IEOR242 Project Report

Predicting Energy Imbalance in the German Electricity Grid

Noranne Gabouge, Tim Schmidtlein, Qing Wen, Yiyang Wu, Xinyu Zhang

May 2020

1 Introduction

Imbalance in the electricity grid occurs on a regular basis as a consequence of diverging supply and demand for power. Traditionally, the main source of supply uncertainty were power plant outages and other failures in the system. However, in light of higher levels of renewable energy ('*Renewables*') in the system, the situation has fundamentally changed: power supply at any given time cannot be deterministically planned, but depends on the weather (solar radiation and wind, mostly) and thus becomes stochastic to a certain degree.

Predicting power imbalance is of great significance for a variety of reasons. First, being able to predict (major) imbalances ahead of time enables Transmission System Operators (TSOs) to prepare appropriate countermeasures to maintain line voltage and frequency, and thus to prevent malfunctions and power outages in the short term. Second, understanding the underlying dynamics and sources of power imbalances allows to make the system more resilient in the longer term, by adjusting balance reserves and market design. Third, from a more individual perspective, predicting imbalances can also be a highly profitable endeavor for energy trading companies, since the European system offers financial rewards for stabilizing the grid by being *on the right side of the market*. It does not surprise then that the prediction and explanation of power imbalance has sparked great attention both in industry and academics in recent years (see [1], for instance).

2 Project Scope

The scope of this project is to analyze the predictive power of three classes of explanatory variables in terms of quarter-hourly imbalance prediction. Specifically, we consider a selection of (i) weather variables, (ii) time variables and (iii) previous imbalances to build our models. All of the data we use is publicly accessible. In practice, imbalance prediction is often done with much less accessible information ([1]). However, our main concern here is to understand if and (if so) to what extent such publicly accessible data can still contribute to imbalance prediction in contrast to the *efficient market hypothesis*.

Weather variables directly influence the power generation of *Renewables* which in Germany today account for about 38% of total power consumption.[2] Thus, if the pattern of power imbalances is different for different levels of renewable energy production, we would expect weather variables to have some predictive power in the models we consider.

Time variables capture important information such as *time of day*, *day of the week*, etc. With trading activity and power demand varying over the course of the day, week, month and year, these variables are suitable to detect corresponding patterns in the imbalance data.

Previous imbalances in the system may imply an increased probability of continued imbalance in the next quarter hour and are therefore included to search for serial correlation in the data.[1]

[1] Note that power imbalance is reported every 15 minutes, however, major unforeseen imbalances may take longer to be fully balanced out.

Our analysis consists of two parts: We first consider binary classification to predict the *sign* of the imbalance and then move on to different regression models to predict the actual imbalance *value*. Our intuition is that binary classification is *easier* to predict, but can still add plenty of value as it can inform a decision about what countermeasures to prepare (excess demand or excess supply?).

3 Data Collection and Pre-Processing

Official **power imbalance data** is collected from regelleistung.net [3], as a quarter-hourly time series for each month over the time period (April 2016 to March 2020). The data includes operational and quality-assured imbalance values (in MW), for each of the 4 TSOs (TransnetBW, TenneT, Amprion, 50Hertz - cf Appendix, figure 1b) as well as for the entire network.
Weather data is obtained from the German Weather Service [4]: we selected 4 weather stations across Germany, based on the repartition of wind and photovoltaics capacities (cf Appendix, figures 1a, 1c), and collected wind (speed, direction, quality level of measurement), temperature (air temperature, relative humidity, quality level), cloud (type of measurement, total cloud cover, quality level) data for each of them, as well as solar data (3 radiation measures, sunshine duration, zenith angle and solar time) for the 2 stations with solar measurement capacities. We get 2 .txt files of hourly weather data (years $\leq$ 2019 and year 2020) for each measurement and each station.

We chose to consider dates over the 4-year period between April 2016 and March 2020. Note that only a subset of all the collected variables will be of potential interest for the purpose of this project : for instance, for power imbalance we consider only operational values.

After data collection, 284 Mo of data (75 files) are to be analyzed, aggregated and preprocessed. Indeed, a number of considerations call for preprocessing.

First, note that the **frequency** is different between power data (quarter-hourly) and weather data (hourly) : more precisely, power data is measured every 15mn, weather data other than solar is reported every hour (00 to 23) while solar data is reported every *solar hour* (which corresponds to a different time depending on the season). In particular, this frequency difference will introduce missing values when the datasets are aggregated. Besides, we have to consider three different date formats (for imbalance data : 2 columns dd.mm.yyyy and hh:mn ; for nonsolar weather data : yyyymmddhh ; for solar data : yyymmddhh:mn).

Second, the **time zone** is UTC for weather data, but Berlin time for power imbalance data. As we expect an influence of working hours on power imbalance, we will convert timestamps to Berlin timezone.

Third, the data will comprise **missing values**, as a result of the following factors :
(1) original missing values from the data source (e.g. indicated by -999 for weather, -1 for clouds when the sky is not observable) ; depending on the weather station, some days are skipped here and there;
(2) adjusting timezones, which can introduce some missing values around the dates of daylight saving time change ;
(3) missing timestamps in weather data ;
(4) difference of frequency between imbalance and weather data (every quarter hour vs every hour vs every solar hour). This will be dealt with by repeating weather data *backwards* over the previous hour : indeed, based on the documentation for weather data, reported measurement time corresponds to either a measurement realized 10 mn before (cloud, temperature, solar) or an average of 6 measurements over the past hour (wind)

Initial preprocessing with Python covers the following steps:

- after loading the different .csv and .txt files : for each type of measurement (imbalance, cloud, wind, solar, temperature for each station), the different files covering the period 2016-2020 are

aggregated. Rows corresponding to observations before April 1st, 2016, 00:00 (12 am) are removed (keeping a 2 h observations before this time for weather data to allow for shifting the timezone without introducing missing values), and only potentially relevant columns are kept. Non solar weather data frames are merged w.r.t. the date

- date columns are converted to date object, for each of the different date formats, and localized by timezone (UTC or Europe/Berlin) ; for solar, minutes are converted to the corresponding quarter hour : e.g 00:26 will be converted to 00:15 (since the actual measurement is made 10 mn before the reported time, according to the documentation)
- the 3 data frames (imbalance, solar, non-solar) are merged under timezone 'Europe/Berlin'
- weather observations are repeated backwards (over the 3 rows before every observation)
- unnecessary date columns are removed and rows out of the time period (April 1st, 2016, 00:00 to March 31st, 2020, 23:45) are removed, before exporting to .csv

Preprocessing in R then consists in converting entries to corresponding data type (factor, numeric, date) and creating the following additional variables :

- **time-related variables**: year, hour of the day, quarter hour of the day, weekday, month
- **imbalance variables** (for the network and for each operator): sign of the imbalance, sign of the imbalance of the previous quarter; value of the imbalance with different lags (lag 1, i.e. previous quarter hour, lags 2,3,12,13,14,15)

The dataset (140k rows and 66 variables) is then split into training (3 first years) and test data (last 25% of the 4-year period) - see tables (Table 4-9) in the Appendix for variables in the final dataset.

4 Analytical Models

4.1 Predicting the sign of the power imbalance

Predicting the sign of power imbalance constitutes a binary classification problem, which we address by training different models : Logistic Regression (all features are included at first, then non significant independent variables are deleted), Linear Discriminant Analysis and CART. Table 1 displays the accuracy of these models on both the training data and the testing data.

Model and Parameters	Accuracy on Training Data	Accuracy on Testing Data
Baseline	0.617	0.531
Logistic Regression	0.828	0.802
LDA	0.826	0.827
CART	0.826	0.800

Table 1: Binary classification performance

Not all independent variables turn out to be significant. These models rely heavily on imbalance features, more specifically the value of the previous quarter imbalance (lag 1), while older lags (older than 3h) yield a much lower accuracy (64.5% for logistic regression, 64.9% for CART, i.e. essentially the same accuracy as the baseline). Weather or time features add little information to the models : for instance, training a Random Forest model using only weather data (see Table 2) results in an accuracy of 61%, which is again comparable to the baseline

Model	Imbalance Features	Weather Features	Time Features
Logistic Regression 1	Previous quarter imbalance (lag 1)	solar	weekday
Logistic Regression 2	Multiple lags apart from the past 3h	solar	weekday
LDA 1	Lag1	Cloud, Temp.	NA
LDA 2	Sign of Lag1	Cloud, Temp.	NA
CART 1	Lag 1	All	All
CART 2	Multiple lags (up to 5 hours, i.e. 20 qh)	All	All
CART 3	Multiple lags apart from the past 3h	All	All
Random Forest	NA	All	NA

Table 2: Trained Models with Corresponding Features

4.2 Predicting the power imbalance value

Predicting the power imbalance value is a regression problem, calling for models such as Linear regression, CART and Random Forest, along with time series models, exploiting the structure of the data. CART and Random Forest use cross-validation to selected the optimal parameters. Table 3 compares the performance of these models and the baseline (mean imbalance). We observe that Random Forest overfits the training data. The "Aggregated AR(1)s" models each of the 4 TSO imbalances as autoregressive models, then add them up to predict the network imbalance. However, we can see that this does not improve predictions compared to the baseline.

	Training Data		Testing Data	
Model and Parameters	RMSE	MAE	RMSE	MAE
Baseline	481	364	554	407
CART	266	201	327	235
Random Forest with mtry=4	319	238	575	418
AR(1)	262	199	304	228
ARIMA(0,1,5)	265	202	306	231
Aggregated AR(1)s	503	364	554	407

Table 3: Regression performance

5 Conclusion & Future Work

Our analysis indicates that weather data and time variables are generally poor predictors of power imbalance in the German electricity grid. This result is consistent across all models we have considered. We interpret this as evidence for market efficiency of the German power market with respect to these variables: All dependency is already fully exploited and the best we can do is a very simple baseline prediction. This does not surprise since this data is not only publicly available, but also free of charge. We reveal, however, that there is still some serial correlation left in the imbalance time series that can be used to build models that significantly outperform the baseline. By far the most relevant lag is lag 1 (i.e. the imbalance of the previous QH). One possible explanation for this is that the time window of 15 minutes is not sufficient to prepare an appropriate response to an imbalance in the system (e.g. startup time of generators, communication between TSO's and balancing energy providers, etc.).
In addition, there is the question of how power prices fit into the bigger picture. We have excluded this aspect from our analysis due to a lack of sufficient data. However, we believe that future work should consider it for mainly two reasons: First, power prices may be used to estimate the monetary value of a certain predictive model (for instance: *How much is it worth to know 15 minutes ahead of time that the next imbalance is going to be negative?*). Second, past prices may also be used as an additional explanatory variable in the models, since they represent material incentives driving the decisions of market participants.

References

[1] Garcia, M. P., Kirschen, D. S. (2006). Forecasting system imbalance volumes in competitive electricity markets. IEEE Transactions on Power Systems, 21(1), 240-248.

[2] https://www.bmwi.de/Redaktion/DE/Dossier/erneuerbare-energien.html

[3] Data basis for imbalance data (RZ_SALDO): Regelleistung.NET.
https://www.regelleistung.net/ext/data/

[4] Data basis for weather data: Deutscher Wetterdienst.
https://opendata.dwd.de/climate_environment/CDC/

Appendix

5.1 Variables in Final Dataset

IMBALANCE VARIABLES	Description
balance_50Hertz	imbalance for 50 Hertz zone
balance_50Hertz_pos	imbalance sign for 50 Hertz zone
balance_Amprion	imbalance for Amprion zone
balance_Amprion_pos	imbalance sign for Amprion zone
balance_network	imbalance in the entire grid
balance_network_pos	imbalance sign in the entire grid
balance_TenneT	imbalance in zone TenneT
balance_TenneT_pos	imbalance sign in zone TenneT
balance_TransnetBW	imbalance in zone TransnetBW
balance_TransnetBW_pos	imbalance sign in zone TransnetBW
imbalance_lag12_network	imbalance in the entire grid with lag 12
imbalance_lag13_network	imbalance in the entire grid with lag 13
imbalance_lag14_network	imbalance in the entire grid with lag 14
imbalance_lag15_network	imbalance in the entire grid with lag 15
imbalance_prev_qh_50Hertz	imbalance for 50 Hertz zone in prev. QH
imbalance_prev_qh_50Hertz_pos	imbalance sign for 50 Hertz zone in prev. QH
imbalance_prev_qh_Amprion	imbalance for Amprion zone in prev. QH
imbalance_prev_qh_Amprion_pos	imbalance sign for Amprion zone in prev. QH
imbalance_prev_qh_network	imbalance in the entire grid in prev. QH
imbalance_prev_qh_network_pos	imbalance sign in the entire grid in prev. QH
imbalance_prev_qh_TenneT	imbalance for TenneT zone in prev. QH
imbalance_prev_qh_TenneT_pos	imbalance sign for TenneT zone in prev. QH
imbalance_prev_qh_TransnetBW	imbalance for TransnetBW zone in prev. QH
imbalance_prev_qh_TransnetBW_pos	imbalance sign for TransnetBW zone in prev. QH

Table 4: Imbalance Variables in Final Data

TIME VARIABLES	Description
date	measurement datetime, format: yyyymmddhh
hour	Hour of the Day, 24 levels
month	Month of the Year, 12 levels
quarterhour	QH of the Day, 96 levels
weekday	Day of the Week, 7 levels
year	Year, 3 levels

Table 5: Time Variables in Final Data

CLOUD VARIABLES	Description
cloud_3379_V_N	total cloud cover in München
cloud_3379_V_N_I	index of total cloud cover in München
cloud_4271_V_N	total cloud cover in Rostock-Warnemünde
cloud_4271_V_N_I	index of total cloud cover in Rostock-Warnemünde
cloud_4928_V_N	total cloud cover in Stuttgart
cloud_4928_V_N_I	index of total cloud cover in Stuttgart
cloud_891_V_N	total cloud cover in Cuxhaven
cloud_891_V_N_I	index of total cloud cover in Cuxhaven

Table 6: Cloud Variables in Final Data

SOLAR VARIABLES	Description
solar_4271_ATMO_LBERG	hourly sum of longwave downward radiation in Rostock-Warnemünde
solar_4271_FD_LBERG	hourly sum of difuse solar rediation in Rostock-Warnemünde
solar_4271_FG_LBERG	hourly sum of solar incoming radiation in Rostock-Warnemünde
solar_4271_SD_LBERG	hourly sum of sunshine duration in Rostock-Warnemünde
solar_4271_WOZ	end of interval in local true solar time in Rostock-Warnemünde
solar_4271_ZENIT	solar zenith angle at mid of degree interval in Rostock-Warnemünde
solar_4928_ATMO_LBERG	hourly sum of longwave downward radiation in Stuttgart
solar_4928_FD_LBERG	hourly sum of diffuse solar radiation in Stuttgart
solar_4928_FG_LBERG	hourly sum of solar incoming radiation in Stuttgart
solar_4928_SD_LBERG	hourly sum of sunshine duration in Stuttgart
solar_4928_WOZ	end of interval in local true solar time in Stuttgart
solar_4928_ZENIT	solar zenith angle at mid of degree interval in Stuttgart

Table 7: Solar Variables in Final Data

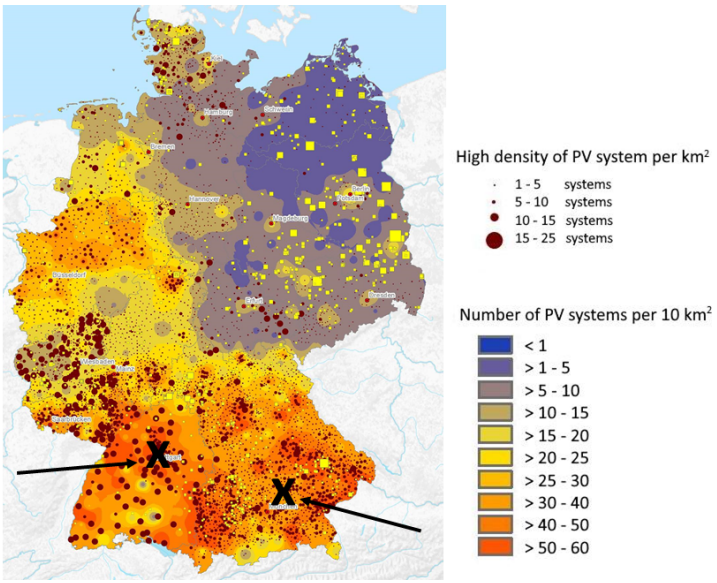
TEMPERATURE VARIABLES	Description
temp_3379_RF_TU	2m relative humidity in München
temp_3379_TT_TU	2m air temperature in München
temp_4271_RF_TU	2m relative humidity in Rostock-Warnemünde
temp_4271_TT_TU	2m air temperature in Rostock-Warnemünde
temp_4928_RF_TU	2m relative humidity in Stuttgart
temp_4928_TT_TU	2m air temperature in Stuttgart
temp_891_RF_TU	2m relative humidity in Cuxhaven
temp_891_TT_TU	2m air temperature in Cuxhaven

Table 8: Temperature Variables in Final Data

WIND VARIABLES	Description
wind_3379_D	wind direction in München
wind_3379_F	wind speed in München
wind_4271_D	wind direction in Rostock-Warnemünde
wind_4271_F	wind speed in Rostock-Warnemünde
wind_4928_D	wind direction in Stuttgart
wind_4928_F	wind speed in Stuttgart
wind_891_D	wind direction in Cuxhaven
wind_891_F	wind speed in Cuxhaven

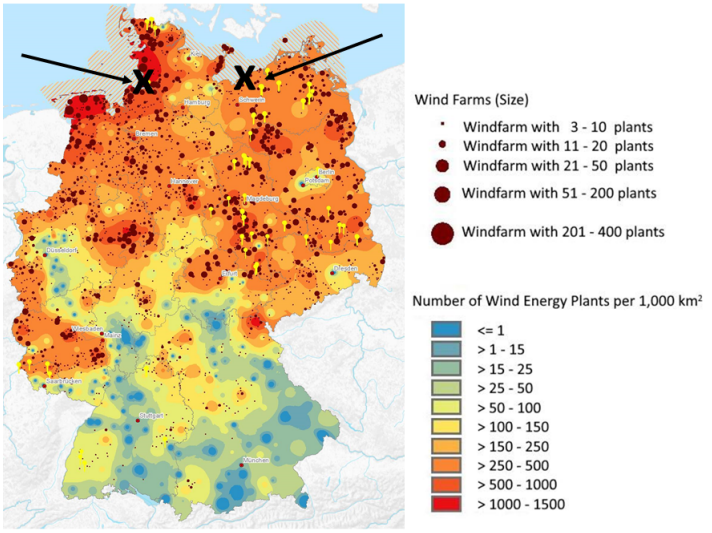
Table 9: Wind Variables in Final Data

5.2 Figures

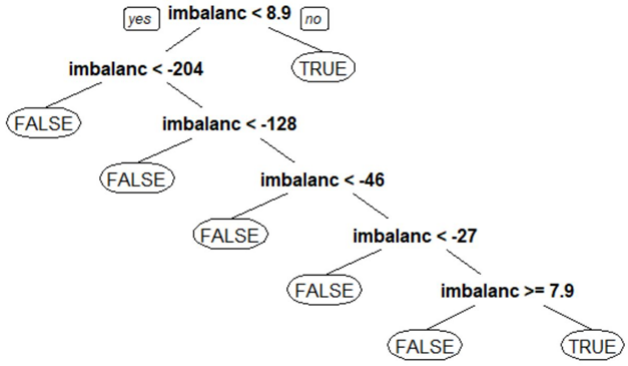


(a) Photovoltaics in Germany [Source: TU Dresden (2015)]

(b) German Electrical Network



(c) Wind capacities in Germany [Source: TU Dresden (2015)]



(d) CART tree for sign imbalance prediction

Figure 1: Selected weather stations and CART tree for imbalance sign prediction

5.3 Code

```
---
title: "IEOR 242 Project"
output: html_notebook
---


```{r}
library(car) # for VIF
library(caTools) # for sample.split
library(rpart)
library(rpart.plot)
library(caret)
library(randomForest)
library(gbm)
library(plyr)
library(dplyr)
library(ggplot2)
library(GGally)
library(MASS)
library(ROCit)
library(data.table)
library(PRROC)
library(forecast) #for time series
```

```{r}
#Load data
data <- read.csv('data.csv',header=TRUE,sep=';',na.strings=(""))

#How many missing values / NA's?
sum(is.na(data)==TRUE) # Total number of NA entries
sum(is.na(filter(data,
                 is.na(
                   data$solar_4271_ATMO_LBERG)==FALSE))) #NA's w/o solar 4271
```


```{r}
#Make continuous variables numeric:
data$balance_network <- sub(".", "", data$balance_network, fixed = TRUE)
data$balance_network <- as.numeric(sub(",", ".",
                                       data$balance_network, fixed = TRUE))

data$balance_Amprion <- sub(".", "", data$balance_Amprion, fixed = TRUE)
data$balance_Amprion <- as.numeric(sub(",", ".",
                                       data$balance_Amprion, fixed = TRUE))

data$balance_TenneT <- sub(".", "", data$balance_TenneT, fixed = TRUE)
data$balance_TenneT <- as.numeric(sub(",", ".",
                                       data$balance_TenneT, fixed = TRUE))
```
```

```
data$balance_50Hertz <- sub(".", "", data$balance_50Hertz, fixed = TRUE)
data$balance_50Hertz <- as.numeric(sub(",", ".",
                                       data$balance_50Hertz, fixed = TRUE))

data$balance_TransnetBW <- sub(".", "", data$balance_TransnetBW, fixed = TRUE)
data$balance_TransnetBW <- as.numeric(sub(",", ".",
                                       data$balance_TransnetBW, fixed = TRUE))

#Convert "date" from factor to datetime
data$date <- strptime(data$date,format="%Y-%m-%d %H:%M:%S",tz="Europe/Berlin")

#Derive new variables from existing ones:
data$balance_network_pos <- as.factor(data$balance_network>0)
data$balance_TransnetBW_pos <- as.factor(data$balance_TransnetBW>0)
data$balance_50Hertz_pos <- as.factor(data$balance_50Hertz>0)
data$balance_Amprion_pos <- as.factor(data$balance_Amprion>0)
data$balance_TenneT_pos <- as.factor(data$balance_TenneT>0)

data$hour <- as.factor(data$date$hour)
data$quarterhour <- as.factor((data$date$hour)*4+((data$date$min)/15+1))
data$weekday <- as.factor(data$date$wday) # 0=Sun, 1=Mon, 2=Tue, ...
data$month <- as.factor(data$date$mon) # 0=Jan, 1=Feb, 2=Mar, ...
data$year <- as.factor(data$date$year+1900)

data$imbalance_prev_qh_network <- shift(data$balance_network,n=1,fill=NA)
data$imbalance_prev_qh_Amprion <- shift(data$balance_Amprion,n=1,fill=NA)
data$imbalance_prev_qh_TenneT <- shift(data$balance_TenneT,n=1,fill=NA)
data$imbalance_prev_qh_50Hertz <- shift(data$balance_50Hertz,n=1,fill=NA)
data$imbalance_prev_qh_TransnetBW <- shift(data$balance_TransnetBW, n=1, fill=NA)

data$imbalance_prev_qh_network_pos <- shift(data$balance_network_pos,n=1,fill=NA)
data$imbalance_prev_qh_Amprion_pos <- shift(data$balance_Amprion_pos,n=1,fill=NA)
data$imbalance_prev_qh_TenneT_pos <- shift(data$balance_TenneT_pos,n=1,fill=NA)
data$imbalance_prev_qh_50Hertz_pos <- shift(data$balance_50Hertz_pos,n=1,fill=NA)
data$imbalance_prev_qh_TransnetBW_pos <- shift(data$balance_TransnetBW_pos, n=1,
                                               fill=NA)

data$imbalance_lag12_network <- shift(data$balance_network,n=12,fill=NA)
data$imbalance_lag13_network <- shift(data$balance_network,n=13,fill=NA)
data$imbalance_lag14_network <- shift(data$balance_network,n=14,fill=NA)
data$imbalance_lag15_network <- shift(data$balance_network,n=15,fill=NA)

data$cloud_3379_V_N[data$cloud_3379_V_N==-1] <- NA
data$cloud_4271_V_N[data$cloud_4271_V_N==-1] <- NA
data$cloud_4928_V_N[data$cloud_4928_V_N==-1] <- NA
data$cloud_891_V_N[data$cloud_891_V_N==-1] <- NA
data$cloud_3379_V_N <- as.factor(data$cloud_3379_V_N)
data$cloud_4271_V_N <- as.factor(data$cloud_4271_V_N)
data$cloud_4928_V_N <- as.factor(data$cloud_4928_V_N)
data$cloud_891_V_N <- as.factor(data$cloud_891_V_N)
```

```
data$timeperiod<-1:nrow(data)

data$date <- as.POSIXct(data$date)
names(data)[names(data)=="solar_42928_ATMO_LBERG"] <- "solar_4928_ATMO_LBERG"
```

```{r}
#Split the data
train <- data[1:(0.75*nrow(data)),]
test <- data[(0.75*nrow(data)):nrow(data),]
```

MODELS BELOW:

```{r}
# Baseline = Always predict pos imbalance

t <- table(train$balance_network_pos)
t

print(paste("Baseline of always predicting positive imbalance has an accuracy of",
      round(t[2]/nrow(train),2),"on the training set."))
```

CART:

```{r}
# CART Classification

mod_cart <- train(balance_network_pos~
                    imbalance_lag12_network+imbalance_lag13_network
                    +cloud_3379_V_N+cloud_4271_V_N+cloud_4928_V_N
                    +cloud_891_V_N+temp_3379_TT_TU+temp_3379_RF_TU
                    +temp_4271_TT_TU+temp_4271_RF_TU+temp_891_TT_TU
                    +temp_891_RF_TU+temp_4928_TT_TU+temp_4928_RF_TU
                    +wind_3379_D+wind_3379_F+wind_4271_D+wind_4271_F
                    +wind_4928_D+wind_4928_F+wind_891_D+wind_891_F
                    +solar_4928_ATMO_LBERG+solar_4928_FD_LBERG
                    +solar_4928_FG_LBERG+solar_4928_SD_LBERG+quarterhour
                    +weekday+year+month
                    ,data=train,na.action=na.rpart,
                    method = "rpart",minbucket=10,
                    tuneGrid = data.frame(cp = seq(0.00001, .0001,
                                                   by=.00001)),
                    trControl = trainControl(method="cv", number=5))

mod_cart_final <- mod_cart$finalModel
mod_cart$results
```

```
# CART Regression

mod_cart <- train(balance_network ~
                    +imbalance_lag1_network
                  +imbalance_lag2_network
                  +imbalance_lag3_network
                  +imbalance_lag4_network
                  +imbalance_lag5_network
                  +imbalance_lag6_network
                  +imbalance_lag7_network
                  +imbalance_lag8_network
                  +imbalance_lag9_network
                  +imbalance_lag10_network
                  #+imbalance_lag11_network
                  #+imbalance_lag12_network
                  #+imbalance_lag13_network
                  #+imbalance_lag14_network
                  #+imbalance_lag15_network
                  #+imbalance_lag16_network
                  #+imbalance_lag17_network
                  #+imbalance_lag18_network
                  #+imbalance_lag19_network
                  #+imbalance_lag20_network
                  +cloud_3379_V_N#+cloud_4271_V_N+cloud_4928_V_N
                  +cloud_891_V_N+temp_3379_TT_TU+temp_3379_RF_TU
                  +temp_4271_TT_TU#+temp_4271_RF_TU+temp_891_TT_TU
                  +temp_891_RF_TU+temp_4928_TT_TU+temp_4928_RF_TU
                  +wind_3379_D#+wind_3379_F+wind_4271_D+wind_4271_F
                  +wind_4928_D+wind_4928_F+wind_891_D+wind_891_F
                  +solar_4928_ATMO_LBERG#+solar_4928_FD_LBERG
                  +solar_4928_FG_LBERG+solar_4928_SD_LBERG
                  +quarterhour
                  +weekday+year+month
                  ,data=train,na.action=na.rpart,
                  method = "rpart",minbucket=10,
                  tuneGrid = data.frame(cp = seq(0.001, .01,
                                                 by=.0001)),
                  trControl = trainControl(method="cv", number=5))

mod_cart_final <- mod_cart$finalModel
mod_cart$results

# CART statistics

``{r}
cart_train <- predict(mod_cart_final)
cart_pred<- predict(mod_cart_final,newdata=test)

#cart_train <- cart_train[2:105191]
sqrt(sum((cart_train-train$balance_network)^2)/nrow(train))

sum(abs(cart_train-train$balance_network))/nrow(train)
```

```
#cart_train <- cart_train[2:105191]
sqrt(sum((cart_pred-test$balance_network)^2)/nrow(test))
sum(abs(cart_pred-test$balance_network))/nrow(test)
```

```{r}
#t <- table(train$balance_network_pos)
#t
length(cart_pred)
length(cart_train)
```

```{r}
table(train$balance_network_pos[2:105192],cart_train)
table(test$balance_network_pos,cart_pred)
```

```
```{r}
table(test$balance_network_pos)
```

```{r}
prp(mod_cart_final)
```

```
LDA

```{r}
#lda1 - imbalance_prev_qh_network accuracy = 0.7789341
mod_lda1<-lda(balance_network_pos~
                    imbalance_prev_qh_network

                    +cloud_3379_V_N+cloud_4271_V_N+cloud_4928_V_N
                    +cloud_891_V_N+temp_3379_TT_TU+temp_3379_RF_TU

                    +temp_4271_TT_TU+temp_4271_RF_TU+temp_891_TT_TU
                    +temp_891_RF_TU+temp_4928_TT_TU+temp_4928_RF_TU

                    +wind_3379_D+wind_3379_F+wind_4271_D+wind_4271_F
                    +wind_4928_D+wind_4928_F+wind_891_D+wind_891_F

                    +solar_4928_ATMO_LBERG+solar_4928_FD_LBERG
                    +solar_4928_FG_LBERG+solar_4928_SD_LBERG
                    +weekday+month, data = train)
```

```{r}
pred_lda1<-predict(mod_lda1, newdata = test, type = "class")
mod_lda1_mat1<-table(pred_lda1$class,test$balance_network_pos)
mod_lda1_mat1
mod_lda1_accuracy <- sum(diag(mod_lda1_mat1)) / sum(mod_lda1_mat1)
mod_lda1_accuracy
```

```

```{r}
#lda2 - imbalance_prev_qh_network_pos accuracy = 0.7849592
mod_lda2<-lda(balance_network_pos~
                    imbalance_prev_qh_network_pos

                    +cloud_3379_V_N+cloud_4271_V_N+cloud_4928_V_N
                    +cloud_891_V_N+temp_3379_TT_TU+temp_3379_RF_TU

                    +temp_4271_TT_TU+temp_4271_RF_TU+temp_891_TT_TU
                    +temp_891_RF_TU+temp_4928_TT_TU+temp_4928_RF_TU

                    +wind_3379_D+wind_3379_F+wind_4271_D+wind_4271_F
                    +wind_4928_D+wind_4928_F+wind_891_D+wind_891_F

                    +solar_4928_ATMO_LBERG+solar_4928_FD_LBERG
                    +solar_4928_FG_LBERG+solar_4928_SD_LBERG
                    +weekday+month, data = train)
```

```{r}
pred_lda2<-predict(mod_lda2, newdata = test, type = "class")
mod_lda2_mat2<-table(pred_lda2$class, test$balance_network_pos)
mod_lda2_mat2
mod_lda2_accuracy2 <- sum(diag(mod_lda2_mat2)) / sum(mod_lda2_mat2)
mod_lda2_accuracy2
```

```{r}
#lda3 - lag12+lag13 accuracy = 0.547516
mod_lda3<-lda(balance_network_pos~
                    imbalance_lag12_network + imbalance_lag13_network

                    +cloud_3379_V_N+cloud_4271_V_N+cloud_4928_V_N
                    +cloud_891_V_N+temp_3379_TT_TU+temp_3379_RF_TU

                    +temp_4271_TT_TU+temp_4271_RF_TU+temp_891_TT_TU
                    +temp_891_RF_TU+temp_4928_TT_TU+temp_4928_RF_TU

                    +wind_3379_D+wind_3379_F+wind_4271_D+wind_4271_F
                    +wind_4928_D+wind_4928_F+wind_891_D+wind_891_F

                    +solar_4928_ATMO_LBERG+solar_4928_FD_LBERG
                    +solar_4928_FG_LBERG+solar_4928_SD_LBERG
                    +weekday+month, data = train)
```

```{r}
pred_lda3<-predict(mod_lda3, newdata = test, type = "class")
mod_lda3_mat3<-table(pred_lda3$class, test$balance_network_pos)
mod_lda3_mat3
```
```

```
mod_lda3_accuracy3 <- sum(diag(mod_lda3_mat3)) / sum(mod_lda3_mat3)
mod_lda3_accuracy3
```

LOGISTIC REGRESSION

```{r}
# Logistic regression model #1
# The accuracy on training data is 0.83
# The accuracy on test data is 0.80

 mod_log1 <- glm(balance_network_pos~
           imbalance_prev_qh_network+cloud_3379_V_N+cloud_4271_V_N
           +cloud_4928_V_N+cloud_891_V_N+temp_3379_TT_TU
           +temp_3379_RF_TU+temp_4271_TT_TU+temp_4271_RF_TU
           +temp_891_TT_TU+temp_891_RF_TU+temp_4928_TT_TU
           +temp_4928_RF_TU+wind_3379_D+wind_3379_F+wind_4271_D
           +wind_4271_F+wind_4928_D+wind_4928_F+wind_891_D
           +wind_891_F+solar_4928_ATMO_LBERG+solar_4928_FD_LBERG
           +solar_4928_FG_LBERG+solar_4928_SD_LBERG+solar_4928_ZENIT
           +quarterhour+weekday+month, data=train,family="binomial")

summary(mod_log1)
vif(mod_log1)

predTrain1 = predict(mod_log1, newdata=train, type="response")
predTest1 = predict(mod_log1, newdata=test, type="response")
table1 <- table(train$balance_network_pos, predTrain1 > 0.5)
table2 <- table(test$balance_network_pos, predTest1 > 0.5)
print(paste("The accuracy on training data is",sum(diag(table1))/sum(table1)))
print(paste("The accuracy on test data is",sum(diag(table2))/sum(table2)))
```

```
```{r}
# Logistic regression model #2 (with lag12 and lag13)
# The accuracy on training data is 0.65
# The accuracy on test data is 0.56

 mod_log2 <- glm(balance_network_pos~
           imbalance_lag12_network +imbalance_lag13_network
           +cloud_3379_V_N+cloud_4271_V_N
           +cloud_4928_V_N+cloud_891_V_N+temp_3379_TT_TU
           +temp_3379_RF_TU+temp_4271_TT_TU+temp_4271_RF_TU
           +temp_891_TT_TU+temp_891_RF_TU+temp_4928_TT_TU
           +temp_4928_RF_TU+wind_3379_D+wind_3379_F+wind_4271_D
           +wind_4271_F+wind_4928_D+wind_4928_F+wind_891_D
           +wind_891_F+solar_4928_ATMO_LBERG+solar_4928_FD_LBERG
           +solar_4928_FG_LBERG+solar_4928_SD_LBERG+solar_4928_ZENIT
           +quarterhour+weekday+month, data=train,family="binomial")
```
```

```
summary(mod_log2)

predTrain2 = predict(mod_log2, newdata=train, type="response")
predTest2 = predict(mod_log2, newdata=test, type="response")
table3 <- table(train$balance_network_pos, predTrain2 > 0.5)
table4 <- table(test$balance_network_pos, predTest2 > 0.5)
print(paste("The accuracy on training data is",sum(diag(table3))/sum(table3)))
print(paste("The accuracy on test data is",sum(diag(table4))/sum(table4)))
```

TIME SERIES

```
```{r}
#Baseline : mean imbalance
mean_network<-mean(train$balance_network)
#RMSE on the train set
sd(train$balance_network - mean_network)    #481.3263
#MAE on the train set
sum(abs(train$balance_network - mean_network))/nrow(train)  #363.9779
#corresponding RMSE on the test set (R2 and OSR2 are 0)
sqrt(sum((test$balance_network - mean_network)^2)/nrow(test))
#554.2683
#corresponding MAE on the test set
sum(abs(test$balance_network - mean_network))/nrow(test)     # 406.5808
#mean((train$balance_network-train$imbalance_prev_qh_network)[-1])#0.001753829

ts_network<-ts(data$balance_network,start=1,frequency=1)
ts_network_train<-ts(train$balance_network,start=1,frequency=1)
ts_network_test<-ts(test$balance_network,start=105193,frequency=1)

#extract a linear trend :

#run linear regression with timeperiod as only independent variable
model_lin_trend_network<-lm(balance_network ~ timeperiod, data = train)
summary(model_lin_trend_network)
#As expected, very small R2 (0.009123),
#but the variable is marked as statistically significant

#create a new times series object without the trend :
trend<-predict(model_lin_trend_network,newdata=data)
data$balance_network_wo_trend<-data$balance_network-trend
train <- data[1:(0.75*nrow(data)),]
test <- data[(0.75*nrow(data)):nrow(data),]
ts_network_wo_trend<-ts(data$balance_network_wo_trend,start=1,frequency=1)
ts_network_wo_trendtrain<-ts(train$balance_network,start=1,frequency=1)

#Autocorrelation and partial autocorrelation
acf(ts_network_train)
pacf(ts_network_train)
#NB : acf(ts_network_wo_trendtrain), pacf(ts_network_wo_trendtrain) are the same
#Show that balance(t) looks like an AR(1)   with coefficient phi~85%;
#indeed the first portion of the acf shows exponential decrease,
```
```

```
#and the partial acf shows only one peak for lag 1;
#for higher lags, the pacf is not significant(less than 0.1 in absolute value)
#-> reasonable to model the balance by an AR(1)

model_network_ar1<-arima(ts_network_train, order = c(1, 0, 0)) # AR(1)
summary(model_network_ar1)
#         ar1   intercept
#   0.838262 125.440825
#Note : same results as linear regression with lag1 (it is exactly the same model),
#but this also gives the variance of the white noise eps_t (square of RMSE)
sd(data$balance_network)
accuracy(model_network_ar1) # from library(forecast)
#RMSE on the training set : 262.449
#MAE on the training set : 198.5911

#MOdel the timeseries as an ARIMA
#using auto.arima to fit an arima
auto_arima_network <- auto.arima(ts_network_train)
summary(auto_arima_network)#-> ARIMA(0,1,5)
auto_arima_network_coeff=auto_arima_network$coef
auto_arima_network_coeff
auto_arima_network_sigma2=auto_arima_network$sigma2
accuracy(auto_arima_network)
#RMSE on the training set : 265.0682

#compare them to predict next 5 values :
forecast(auto_arima_network,5)$mean
forecast(model_network_ar1,5)$mean
test$balance_network[1:5]

arima_test <- Arima(ts_network_test, model=auto_arima_network)
ar1_test <- Arima(ts_network_test, model=model_network_ar1)
accuracy(ar1_test)#RMSE 303.5319
accuracy(arima_test) #RMSE 305.5285
```