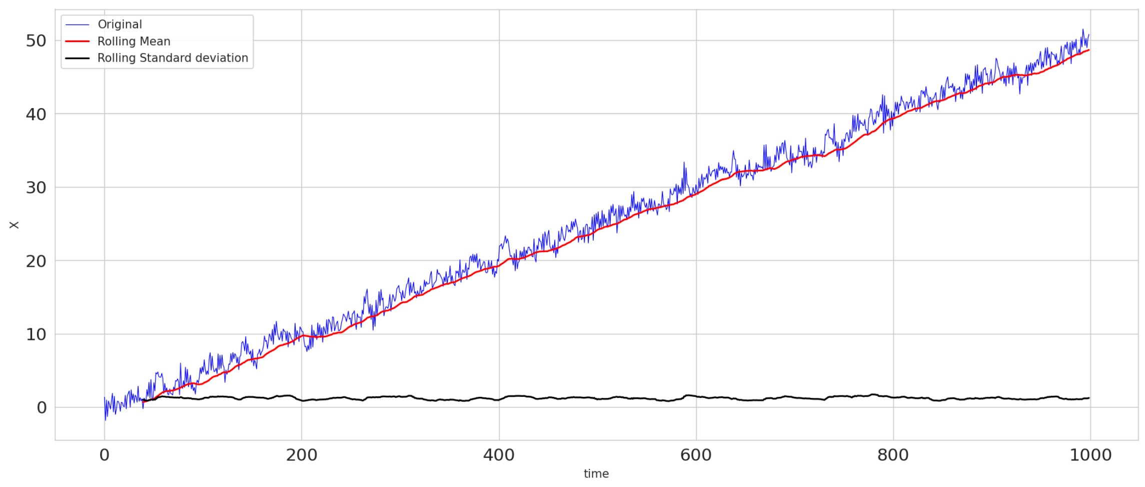

Figure 2: The time series after a log-transformation. This series is clearly not stationary. The mean increases over time.

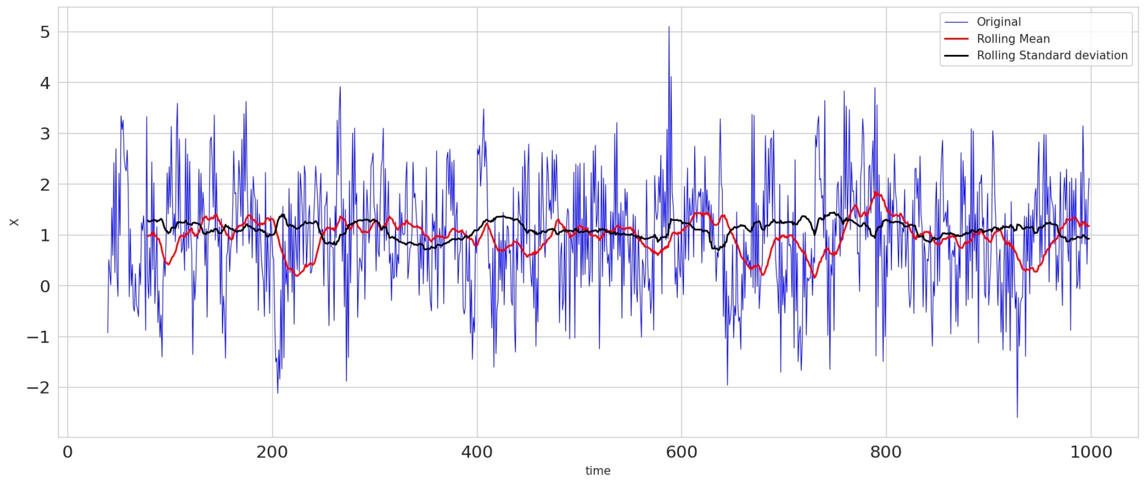

Figure 3: The result from smoothing and differencing. We can now consider this as a stationary time series.

Test Statistic	-0.2312
p-value	0.9347
Lags	22.0000
Observations	975.0000
Critical Value (1%)	-3.4371
Critical Value (5%)	-2.8645
Critical Value (10%)	-2.5684

I will use the moving average smoothing method which we used in the exercise class to eliminate the trend. First i calculated the rolling mean with the pandas function and then subtracted this from the original series (code: `timeseries/Q1.2.py`). Then i dropped all the nan values that resulted from averaging over the time window (i tried out different sizes of that window and ended up with a value of 40). To remove seasonality i applied differencing. In figure 3 we see the result and that we eliminated the trend. Let's take a look at the test statistics:

Test Statistic	-10.7860
p-value	0.0000
Lags	1.0000
Observations	937.0000
Critical Value (1%)	-3.4373
Critical Value (5%)	-2.8646
Critical Value (10%)	-2.5684

The value is lower than all the critical values. Therefore we can say at least with 99% confidence this is now a stationary time-series.

- For answering this question we need to find the p-value. For this value to find we look at the plot of the partial autocorrelation function (PACF). From figure 4 we find a p-value of 3. That

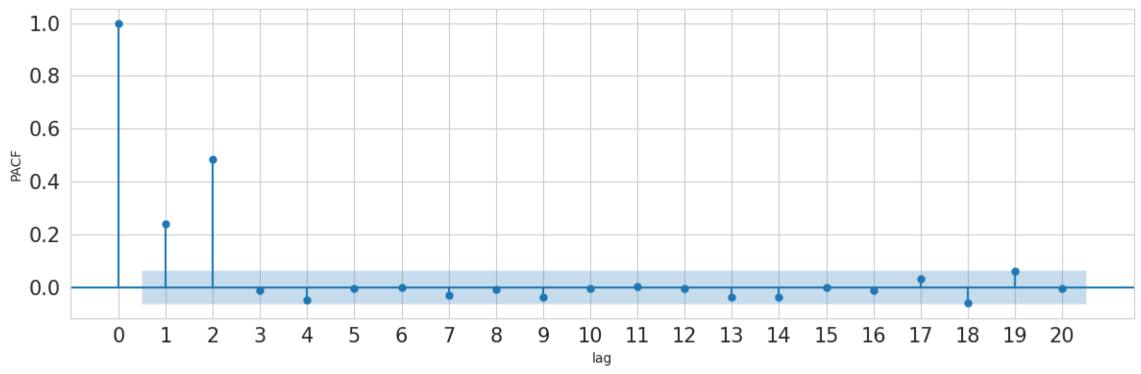

Figure 4: Plot of the partial autocorrelation function. The x-axis are the lags. One can see that the first time the PACF crosses the confidence interval (blue) is at lag 3.

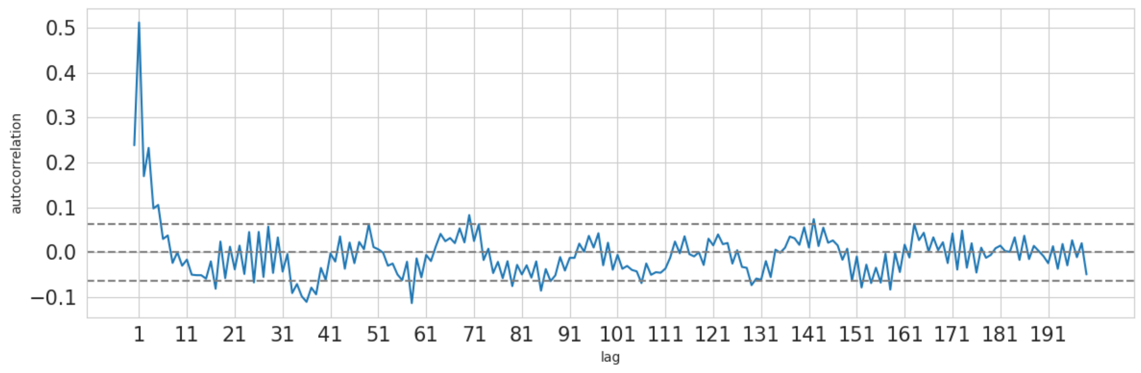

Figure 5: The autocorrelation function to identify the ARIMA model.

means that 3 values of at prior times are expected to directly affect a given current value of the time series.

- To find which ARIMA model would be appropriate we look at the autocorrelation function in. We clearly can see that the function decays to zero while alternating between + and -. This seems to be an AR model (for which we already identified the p-value as 3). Maybe one could also try a mixed AR and MA model. For this i also identified the q value. The function crosses the confidence interval at lag 7 $\implies q = 7$.

1.3

- In figure 6 we see the mean increasing over time. Here the results from the test statistic:

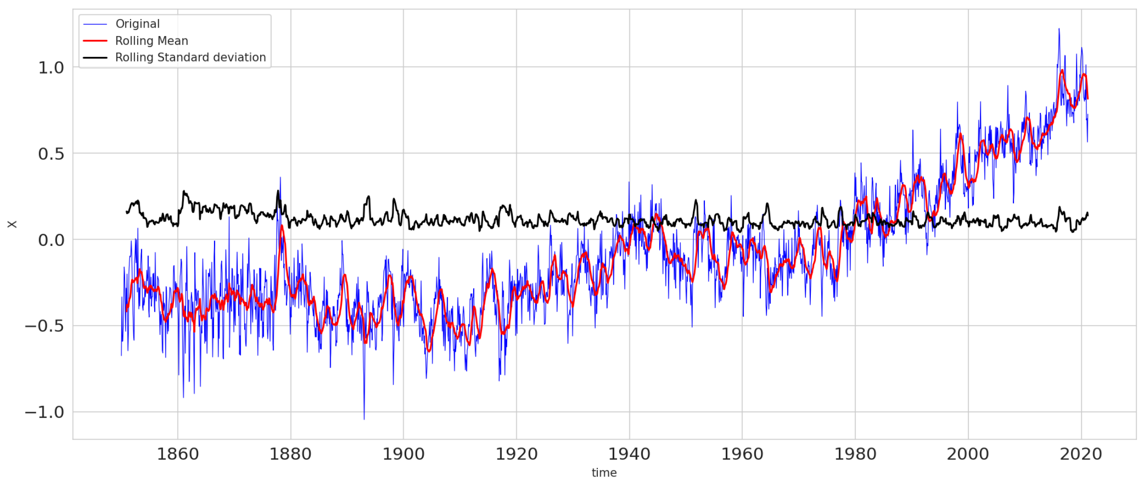

Figure 6: Average temperature anomaly raw data.

Test Statistic	-0.5905
p-value	0.8730
Lags	23.0000
Observations	2031.0000
Critical Value (1%)	-3.4335
Critical Value (5%)	-2.8629
Critical Value (10%)	-2.5675

After differencing the time-series looks stationary as we ca see in figure 7. Let's support that with the statistics:

Test Statistic	-15.7773
p-value	0.0000
Lags	22.0000
Observations	2031.0000
Critical Value (1%)	-3.4335
Critical Value (5%)	-2.8629
Critical Value (10%)	-2.5675

The test statistic value is clearly under the critical value of 1% therefore we can say whit at least 99% certainty that this is now a stationary time series. To find the p and q we look at the ACF and PACF in figure 8 and 9 respectively. And we find for $p = 12$ and $q = 2$.

We use a KDE to see the residuals and its variance. In figure 10 we can see that the mean of the residuals is near zero with a good uniform variance. That means we can use the model to predict future values.

- Here we try to give a prediction for the average temperature anomaly from the years 2010-2021. In figure 11 we see in orange the measured values and in blue the prediction with the ARIMA model. The resulted RMSE of only 0.1188 seems a bit low to me.

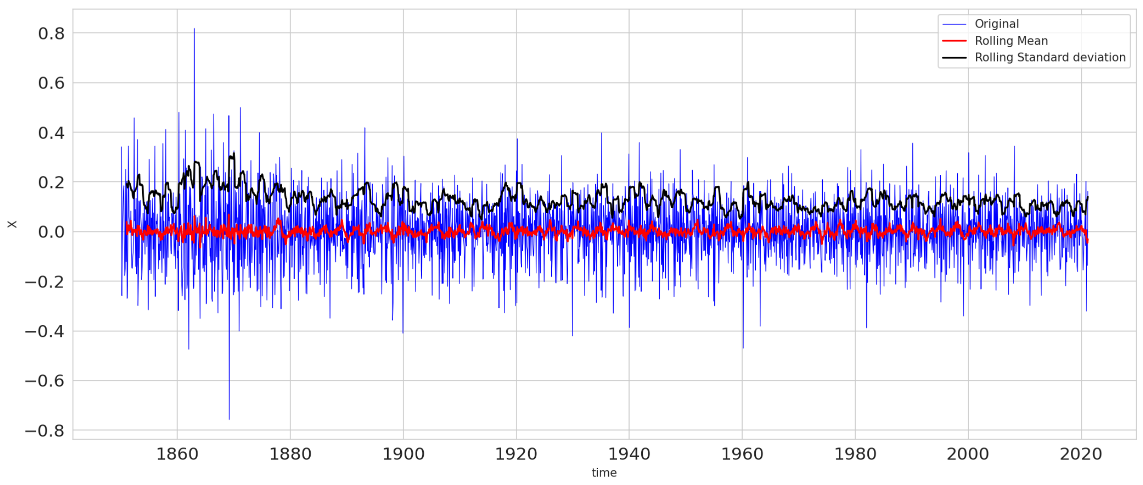

Figure 7: Average temperature anomaly after differencing.

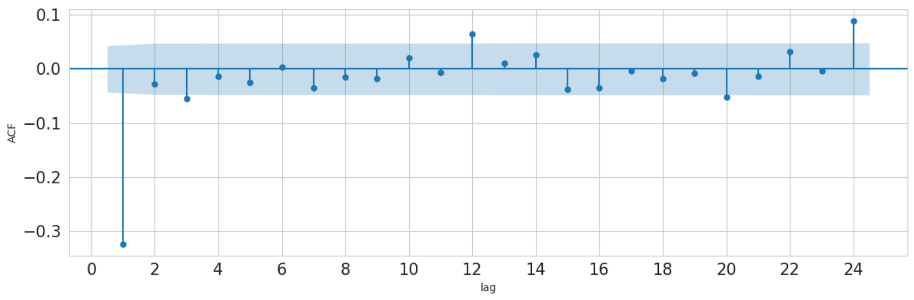

Figure 8: The autocorrelation function. One can see that the first time the ACF crosses the confidence interval (blue) is at lag 2.

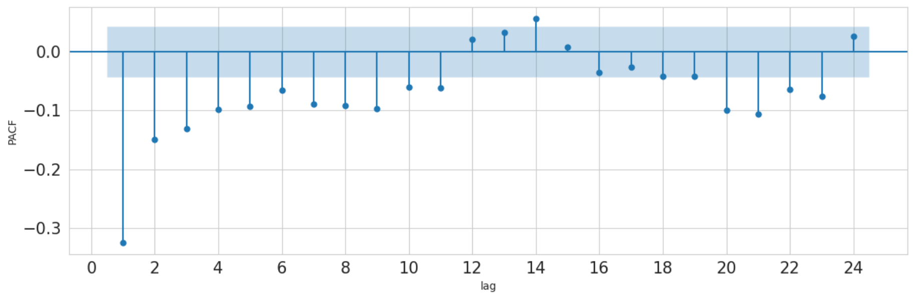

Figure 9: The partial autocorrelation function. One can see that the first time the PACF crosses the confidence interval (blue) is at lag 12.

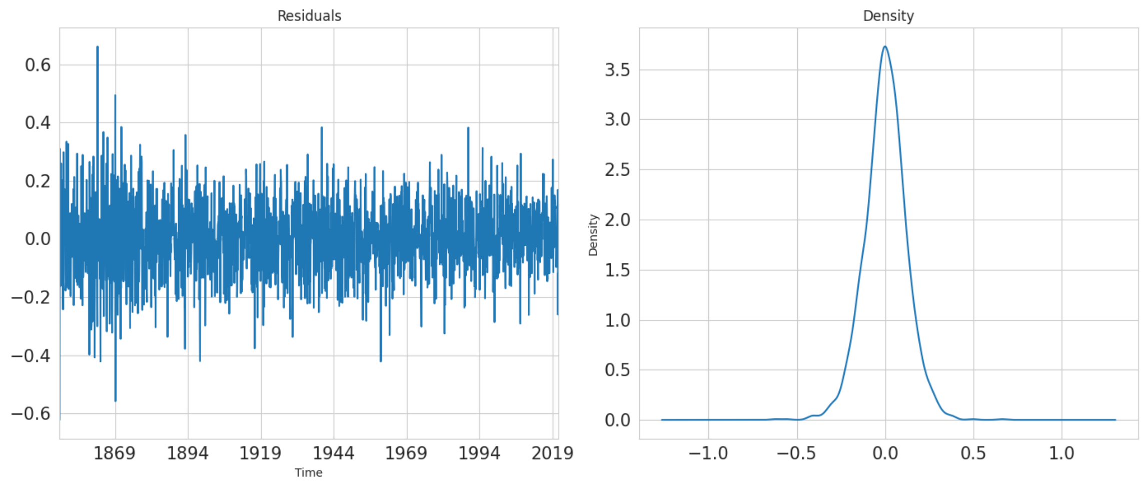

Figure 10: Residuals and density.

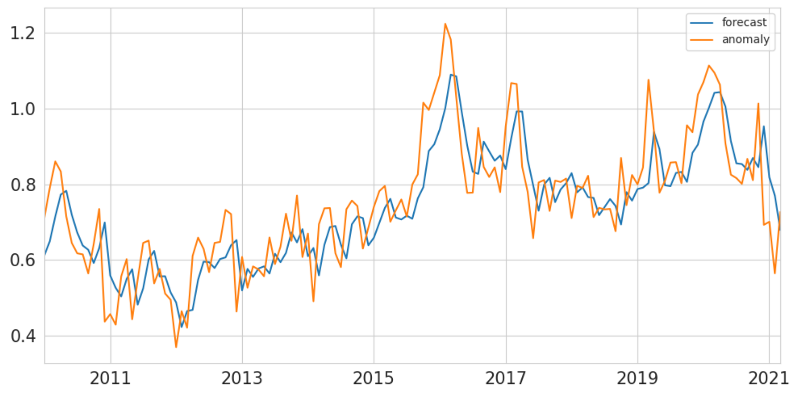

Figure 11: Forecast 2010 - 2021

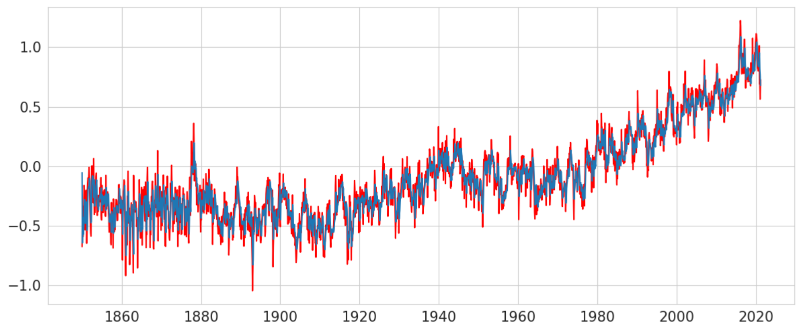

Figure 12: Forecast 2010 - 2021. This resulted in a RMSE of 0.1188.

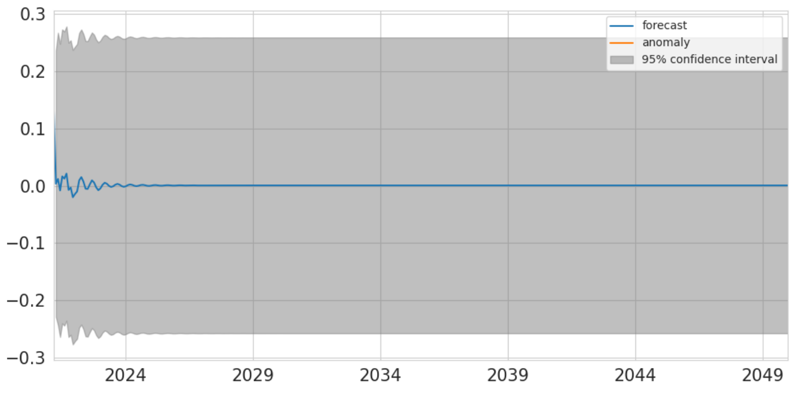

Figure 13: Forecast 2021 - 2050. The model always ended up giving me decaying results. Doesn't matter what parameters of the ARIMA model i tweak'd.

- I tried a lot of different models and values for p ad q. But all gave some very fast decaying solution for my prediction like in figure

2 Image classification

2.1 Image classification

If better means higher accuracy then i would choose a CNN for image classification. RF performs well on categorical data while a CNN handles numerical input very well. Here we have pixels with numerical values between 0 and 255. We can also tune more parameters in a CNN like kind/number of layers, epochs and learning rate. Also different activation functions for the neurons can be chosen. Finally a CNN learns how to apply a filter to an input during training such that certain features in the image can be recognized. A RF can not take advantage of such structures in images.

On the other hand a CNN needs a lot of data to perform well. I don't know exactly if our dataset is large enough but if i compare with other models they used hundred thousands or even millions of images to train a CNN.

2.2 Classification performance

- Accuracy CNN: 99.1%, that means the error rate was 1%. It misclassified 10/1000 images from my test set.
- Accuracy RF: 99.6%, that means the error rate was 0.4%. It misclassified 5/1000 images from my test set.
- Both models performed surprisingly good. I checked many times if i excluded any test data from the training. No model performed significantly better.
- Both models agreed in all images only the CNN misclassified image 001212.jpg where it predicted the class 'headCT' instead of 'Hand'. As far as i can tell the RF made no prediction error on the unlabeled dataset. The predictions with the images can be found in figure A1 in

the appendix. I included the corresponding files in the folder `report/data` in the repository (`predictions_cnn.csv`, `predictions_rf.csv`).

2.3 Hyper parameter tuning

- For the random forest i pretty much left the default parameters. I played around with the depth of the tree (max_depth) and the number of trees in the forest (n_estimators). Nothing gave me significant improvement. The only thing i did was switching off bootstrapping.
- The CNN i started from an older task i did with image recognition. For the optimizer i choose adam with a learning rate of 0.001. I played around with number of layers and filters in the Conv2D layers. For performance reason i did a pooling layer at the end of two consecutive convolutional layers. I also played around with the kernel size and different activation function such as relu and sigmoid. After a flattening layer i applied a dense layer followed by a dropout layer (quiet high 50%) and a dense layer at the end with a softmax of 6 unit (one per class) which gave me the probabilities for each class. A detailed plot of the model built with keras can be found in the appendiy in figure A2.

2.4 Interpretation of model performance

For the training and validation accuracy and loss i made a plot in figure 14. The evaluation of the random forest model i did with the sklearn integrated method `precision_recall_fscore_support()`. This module provides information about precision, recall and f-score. I couldn't do a out-of-bag (OOB) error estimation because i turned bootstrapping of. All this can be seen in the script `classification/Q2.py`. I couldn't see much under- or overfitting on the models or bias. But to prevent overfitting in random forest i would probably turn on bootstrapping or growing a larger forest.

To avoid bias i would look at the bias-variance trade-off. As we learned in class, more flexible methods tends to have lower bias but higher variance and more restrictive methods tend to have lower variance but higher bias. Then i would choose the mothod with the right amount of flexibility to minimize test error (via cross validation).

2.5 Bonus Question

I implemented boosting trees with the help of this colab notebook. That got me an surprisingly high accuracy of 99.7%. This is better then the random forest approach. Even with the default settings and without tweaking anything at the parameters.

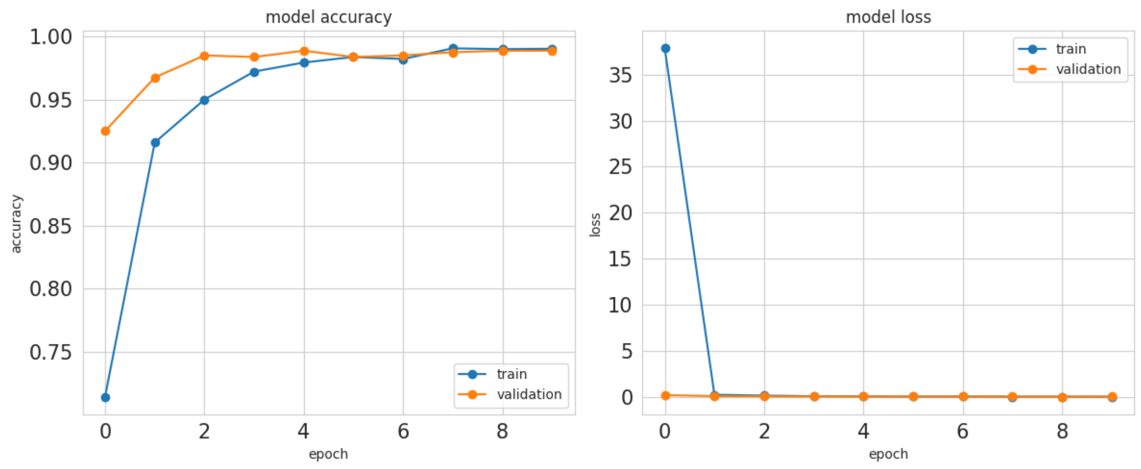

Figure 14: CNN training and validation accuracy and loss in the training phase during 10 epochs.

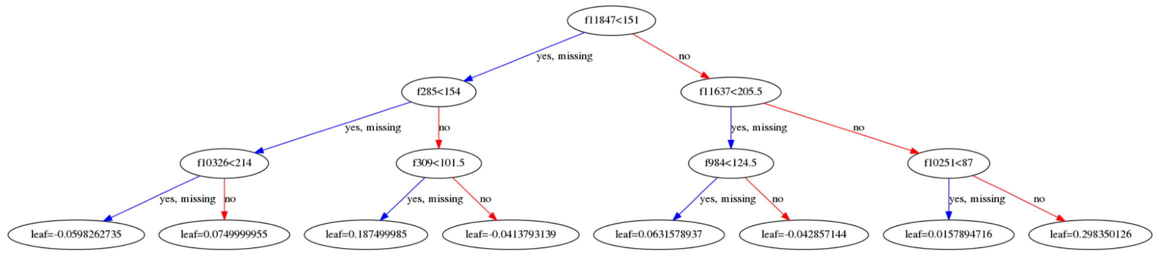

Figure 15: This is the tree from the boosting model. I have no idea what the numbers or the arrows or the colors or anything mean. It seems like someone plotted this just because it looks good and he would get the bonus points... I would never do that myself.

A Additional Plots

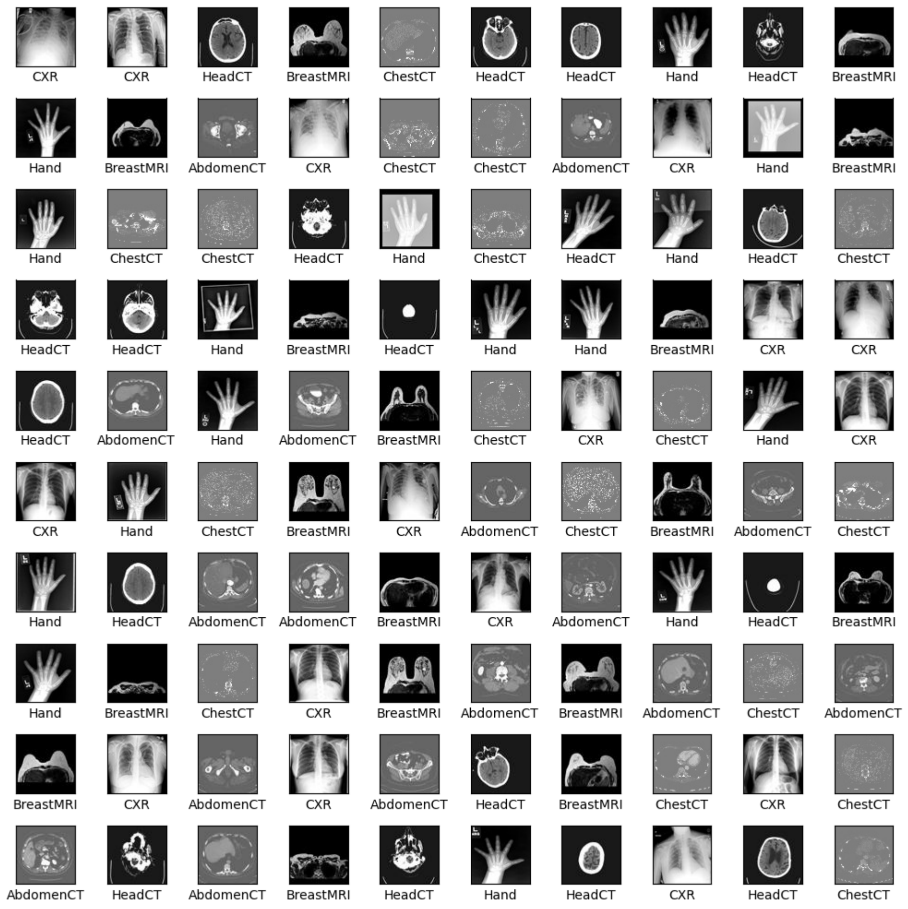

Figure A1: The prediction from the convolutional neural network. It misclassified one image. Can you find it?

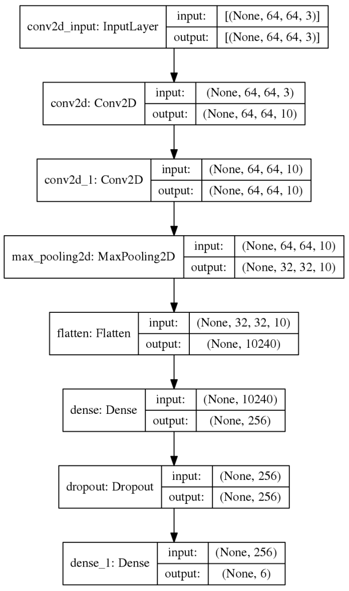

Figure A2: The CNN model built with tensorflow.