12.05.2020

Basel Air Pollution Prediction

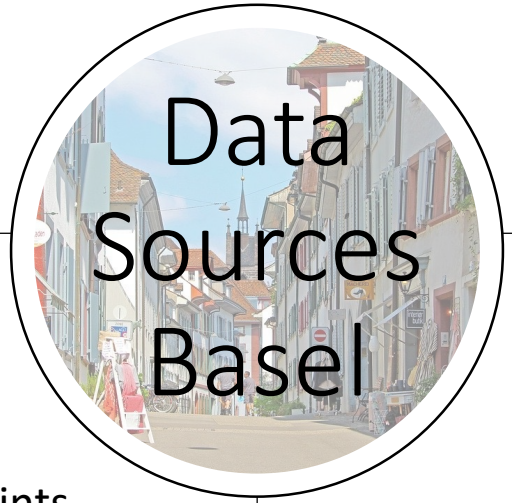
Final Project

„Data Science at Home“

by

Machine Learning Graz

Bianca Köck & Bernhard Lutzer



Traffic

[Swiss Open Data]
12.08.2014-03.05.2020
408 MB
only used one measurement point,
close to the weather measurement points

Weather

[Meteoblue.com]
1.1.2014-03.05.2020
4 MB

The reason we took Basel – the only free data available

Air Quality

[Local government]	[Swiss government]
12.08.2014-03.05.2020	01.01.2014-03.05.2020
2,7 MB	3,6 MB
only took one measurement point – St. Johann	only took one measurement point, close to the weather measurement points

some measurement points lack in both air quality data frames

= 45 000 Samples in the joined data frames by time stamp (join method inner)

Objectives

- Exploratory Data Analysis
- Clean & Merge Data
- Model building on air pollution based on weather and traffic data

Exploratory Data Analysis & Prepare for Merging

Traffic Data

SiteCode	Site Name	Direction Name	Lane Code	Lane Name	Date	Time From	TimeTo	Values Approved	Values Edited	Traffic Type	Total
MR	PW	PW+	Lief	Lief+	Lief+ Aufl.	LW	LW+	Sattelzug	Bus	andere	

df_verkehr.Date+" "+df_verkehr.TimeTo

df_verkehr['timestamp']= pd.to_datetime(df_verkehr.Date+" „ +df_verkehr.TimeTo, dayfirst=True)

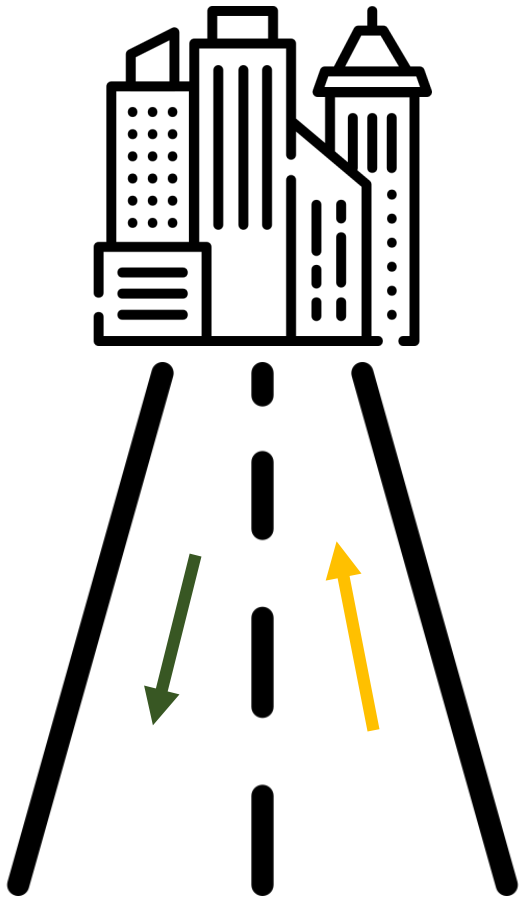
Convert to Timestamp

0	2014-08-21 01:00:00
1	2014-08-21 02:00:00
2	2014-08-21 03:00:00
3	2014-08-21 04:00:00
4	2014-08-21 05:00:00
5	2014-08-21 06:00:00
6	2014-08-21 07:00:00
7	2014-08-21 08:00:00

Site Selection close to the weather station

df_verkehr_stjohann = df_verkehr[df_verkehr.SiteName == "660 Flughafenstrasse, Grenze CH-F"]

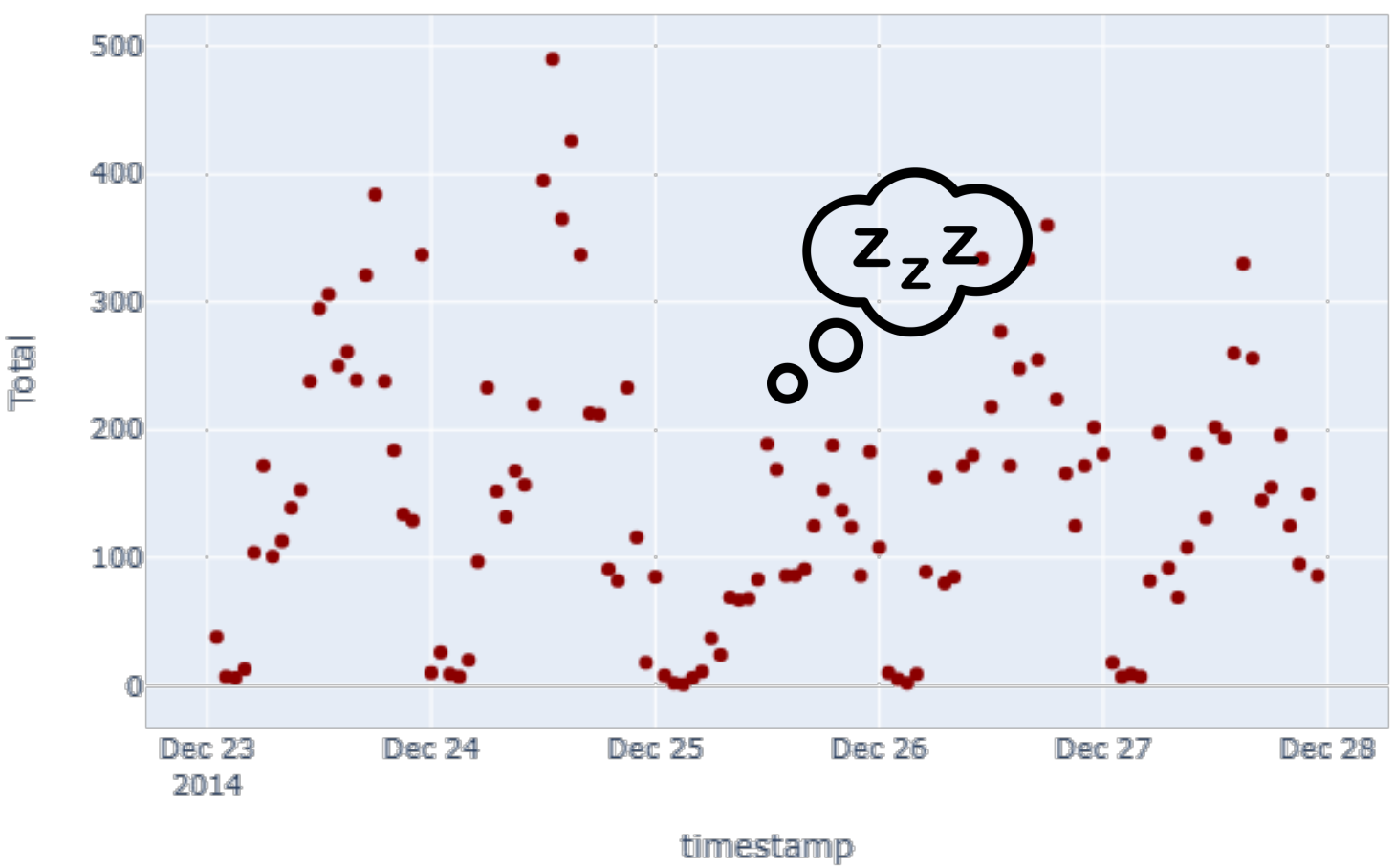
Traffic at Lanes



	TimeTo	Lane Name	Total
0	00:00	Spur 1	211.587021
1	00:00	Spur 2	60.489184
2	01:00	Spur 1	54.068338
3	01:00	Spur 2	12.903147
4	02:00	Spur 1	7.809735

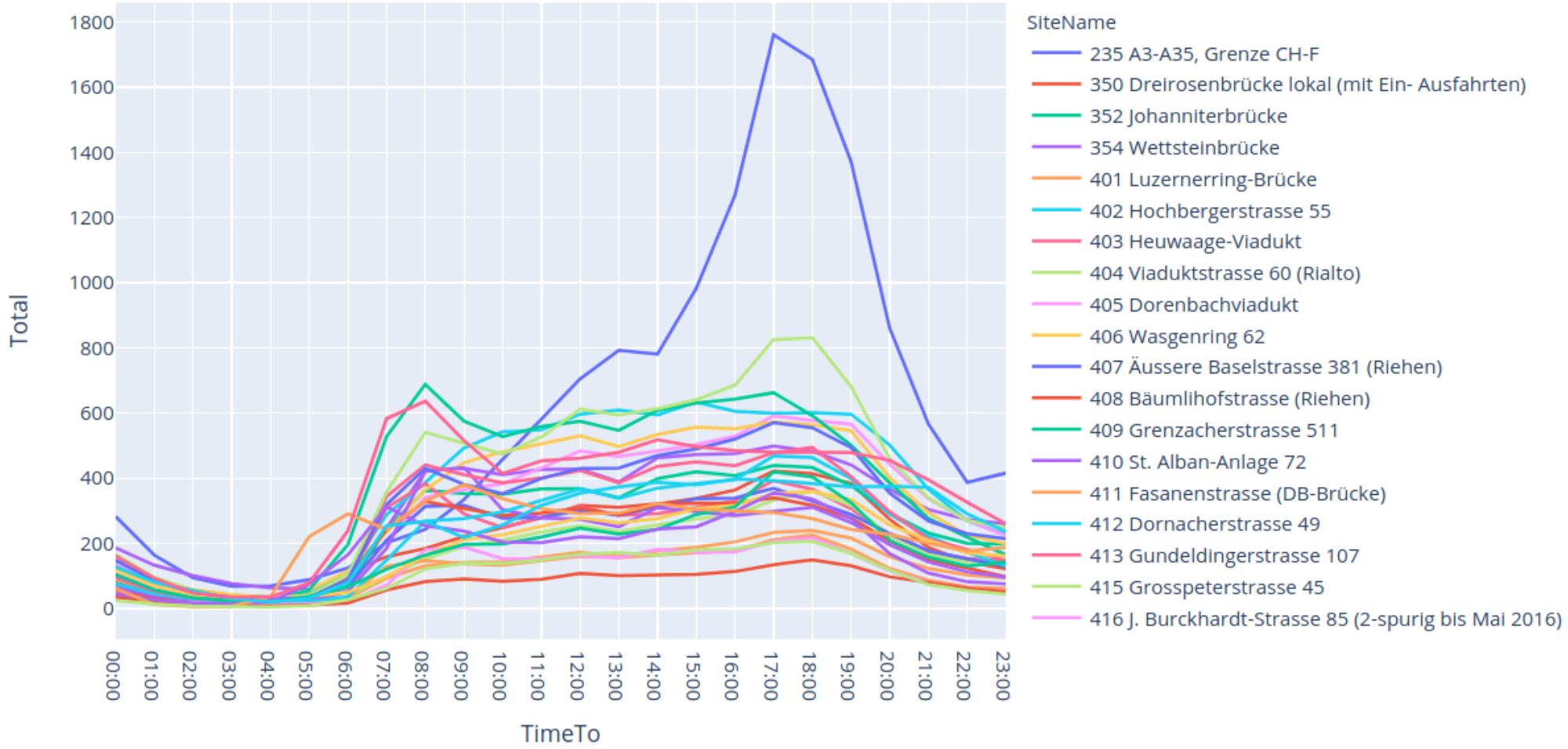

```
df_grouped_mean = df_verkehr_stjohann.groupby(["TimeTo", "LaneName"]).Total.mean().reset_index()
px.line(df_grouped_mean, x=df_grouped_mean.TimeTo, y="Total", color="LaneName", color_discrete_sequence=["darkgreen", "yellow"])
```

Traffic at Christmas 2014



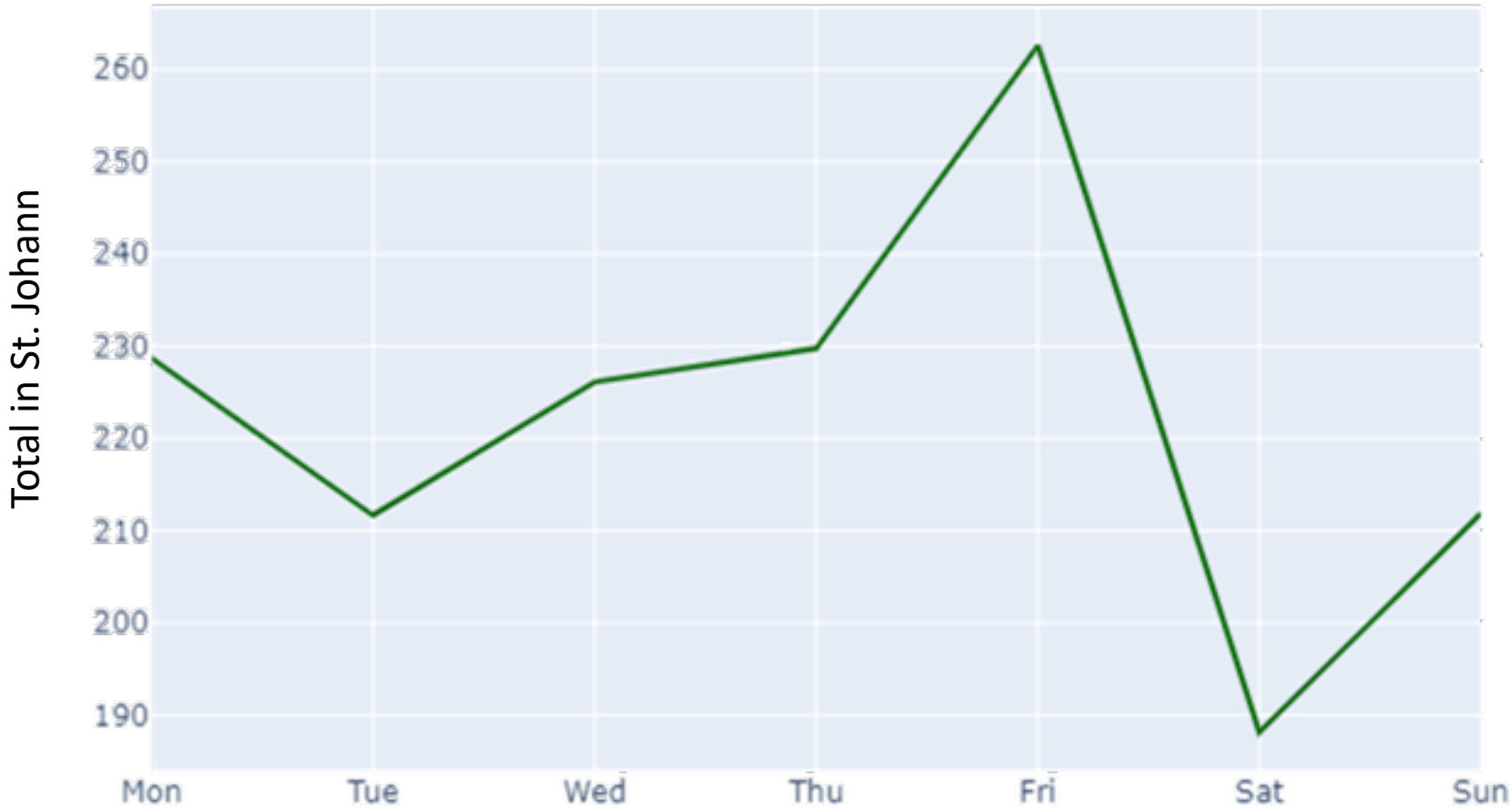
```
df_christmas = df_verkehr_stjohann_index[df_verkehr_stjohann_index.LaneName == "Spur 1"].loc['2014-12-23 01:00:00' : '2014-12-27 23:00:00'].Total.reset_index()

px.scatter(df_christmas.sort_index(), x="timestamp", y="Total")
```

Traffic depending on Locations within the city




```
df_v1 = df_verkehr[df_verkehr.LaneName == "Spur 2"].groupby(["SiteName", "TimeTo"]).Total.mean().reset_index()
px.line(df_v1, x="TimeTo", y="Total", color="SiteName")
```

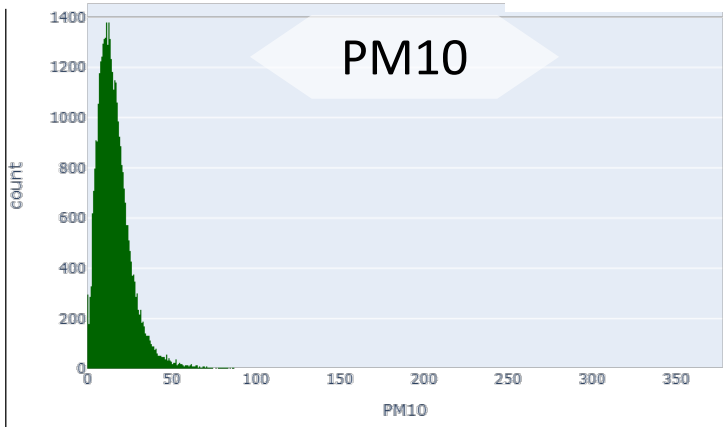
Change of the mean traffic during the weekdays

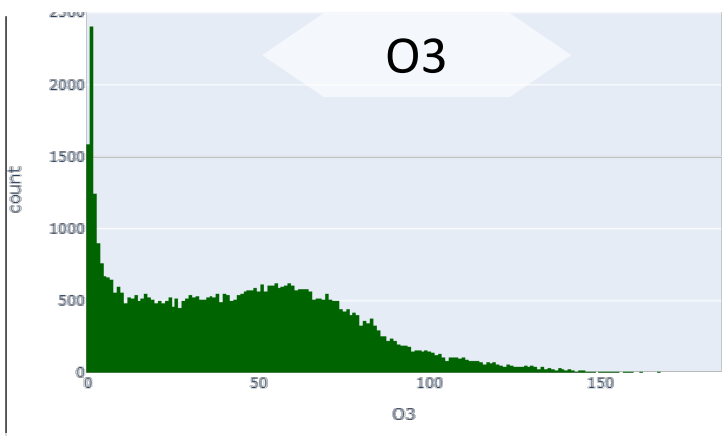


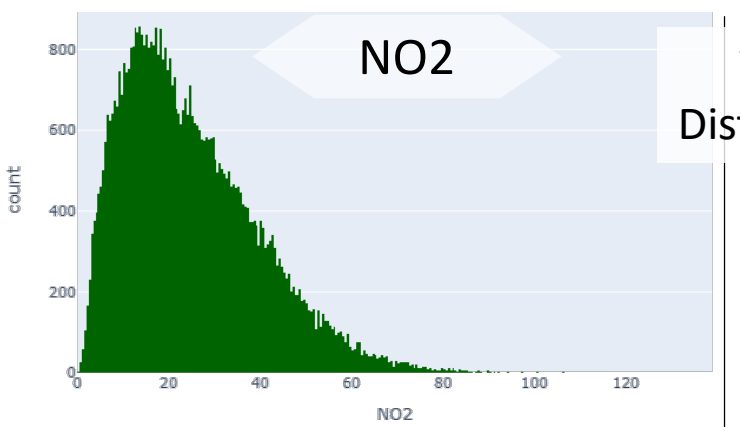

```
df_verkehr_stjohann_index = df_verkehr_stjohann.set_index("timestamp")
df_christmas = df_verkehr_stjohann_index[df_verkehr_stjohann_index.LaneName ==
"Spur 1"].loc['2014-12-23 01:00:00' : '2014-12-27 23:00:00'].Total.reset_index()
px.scatter(df_christmas.sort_index(), x="timestamp", y="Total")
```

Air quality in Basel St. Johann

px.histogram(df_basel_feld_luft, x="PM10", color_discrete_sequence=["darkgreen"])

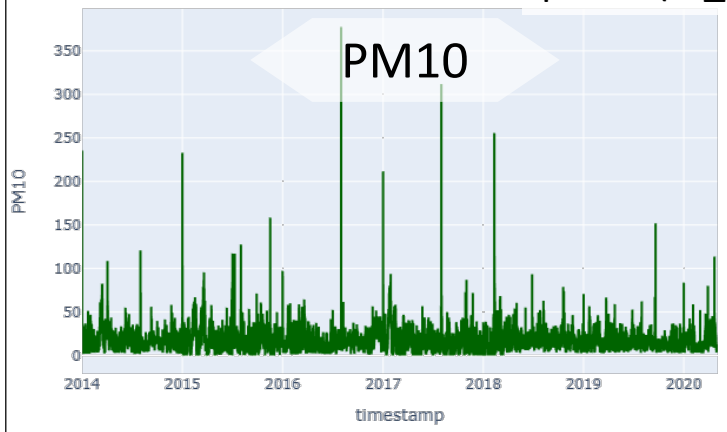


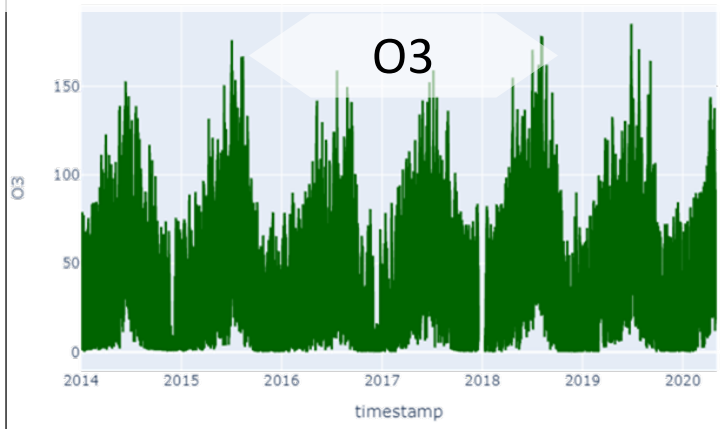


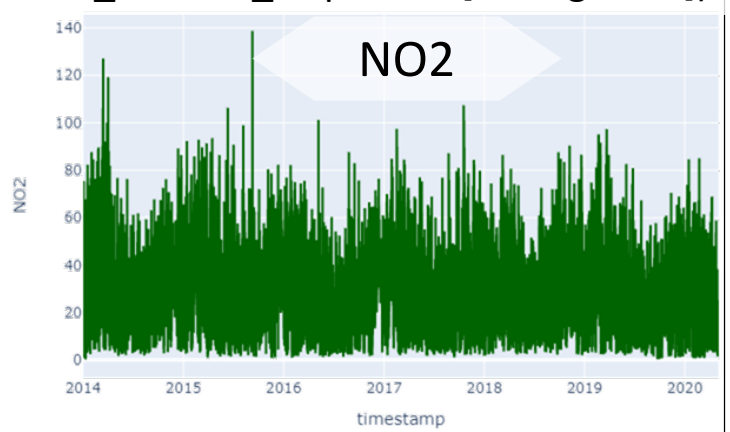


Value Distribution

px.line(df_basel_feld_luft, x="timestamp", y="PM10", color_discrete_sequence=["darkgreen"])



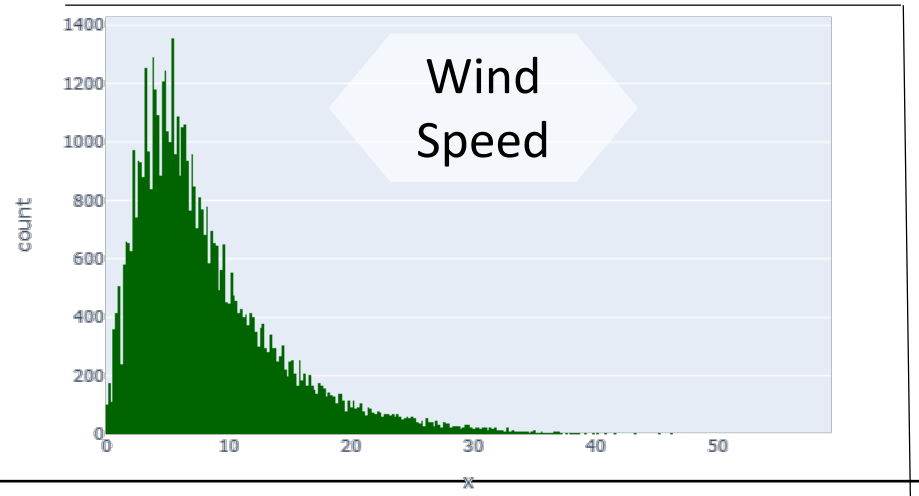




Over Time

	timestamp	PM10	PM2.5	O3	NO2
0	2014-01-01 01:00:00	26.719	NaN	10.247	33.710
1	2014-01-01 02:00:00	67.560	NaN	4.997	34.824

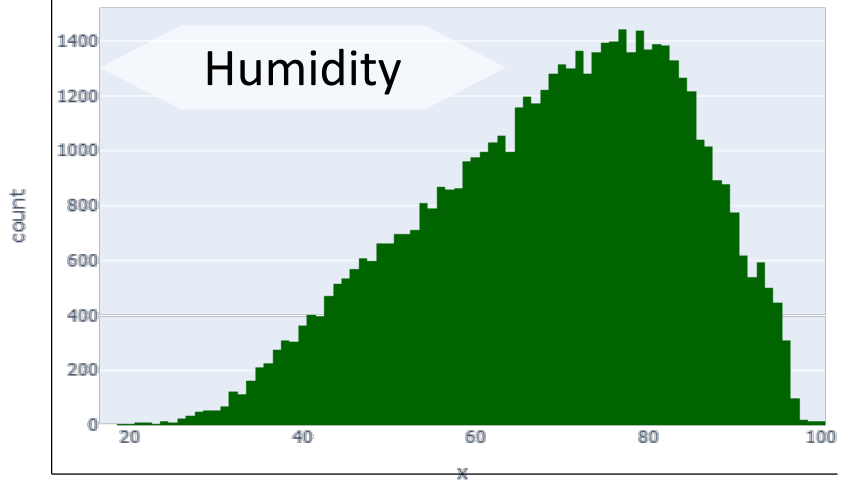
Weather

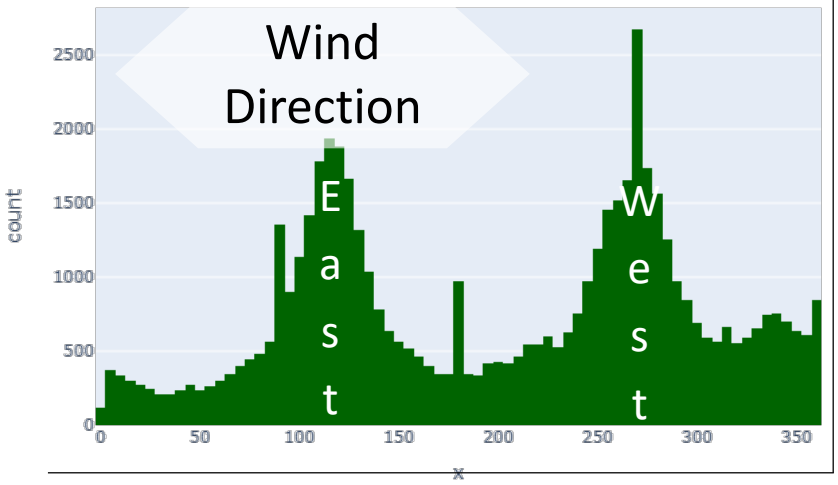


	Time stamp	temp	humidity	pressure	precipitation	Cloud cover	sunshine duration	Shortwave radiation	wind speed	wind direction
0	20140101T0000	1.690529	45.0	1017.8	0.0	17.0	0.0	0.0	5.145039	111.25050
1	20140101T0100	2.080528	41.0	1018.1	0.0	100.0	0.0	0.0	5.491976	104.03624

```
df_basel_wetter['timestamp']=
pd.to_datetime
(df_basel_wetter['timestamp'])
```

	Time stamp
0	2014-01-01 00:00:00
1	2014-01-01 01:00:00

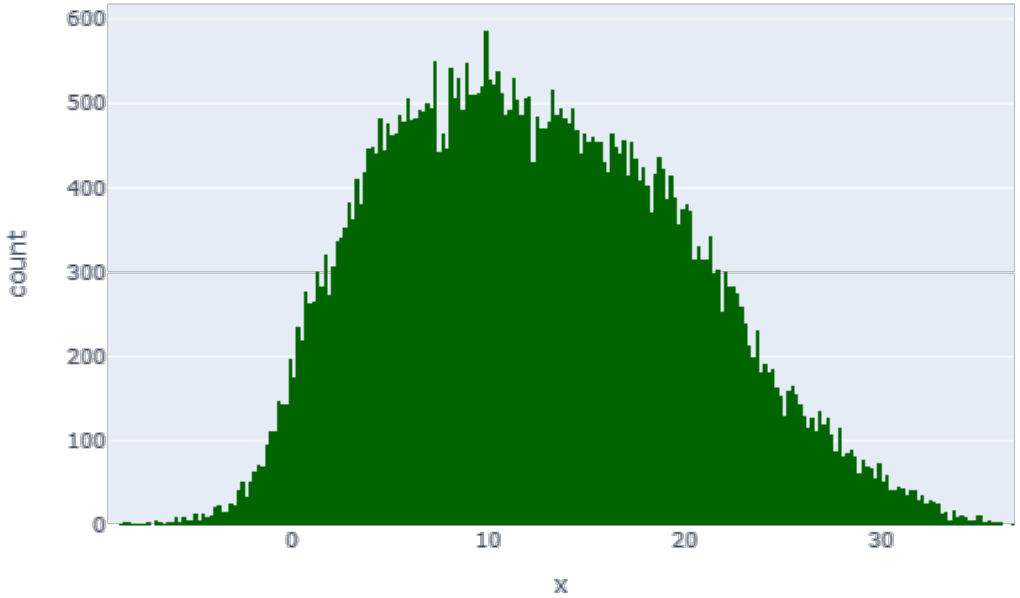




Temperature Insights 🌡️

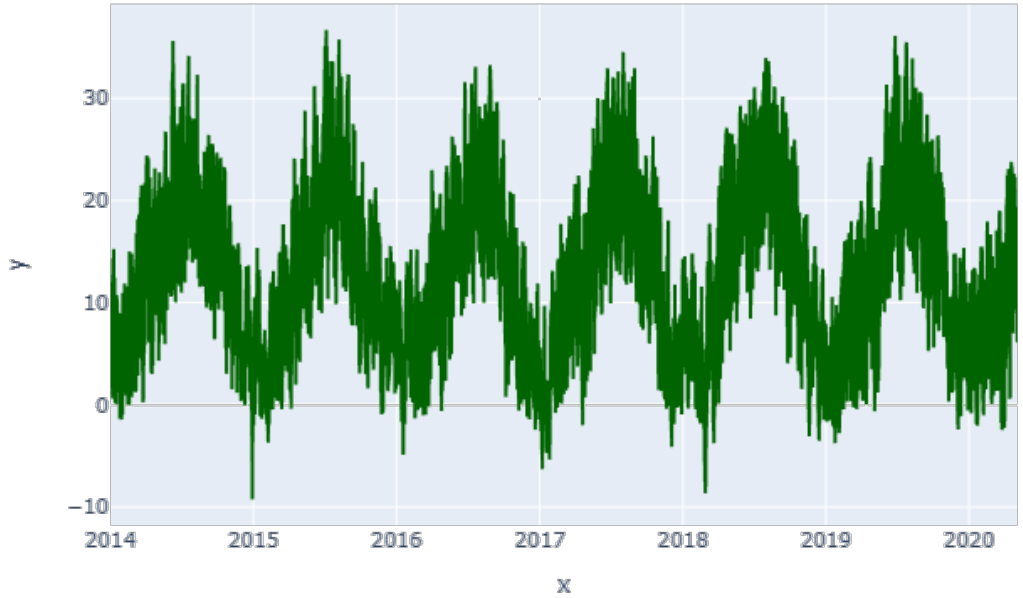
🥵 On 05.07.2015, 17:00 it was **36,7 °C**. Pretty hot.
df_basel_wetter.timestamp[df_basel_wetter.temp == df_basel_wetter.temp.max()]

Degree Distribution



px.histogram(x=df_basel_wetter.temp,
color_discrete_sequence=["darkgreen"])

Over Time



px.line(x=df_basel_wetter.timestamp, y=df_basel_wetter.temp,
color_discrete_sequence=["darkgreen"])

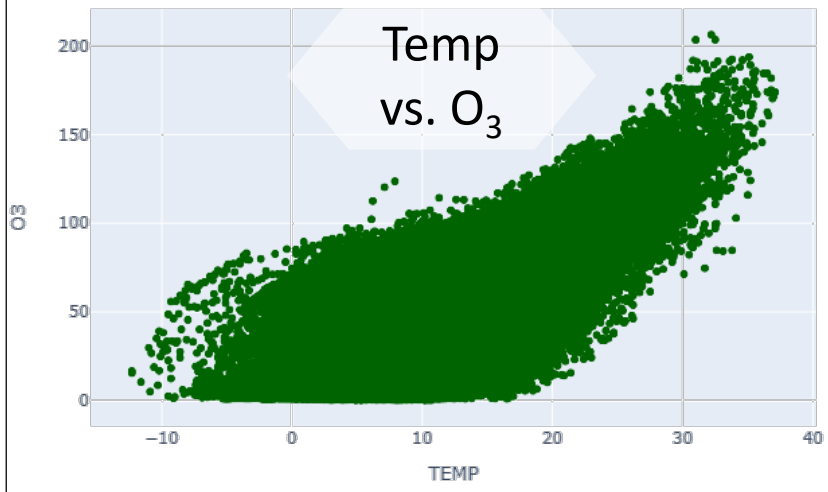
Air Quality Data Swiss Government

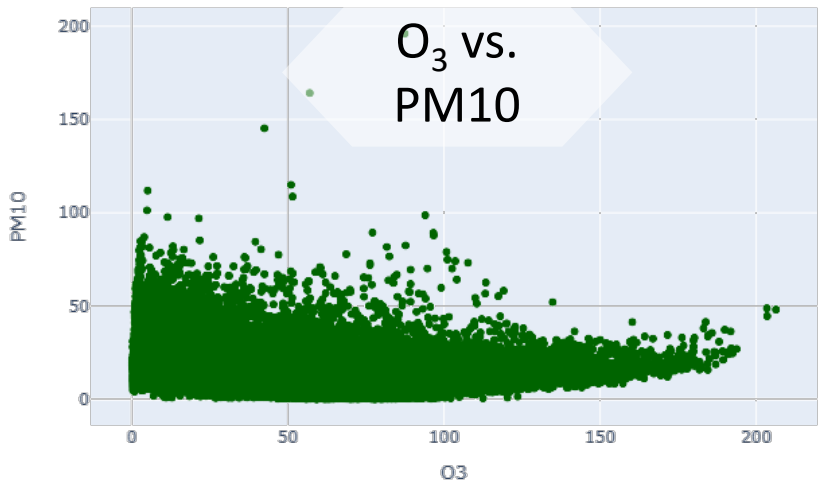
	Datum/ Zeit	O3 [ug/m3]	NO2 [ug/m3]	SO2 [ug/m3]	PM10 [ug/m3]	PM2.5 [ug/m3]	EC [ug/m3]	CPC [1/cm3]	NOX [ug/m3 eq. NO2]	TEMP [C]	PREC [mm]	RAD [W/m2]
0	01.01.2014 01:00	7.1	36.7	5.3	35.6	NaN	1.4	17257.8	43.2	0.5	0.0	0.0
1	01.01.2014 02:00	4.7	35.5	6.3	57.3	NaN	1.5	20101.9	47.1	0.3	0.0	0.0

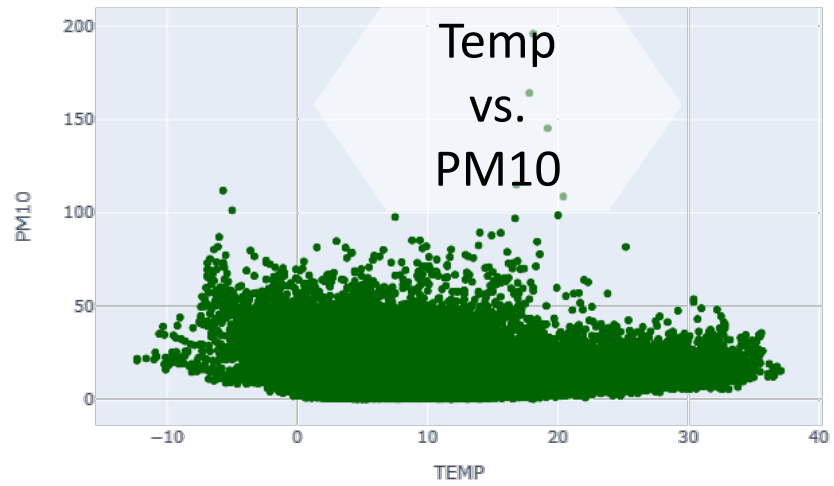
Changing Column Titles & Timestamp

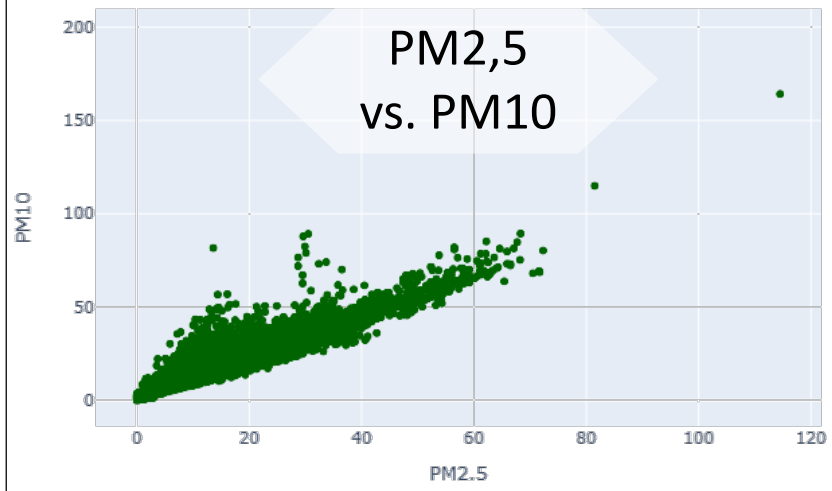
```
df_basel_bin.columns = ['timestamp', 'O3', 'NO2', 'SO2', 'PM10', 'PM2.5', 'EC', 'CPC', 'NOX', 'TEMP', 'PREC', 'RAD']
df_basel_bin['timestamp']= pd.to_datetime(df_basel_bin['timestamp'], dayfirst=True)
```

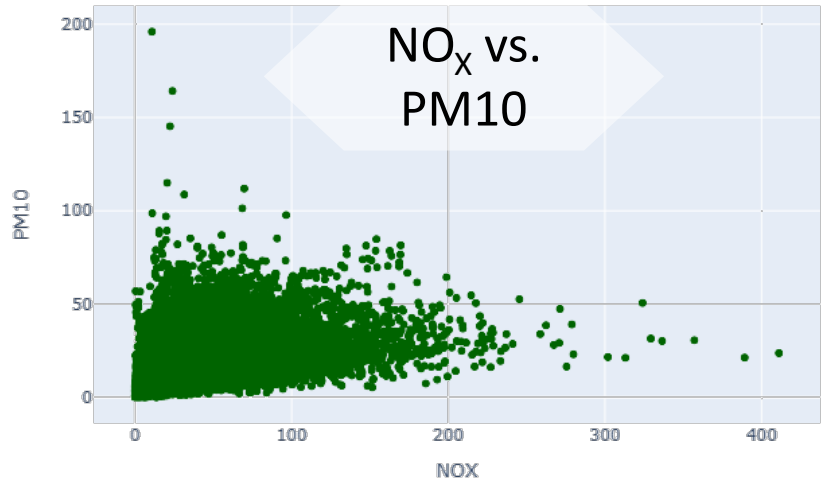
	timestamp	O3	NO2	SO2	PM10	PM2.5	EC	CPC	NOX	TEMP	PREC	RAD
0	2014-01-01 01:00:00	7.1	36.7	5.3	35.6	NaN	1.4	17257.8	43.2	0.5	0.0	0.0
1	2014-01-01 02:00:00	4.7	35.5	6.3	57.3	NaN	1.5	20101.9	47.1	0.3	0.0	0.0

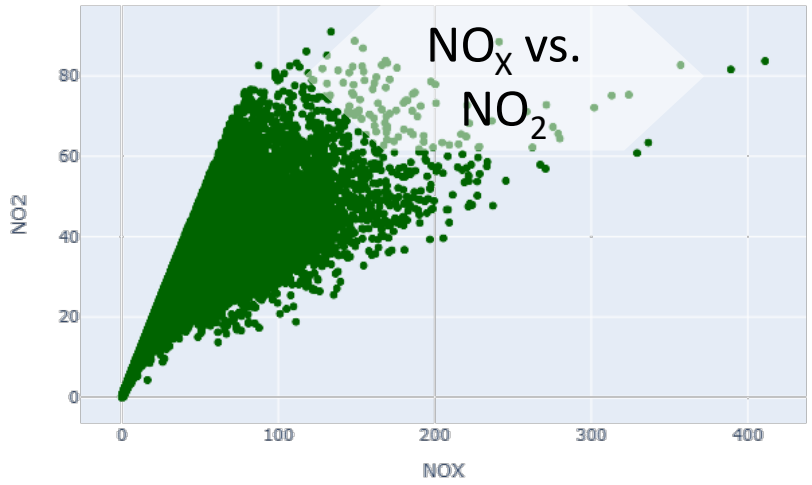










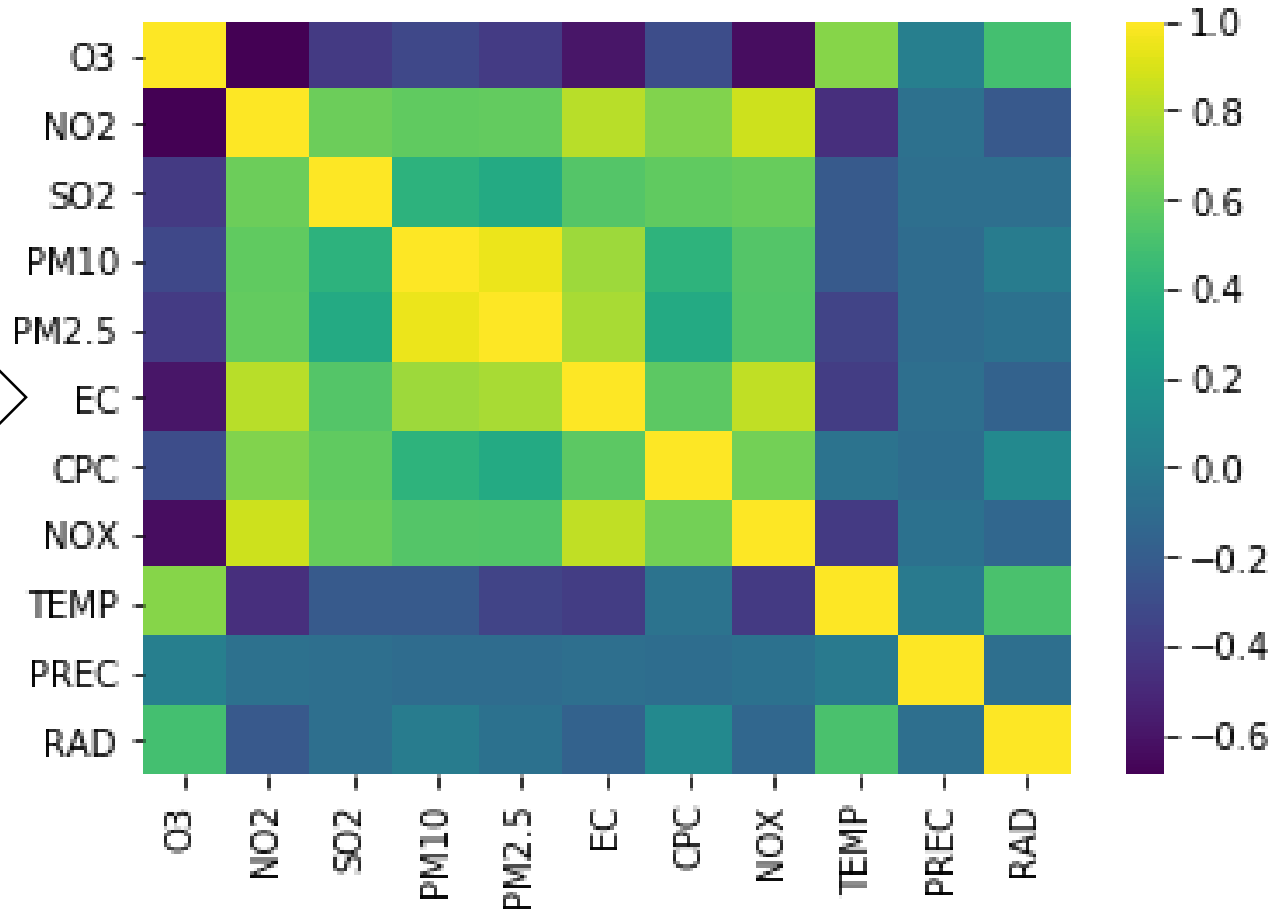


Correlation of Variables

px.scatter(df_basel_bin, x="O3", y="PM10" , color_discrete_sequence=["darkgreen"])

Correlation Matrix

	O3	NO2	SO2	PM10	PM2.5	EC	CPC	NOX	TEMP	PREC	RAD
O3	1.000000	-0.689528	-0.399333	-0.320907	-0.396390	-0.594459	-0.293816	-0.633653	0.692559	0.035824	0.492596
NO2	-0.689528	1.000000	0.622611	0.589322	0.596209	0.819590	0.677649	0.871607	-0.461351	-0.067927	-0.222471
SO2	-0.399333	0.622611	1.000000	0.397920	0.337779	0.545976	0.589424	0.606417	-0.216469	-0.080991	-0.078979
PM10	-0.320907	0.589322	0.397920	1.000000	0.947483	0.755268	0.399956	0.547127	-0.216881	-0.097354	0.012501
PM2.5	-0.396390	0.596209	0.337779	0.947483	1.000000	0.776438	0.336360	0.541758	-0.352146	-0.095720	-0.058755
EC	-0.594459	0.819590	0.545976	0.755268	0.776438	1.000000	0.577492	0.841022	-0.390973	-0.080603	-0.167850
CPC	-0.293816	0.677649	0.589424	0.399956	0.336360	0.577492	1.000000	0.641830	-0.046931	-0.091947	0.109030
NOX	-0.633653	0.871607	0.606417	0.547127	0.541758	0.841022	0.641830	1.000000	-0.403684	-0.056545	-0.131912
TEMP	0.692559	-0.461351	-0.216469	-0.216881	-0.352146	-0.390973	-0.046931	-0.403684	1.000000	-0.002907	0.515680
PREC	0.035824	-0.067927	-0.080991	-0.097354	-0.095720	-0.080603	-0.091947	-0.056545	-0.002907	1.000000	-0.081704
RAD	0.492596	-0.222471	-0.078979	0.012501	-0.058755	-0.167850	0.109030	-0.131912	0.515680	-0.081704	1.000000

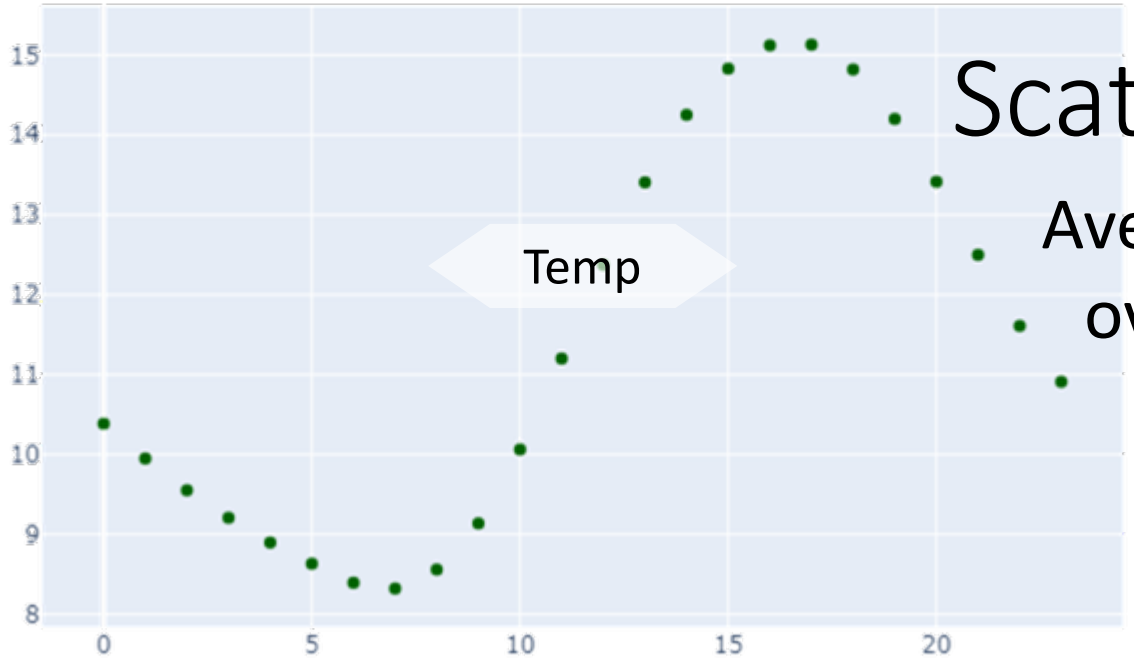


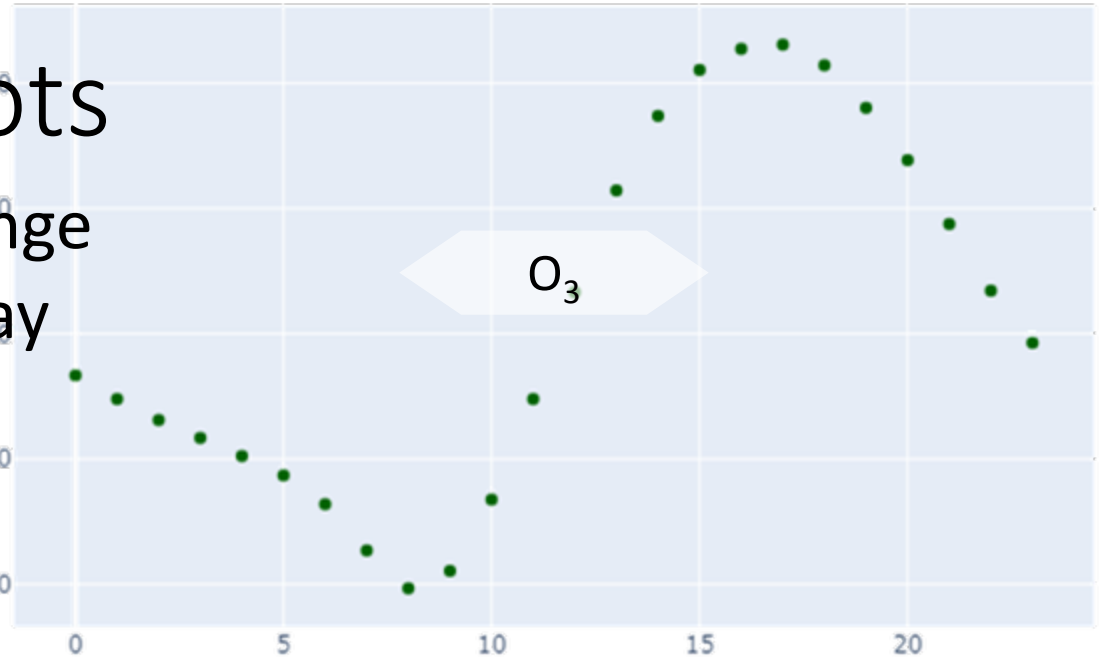

```
import matplotlib as plt
import seaborn as sns
sns.heatmap(df_basel_bin.corr(), cmap='viridis')
```

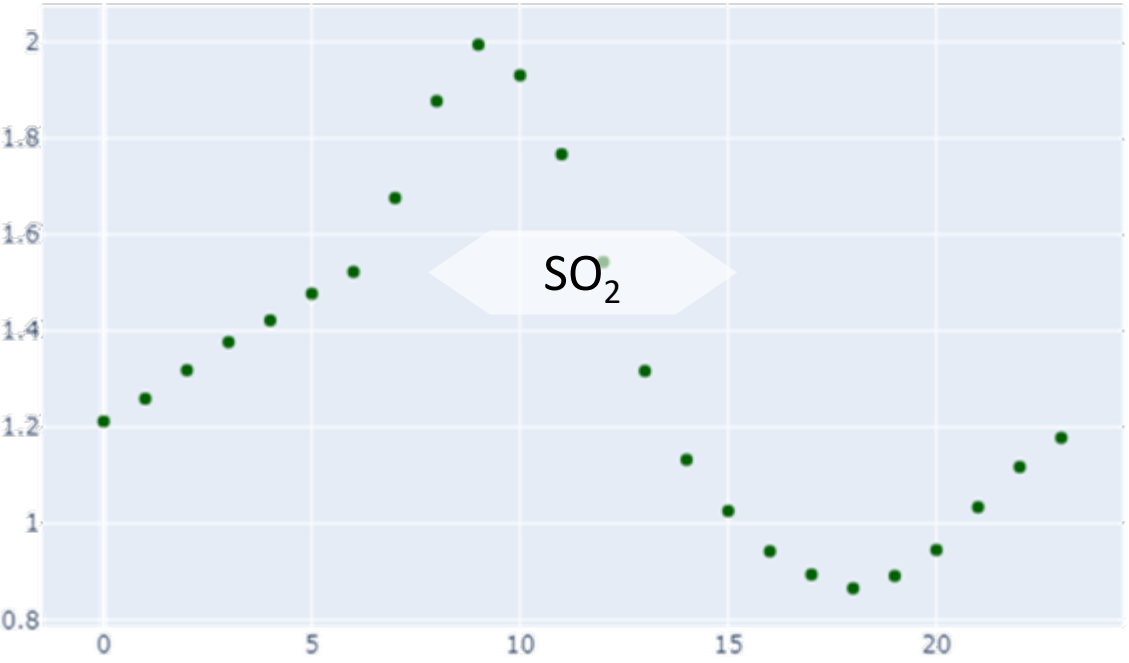
```
df_basel_bin.corr()
```

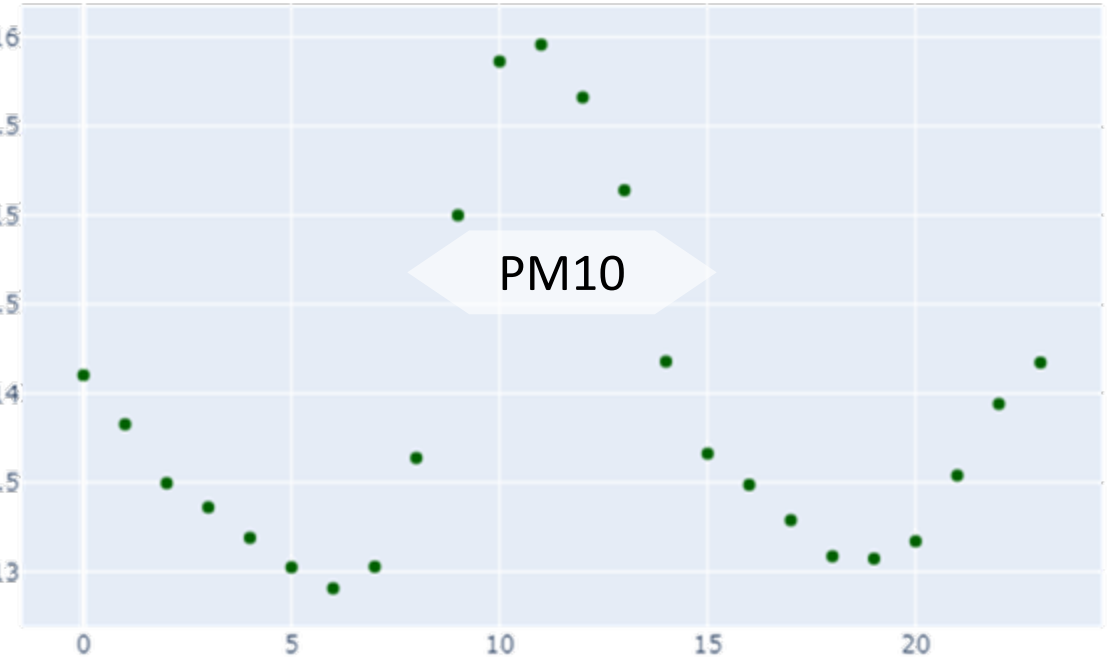
Scatter Plots

Average change over one day










```
px.scatter(x=df_basel_bin2.index.hour.unique(), y=df_basel_bin2.groupby(df_basel_bin2.index.hour).SO2.mean() , color_discrete_sequence=["darkgreen"])
```

Merging & Modelling

Merged Data Frame

Time stamp	O3	NO2	SO2	PM10	PM2.5	EC	CPC	NOX
2014-08-12 00:00:00	47.7	4.4	0.3	7.6	NaN	0.1	3616.1	4.6
2014-08-12 01:00:00	44.7	5.7	0.4	8.4	NaN	0.1	4250.2	5.9

df_basel

Traffic In	Traffic Out
328	56
48	12

df_verkehr_stjohann

temp	humidity	Pre-ssure	precipi tation	Cloud cover	Sunshine duration	Shortwave radiation	Wind speed	Wind direction
18.0105	57.0	1015.4	0.0	100.0	0.0	0.0	5.39419	200.225
17.7605	56.0	1015.8	0.0	100.0	0.0	0.0	2.51321	237.995

df_basel_wetter

Joining & Renaming DF_Verkehr In- & Out

```
df_all = df_basel_bin.join(df_verkehr_stjohann[df_verkehr_stjohann.LaneName == "Spur 1"].Total, how="inner")
df_all.rename(columns={'Total':'TrafficIn'}, inplace=True)
```

Joining Basel Weather

```
df_all = df_all.join(df_basel_wetter, how="inner")
```

Only using rows with complete data

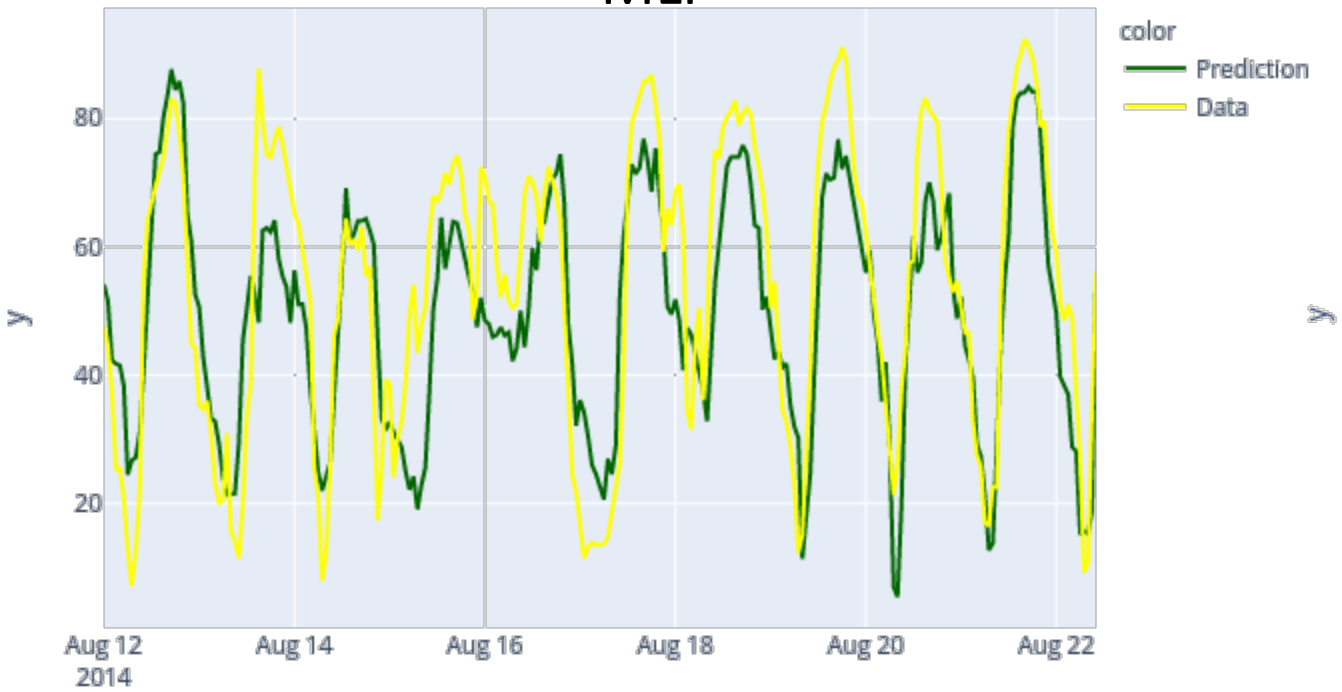
```
df_all_clean = df_all[df_all['O3'].isna() == False]
```

Model Creation

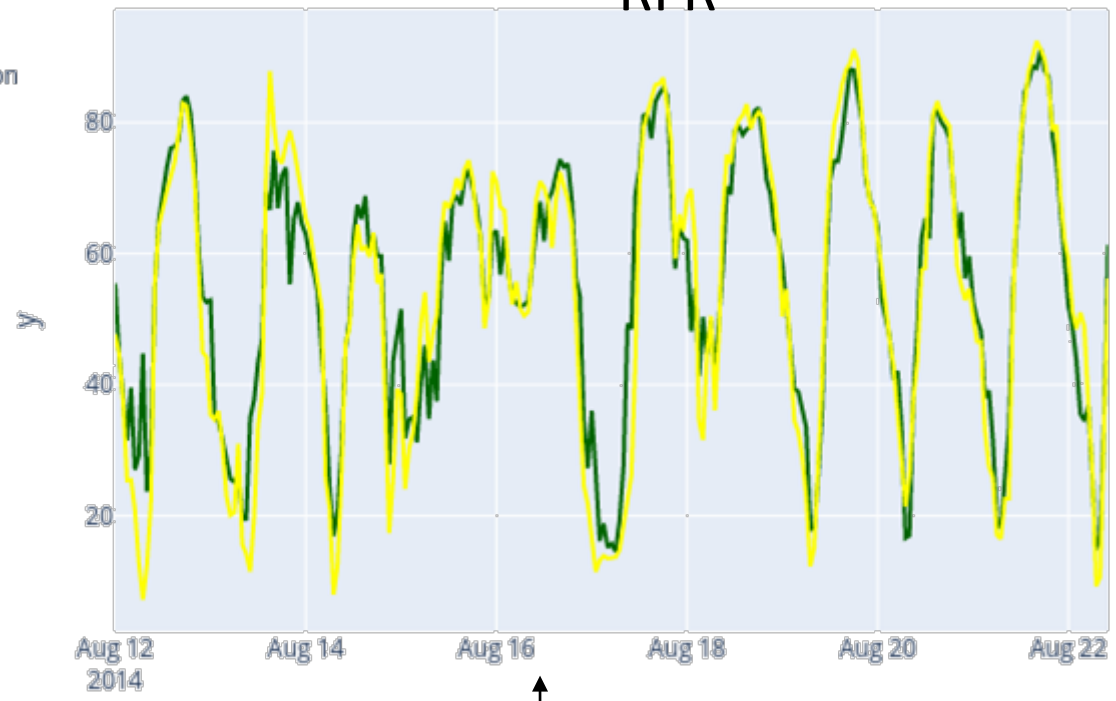
	MLP Regression	Linear Model	Stochastic Gradiant	Random Forest
Creating Trainings Data & Fitting Model – Using Standard Hyper Parameter from Scikit-Learn	from sklearn.neural_network import MLPRegressor mlp = MLPRegressor() X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42) mlp.fit(X_train, y_train)	from sklearn import linear_model lm = linear_model.LinearRegression() lm.fit(X_train, y_train)	from sklearn.linear_model import SGDRegressor sgd = SGDRegressor(verbose=True) sgd.fit(X_train, y_train)	from sklearn.ensemble import RandomForestRegressor rfr = RandomForestRegressor(verbose=True) rfr.fit(X_train, y_train)
Calculating R^2-Score of Test Data	y_predsgd = sgd.predict(X_test) r2_score(y_test, y_predsgd)			
	0.756402066994691 ✓	0.6543257157570724	-2.369159908684308e+25	0.8394071700583421 ✓

Prediction vs. Data

MLP



RFR




```
df_plot1 = pd.DataFrame(data=rfr.predict(X), index=df.timestamp, columns={'value‘})
df_plot1['type'] = 'Prediction‘
df_plot2 = pd.DataFrame()
df_plot2['value'] = y
df_plot2.index = df.timestamp
df_plot2['type'] = 'Data‘
df_plot = pd.concat([df_plot1, df_plot2])
df_plot = df_plot.sort_values('timestamp’)
px.line(x=df_plot.index[0:500], y=df_plot.value[0:500], color=df_plot.type[0:500])
```

Better Prediction ✓

Cross Validation RFR ✚

{'fit_time': array([39.7163496 , 41.22160959, 39.94468474]),
 'score_time': array([0.3892498 , 0.39804006, 0.37150788]),
 'test_r2': array([**0.69184419, 0.70089555, 0.75655615**]),
 'train_r2': array([0.9819971 , 0.98168873, 0.98054151]),
 'test_neg_mean_squared_error': array([-352.65556057, -330.95266442, -322.5348811]),
 'train_neg_mean_squared_error': array([-22.01922145, -22.91095844, -21.93464345])}

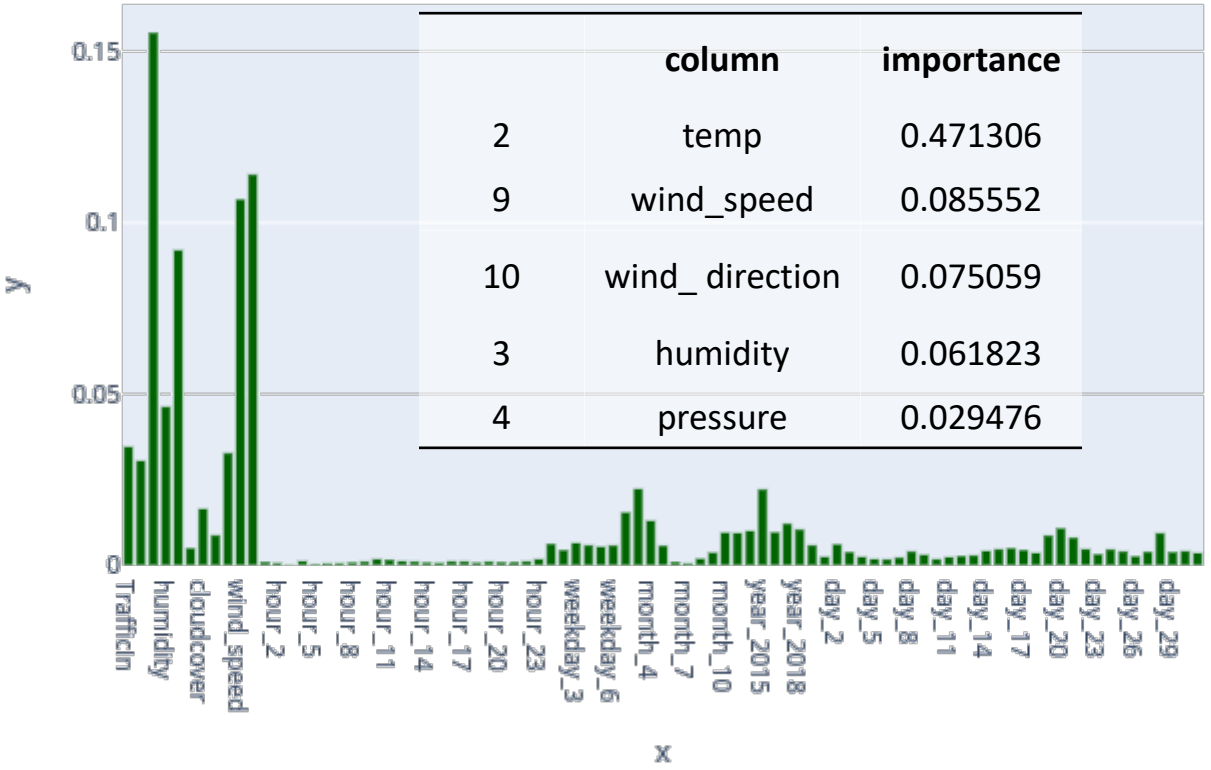
Not as good as before ->

Overfitting?

```
from sklearn.model_selection import cross_validate
scores = cross_validate(rfr, X, y, cv=3,
...                     scoring=('r2', 'neg_mean_squared_error'),
...                     return_train_score=True)
scores
```

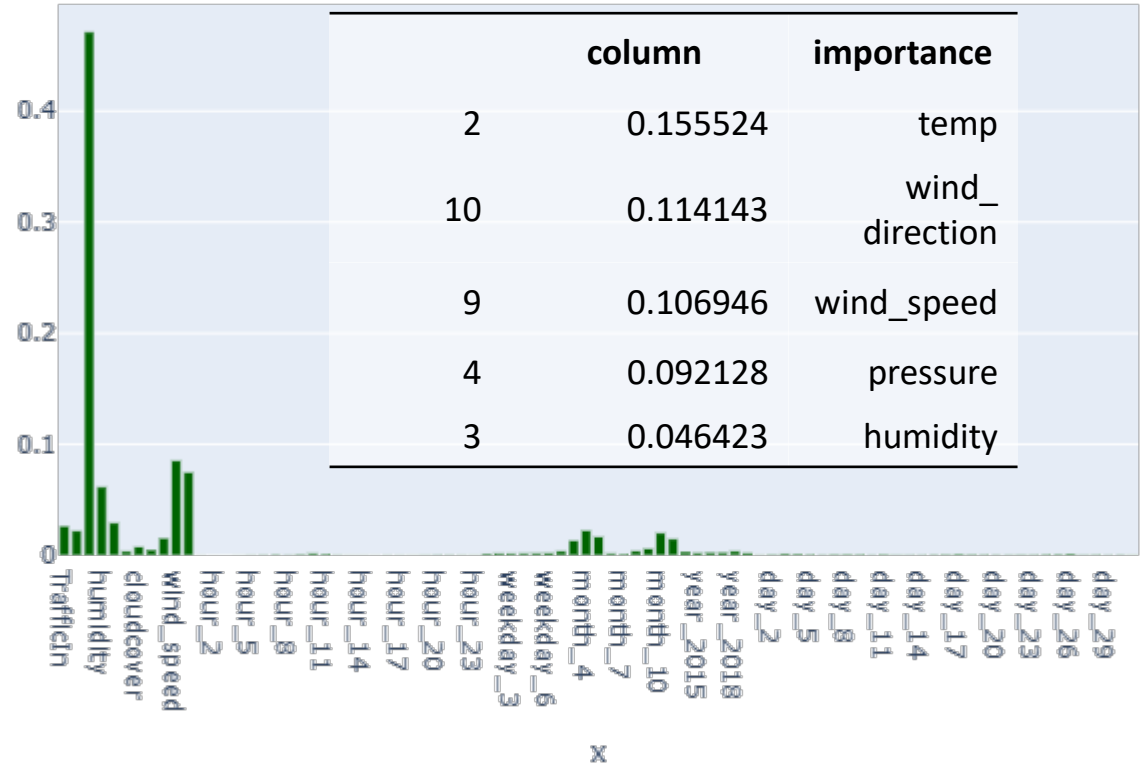
Higest influence of variables on…

PM10



	column	importance
2	temp	0.471306
9	wind_speed	0.085552
10	wind_ direction	0.075059
3	humidity	0.061823
4	pressure	0.029476

O_3



	column	importance
2	0.155524	temp
10	0.114143	wind_ direction
9	0.106946	wind_speed
4	0.092128	pressure
3	0.046423	humidity

```
px.bar(y=rfr.feature_importances_, x=df[df.columns[9:]].columns, color_discrete_sequence=["darkgreen"])
```