

Prosjektoppgave

December 21, 2022

Contents

1 Innledning	2
2 Datainnhenting	3
2.1 Værdata	3
2.2 UV-data	5
2.3 Lufttrykk	6
2.4 Trafikkdata	7
2.5 Helligdager	9
2.6 Strømforbruk	9
2.7 Forurensing	10
2.8 Eksport av CSV	12
3 Visualisering og deskriptiv statistikk	13
4 Enkel lineær regresjon	25
5 Multippel regresjon	32
6 ”Fake”-data Simulasjon	33
6.1 Normalfordeling	33
6.2 ****Bootstrapping	35
7 Tidsrekke-regresjon	36
8 Diagnostikk og evaluering av modell	51
9 Diskusjon om identifikasjon og kausalitet	52
10 Kilder	53
11 Word Count	54

1 Innledning

I denne prosjektoppgaven vil jeg ta for meg data knyttet til vær, stråling og forurensing i Trondheim, dette for å se en sammenheng mellom disse og om det være mulig å predikere når det er høyest andel av svevestøv basert på disse variablene. Svevestøv er i dag et problem i byer, spesielt for personer med astma og andre luftveissykdommer, disse partiklene deles ofte inn i de to størrelsene PM2.5 og PM10 der partiklene er henholdsvis 2.5 og 10 μm i diameter.

2 Datainnhenting

Starter med å importere de relevante pakkene, for deretter å innhente værdata fra Metrologisk institutt. Målingene er gjort på Voll målestasjon i Trondheim.

```
[ ]: import matplotlib
      import pandas as pd
      import matplotlib.pyplot as plt
      import numpy as np
      import seaborn as sns
      import scipy.stats as sts
      import statsmodels as sms
      import statsmodels.formula.api as smf
      from statsmodels.tsa.stattools import adfuller
      import holidays
      from IPython.display import display, Image
```

2.1 Værdata

```
[ ]: weather = pd.read_csv("data/Temperatur_regn.csv", sep=";", decimal=",")
```

```
[ ]: weather.tail()
```

```
[ ]:                                     Navn   Stasjon \
2553          Trondheim - Voll   SN68860
2554          Trondheim - Voll   SN68860
2555          Trondheim - Voll   SN68860
2556          Trondheim - Voll   SN68860
2557 Data er gyldig per 16.11.2022 (CC BY 4.0), Met...     NaN

    Tid(norsk normaltid)  Maksimumstemperatur (døgn) \
2553        28.12.2021             -4.5
2554        29.12.2021             -4.5
2555        30.12.2021            -0.4
2556        31.12.2021             2.4
2557           NaN                  NaN

    Minimumstemperatur (døgn) Høyeste middelvind (døgn)  Nedbør (døgn)
2553            -10.1                 1,6               0.0
2554             -9.7                 2,4               0.0
2555             -6.5                 2,9               0.0
2556             -1.9                 4,8               3.2
2557             NaN                  NaN               NaN
```

```
[ ]: weather.drop(weather.tail(1).index, inplace=True)
```

```
[ ]: weather.rename(columns={"Navn" : "Station_Name"}, inplace=True)
weather.rename(columns={"Stasjon" : "StationID"}, inplace=True)
weather.rename(columns={"Tid(norsk normaltid)" : "Time"}, inplace=True)
weather.rename(columns={"Maksimumstemperatur (døgn)" : "Max_Temp"}, ↵
    ↵inplace=True)
weather.rename(columns={"Minimumstemperatur (døgn)" : "Min_Temp"}, inplace=True)
weather.rename(columns={"Høyeste middelvind (døgn)" : "Max_Wind"}, inplace=True)
weather.rename(columns={"Nedbør (døgn)" : "Rain"}, inplace=True)
```

Endrer navnet på kolonnene for lettere koding, unngår mellomrom av samme grunn.

```
[ ]: weather.set_index("Time", inplace=True)
weather.index = pd.to_datetime(weather.index, format="%d.%m.%Y").date
```

Setter indeksen til dag ved bruk av pandas sin innebygde datetime funksjon. Formatet er europeisk med dag, måned, år.

```
[ ]: year = pd.DatetimeIndex(weather.index).year.to_numpy()
month = pd.DatetimeIndex(weather.index).month.to_numpy()
```

```
[ ]: weather.insert(0, "Month", month)
weather.insert(1, "Year", year)
```

Legger til måned og år som variabler for å kunne se om tid på året påvirker de andre variablene.

```
[ ]: weather.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2557 entries, 2015-01-01 to 2021-12-31
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Month       2557 non-null   int64  
 1   Year        2557 non-null   int64  
 2   Station_Name 2557 non-null   object  
 3   StationID    2557 non-null   object  
 4   Max_Temp     2557 non-null   float64 
 5   Min_Temp     2557 non-null   float64 
 6   Max_Wind     2557 non-null   object  
 7   Rain         2557 non-null   float64 
dtypes: float64(3), int64(2), object(3)
memory usage: 179.8+ KB
```

Lager en variabel for daglig gjennomsnittstemperatur

```
[ ]: mean_temp = (weather["Max_Temp"]+weather["Min_Temp"])/2
[ ]: weather.insert(6, "Mean_Temp", mean_temp)
```

Bruk funksjonen insert for å legge til gjennomsnittstemperaturen ved siden av de andre temperaturvariablene.

```
[ ]: weather["Is_Rain"] = np.where(weather.Rain>0, 1, 0)
```

Legger til en binomisk variabel for om det har regnet eller ikke.

```
[ ]: weather["Max_Wind"] = weather.Max_Wind.str.replace(',', '.')
      weather["Max_Wind"] = weather.Max_Wind.str.replace('-', '0')
```

Ser ovenfor at vindvariabelen har blitt importert som et object og ikke som et float-tall. Ser gjennom datasettet og finner at årsaken til dette er at vindstille er satt til “-” og ikke “0”. I tillegg må desimalformen endres fra “,” til “.”, da dette ikke ble gjort ved importeringen. Omgjør deretter hele kolonnen til float64.

```
[ ]: weather["Max_Wind"] = weather.Max_Wind.astype("float64")
```

2.2 UV-data

```
[ ]: uv = pd.read_table("https://raw.githubusercontent.com/uvnrpa/Daily_Doses/master/  
    ↪TRH_daily.txt", skiprows=31)
```

Historisk UV-data er kun publisert som en txt-fil, det var derfor nødvendig å importere den med "read_table", selv om "read_fwf" også hadde vært en mulighet. Dette datasettet slutter 31 desember 2021, det var derfor bare å ta lengden av hoveddatasettet og kopiere dette ut fra bunnen av UV-datasettet.

```
[ ]: length_period = len(weather)
```

```
[ ]: uv = uv.tail(length_period)
```

```
[ ]: weather["UVA"] = uv["UVA"].to_numpy()  
weather["UVB"] = uv["UVB"].to_numpy()
```

Velger å kun se på UVA og UVB stråling da de andre målingene i datasettet tar i større grad for seg biologiske faktorer som grad av fotosyntese og omgjøring av 7-DH7 til vitamin D3.

For å unngå NaN-verdier måtte pandas seriene gjøres om til numpy før det ble lagt til i hoveddatasettet. Dette har nok noe med forskjellen i indekseringen å gjøre. Dette steget gjentas gjennom databehandlingen.

```
[ ]: weather.tail()
```

[]:	Month	Year	Station_Name	StationID	Max_Temp	Min_Temp	\
2021-12-27	12	2021	Trondheim - Voll	SN68860	0.2	-7.3	
2021-12-28	12	2021	Trondheim - Voll	SN68860	-4.5	-10.1	
2021-12-29	12	2021	Trondheim - Voll	SN68860	-4.5	-9.7	
2021-12-30	12	2021	Trondheim - Voll	SN68860	-0.4	-6.5	
2021-12-31	12	2021	Trondheim - Voll	SN68860	2.4	-1.9	

	Mean_Temp	Max_Wind	Rain	Is_Rain	UVA	UVB
2021-12-27	-3.55	4.2	1.2	1	24021.0	57.144
2021-12-28	-7.30	1.6	0.0	0	23988.0	46.431
2021-12-29	-7.10	2.4	0.0	0	19061.0	32.673
2021-12-30	-3.45	2.9	0.0	0	24308.0	30.534
2021-12-31	0.25	4.8	3.2	1	17993.0	19.491

```
[ ]: weather = weather.drop("StationID", axis=1)
```

2.3 Lufttrykk

```
[ ]: pressure = pd.read_csv("data/Lufttrykk.csv", sep=";", decimal=",")
pressure.drop(pressure.tail(1).index, inplace=True)
```

```
[ ]: pressure.columns
```

```
[ ]: Index(['Navn', 'Stasjon', 'Tid(norsk normaltid)',
           'Høyeste lufttrykk i havnivå (døgn)',
           'Laveste lufttrykk i havnivå (døgn)'],
           dtype='object')
```

```
[ ]: weather["Max_Pressure"] = pressure["Høyeste lufttrykk i havnivå (døgn)"].
      ↪to_numpy()
weather["Min_Pressure"] = pressure["Laveste lufttrykk i havnivå (døgn)"].
      ↪to_numpy()
```

```
[ ]: weather.tail()
```

	Month	Year	Station_Name	Max_Temp	Min_Temp	Mean_Temp	\
2021-12-27	12	2021	Trondheim - Voll	0.2	-7.3	-3.55	
2021-12-28	12	2021	Trondheim - Voll	-4.5	-10.1	-7.30	
2021-12-29	12	2021	Trondheim - Voll	-4.5	-9.7	-7.10	
2021-12-30	12	2021	Trondheim - Voll	-0.4	-6.5	-3.45	
2021-12-31	12	2021	Trondheim - Voll	2.4	-1.9	0.25	

	Max_Wind	Rain	Is_Rain	UVA	UVB	Max_Pressure	\
2021-12-27	4.2	1.2	1	24021.0	57.144	1007.9	
2021-12-28	1.6	0.0	0	23988.0	46.431	1001.6	
2021-12-29	2.4	0.0	0	19061.0	32.673	1000.6	
2021-12-30	2.9	0.0	0	24308.0	30.534	996.4	
2021-12-31	4.8	3.2	1	17993.0	19.491	1009.9	

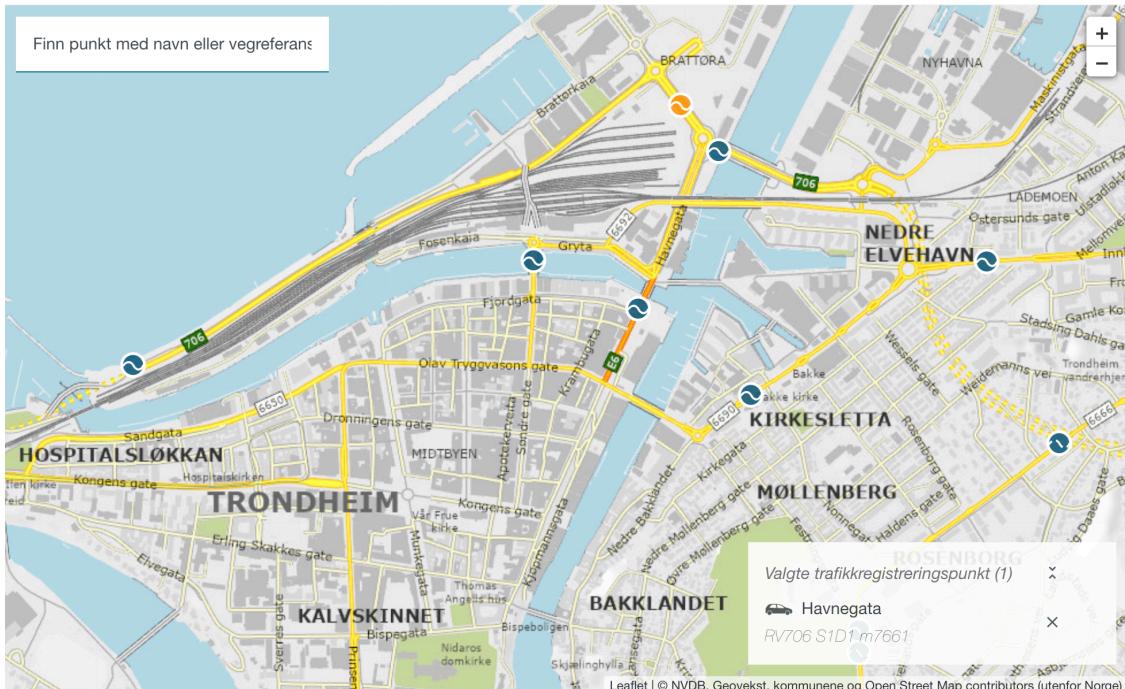
	Min_Pressure
2021-12-27	1002.2
2021-12-28	998.7

2021-12-29	996.8
2021-12-30	987.3
2021-12-31	995.1

2.4 Trafikkdata

Velger å legge til trafikkdata fra Havnegata i Trondheim, grunnen til dette er delvis fordi det var den målestasjonen med flest registrerte datoer, i tillegg til at beliggenheten burde føre til et representativt bilde av trafikken i Trondheim sentrum. Målestasjonen er uthevet øverst i bildet i oransje:

```
[ ]: display(Image(filename="data/veidata_kart.png", width=1500))
```



Importerer csv-fil lastet ned fra Statens Vegvesen, Havnegata som hoveddatasettet, og Innherredsveien for å komplettere.

```
[ ]: veidata_havnegata = pd.read_csv("data/veidata_havnegata.csv", encoding="latin1", decimal=",", sep=";")
veidata_innherred_saxe = pd.read_csv("data/veidata_innherred_saxe.csv", encoding="latin1", decimal=",", sep=";")
```

Ettersom det var kun de dagene med registrerte data som var inkludert i CSV-filen er det nødvendig å sette inn disse dagene med NaN-verdier, slik at en lettere kan sette inn data fra Innherredsveien.

```
[ ]: veidata_havnegata = veidata_havnegata.loc[(veidata_havnegata['Felt'] == "Totalt")]
veidata_havnegata = veidata_havnegata[['Dato", "Volum"]]
veidata_havnegata["Volum"] = veidata_havnegata.Volum.str.replace('-', '0')
veidata_havnegata["Volum"] = veidata_havnegata.Volum.astype("float64")
veidata_havnegata.set_index("Dato", inplace=True)
veidata_havnegata.index = pd.to_datetime(veidata_havnegata.index, format="%Y-%m-%d").date

idx = pd.date_range('2015-01-01', '2021-12-31')
veidata_havnegata = veidata_havnegata.reindex(idx, fill_value=0)
veidata_havnegata["Volum"].replace(0, np.nan, inplace=True)
veidata_innherred_saxe = veidata_innherred_saxe.loc[(veidata_innherred_saxe['Felt'] == "Totalt")]
```

```
[ ]: veidata_havnegata.isna().sum()
```

```
[ ]: Volum    70
      dtype: int64
```

```
[ ]: veidata_innherred_saxe = veidata_innherred_saxe[['Dato", "Volum"]]
veidata_innherred_saxe["Volum"] = veidata_innherred_saxe.Volum.str.replace('-', '0')
veidata_innherred_saxe["Volum"] = veidata_innherred_saxe.Volum.astype("float64")
veidata_innherred_saxe.set_index("Dato", inplace=True)
veidata_innherred_saxe.index = pd.to_datetime(veidata_innherred_saxe.index, format="%Y-%m-%d").date

idx = pd.date_range('2015-01-01', '2021-12-31')
veidata_innherred_saxe = veidata_innherred_saxe.reindex(idx, fill_value=0)
veidata_innherred_saxe["Volum"].replace(0, np.nan, inplace=True)

oppjusteringsfaktor = veidata_havnegata.Volum.mean()/veidata_innherred_saxe.Volum.mean()
veidata_havnegata["Volum"] = fillna(veidata_innherred_saxe["Volum"]*oppjusteringsfaktor, inplace=True)
```

Ser det er en viss forskjell mellom data fra Havnegata og Innherredsveien, tar derfor og dividerer gjenomsnittene på hverandre for å få en oppjusteringsfaktor. Multipliserer denne deretter med de dataene jeg fyller inn i datasettet fra Havnegata. Selv om det vil være et lite avvik fra de reelle bilpasseringene disse dagene så vil det være et godt estimat, i tillegg til at det kun vil gjelde 70 datapunkter i et sett med 2557.

```
[ ]: veidata_havnegata[veidata_havnegata["Volum"].isna()]
veidata_havnegata["Volum"] = (veidata_havnegata["Volum"].ffill()+veidata_havnegata["Volum"].bfill())/2
veidata_havnegata["Volum"] = veidata_havnegata["Volum"].round(0).astype("int")
```

```
veidata_havnegata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2557 entries, 2015-01-01 to 2021-12-31
Freq: D
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype  
---  --     --     --    
 0   Volum   2557 non-null   int64  
dtypes: int64(1)
memory usage: 40.0 KB
```

```
[ ]: weather["Traffic"] = veidata_havnegata["Volum"]
```

2.5 Helligdager

Legger til en variabel med helligdager ettersom det kan tenkes at dette kan påvirke nivået av svevestøv og trafikk. Dette importeres ved bruk av pakken “holidays”.

```
[ ]: holiday_list = []
for holiday in holidays.Norway(years=[2015, 2016, 2017, 2018, 2019, 2020, 2021]).items():
    holiday_list.append(holiday)

holidays_df = pd.DataFrame(holiday_list, columns=["date", "holiday"])
holidays_df.set_index("date", inplace=True)
holidays_df.index = pd.to_datetime(holidays_df.index, format="%Y-%m-%d").date
holidays_df = holidays_df.sort_index()

holiday_dates = holidays_df

idx = pd.date_range('2015-01-01', '2021-12-31')
holidays_df = holidays_df.reindex(idx, fill_value=0)
holidays_df["Is_Holiday"] = np.where(holidays_df.holiday == 0, 0, 1)
holidays_df["holiday"].replace(0, np.nan, inplace=True)
```

2.6 Strømforbruk

Legger ved strømforbruket i midt-Norge, dette gjøres ved å laste ned daglige data for hvert år for deretter å bruke concat for å slå dette sammen.

```
[ ]: power2015 = pd.read_csv("data/poweruse_2015.csv")
power2016 = pd.read_csv("data/poweruse_2016.csv")
power2017 = pd.read_csv("data/poweruse_2017.csv")
power2018 = pd.read_csv("data/poweruse_2018.csv")
power2019 = pd.read_csv("data/poweruse_2019.csv")
power2020 = pd.read_csv("data/poweruse_2020.csv")
power2021 = pd.read_csv("data/poweruse_2021.csv")
```

```
[ ]: power_list = [power2015, power2016, power2017, power2018, power2019, power2020,  
↳ power2021]

[ ]: daily_power = pd.DataFrame()  
daily_power = pd.concat(power_list, ignore_index=True)

[ ]: daily_power.drop(daily_power.head(3).index, inplace=True)  
daily_power.drop(daily_power.tail(2).index, inplace=True)  
daily_power.reset_index(inplace=True)

[ ]: weather_power = weather.copy()

[ ]: weather_power["Max_Power"] = daily_power["Max Total Load [MW] - BZN|NO3"].  
↳ to_numpy()  
weather_power["Min_Power"] = daily_power["Min Total Load [MW] - BZN|NO3"].  
↳ to_numpy()
```

2.7 Forurensing

```
[ ]: pollution = pd.read_csv("data/luftkvalitet.csv", skiprows=3)
```

```
[ ]: pollution.isna().sum()
```

Tid	0
E6-Tiller PM10 µg/m³ Day	330
Dekning	1
E6-Tiller PM2.5 µg/m³ Day	330
Dekning.1	1
Elgeseter PM10 µg/m³ Day	132
Dekning.2	1
Elgeseter PM2.5 µg/m³ Day	132
Dekning.3	1
Elgeseter mobil PM10 µg/m³ Day	2514
Dekning.4	2513
Elgeseter mobil PM2.5 µg/m³ Day	2514
Dekning.5	2513
Omkjøringsvegen PM10 µg/m³ Day	1871
Dekning.6	1427
Omkjøringsvegen PM2.5 µg/m³ Day	1892
Dekning.7	1427
Torvet PM10 µg/m³ Day	268
Dekning.8	35
Torvet PM2.5 µg/m³ Day	272
Dekning.9	34
Åsveien skole PM10 µg/m³ Day	2121
Dekning.10	2113

Åsveien skole PM2.5 $\mu\text{g}/\text{m}^3$ Day	2121
Dekning.11	2113
Bakke kirke PM10 $\mu\text{g}/\text{m}^3$ Day	539
Dekning.12	445
Bakke kirke PM2.5 $\mu\text{g}/\text{m}^3$ Day	539
Dekning.13	445
dtype: int64	

Velger å bruke målestasjonen i Elgeseter ettersom denne har færrest NaN-verdier, i tillegg til at det er den det er mest relevant å se på som følge av sin nærhet til Adolf Øien bygget. Grunnen til disse NaN-verdiene kan være manglende dekningen på målestasjonene. For å få fullstendige data velger jeg å fylle disse verdiene med målinger fra andre stasjoner i nærheten, jeg velger hovedsaklig de med likest lokasjon og omgivelser og unnlater å ta med målinger ved E6. De verdiene som fremdeles manglet etter dette fylte jeg først ved å ta snittet av den forrige og den neste verdien, dette vil føre til et någenlunde greit estimat for de fem dagene det gjelder. Til slutt fyllte jeg den siste verdien med den nest siste.

```
[ ]: pollution['Elgeseter PM10  $\mu\text{g}/\text{m}^3$  Day'].fillna(pollution['Torvet PM10  $\mu\text{g}/\text{m}^3$  Day'], inplace=True)
print(pollution["Elgeseter PM10  $\mu\text{g}/\text{m}^3$  Day"].isna().sum())
pollution['Elgeseter PM10  $\mu\text{g}/\text{m}^3$  Day'].fillna(pollution['Bakke kirke PM10  $\mu\text{g}/\text{m}^3$  Day'], inplace=True)
print(pollution["Elgeseter PM10  $\mu\text{g}/\text{m}^3$  Day"].isna().sum())
pollution['Elgeseter PM10  $\mu\text{g}/\text{m}^3$  Day'].fillna(pollution['Åsveien skole PM10  $\mu\text{g}/\text{m}^3$  Day'], inplace=True)
print(pollution["Elgeseter PM10  $\mu\text{g}/\text{m}^3$  Day"].isna().sum())
pollution["Elgeseter PM10  $\mu\text{g}/\text{m}^3$  Day"] = (pollution["Elgeseter PM10  $\mu\text{g}/\text{m}^3$  Day"].
    ffill()+pollution["Elgeseter PM10  $\mu\text{g}/\text{m}^3$  Day"].bfill())/2
print(pollution["Elgeseter PM10  $\mu\text{g}/\text{m}^3$  Day"].isna().sum())
pollution["Elgeseter PM10  $\mu\text{g}/\text{m}^3$  Day"].ffill(inplace=True)
print(pollution["Elgeseter PM10  $\mu\text{g}/\text{m}^3$  Day"].isna().sum())
```

27
13
6
1
0

```
[ ]: pollution['Elgeseter PM2.5  $\mu\text{g}/\text{m}^3$  Day'].fillna(pollution['Torvet PM2.5  $\mu\text{g}/\text{m}^3$  Day'], inplace=True)
pollution['Elgeseter PM2.5  $\mu\text{g}/\text{m}^3$  Day'].fillna(pollution['Bakke kirke PM2.5  $\mu\text{g}/\text{m}^3$  Day'], inplace=True)
pollution['Elgeseter PM2.5  $\mu\text{g}/\text{m}^3$  Day'].fillna(pollution['Åsveien skole PM2.5  $\mu\text{g}/\text{m}^3$  Day'], inplace=True)
pollution["Elgeseter PM2.5  $\mu\text{g}/\text{m}^3$  Day"] = (pollution["Elgeseter PM2.5  $\mu\text{g}/\text{m}^3$  Day"].
    ffill()+pollution["Elgeseter PM2.5  $\mu\text{g}/\text{m}^3$  Day"].bfill())/2
pollution["Elgeseter PM2.5  $\mu\text{g}/\text{m}^3$  Day"].ffill(inplace=True)
```

```
print(pollution["Elgeseter PM2.5 µg/m³ Day"].isna().sum())
```

```
0
```

```
[ ]: weather_power["Pollution_PM25"] = pollution["Elgeseter PM2.5 µg/m³ Day"].  
      ↵to_numpy()  
weather_power["Pollution_PM10"] = pollution["Elgeseter PM10 µg/m³ Day"].  
      ↵to_numpy()
```

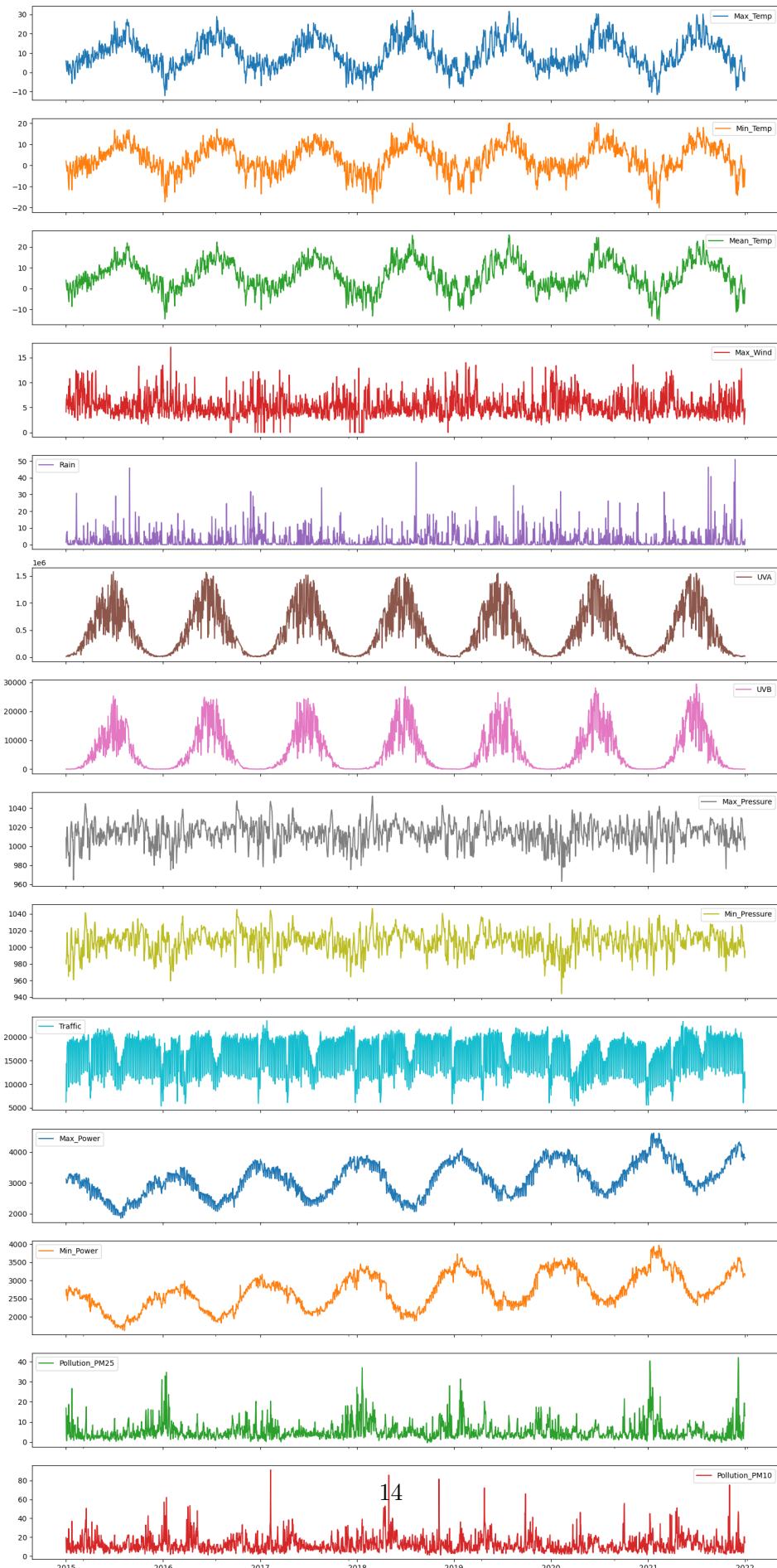
2.8 Eksport av CSV

Til slutt eksporterer jeg den ferdige CSV-filen, god praksis for å holde styr på de orginale tallene om det skulle skje endringer av datasettet gjennom oppgavgen.

```
[ ]: weather_power.to_csv("data/weather_power_TRD.csv")
```

3 Visualisering og deskriptiv statistikk

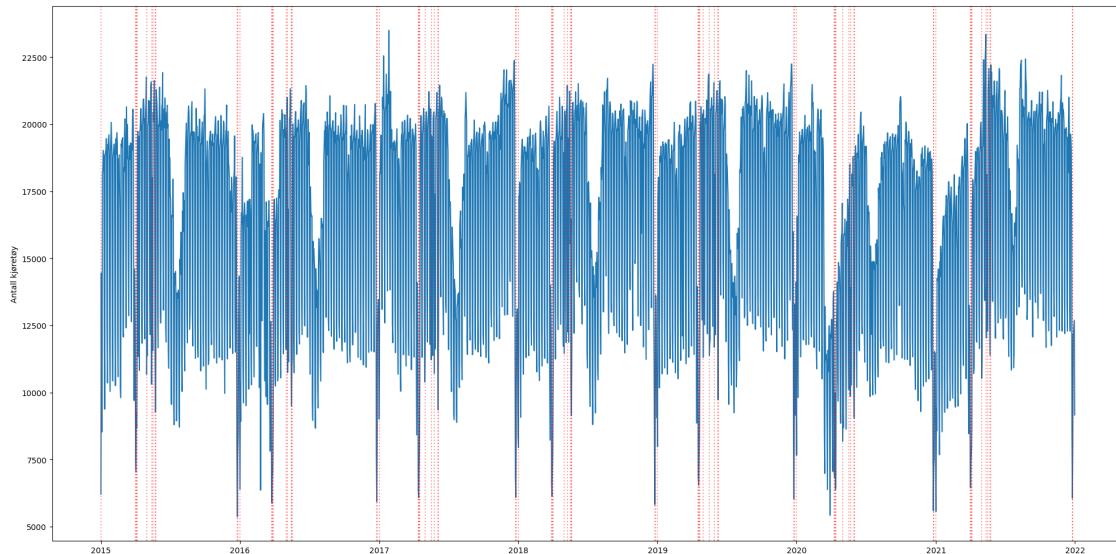
```
[ ]: weather_power.drop(["Is_Rain", "Year", "Month"],axis=1).plot(subplots = True,  
    ↪figsize=(15,30))  
  
plt.tight_layout()  
plt.show()
```



Legg merke til de årlige nedgangene i trafikk rundt nyttår, dette kan visualiseres ved å legge til helligdagene i grafen. De røde strekene under markerer de norske helligdagene, som en ser så blir trafikken sterkt påvirket av dette, spesielt rundt jul og påske. En kan også se den årlige nedgangen som mest sannsynlig viser avvikling av fellesferien. I tillegg er det en tydelig nedgang i mars 2020, dette på grunn av covid.

```
[ ]: weather_power.Traffic.plot(figsize=(20,10))
for (dates, holiday_names) in holiday_dates.iterrows():
    col = (np.random.random(), np.random.random(), np.random.random())
    plt.axvline(x=dates, color="red", linestyle=':', alpha=0.4,
    label=holiday_names[0])

# plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.05),
#            fancybox=True, shadow=True, ncol=5)
plt.ylabel("Antall kjøretøy")
plt.tight_layout()
plt.show()
```



Kan også tydliggjøre dagene med minst trafikk ved å printe ut disse:

```
[ ]: print(weather_power.Traffic.sort_values().head(20))
```

2015-12-25	5369
2020-03-29	5418
2021-01-01	5561
2020-12-25	5589
2018-12-25	5818

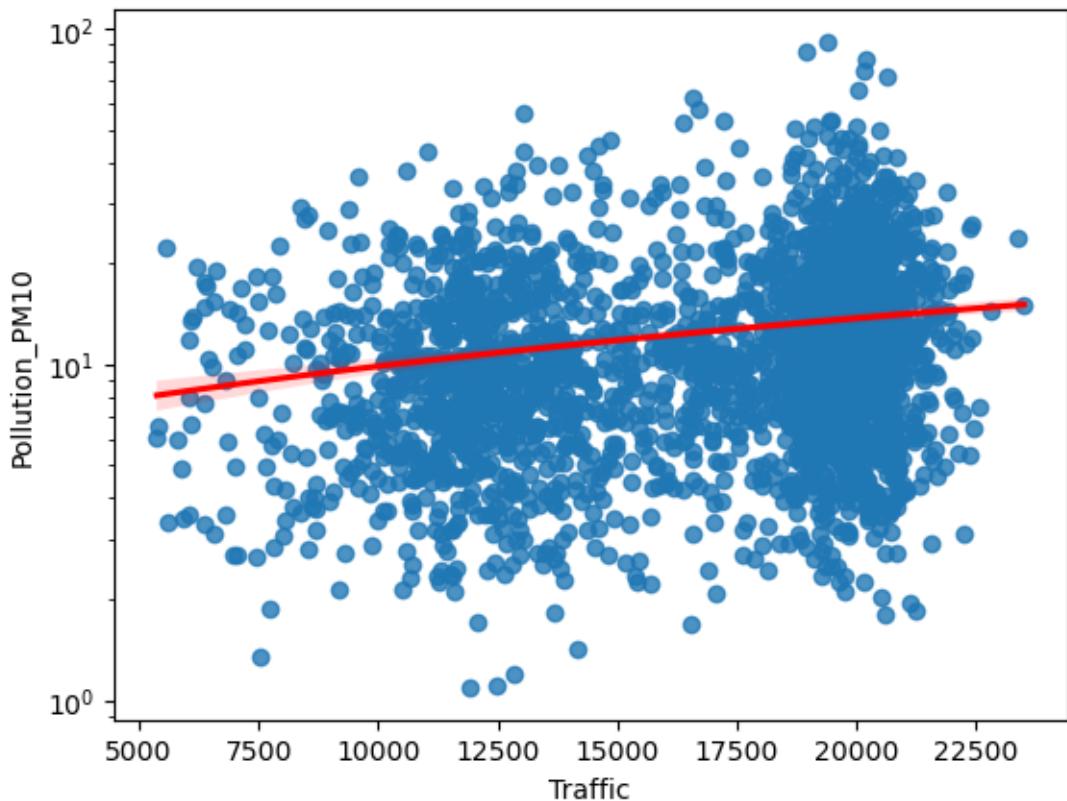
```

2016-03-25    5868
2016-12-25    5931
2016-03-24    6029
2019-12-25    6032
2021-12-25    6057
2017-04-14    6082
2017-12-25    6089
2018-03-30    6135
2015-01-01    6206
2017-04-13    6348
2016-02-24    6357
2020-04-12    6380
2020-03-22    6383
2016-01-01    6394
2021-04-02    6440
Name: Traffic, dtype: int64

```

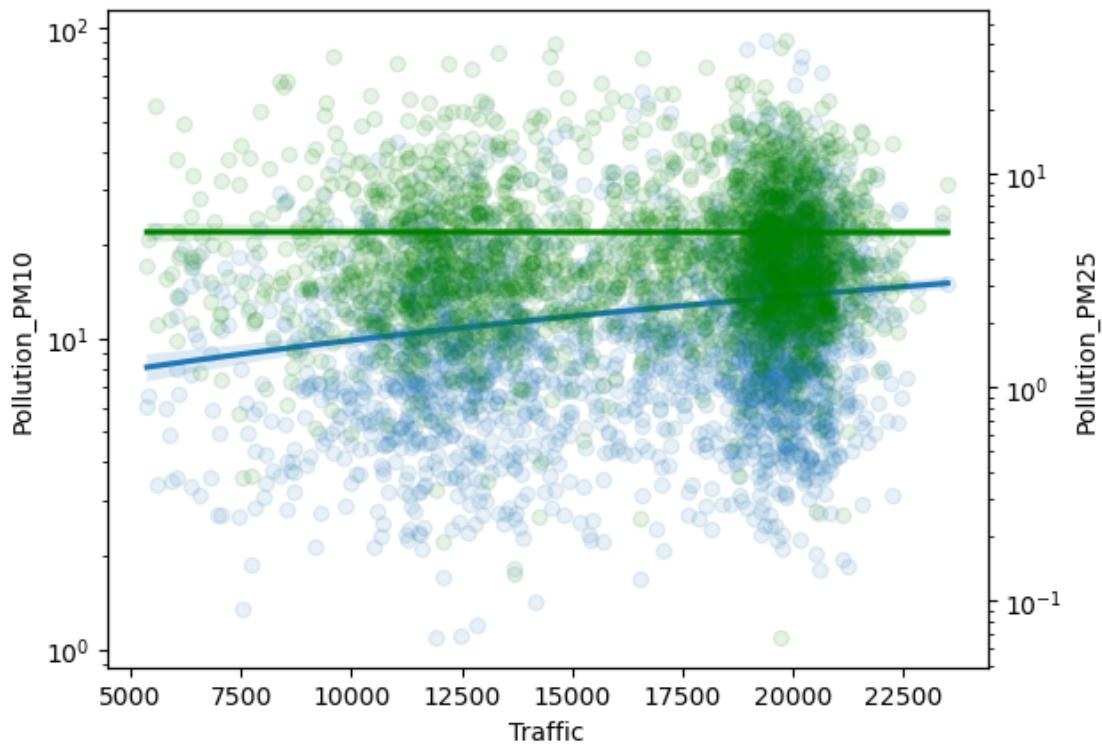
Det vil også være mulig å se på den logaritmiske sammenhengen mellom noen av variablene. Dette for å lettere visualisere sammenhengen mellom variabler hvor grunnlaget kan være logaritmisk.

```
[ ]: ax = sns.regplot(x="Traffic", y="Pollution_PM10", data=weather_power,
                     line_kws={'color': 'red'})
ax.set_yscale('log')
```

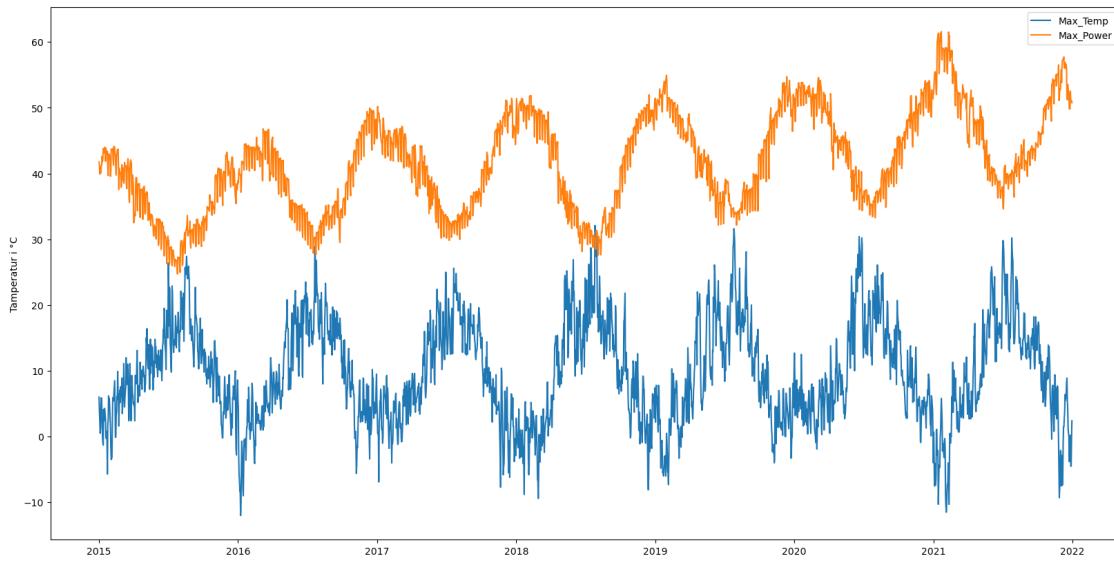


Her vil man letter kunne se sammenhengen mellom trafikk og svevestøv. Nedenfor illustreres forskjellen mellom svevestøv på 10 mikrometer og på 2.5 mikrometer, ser at utslippet på 10 mikrometer påvirkes i høyere grad av trafikk.

```
[ ]: ax = sns.regplot(x="Traffic", y="Pollution_PM10", data=weather_power, scatter_kws={'alpha':0.1})
ax2 = plt.twinx()
ax2 = sns.regplot(x = "Traffic" , y="Pollution_PM25", data=weather_power, color="g", scatter_kws={'alpha':0.1})
ax2.set_yscale('log')
ax.set_yscale('log')
```

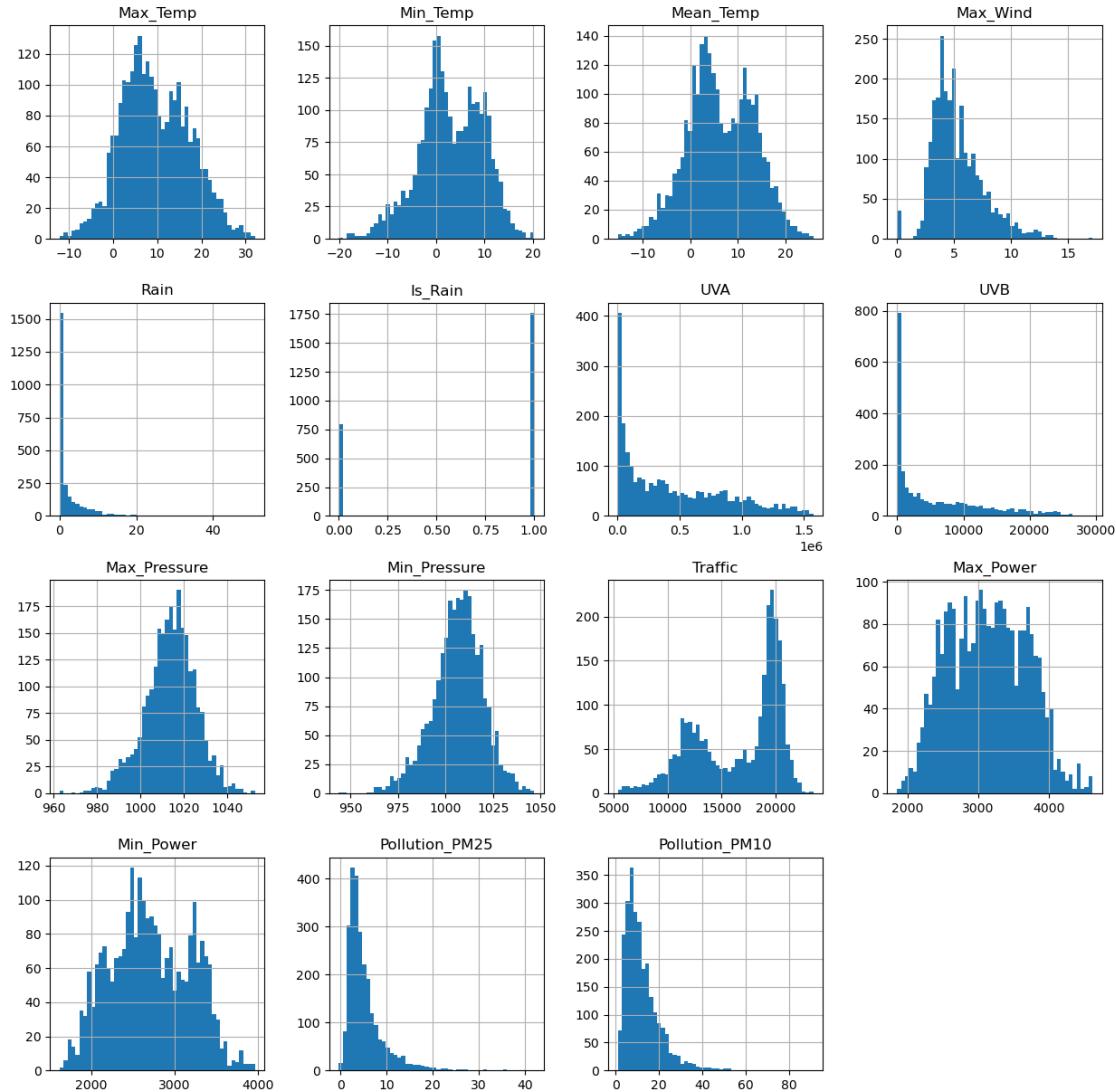


```
[ ]: weather_power.Max_Temp.plot()
(weather_power.Max_Power/75).plot(figsize=(20,10))
plt.ylabel("Temperatur i °C")
plt.legend()
plt.show()
```



En enkel måte å vise den iverse sammenhengen mellom maksimal temperatur og maksimal strømforbruk i løpet av dagen. Dette er ikke overraskende da mesteparten av strømforbruken i en husholdning går til oppvarming.

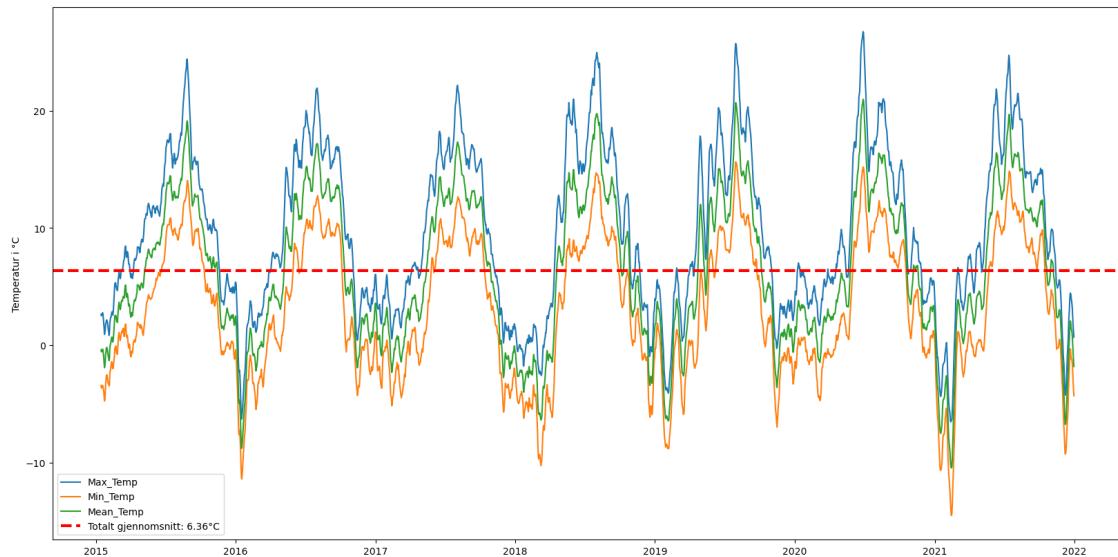
```
[ ]: weather_power.drop(["Year", "Month"], axis=1).hist(bins=50, figsize=(15,15))  
plt.show()
```



Gjennom dette histogrammet kan en se at begge målenhetene for forurensing har en positiv skjevhets(skewness), de er samlet forholdsvis lavt med noen høyere ytterpunkter. Muligens ikke så overraskende for de som har bodd i Trondheim ser en at det regner nesten dobbelt så mange dager enn det har vært opphold, men samtidig har også dette en positiv skjevhets, noe som innebærer at det ikke har vært for høy nedbør når det har regnet. Videre kan det se ut som at de andre variablene er relativt normalfordelt, i tillegg til noen bimodale fordelinger.

```
[ ]: weather_power.Max_Temp.rolling(14).mean().plot(figsize=(20,10))
weather_power.Min_Temp.rolling(14).mean().plot()
weather_power.Mean_Temp.rolling(14).mean().plot()
plt.axhline(y=np.nanmean(weather_power.Mean_Temp), color='red', linestyle='--',
            linewidth=3, label=('Totalt gjennomsnitt: ' + str(round(np.
            nanmean(weather_power.Mean_Temp),2))+ '°C'))
plt.legend()
```

```
plt.ylabel("Temperatur i °C")
plt.show()
```



```
[ ]: weather_power.describe().transpose()
```

	count	mean	std	min	\
Month	2557.0	6.522487	3.449499	1.00000	
Year	2557.0	2018.000000	2.000391	2015.00000	
Max_Temp	2557.0	9.636840	7.696523	-12.00000	
Min_Temp	2557.0	3.079937	6.585082	-20.10000	
Mean_Temp	2557.0	6.358389	6.993643	-15.20000	
Max_Wind	2557.0	5.286117	2.227827	0.00000	
Rain	2557.0	2.453109	4.634751	0.00000	
Is_Rain	2557.0	0.688698	0.463117	0.00000	
UVA	2557.0	461903.896402	419709.313856	3683.10000	
UVB	2557.0	6067.404159	6915.824940	-20.35200	
Max_Pressure	2557.0	1013.730583	11.677193	962.90000	
Min_Pressure	2557.0	1006.403989	13.358389	944.00000	
Traffic	2557.0	16559.910442	3992.502751	5369.00000	
Max_Power	2557.0	3136.133750	547.332617	1854.00000	
Min_Power	2557.0	2710.414548	477.499468	1620.00000	
Pollution_PM25	2557.0	5.303824	4.427758	-0.39186	
Pollution_PM10	2557.0	12.392685	8.732117	1.09158	
		25%	50%	75%	max
Month		4.000000	7.000000	10.000000	1.200000e+01
Year		2016.000000	2018.000000	2020.000000	2.021000e+03
Max_Temp		4.000000	8.800000	15.200000	3.210000e+01

Min_Temp	-1.100000	2.800000	8.300000	2.030000e+01
Mean_Temp	1.450000	5.700000	11.800000	2.580000e+01
Max_Wind	3.800000	4.800000	6.400000	1.710000e+01
Rain	0.000000	0.400000	2.900000	5.100000e+01
Is_Rain	0.000000	1.000000	1.000000	1.000000e+00
UVA	77954.000000	347970.000000	772490.000000	1.575500e+06
UVB	303.030000	3126.500000	10185.000000	2.947300e+04
Max_Pressure	1007.000000	1014.300000	1021.300000	1.052600e+03
Min_Pressure	998.800000	1007.300000	1015.000000	1.046500e+03
Traffic	12838.000000	18472.000000	19838.000000	2.349700e+04
Max_Power	2680.000000	3130.000000	3578.000000	4.618000e+03
Min_Power	2357.000000	2671.000000	3109.000000	3.965000e+03
Pollution_PM25	2.613866	3.984580	6.313460	4.197452e+01
Pollution_PM10	6.593005	10.157918	15.754641	9.125000e+01

```
[ ]: days_above_limit_num = weather_power.Pollution_PM10.where(weather_power.
    ↪Pollution_PM10>50).sort_values(ascending=False).count()
days_above_limit = weather_power.Pollution_PM10.where(weather_power.
    ↪Pollution_PM10>50).sort_values(ascending=False).head(days_above_limit_num)
print("Antall overskridelser av døgnmiddel på 50 µg/m³:", days_above_limit_num, ↪
    ↪"\nDette inntraff følgende dager med døgnmiddel:\n", days_above_limit)
```

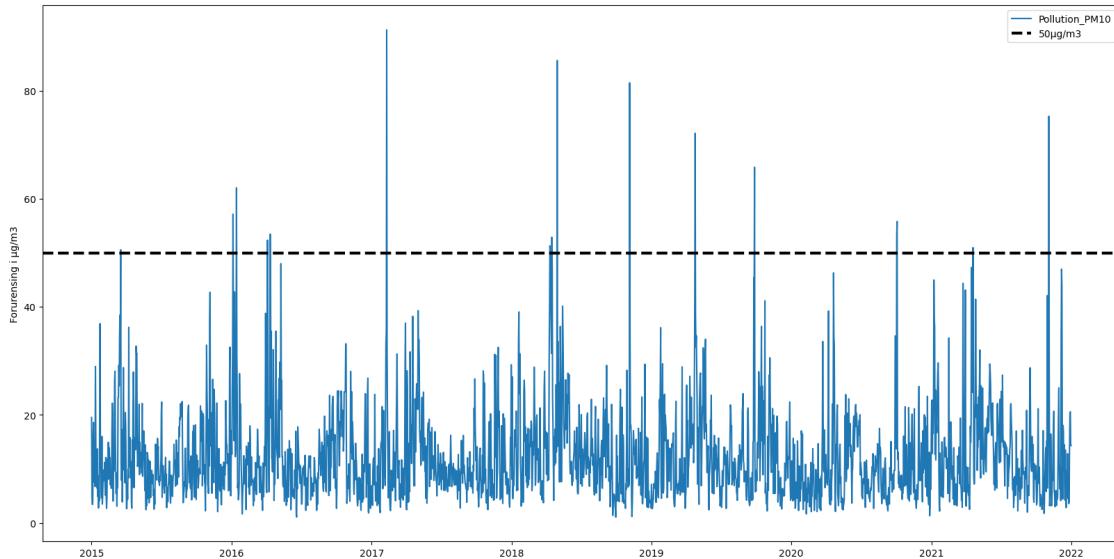
Antall overskridelser av døgnmiddel på 50 µg/m³: 16

Dette inntraff følgende dager med døgnmiddel:

```
2017-02-09      91.250005
2018-04-30      85.571233
2018-11-05      81.436170
2021-11-03      75.269673
2019-04-25      72.108251
2019-09-27      65.834444
2016-01-14      62.022458
2016-01-05      57.150666
2020-10-03      55.797206
2016-04-11      53.456952
2020-10-02      53.242974
2018-04-16      52.881548
2016-04-04      52.330207
2018-04-11      51.291155
2021-04-19      50.960609
2015-03-18      50.566502
```

Name: Pollution_PM10, dtype: float64

```
[ ]: weather_power.Pollution_PM10.plot(figsize=(20,10))
plt.axhline(y=50, color='black', linestyle='--', linewidth=3, label='50µg/m³')
plt.ylabel("Forurensing i µg/m³")
plt.legend()
plt.show()
```



Grenseverdien på 50 $\mu\text{g}/\text{m}^3$ ble oversteget totalt 16 ganger i løpet av seks år, dette er langt under grensen på 25 ganger per år. Likevel er det relevant å se hvilke dager dette er og se hvilke faktorer som gjorde at graden av svevestøv var høyere enn normalt.

```
[ ]: diff_Temp = weather_power.Max_Temp - weather_power.Min_Temp
```

```
[ ]: diff_Temp.describe()
```

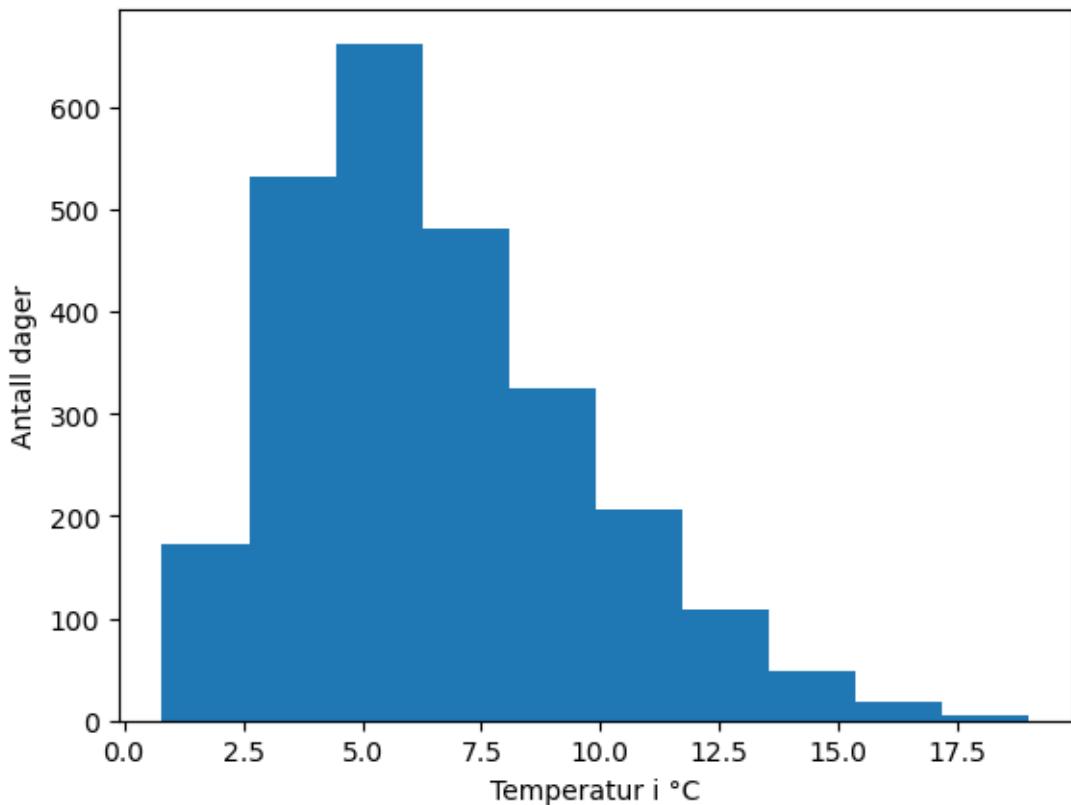
```
[ ]: count      2557.000000
mean        6.556903
std         3.091183
min         0.800000
25%        4.200000
50%        5.900000
75%        8.400000
max        19.000000
dtype: float64
```

```
[ ]: diff_Temp.sort_values(ascending=False).head(15)
```

2018-05-08	19.0
2018-05-27	17.7
2020-06-12	17.4
2019-04-23	17.3
2021-06-01	17.3
2016-06-20	17.1
2021-07-26	17.1
2018-07-27	17.0
2018-05-30	16.8

```
2019-04-22    16.6
2018-07-05    16.2
2020-07-26    16.0
2021-08-06    16.0
2019-04-20    15.9
2021-07-04    15.9
dtype: float64
```

```
[ ]: plt.hist(diff_Temp)
plt.ylabel("Antall dager")
plt.xlabel("Temperatur i °C")
plt.show()
```



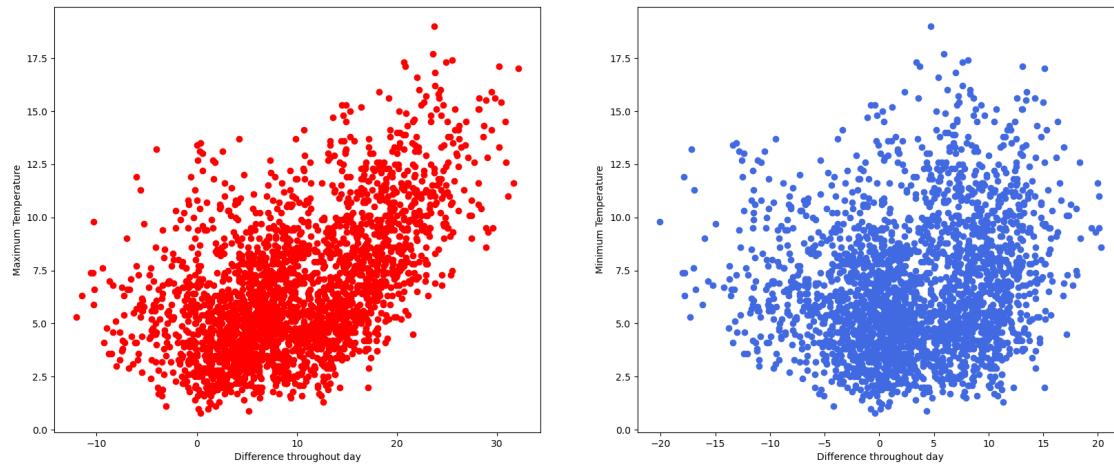
Ser her at forskjellen mellom minimum og maksimumstemperatur gjennom fra 2015 til 2021 lå mellom 0,8 og 19 grader med et snitt på 6,55. Ser også her at histogrammet er relativt skjevfordelt med en positiv skjevhets. Dette innebærer at det vanligvis vil være temperaturforskjell under snittet, men at det inntreffer dager hvor det er større forskjeller. De største temperaturforskjellene inntraff i mai 2018.

```
[ ]: fig, (ax1, ax2) = plt.subplots(1,2, figsize=(20,8))
ax1.scatter(weather_power.Max_Temp, diff_Temp, color="red")
```

```

ax2.scatter(weather_power.Min_Temp, diff_Temp, color="royalblue")
ax1.set_ylabel('Maximum Temperature')
ax1.set_xlabel('Difference throughout day')
ax2.set_ylabel('Minimum Temperature')
ax2.set_xlabel('Difference throughout day')
plt.show()

```



Ser med disse grafene at korrelasjonen mellom temperaturforskjell og temperatur er høyest de dagene den maksimale temperaturen er høy. Dette tyder på at det er den høye dagstemperaturen som drar forskjellen opp. Kan også se dette med en enkel korrelasjonsmatrise.

```

[ ]: max_corr = np.corrcoef([weather_power.Max_Temp, diff_Temp])
min_corr = np.corrcoef([weather_power.Min_Temp, diff_Temp])

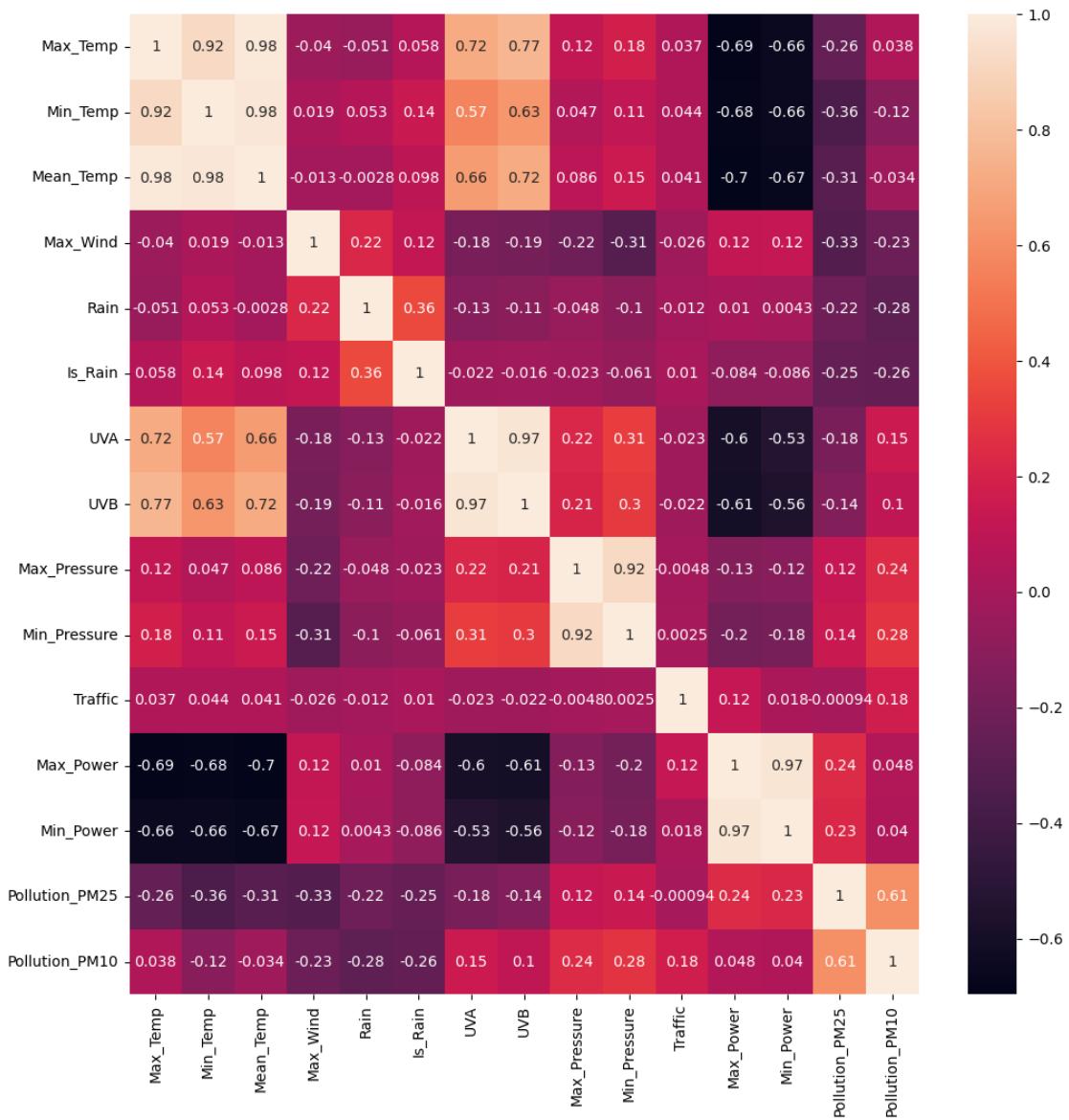
print("Korrelasjon mellom maksimumstemperatur og temperaturforskjell er:", ↪
      max_corr[1,0],
      "til forskjell fra korrelasjonen mellom temperaturforskjell og ↪
      minimumstemperaturen som er:", ↪
      min_corr[1,0])

```

Korrelasjon mellom maksimumstemperatur og temperaturforskjell er:
0.5344076235487902 til forskjell fra korrelasjonen mellom temperaturforskjell og
minimumstemperaturen som er: 0.15518375266509873

4 Enkel lineær regresjon

```
[ ]: plt.figure(figsize=(12,12))
sns.heatmap(weather_power.drop(["Year", "Month"], axis=1),
            corr_numeric_only=True, annot=True)
plt.show()
```



Starter med en korrelasjonsmatrise for å se hvilke variabler som har størst påvirkning på hverandre. Velger å se på vind og svevestøv med størrelse 10 .

```
[ ]: linmod1 = smf.ols(formula = "Pollution_PM25 ~ Max_Wind", data=weather_power).
      fit()
```

```
[ ]: linmod1.summary()
```

```
[ ]: <class 'statsmodels.iolib.summary.Summary'>
"""
                OLS Regression Results
=====
Dep. Variable:      Pollution_PM25    R-squared:                 0.110
Model:                          OLS    Adj. R-squared:            0.110
Method:                     Least Squares    F-statistic:              317.0
Date:             Wed, 21 Dec 2022    Prob (F-statistic):       6.24e-67
Time:                  21:58:40    Log-Likelihood:           -7282.8
No. Observations:          2557    AIC:                      1.457e+04
Df Residuals:             2555    BIC:                      1.458e+04
Df Model:                           1
Covariance Type:        nonrobust
=====
              coef      std err      t      P>|t|      [0.025      0.975]
-----
Intercept    8.7940     0.213     41.338     0.000      8.377      9.211
Max_Wind   -0.6603     0.037    -17.803     0.000     -0.733     -0.588
=====
Omnibus:            1608.213    Durbin-Watson:            0.732
Prob(Omnibus):        0.000    Jarque-Bera (JB):       20852.558
Skew:                   2.800    Prob(JB):                  0.00
Kurtosis:                  15.820    Cond. No.                 15.2
=====
```

Notes:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

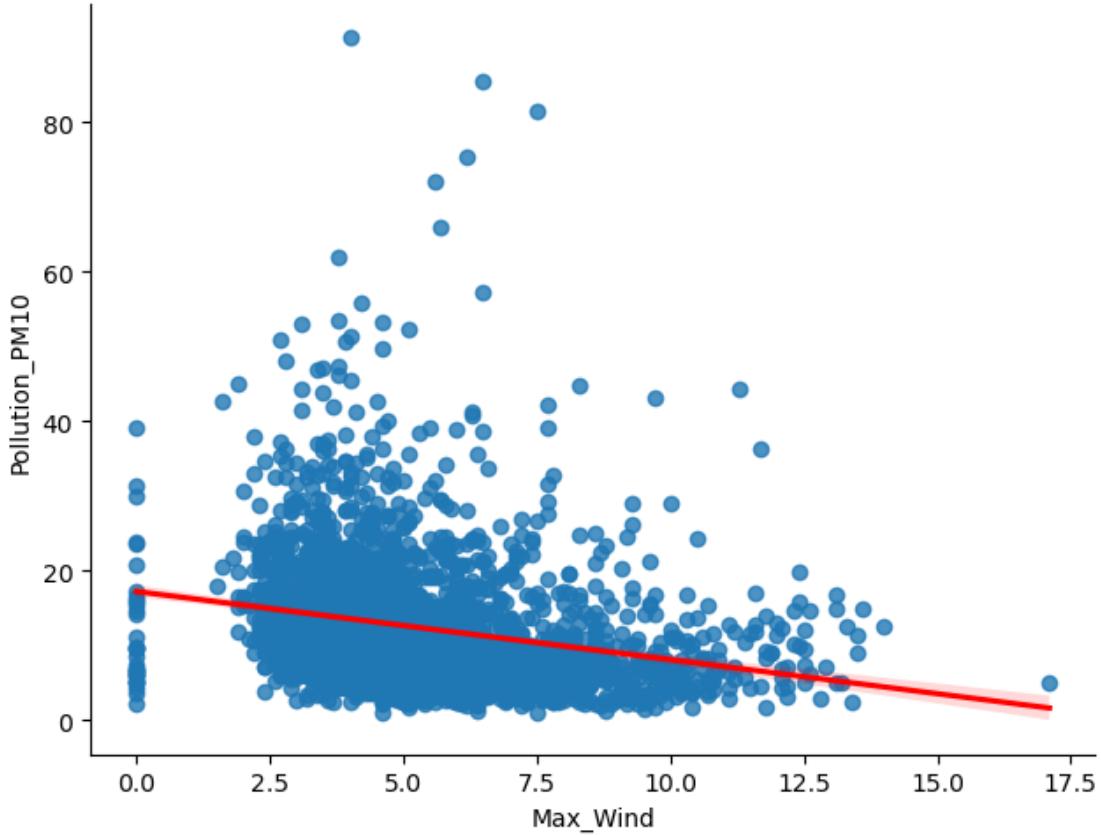
```
[ ]: print("Gjennomsnittlig nivå av svevestøv:", weather_power.Pollution_PM25.mean())
```

Gjennomsnittlig nivå av svevestøv: 5.303824480872468

Fra korrelasjonsmatrisen så det ut til at vindstyrken påvirket svevestøvet på $2.5 \mu\text{m}$ negativt, dette sjekket jeg videre gjennom en lineær regresjon ved bruk av minste kvadrats metode. Her ser en at p-verdien er 0, noe som betyr at vi kan avvise nullhypotesen om at nivået av svevestøv ikke blir påvirket av vindstyrken. Samtidig kan en se under coef at en sekundmeter kraftigere vind fører til -0,66 mikrogram per kubikkmeter. Til sammenligning er det gjennomsnittlige nivået på $5,3 \mu\text{m}/\text{m}^3$. Samtidig er r^2 kun på 0,11, dette kan bety at forklaringsgraden er relativt lav, men ikke nødvendigvis at modellen er dårlig.

```
[ ]: sns.lmplot(x="Max_Wind", y="Pollution_PM10", data=weather_power, height=5, aspect=1.3, line_kws={'color': 'red'})
```

```
plt.show()
```



Visuell fremstilling av korrelasjonen mellom vind og svevestøv fremstilt ved hjelp av Seaborn-pakken. Ser at det er en negativ trend slik som modellen ovenfor viste, samtidig kan en se at det er endel uteliggere både på nivået av svevestøv og vind som kan påvirke modellen.

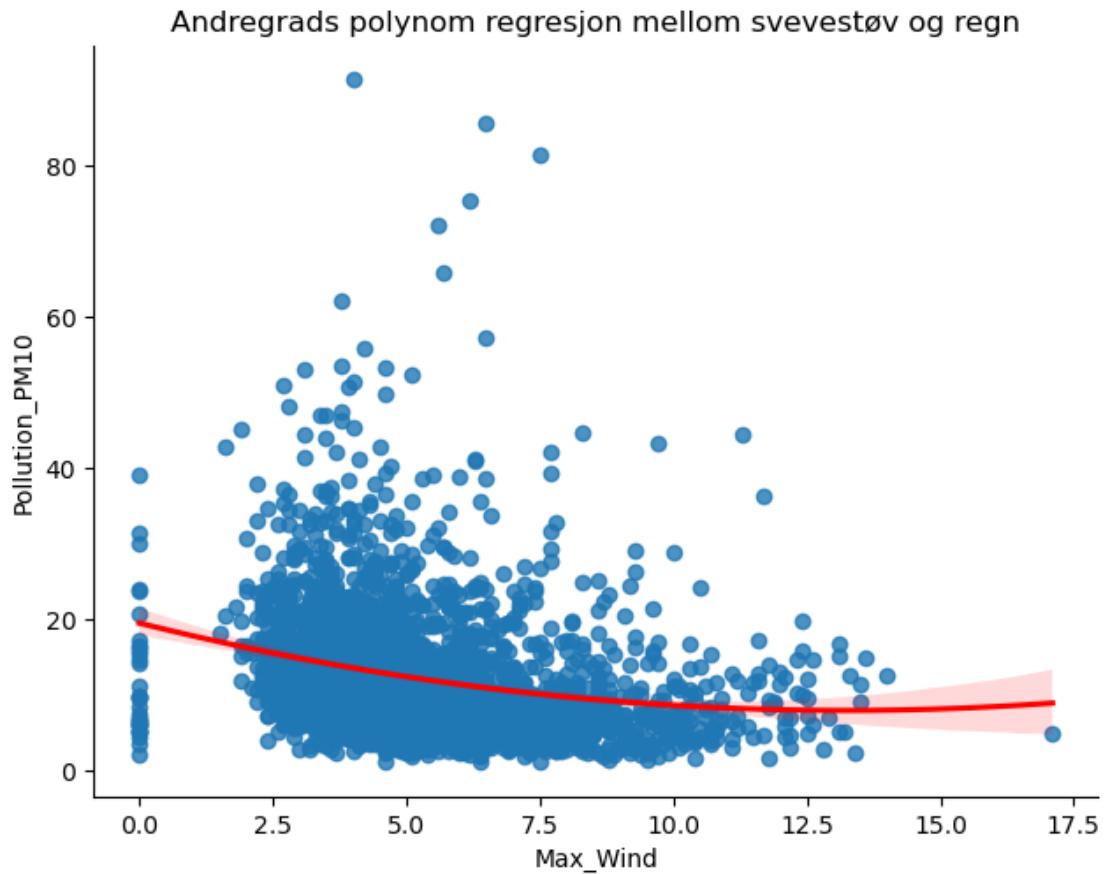
Det er også mulig å kjøre en polynom regresjon som er gjort i figurene under. Dette for å bedre vise den faktiske korrelasjonen mellom den avhngige og uavhengige variabelen. Dette vises spesielt godt når man sammenligner måned som avhengig og temperatur som uavhengig variabel. En polynom regresjonslinje vil forklare sammenhengen i en mye større grad enn en rett linje. Utifra dette vil det være mulig å se om en enkel lineær regresjon er nok til å forklare sammenhengen.

```
[ ]: sns.lmplot(x="Max_Wind", y="Pollution_PM10", data=weather_power, height=5, aspect=1.3, order = 2, line_kws={'color': 'red'})
plt.title("Andregrads polynom regresjon mellom svevestøv og regn")

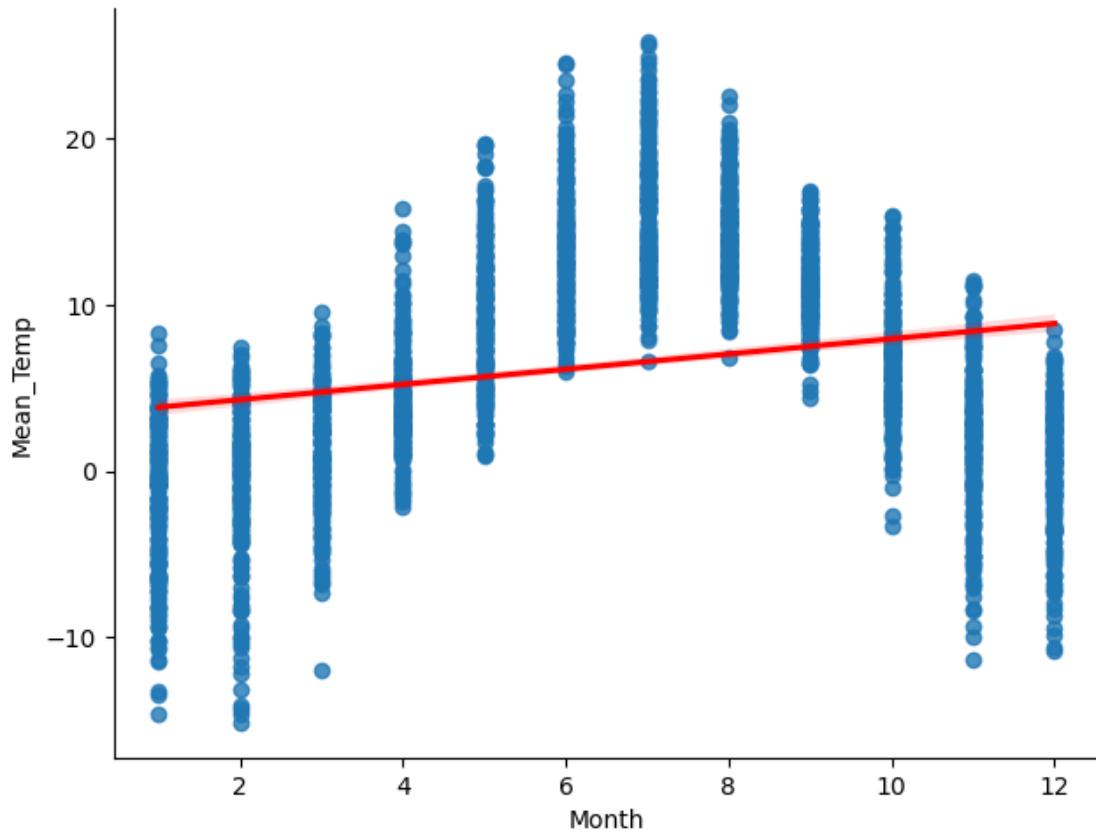
sns.lmplot(x="Month", y="Mean_Temp", data=weather_power, height=5, aspect=1.3, order = 1, line_kws={'color': 'red'})
plt.title("Førstegrads lineær regresjon mellom temperatur og måned")

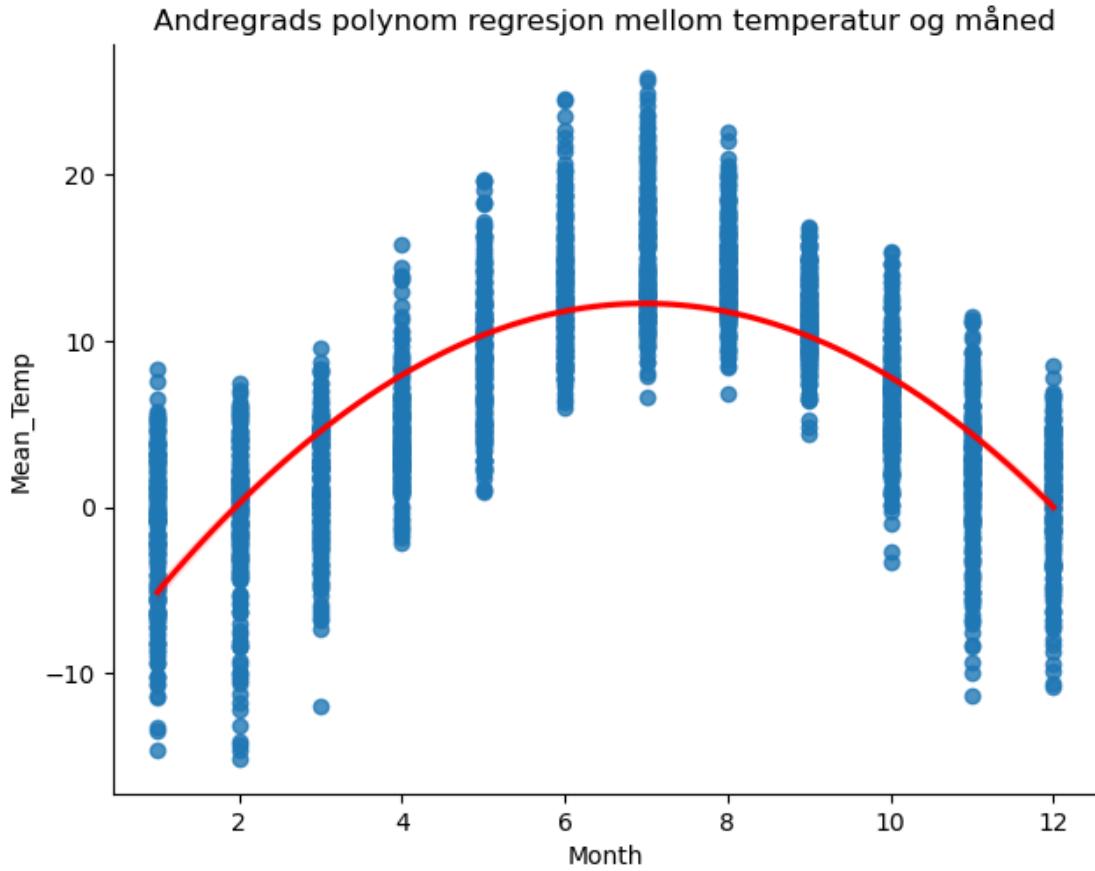
sns.lmplot(x="Month", y="Mean_Temp", data=weather_power, height=5, aspect=1.3, order = 2, line_kws={'color': 'red'})
```

```
plt.title("Andregrads polynom regresjon mellom temperatur og måned")  
plt.show()
```



Førstegrads lineær regresjon mellom temperatur og måned



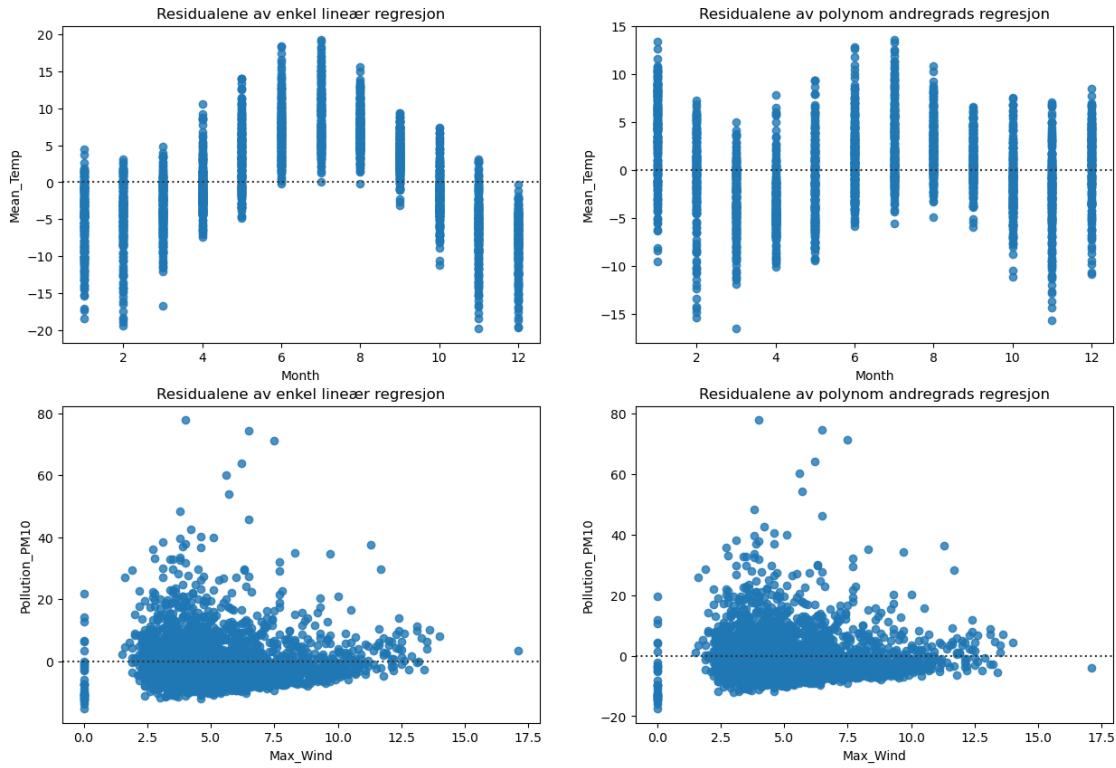


I tillegg kan man se på residualene fra den linære regresjonen, disse burde være tilfeldig spredt rundt $y = 0$, om de ikke er det kan det bety at det er et underliggende mønster som regresjonsmodellen ikke fanger opp. Gjør en residualanalyse av både temperatur mot måned, samt vindstyrke mot nivå av svevestøv.

```
[ ]: fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2,2, figsize=(15,10))
sns.residplot(x="Month", y="Mean_Temp", data=weather_power, order=1, ax=ax1)
ax1.set_title("Residualene av enkel lineær regresjon")
sns.residplot(x="Month", y="Mean_Temp", data=weather_power, order=2, ax=ax2)
ax2.set_title("Residualene av polynom andregrads regresjon")

sns.residplot(x="Max_Wind", y="Pollution_PM10", data=weather_power, order=1, ax=ax3)
ax3.set_title("Residualene av enkel lineær regresjon")
sns.residplot(x="Max_Wind", y="Pollution_PM10", data=weather_power, order=2, ax=ax4)
ax4.set_title("Residualene av polynom andregrads regresjon")

plt.show()
```



Som en kan se på de øverste figurene så er det et klart mønster i residualene i figuren til venstre, dette motvirkes ved å gjøre regresjonsmodellen polynom i figuren øverst til høyre, men fremdeles ser det ut til å være et underliggende mønster som ikke forklares med en slik regresjonsmodell.

I figurene nederst er det vanskelig å se noe klart mønster i residualene, og det virker ikke ha skjedd en forbedring ved å gjøre regresjonsmodellen polynom.

5 Multippel regresjon

Setter opp en multippel regresjonsanalyse hvor jeg kontrollerer for forskjellige variabler.

```
[ ]: regvari = "Pollution_PM10 ~ Mean_Temp + Max_Wind + Traffic"
multireg1 = smf.ols(regvari, data=weather_power).fit()
multireg1.summary()
```

```
[ ]: <class 'statsmodels.iolib.summary.Summary'>
"""
=====
              OLS Regression Results
=====
Dep. Variable:      Pollution_PM10    R-squared:                 0.085
Model:                          OLS    Adj. R-squared:            0.084
Method:                     Least Squares    F-statistic:             79.05
Date:                Wed, 21 Dec 2022    Prob (F-statistic):       6.78e-49
Time:                      21:58:41    Log-Likelihood:          -9055.2
No. Observations:        2557    AIC:                  1.812e+04
Df Residuals:           2553    BIC:                  1.814e+04
Df Model:                           3
Covariance Type:      nonrobust
=====
            coef      std err          t      P>|t|      [0.025      0.975]
-----
Intercept    11.2568      0.826     13.630      0.000      9.637     12.876
Mean_Temp   -0.0550      0.024     -2.326      0.020     -0.101     -0.009
Max_Wind    -0.8966      0.074    -12.078      0.000     -1.042     -0.751
Traffic     0.0004  4.15e-05      9.069      0.000      0.000      0.000
=====
Omnibus:            1485.181    Durbin-Watson:            0.915
Prob(Omnibus):      0.000    Jarque-Bera (JB):        18767.171
Skew:                   2.515    Prob(JB):                  0.00
Kurtosis:                  15.282    Cond. No.            8.52e+04
=====
```

Notes:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
- [2] The condition number is large, 8.52e+04. This might indicate that there are strong multicollinearity or other numerical problems.

"""

```
[ ]:
```

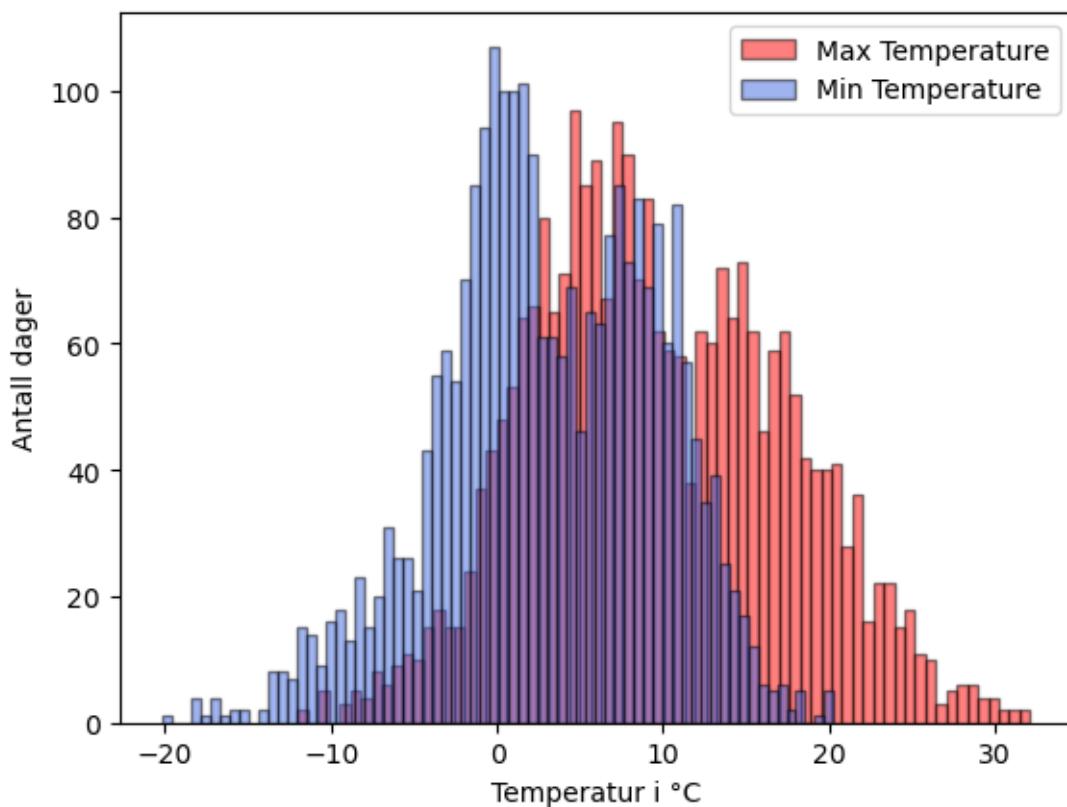
```
[ ]:
```

```
[ ]:
```

6 ”Fake”-data Simulasjon

6.1 Normalfordeling

```
[ ]: plt.hist(weather_power.Max_Temp, label='Max Temperature', alpha=0.5, □  
    ↵color="red", bins=70, edgecolor='black')  
plt.hist(weather_power.Min_Temp, label='Min Temperature', alpha=0.5, □  
    ↵color="royalblue", bins=70, edgecolor='black')  
plt.ylabel("Antall dager")  
plt.xlabel("Temperatur i °C")  
plt.legend()  
  
plt.show()
```



Ved å legge inn både daglige maksimum og minimums verdier i et histogram kan en tenke at fordelingen ikke er normalfordelt, men heller har to dittinkte topper. Dette kan vises testes ved å kjøre en simulasjon. Velger å se på minimumstemperaturen.

```
[ ]: min_temp_mean = weather_power.Min_Temp.mean()  
min_temp_std = weather_power.Min_Temp.std()  
  
print(min_temp_mean, min_temp_std)
```

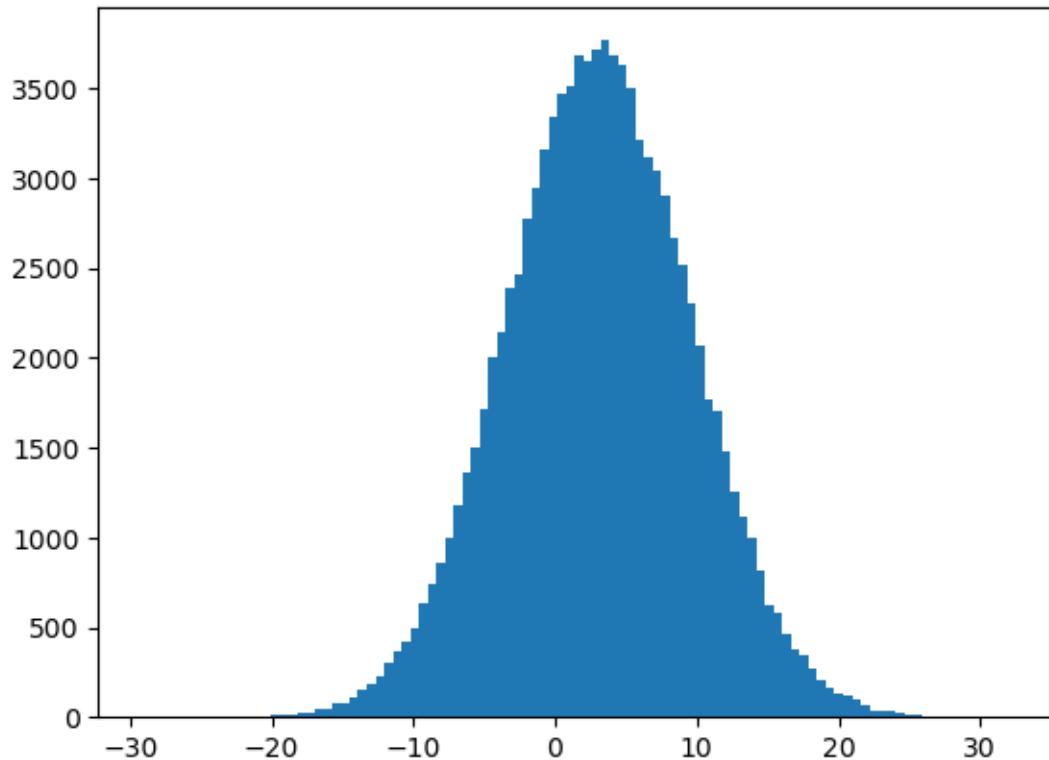
```
3.079937426671881 6.585082396464526
```

```
[ ]: n=100000
min_temp_sim = np.random.normal(min_temp_mean, min_temp_std, n)

normEst = np.sum(min_temp_sim>10)/n

realProp = np.sum(weather_power.Min_Temp>10)/weather_power.Min_Temp.count()
```

```
[ ]: plt.hist(min_temp_sim, bins=100)
plt.show()
```



```
[ ]: normEst
```

```
[ ]: 0.14614
```

```
[ ]: realProp
```

```
[ ]: 0.16112631990614001
```

```
[ ]: realProp/normEst
```

```
[ ]: 1.1025476933498017
```

Ser med simulasjonen at antallat tilfeller med minimumstemperatur ved simulasjonen (14,67%) ikke er langt unna det reelle antallet tilfeller (16,1%). Derimot kan en se på histogrammet av simulasjonen at den ser klart annerledes ut enn histogrammet av den reelle temperaturen. Velger å kjøre en Shapiro-Wilk test for å vise at den ikke er normalfordelt.

```
[ ]: stat, p = sts.shapiro(weather_power.Max_Temp)
print('Resultat av testen =', stat, 'p=', round(p,5))

a = 0.05
if p > a:
    print('Minimumstemperaturen er normalfordelt.')
else:
    print('Minimumstemperaturen er ikke normalfordelt.)
```

```
Resultat av testen = 0.9926794767379761 p= 0.0
Minimumstemperaturen er ikke normalfordelt.
```

6.2 ****Bootstrapping

Bootstrapping er en metode for å

```
[ ]: boot = weather_power.Max_Temp.sample(100, random_state=67, replace=True)

[ ]: N=10000

boots = []

for i in range(N):
    boot = weather_power.Max_Temp.sample(n=100, replace=True)
    boots.append(boot.mean())

boots = pd.Series(boots)
```

```
[ ]: print("Estimert standardavvik:", boots.std())
print("Faktisk standardsavvik:", weather_power.Max_Temp.std())
```

```
Estimert standardavvik: 0.7584987673189126
Faktisk standardsavvik: 7.696523046551285
```

7 Tidsrekke-regresjon

For å kjøre en tidsrekke-regresjon, i dette tilfellet en ARIMA-modellen, må dataene helst være stasjonære, de må ikke være påvirket av tidsaspektet. For å sjekke dette kan en kjøre en adfuller-test hvor nullhypotesen er at dataen er ikke-stasjonær.

```
[ ]: dfest1 = adfuller(weather_power.Pollution_PM10, autolag="AIC")
dfest2 = adfuller(weather_power.Mean_Temp, autolag="AIC")
dfest3 = adfuller(weather_power.Traffic, autolag="AIC")
dfest4 = adfuller(weather_power.Max_Wind, autolag="AIC")

print(dfest1)
print(dfest2)
print(dfest3)
print(dfest4)

if dfest1[1] < 0.05:
    print("Pollution_10 er stasjonært.")
else:
    print("Pollution_PM10 er ikke-stasjonært.")
```

```
(-9.867237263978975, 4.083234098768378e-17, 12, 2544, {'1%': -3.432923078043983,
'5%': -2.8626767808549087, '10%': -2.56737515100891}, 17167.13140788882)
(-3.134388277644208, 0.024110773543148684, 24, 2532, {'1%': -3.4329352851231945,
'5%': -2.862682171580326, '10%': -2.5673780210899224}, 11241.815476538892)
(-6.766177758107493, 2.7135681095727288e-09, 27, 2529, {'1%':
-3.432938355012086, '5%': -2.8626835272597217, '10%': -2.567378742868999},
44971.15757234195)
(-7.615737660374975, 2.194401282578652e-11, 23, 2533, {'1%': -3.432934263444484,
'5%': -2.8626817204012203, '10%': -2.5673777808771043}, 10497.99996055101)
Pollution_10 er stasjonært.
```

Sjekker p-verdi under 0.05 og ser av testen at alle av de foregående variablene er stasjonære, dette innebærer at de kan brukes i en ARIMA-modell uten justering. Velger videre å se på Pollution_PM10.

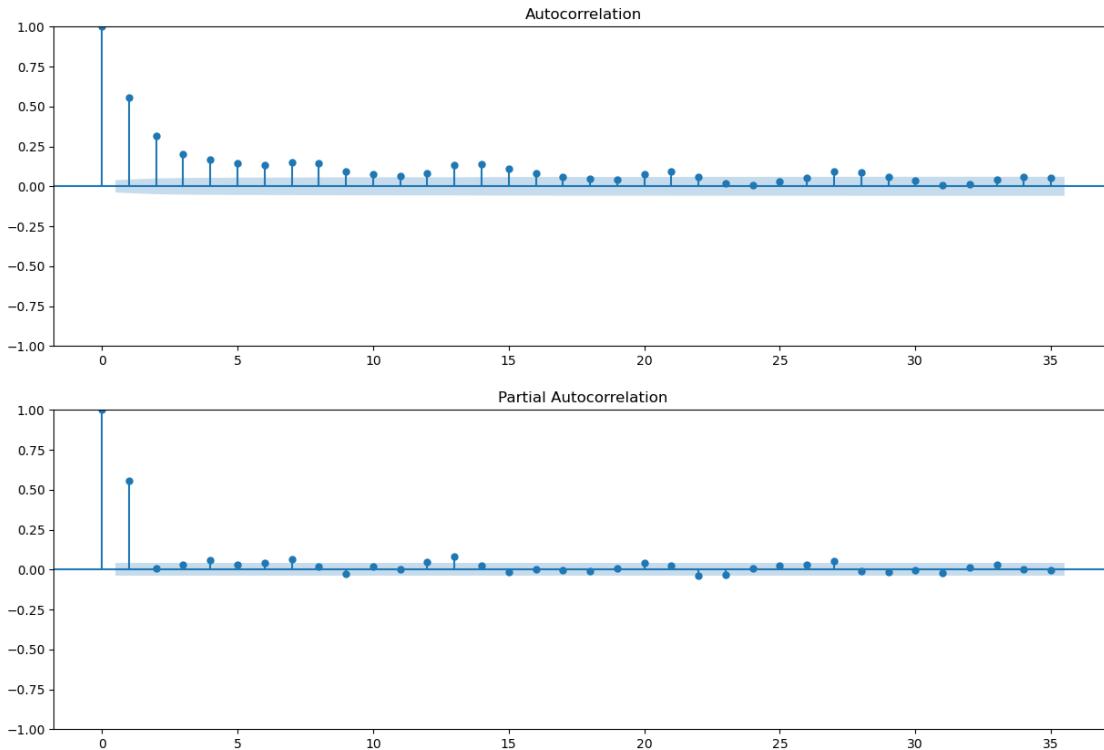
```
[ ]: from statsmodels.tsa.stattools import acf, pacf
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

[ ]: fig, ax = plt.subplots(2,1, figsize=(15,10))
plot_acf(weather_power.Pollution_PM10, ax=ax[0])
plot_pacf(weather_power.Pollution_PM10, ax=ax[1])

plt.show()
```

```
/Users/vrognaas/opt/anaconda3/envs/MET2010/lib/python3.10/site-
packages/statsmodels/graphics/tsaplots.py:348: FutureWarning: The default method
'yw' can produce PACF values outside of the [-1,1] interval. After 0.13, the
default will change to unadjusted Yule-Walker ('ywm'). You can use this method
```

```
now by setting method='ywmm'.  
warnings.warn(
```



Ser av autokorrelasjonsplottet at det ser ut til å være en positiv autoregressiv korrelsjon på 1 AR. Dette innebærer at en dags forurensing påvirker den neste dagen. Dette brukes under i ARIMA-modellen.

```
[ ]: from statsmodels.tsa.arima.model import ARIMA
```

```
[ ]: arimaModPM10 = ARIMA(weather_power.Pollution_PM10, order=(1,0,0)).fit()
```

```
/Users/vrognaas/opt/anaconda3/envs/MET2010/lib/python3.10/site-  
packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency  
information was provided, so inferred frequency D will be used.  
    self._init_dates(dates, freq)  
/Users/vrognaas/opt/anaconda3/envs/MET2010/lib/python3.10/site-  
packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency  
information was provided, so inferred frequency D will be used.  
    self._init_dates(dates, freq)  
/Users/vrognaas/opt/anaconda3/envs/MET2010/lib/python3.10/site-  
packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency  
information was provided, so inferred frequency D will be used.  
    self._init_dates(dates, freq)
```

```
[ ]: arimaModPM10.summary()

[ ]: <class 'statsmodels.iolib.summary.Summary'>
"""

                    SARIMAX Results
=====

Dep. Variable:      Pollution_PM10    No. Observations:                  2557
Model:              ARIMA(1, 0, 0)    Log Likelihood:                 -8696.254
Date:                Wed, 21 Dec 2022   AIC:                            17398.509
Time:                      21:58:43     BIC:                            17416.049
Sample:             01-01-2015   HQIC:                           17404.869
                   - 12-31-2021

Covariance Type:            opg
=====

            coef    std err          z      P>|z|      [0.025      0.975]
-----
const      12.3927     0.437     28.363      0.000     11.536     13.249
ar.L1       0.5558     0.010     58.214      0.000      0.537     0.575
sigma2      52.6687     0.616     85.525      0.000     51.462     53.876
=====

===
Ljung-Box (L1) (Q):           0.04    Jarque-Bera (JB):
24481.73
Prob(Q):                     0.83    Prob(JB):
0.00
Heteroskedasticity (H):       0.99    Skew:
2.17
Prob(H) (two-sided):         0.91    Kurtosis:
17.53
=====
===
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
"""


```

Lager en enkel ARIMA-modell med en AR, dette kan brukes for skape en “baseline” hvor en senere kan teste modeller mot ved å endre MA-variabelen og se om forklaringsgraden til modellen forbedrer seg.

```
[ ]: arimaModPM10_1 = ARIMA(weather_power.Pollution_PM10, order=(1,0,1)).fit()
```

```
/Users/vrognaas/opt/anaconda3/envs/MET2010/lib/python3.10/site-
packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency
information was provided, so inferred frequency D will be used.
    self._init_dates(dates, freq)
/Users/vrognaas/opt/anaconda3/envs/MET2010/lib/python3.10/site-
```

```
packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency information was provided, so inferred frequency D will be used.
```

```
    self._init_dates(dates, freq)
/Users/vrognaas/opt/anaconda3/envs/MET2010/lib/python3.10/site-
packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency information was provided, so inferred frequency D will be used.
    self._init_dates(dates, freq)
```

```
[ ]: arimaModPM10_1.summary()
```

```
[ ]: <class 'statsmodels.iolib.summary.Summary'>
```

```
"""
```

SARIMAX Results

```
=====
Dep. Variable:          Pollution_PM10    No. Observations:                  2557
Model:                 ARIMA(1, 0, 1)    Log Likelihood:                    -8696.176
Date:                 Wed, 21 Dec 2022   AIC:                            17400.351
Time:                     21:58:43      BIC:                            17423.737
Sample:                01-01-2015   HQIC:                           17408.832
                           - 12-31-2021
Covariance Type:            opg
=====
```

	coef	std err	z	P> z	[0.025	0.975]
const	12.3928	0.441	28.107	0.000	11.529	13.257
ar.L1	0.5662	0.022	25.434	0.000	0.523	0.610
ma.L1	-0.0151	0.026	-0.578	0.563	-0.066	0.036
sigma2	52.6603	0.618	85.243	0.000	51.449	53.871

```
====
```

```
Ljung-Box (L1) (Q):             0.00    Jarque-Bera (JB):
```

```
24432.15
```

```
Prob(Q):                      0.99    Prob(JB):
```

```
0.00
```

```
Heteroskedasticity (H):        0.99    Skew:
```

```
2.17
```

```
Prob(H) (two-sided):           0.91    Kurtosis:
```

```
17.51
```

```
====
```

```
Warnings:
```

```
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

```
"""
```

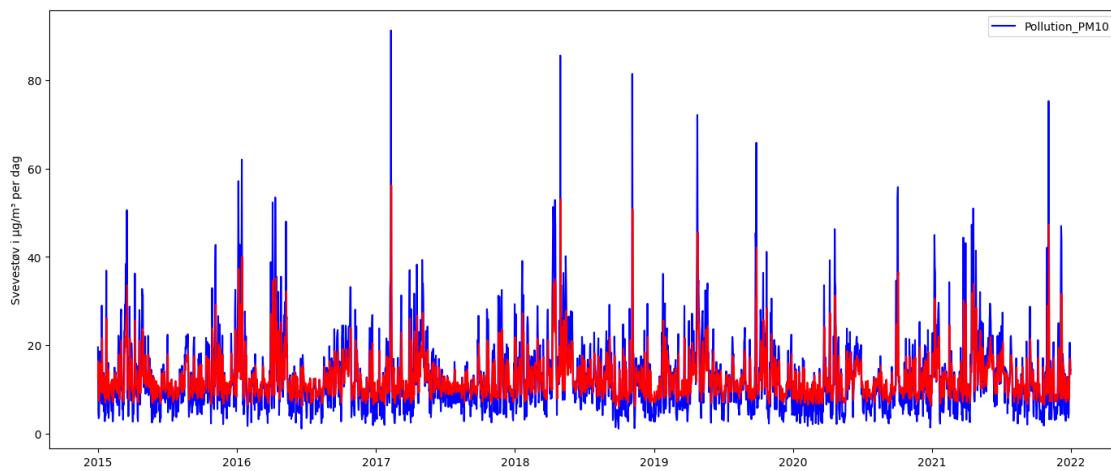
```
[ ]: arimaModPM10_1.aic < arimaModPM10.aic and arimaModPM10_1.bic < arimaModPM10.bic
```

```
[ ]: False
```

Ser at ved å legge til en moving average øker både AIC og BIC, dette vil tilsi et informasjonstap i modellen og lavere forklaringsevne. Det vil derfor være bedre å benytte seg av den første modellen.

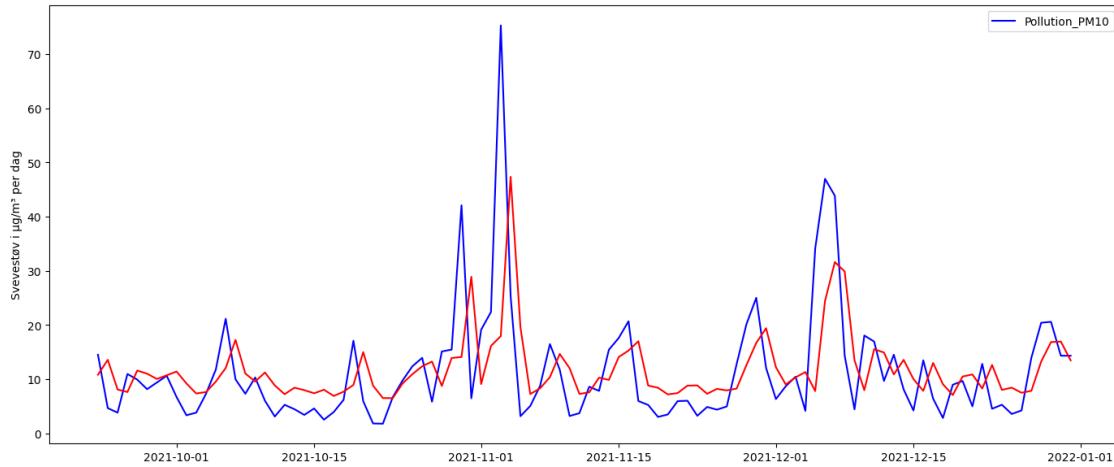
```
[ ]: fitted = arimaModPM10.fittedvalues
```

```
[ ]: fig, ax = plt.subplots(figsize=(17,7))
weather_power.Pollution_PM10.plot(ax=ax, color='b')
fitted.plot(ax=ax, color='r')
plt.ylabel("Svevestøv i  $\mu\text{g}/\text{m}^3$  per dag")
plt.legend()
plt.show()
```



Ser at ARIMA-modellen klarer fange tidsserien relativt bra, men den treffer ikke helt på de aller høyeste dagene, ser også at den konstant ligger litt høyere enn de laveste nivåene av svevestøv.

```
[ ]: fig, ax = plt.subplots(figsize=(17,7))
weather_power.Pollution_PM10.tail(100).plot(ax=ax, color='b')
fitted.tail(100).plot(ax=ax, color='r')
plt.ylabel("Svevestøv i  $\mu\text{g}/\text{m}^3$  per dag")
plt.legend()
plt.show()
```

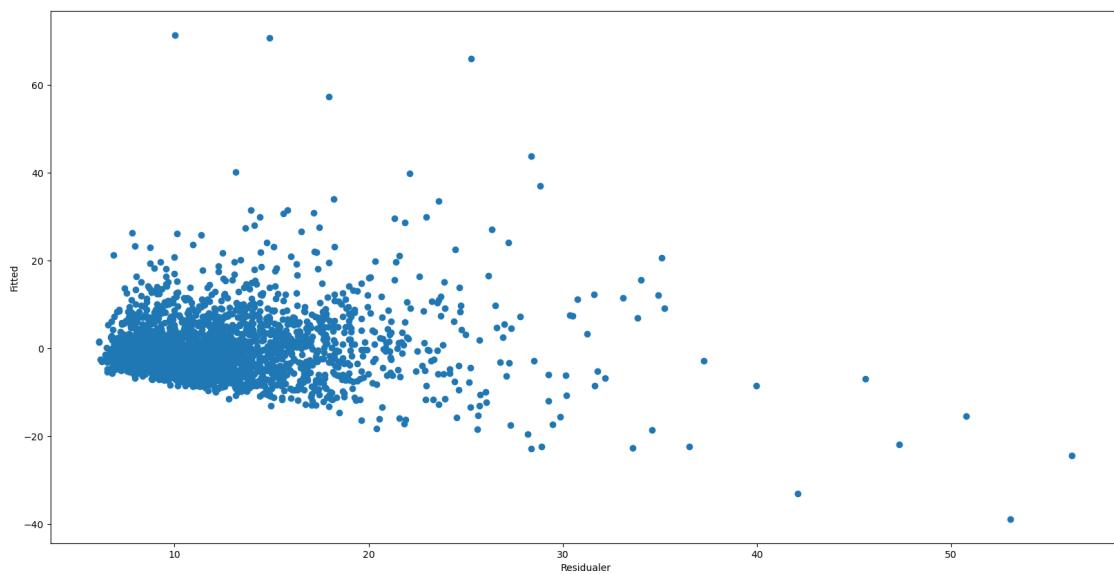


Kan også se på de siste hundre dagene for å sjekke hvor bra modellen treffer:

Videre kan en også se på residualene for å sjekke om det er et mønster som ligger igjen, eller om modellen har fanget opp det meste.

```
[ ]: resid_arima = arimaModPM10.resid
```

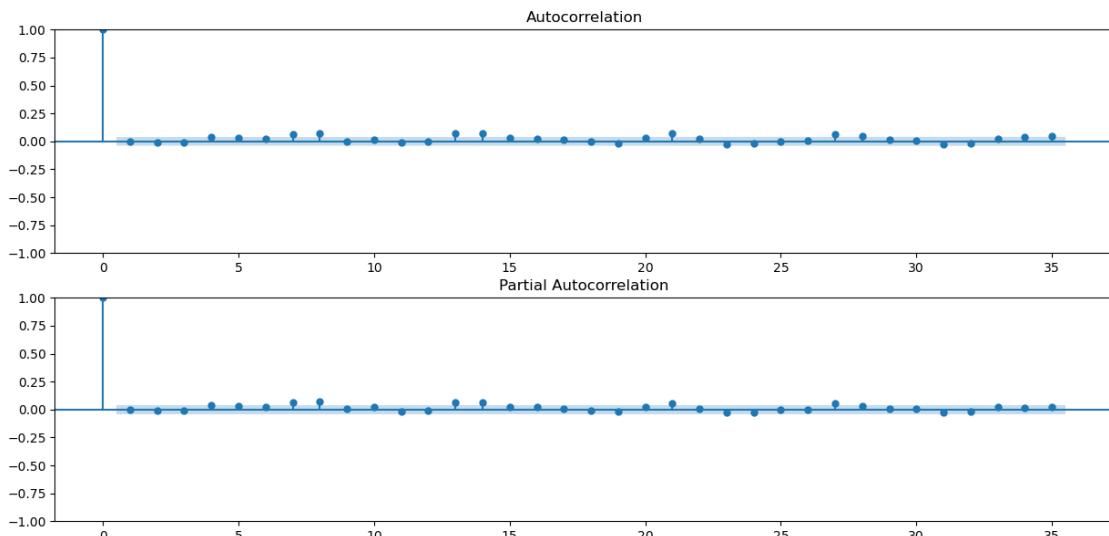
```
[ ]: fig, ax = plt.subplots(figsize=(20,10))
ax.scatter(x=fitted, y=resid_arima)
plt.ylabel('Fitted')
plt.xlabel('Residualer')
plt.show()
```



Ser av dette plottet at det ikke er noen klar trend i residualene, de ligger også nokså samlet rundt null, med unntak av noen uteliggere. Dette vil si at modellen har fanget opp det meste relativt bra.

```
[ ]: fig, ax = plt.subplots(2,1, figsize=(15,7))
plot_acf(resid_arima, ax=ax[0])
plot_pacf(resid_arima, ax=ax[1])
plt.show()
```

```
/Users/vrognaas/opt/anaconda3/envs/MET2010/lib/python3.10/site-
packages/statsmodels/graphics/tsaplots.py:348: FutureWarning: The default method
'yw' can produce PACF values outside of the [-1,1] interval. After 0.13, the
default will change to unadjusted Yule-Walker ('ywm'). You can use this method
now by setting method='ywm'.
warnings.warn(
```



Ser også av autokorrelasjonen av residualene at det ikke er mye autokorrelasjon, dette gir også en pekepinn på at modellen fanger opp de tendensene i tidsserien. Kan derfor prøve å forutsi de neste 30 dagene ved hjelp av ARIMA.

```
[ ]: forecast_arima = arimaModPM10.get_forecast(steps=30).summary_frame()
[ ]: forecast_arima = forecast_arima.reset_index()
forecast_arima = forecast_arima.rename(columns={"index" : "date"})
[ ]: forecast_arima
[ ]: 
```

	Pollution_PM10	date	mean	mean_se	mean_ci_lower	mean_ci_upper
0		2022-01-01	13.473111	7.257323	-0.750981	27.697202
1		2022-01-02	12.993240	8.303091	-3.280520	29.267000

```

2          2022-01-03  12.726507  8.600519   -4.130200  29.583215
3          2022-01-04  12.578246  8.690354   -4.454536  29.611028
4          2022-01-05  12.495836  8.717923   -4.590979  29.582651
5          2022-01-06  12.450029  8.726423   -4.653446  29.553504
6          2022-01-07  12.424567  8.729047   -4.684051  29.533186
7          2022-01-08  12.410415  8.729858   -4.699793  29.520622
8          2022-01-09  12.402548  8.730109   -4.708150  29.513247
9          2022-01-10  12.398176  8.730186   -4.712674  29.509026
10         2022-01-11  12.395745  8.730210   -4.715152  29.506642
11         2022-01-12  12.394394  8.730217   -4.716517  29.505306
12         2022-01-13  12.393643  8.730220   -4.717273  29.504559
13         2022-01-14  12.393226  8.730220   -4.717691  29.504143
14         2022-01-15  12.392994  8.730220   -4.717924  29.503911
15         2022-01-16  12.392865  8.730220   -4.718053  29.503783
16         2022-01-17  12.392793  8.730221   -4.718125  29.503711
17         2022-01-18  12.392753  8.730221   -4.718164  29.503671
18         2022-01-19  12.392731  8.730221   -4.718187  29.503649
19         2022-01-20  12.392719  8.730221   -4.718199  29.503637
20         2022-01-21  12.392712  8.730221   -4.718206  29.503630
21         2022-01-22  12.392708  8.730221   -4.718210  29.503626
22         2022-01-23  12.392706  8.730221   -4.718212  29.503624
23         2022-01-24  12.392705  8.730221   -4.718213  29.503623
24         2022-01-25  12.392704  8.730221   -4.718213  29.503622
25         2022-01-26  12.392704  8.730221   -4.718214  29.503622
26         2022-01-27  12.392704  8.730221   -4.718214  29.503622
27         2022-01-28  12.392704  8.730221   -4.718214  29.503621
28         2022-01-29  12.392704  8.730221   -4.718214  29.503621
29         2022-01-30  12.392704  8.730221   -4.718214  29.503621

```

```
[ ]: forecast_arima.info()
```

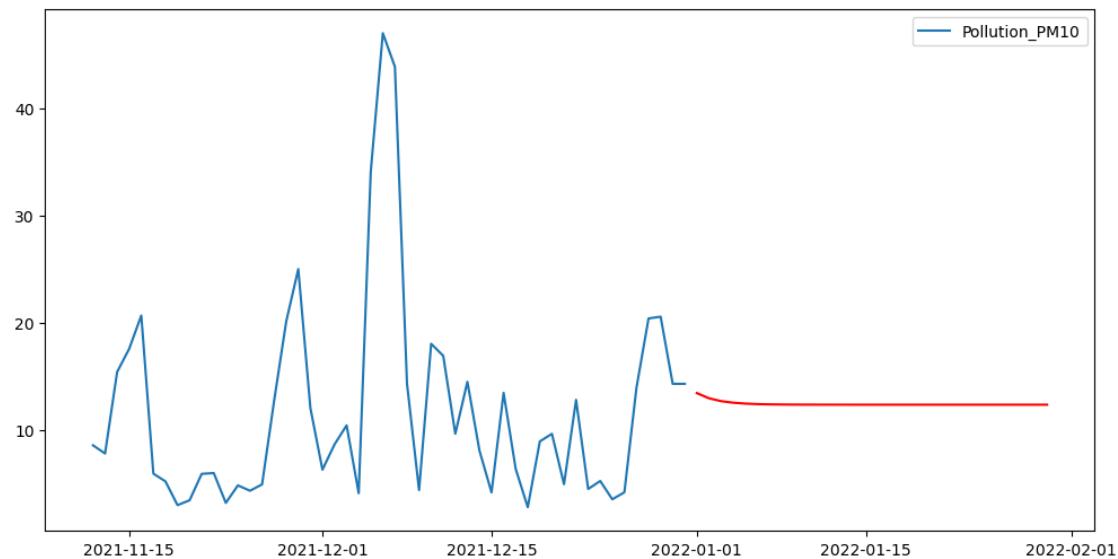
```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   date        30 non-null    datetime64[ns]
 1   mean         30 non-null    float64 
 2   mean_se      30 non-null    float64 
 3   mean_ci_lower 30 non-null    float64 
 4   mean_ci_upper 30 non-null    float64 
dtypes: datetime64[ns](1), float64(4)
memory usage: 1.3 KB

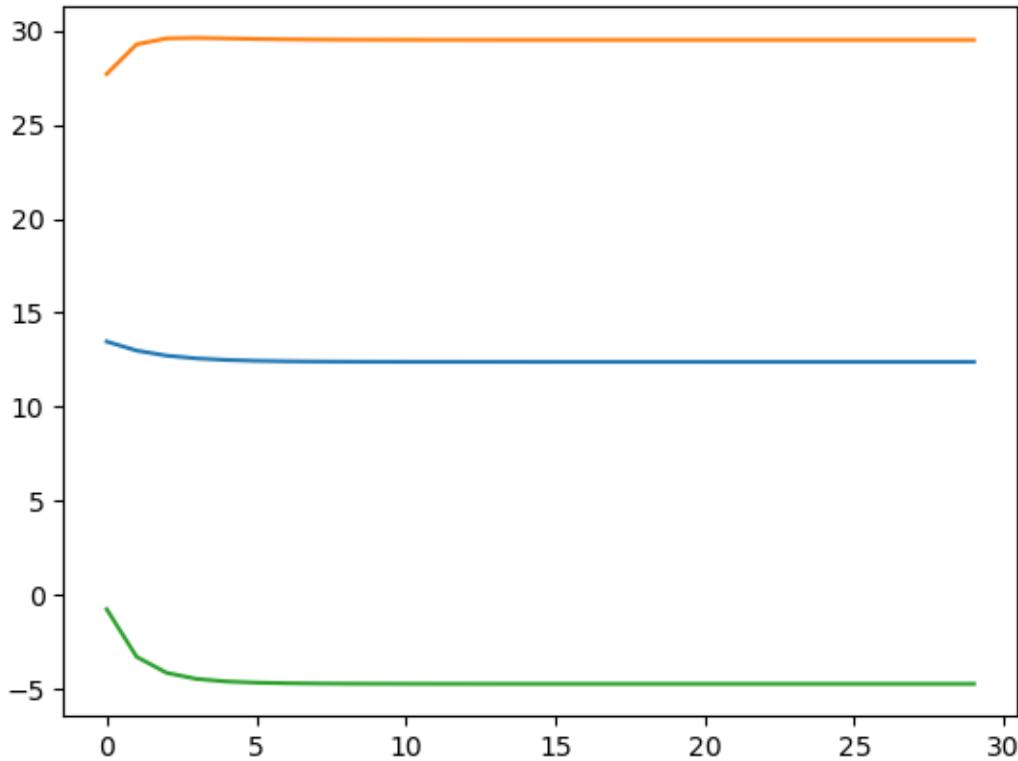
```

```
[ ]: fig, ax = plt.subplots(figsize=(12,6))
ax.plot(forecast_arima["date"], forecast_arima["mean"], color="red")
weather_power.Pollution_PM10.tail(50).plot(ax=ax)
```

```
plt.legend()  
plt.show()
```



```
[ ]: plt.plot(forecast_arima["mean"])  
plt.plot(forecast_arima.mean_ci_upper)  
plt.plot(forecast_arima.mean_ci_lower)  
plt.show()
```



Ser at ARIMA-modellen legge opp til relativt flat utvikling, men veldig stor spredning, den treffer dårlig på dagene fremover. Kan derfor prøve å lege til en ekogen, en forklarende variabel og se om dette forbedrer modellen noe. Velger å legge til trafikk da dette kan tenkes påvirker nivået av svevestøv.

```
[ ]: arimaModPM10_2 = ARIMA(weather_power.Pollution_PM10, weather_power.Traffic,
    ↴order=(1,1,0)).fit()
```

```
/Users/vrognaas/opt/anaconda3/envs/MET2010/lib/python3.10/site-
packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency
information was provided, so inferred frequency D will be used.
    self._init_dates(dates, freq)
/Users/vrognaas/opt/anaconda3/envs/MET2010/lib/python3.10/site-
packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency
information was provided, so inferred frequency D will be used.
    self._init_dates(dates, freq)
/Users/vrognaas/opt/anaconda3/envs/MET2010/lib/python3.10/site-
packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency
information was provided, so inferred frequency D will be used.
    self._init_dates(dates, freq)
```

```
[ ]: arimaModPM10_2.summary()
```

```
[ ]: <class 'statsmodels.iolib.summary.Summary'>
"""
                SARIMAX Results
=====
Dep. Variable:      Pollution_PM10    No. Observations:                  2557
Model:              ARIMA(1, 1, 0)   Log Likelihood:                   -8870.939
Date:              Wed, 21 Dec 2022 AIC:                            17747.879
Time:                      21:58:44   BIC:                            17765.417
Sample:             01-01-2015   HQIC:                           17754.239
                           - 12-31-2021
Covariance Type:            opg
=====
                                         coef      std err       z     P>|z|      [0.025      0.975]
-----
Traffic          0.0005  4.14e-05    11.541      0.000      0.000      0.001
ar.L1           -0.2388     0.009    -27.399      0.000     -0.256     -0.222
sigma2          60.4362     0.731    82.706      0.000    59.004    61.868
=====
===
Ljung-Box (L1) (Q):                 6.20    Jarque-Bera (JB):
12298.18
Prob(Q):                         0.01    Prob(JB):
0.00
Heteroskedasticity (H):            0.99    Skew:
0.50
Prob(H) (two-sided):               0.88    Kurtosis:
13.70
=====
===
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
"""

```

Har laget modellen, og ser nå på hva den predikterer basert på et høyt nivå av trafikk fremover, og et lavt nivå.

```
[ ]: weather_power.Traffic.mean()
```

```
[ ]: 16559.91044192413
```

```
[ ]: weather_power.Traffic.sort_values().tail(10)
```

```
[ ]: 2021-08-12      22276
2021-05-10      22372
2017-12-21      22378
2021-08-13      22387
```

```

2021-05-06    22400
2021-08-24    22427
2017-01-12    22550
2021-05-11    22794
2021-05-12    23354
2017-01-26    23497
Name: Traffic, dtype: int64

```

```
[ ]: traffic_high = np.linspace(17000, 25000, 30)
traffic_low = np.linspace(15000, 7000, 30)
```

```
[ ]: forecast_highTraffic = arimaModPM10_2.get_forecast(exog=traffic_high, steps=30).
    ↪summary_frame()
forecast_lowTraffic = arimaModPM10_2.get_forecast(exog=traffic_low, steps=30).
    ↪summary_frame()
```

```
[ ]: forecast_highTraffic
```

Pollution_PM10	mean	mean_se	mean_ci_lower	mean_ci_upper
2022-01-01	17.684363	7.774070	2.447465	32.921261
2022-01-02	17.912466	9.770096	-1.236571	37.061503
2022-01-03	18.021440	11.658326	-4.828459	40.871339
2022-01-04	18.158862	13.230375	-7.772196	44.089920
2022-01-05	18.289491	14.645339	-10.414846	46.993828
2022-01-06	18.421742	15.932774	-12.805923	49.649406
2022-01-07	18.553605	17.124215	-15.009240	52.116450
2022-01-08	18.685561	18.237869	-17.060005	54.431127
2022-01-09	18.817495	19.287354	-18.985024	56.620014
2022-01-10	18.949434	20.282601	-20.803734	58.702602
2022-01-11	19.081372	21.231248	-22.531109	60.693853
2022-01-12	19.213310	22.139283	-24.178886	62.605507
2022-01-13	19.345248	23.011514	-25.756491	64.446987
2022-01-14	19.477186	23.851871	-27.271621	66.225994
2022-01-15	19.609124	24.663611	-28.730664	67.948913
2022-01-16	19.741063	25.449472	-30.138987	69.621112
2022-01-17	19.873001	26.211784	-31.501151	71.247152
2022-01-18	20.004939	26.952543	-32.821074	72.830951
2022-01-19	20.136877	27.673480	-34.102148	74.375901
2022-01-20	20.268815	28.376107	-35.347333	75.884963
2022-01-21	20.400753	29.061752	-36.559233	77.360740
2022-01-22	20.532691	29.731589	-37.740152	78.805534
2022-01-23	20.664629	30.386663	-38.892137	80.221395
2022-01-24	20.796567	31.027911	-40.017021	81.610156
2022-01-25	20.928506	31.656172	-41.116451	82.973462
2022-01-26	21.060444	32.272204	-42.191914	84.312802
2022-01-27	21.192382	32.876696	-43.244758	85.629521
2022-01-28	21.324320	33.470272	-44.276207	86.924847

```

2022-01-29      21.456258  34.053502     -45.287380    88.199896
2022-01-30      21.588196  34.626911     -46.279303    89.455695

```

```
[ ]: forecast_lowTraffic
```

```

[ ]: Pollution_PM10      mean      mean_se  mean_ci_lower  mean_ci_upper
2022-01-01      16.727812  7.774070      1.490914    31.964710
2022-01-02      16.692038  9.770096     -2.456998    35.841075
2022-01-03      16.537137  11.658326     -6.312762    39.387036
2022-01-04      16.410682  13.230375     -9.520376    42.341740
2022-01-05      16.277435  14.645339    -12.426902    44.981771
2022-01-06      16.145809  15.932774    -15.081855    47.373473
2022-01-07      16.013796  17.124215    -17.549049    49.576641
2022-01-08      15.881876  18.237869    -19.863690    51.627442
2022-01-09      15.749934  19.287354    -22.052585    53.552452
2022-01-10      15.617997  20.282601    -24.135172    55.371165
2022-01-11      15.486058  21.231248    -26.126423    57.098539
2022-01-12      15.354120  22.139283    -28.038076    58.746317
2022-01-13      15.222182  23.011514    -29.879557    60.323921
2022-01-14      15.090244  23.851871    -31.658564    61.839052
2022-01-15      14.958306  24.663611    -33.381483    63.298094
2022-01-16      14.826368  25.449472    -35.053682    64.706417
2022-01-17      14.694430  26.211784    -36.679722    66.068581
2022-01-18      14.562491  26.952543    -38.263521    67.388504
2022-01-19      14.430553  27.673480    -39.808471    68.669578
2022-01-20      14.298615  28.376107    -41.317533    69.914763
2022-01-21      14.166677  29.061752    -42.793309    71.126664
2022-01-22      14.034739  29.731589    -44.238104    72.307582
2022-01-23      13.902801  30.386663    -45.653965    73.459567
2022-01-24      13.770863  31.027911    -47.042725    74.584451
2022-01-25      13.638925  31.656172    -48.406032    75.683881
2022-01-26      13.506987  32.272204    -49.745372    76.759345
2022-01-27      13.375048  32.876696    -51.062091    77.812188
2022-01-28      13.243110  33.470272    -52.357416    78.843637
2022-01-29      13.111172  34.053502    -53.632466    79.854811
2022-01-30      12.979234  34.626911    -54.888265    80.846733

```

```

[ ]: forecast_highTraffic["date"] = pd.date_range(start='1/1/2022', freq='D',
                                                periods=30)
forecast_lowTraffic["date"] = pd.date_range(start='1/1/2022', freq='D',
                                             periods=30)

```

```
[ ]: forecast_lowTraffic.head(5)
```

```

[ ]: Pollution_PM10      mean      mean_se  mean_ci_lower  mean_ci_upper      date
2022-01-01      16.727812  7.774070      1.490914    31.964710  2022-01-01
2022-01-02      16.692038  9.770096     -2.456998    35.841075  2022-01-02

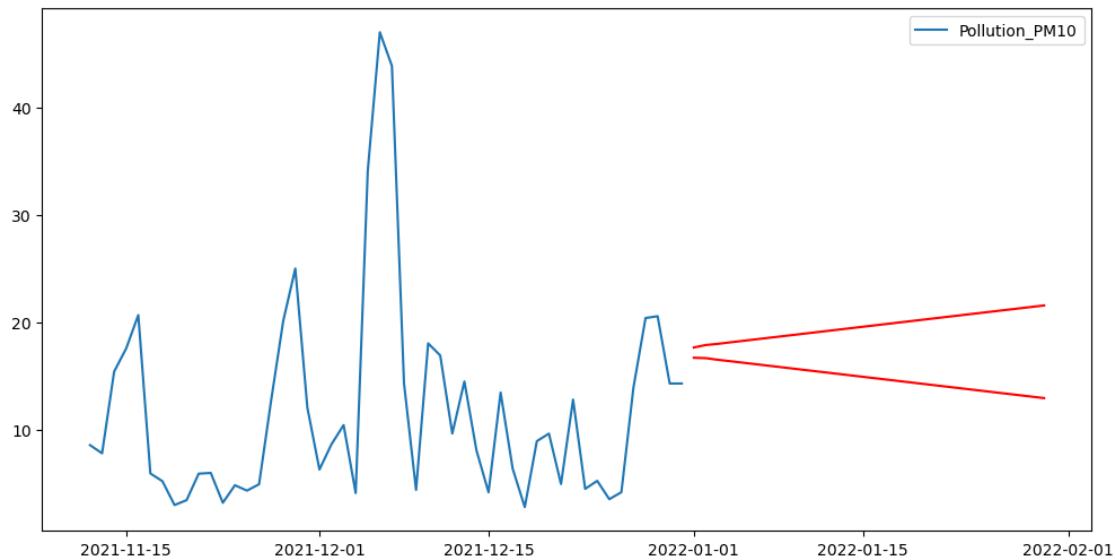
```

```

2022-01-03      16.537137  11.658326      -6.312762      39.387036 2022-01-03
2022-01-04      16.410682  13.230375      -9.520376      42.341740 2022-01-04
2022-01-05      16.277435  14.645339     -12.426902      44.981771 2022-01-05

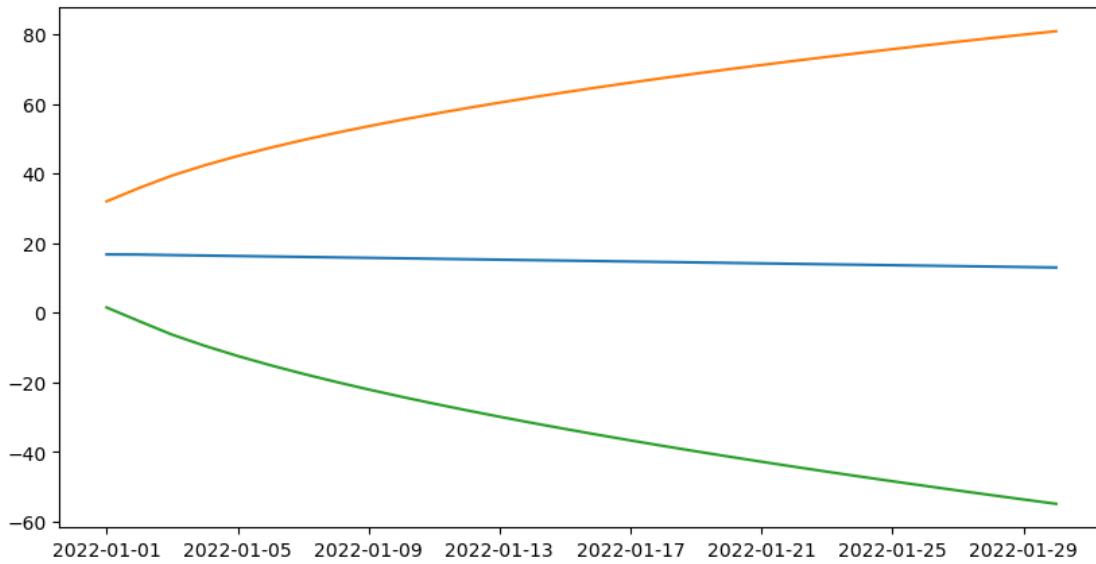
```

```
[ ]: fig, ax = plt.subplots(figsize=(12,6))
ax.plot(forecast_lowTraffic["date"], forecast_lowTraffic["mean"], color="red")
ax.plot(forecast_highTraffic["date"], forecast_highTraffic["mean"], color="red")
weather_power.Pollution_PM10.tail(50).plot(ax=ax)
plt.legend()
plt.show()
```

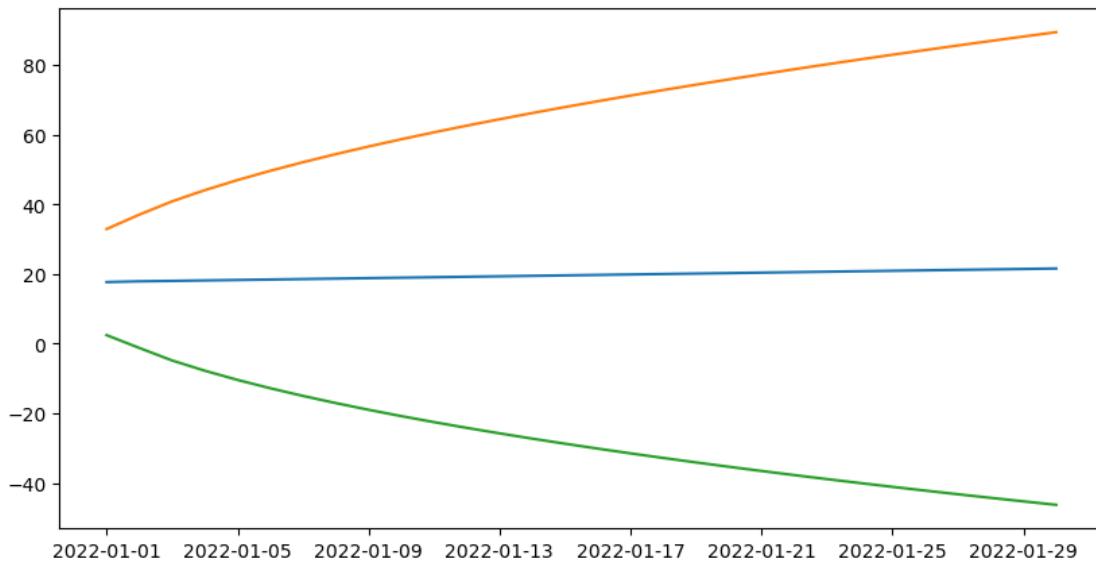


Etter å ha lagt inn dataen i plottet ser man at trafikk påvirker svevestøvet til en viss grad, men fremdeles er usikkerheten relativt høy om en ser på konfidensintervallet.

```
[ ]: plt.figure(figsize=(10,5))
plt.plot(forecast_lowTraffic["mean"])
plt.plot(forecast_lowTraffic.mean_ci_upper)
plt.plot(forecast_lowTraffic.mean_ci_lower)
plt.show()
```



```
[ ]: plt.figure(figsize=(10,5))
plt.plot(forecast_highTraffic["mean"])
plt.plot(forecast_highTraffic.mean_ci_upper)
plt.plot(forecast_highTraffic.mean_ci_lower)
plt.show()
```



Ser her klart at den høye usikkerheten fremdeles eksisterer og det er nesten samme konfidensintervall om det er høyt eller lavt nivå av trafikk fremover. ARIMA-modellen klarer ikke med høy sikkerhet å predikere en måned fremover.

8 Diagnostikk og evaluering av modell

9 Diskusjon om identifikasjon og kausalitet

10 Kilder

UV-data hentet fra: https://github.com/uvnrpa/Daily_Doses
Takk til DSA, NILU og UIO som tilbyr dette gratis.

Fakta knyttet til svevestøv hentet fra: <https://www.fhi.no/nettpub/luftkvalitet/temakapitler/svevestov/>

Trafikkdata hentet fra: <https://www.vegvesen.no/trafikkdata/start/eksport>

Værdata hentet fra: <https://seklima.met.no/>

Strømforbruk hentet fra: <https://transparency.entsoe.eu/dashboard/show>

11 Word Count

```
[ ]: import json

with open('Prosjektoppgave.ipynb') as json_file:
    data = json.load(json_file)

wordCount = 0
for each in data['cells']:
    cellType = each['cell_type']
    if cellType == "markdown":
        content = each['source']
        for line in content:
            temp = [word for word in line.split() if "#" not in word]
            wordCount = wordCount + len(temp)

print("Antall ord ekskludert kodeblokker:", wordCount)
```

Antall ord ekskludert kodeblokker: 2209

```
[ ]: dummy_months = pd.get_dummies(weather_power.Month, prefix="Month ")
```

```
[ ]:
```