

Prosjektoppgave

November 17, 2022

1 Prosjektoppgave MET2010

1.1 Innholdsfortegnelse

1. Innledning
2. Datainnhenting og strukturering
3. Visualisering og deskriptiv statistikk
4. Simulasjon
- 5.
- 6.
7. Kilder
8. Ordtelling

1.2 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 også kunne predikere når det er høyest andel av svevestøv. 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 . Ifølge FHI så vil begge disse størrelse

1.3 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 pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from scipy.stats import shapiro
```

```
[ ]: 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={"Tid(norsk normaltid)" : "Tid"}, 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("Tid", 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.

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

```
<class 'pandas.core.frame.DataFrame'>
Index: 2557 entries, 2015-01-01 to 2021-12-31
Data columns (total 6 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Navn        2557 non-null   object
1   Stasjon     2557 non-null   object
2   Max_Temp    2557 non-null   float64
3   Min_Temp    2557 non-null   float64
4   Max_Wind    2557 non-null   object
5   Rain        2557 non-null   float64
```

```
dtypes: float64(3), object(3)
memory usage: 139.8+ KB
```

```
[ ]: 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")
```

```
[ ]: 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()
```

```
[ ]:
```

		Navn	Stasjon	Max_Temp	Min_Temp	Max_Wind	Rain	\
2021-12-27	Trondheim	- Voll	SN68860	0.2	-7.3	4.2	1.2	
2021-12-28	Trondheim	- Voll	SN68860	-4.5	-10.1	1.6	0.0	
2021-12-29	Trondheim	- Voll	SN68860	-4.5	-9.7	2.4	0.0	
2021-12-30	Trondheim	- Voll	SN68860	-0.4	-6.5	2.9	0.0	
2021-12-31	Trondheim	- Voll	SN68860	2.4	-1.9	4.8	3.2	

	Is_Rain	UVA	UVB
--	---------	-----	-----

2021-12-27	1	24021.0	57.144
2021-12-28	0	23988.0	46.431
2021-12-29	0	19061.0	32.673
2021-12-30	0	24308.0	30.534
2021-12-31	1	17993.0	19.491

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

```
[ ]: 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()
```

```
[ ]:
      Navn  Max_Temp  Min_Temp  Max_Wind  Rain  Is_Rain  \
2021-12-27  Trondheim - Voll      0.2    -7.3      4.2    1.2      1
2021-12-28  Trondheim - Voll     -4.5   -10.1      1.6    0.0      0
2021-12-29  Trondheim - Voll     -4.5    -9.7      2.4    0.0      0
2021-12-30  Trondheim - Voll     -0.4    -6.5      2.9    0.0      0
2021-12-31  Trondheim - Voll      2.4    -1.9      4.8    3.2      1
```

	UVA	UVB	Max_Pressure	Min_Pressure
2021-12-27	24021.0	57.144	1007.9	1002.2
2021-12-28	23988.0	46.431	1001.6	998.7
2021-12-29	19061.0	32.673	1000.6	996.8
2021-12-30	24308.0	30.534	996.4	987.3
2021-12-31	17993.0	19.491	1009.9	995.1

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

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

```
[ ]: Tid
      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 µg/m³ Day	2121
Dekning.11	2113
Bakke kirke PM10 µg/m³ Day	539
Dekning.12	445
Bakke kirke PM2.5 µg/m³ 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 µg/m³ Day'].fillna(pollution['Torvet PM10 µg/m³ Day'], inplace=True)
print(pollution["Elgeseter PM10 µg/m³ Day"].isna().sum())
pollution['Elgeseter PM10 µg/m³ Day'].fillna(pollution['Bakke kirke PM10 µg/m³ Day'], inplace=True)
print(pollution["Elgeseter PM10 µg/m³ Day"].isna().sum())
pollution['Elgeseter PM10 µg/m³ Day'].fillna(pollution['Åsveien skole PM10 µg/m³ Day'], inplace=True)
print(pollution["Elgeseter PM10 µg/m³ Day"].isna().sum())
pollution["Elgeseter PM10 µg/m³ Day"] = (pollution["Elgeseter PM10 µg/m³ Day"].ffill()+pollution["Elgeseter PM10 µg/m³ Day"].bfill())/2
```

```
print(pollution["Elgeseter PM10 µg/m³ Day"].isna().sum())
pollution["Elgeseter PM10 µg/m³ Day"].ffill(inplace=True)
print(pollution["Elgeseter PM10 µg/m³ Day"].isna().sum())
```

27
13
6
1
0

```
[ ]: pollution['Elgeseter PM2.5 µg/m³ Day'].fillna(pollution['Torvet PM2.5 µg/m³ Day'], inplace=True)
pollution['Elgeseter PM2.5 µg/m³ Day'].fillna(pollution['Bakke kirke PM2.5 µg/m³ Day'], inplace=True)
pollution['Elgeseter PM2.5 µg/m³ Day'].fillna(pollution['Åsveien skole PM2.5 µg/m³ Day'], inplace=True)
pollution["Elgeseter PM2.5 µg/m³ Day"] = (pollution["Elgeseter PM2.5 µg/m³ Day"].ffill()+pollution["Elgeseter PM2.5 µg/m³ Day"].bfill())/2
pollution["Elgeseter PM2.5 µg/m³ Day"].ffill(inplace=True)
print(pollution["Elgeseter PM2.5 µg/m³ Day"].isna().sum())
```

0

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

```
[ ]: 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)
```

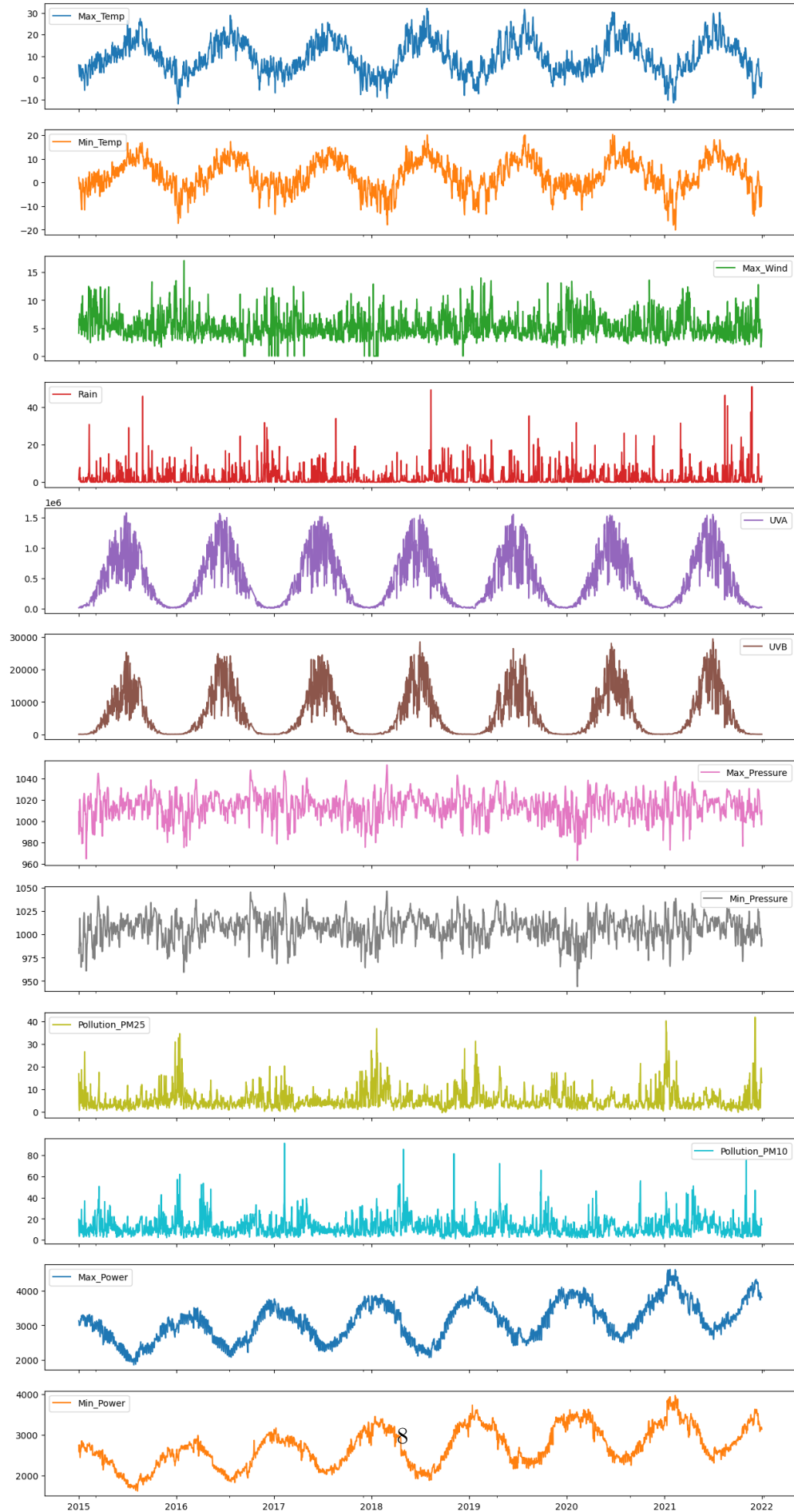
```
[ ]: weather_power = weather.copy()
```

```
[ ]: weather_power["Max_Power"] = daily_power["Max Total Load [MW] - BZN|N03"].  
      ↪to_numpy()  
weather_power["Min_Power"] = daily_power["Min Total Load [MW] - BZN|N03"].  
      ↪to_numpy()
```

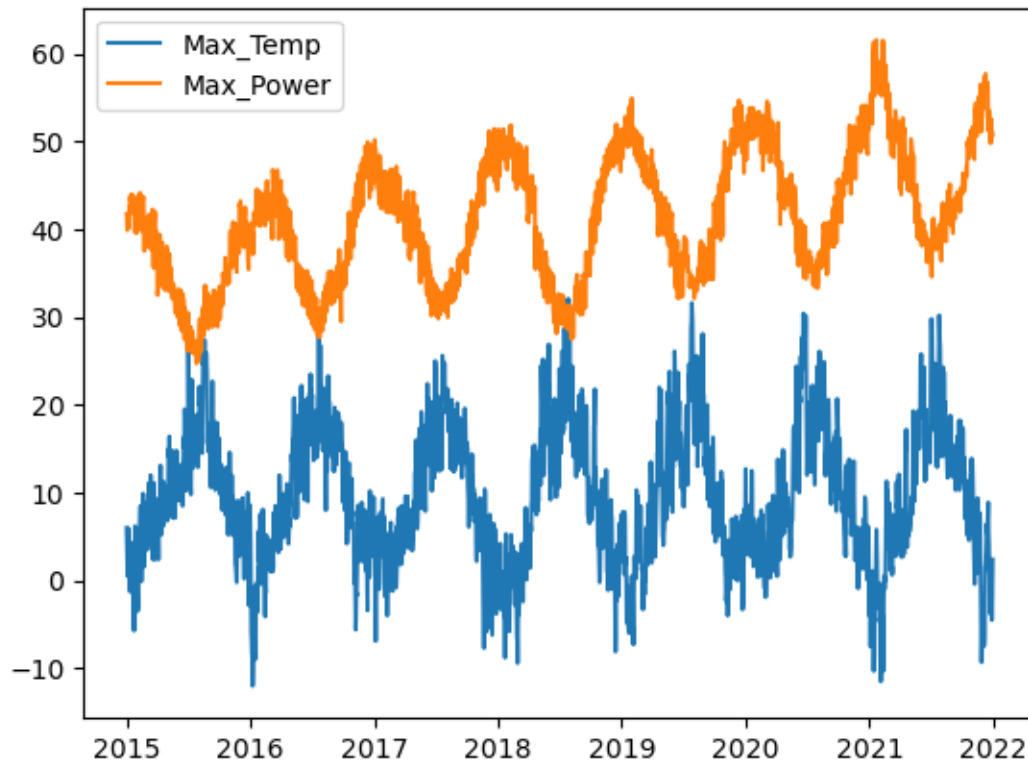
```
[ ]: weather_power.to_csv("data/weather_power_TRD.csv")
```

1.4 3. Visualisering og deskriptiv statistikk

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

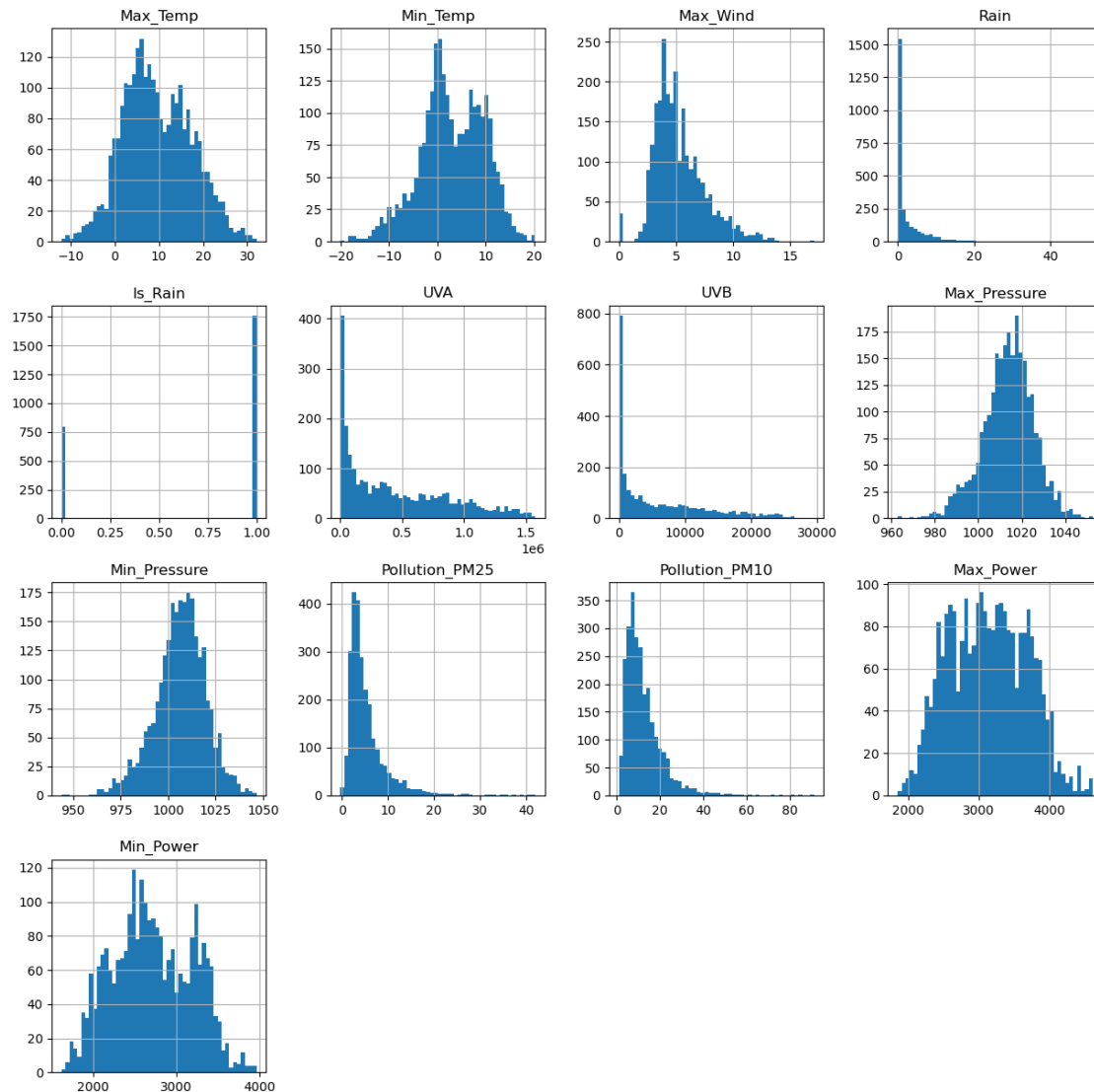



```
[ ]: weather_power.Max_Temp.plot()  
(weather_power.Max_Power/75).plot()  
plt.legend()  
plt.show()
```



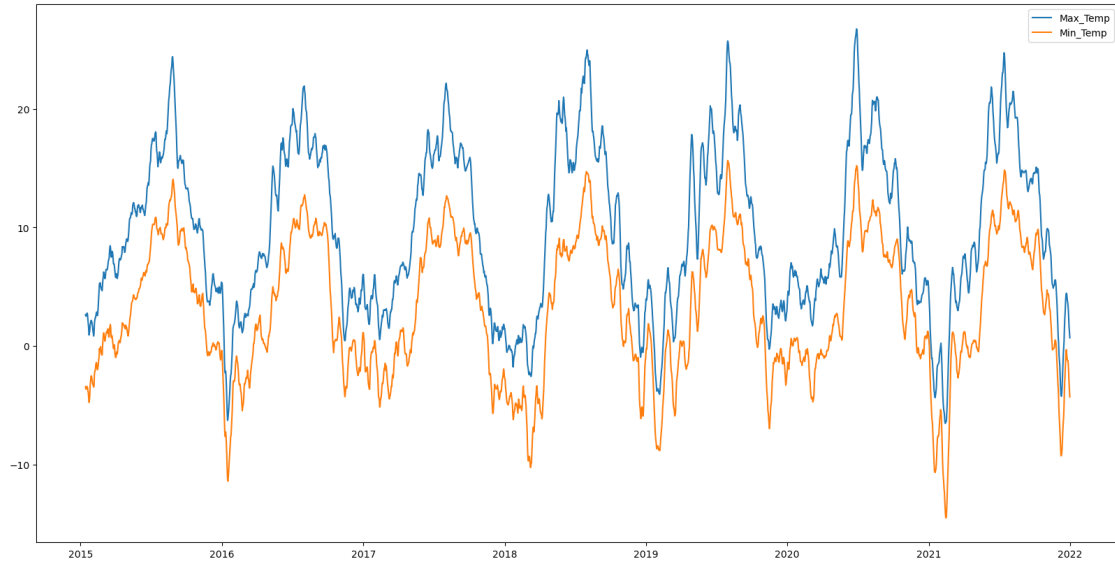
En enkel måte å vise den iverse sammenhengen mellom maksimal temperatur og maksimal strømforbruk i løpet av dagen. Dette er ikke overraskende da mesteparten av strømforbruken i en husholdning går til oppvarming.

```
[ ]: weather_power.hist(bins=50, figsize=(15,15))  
plt.show()
```



Gjennom dette histogrammet kan en se at begge målenhetene for forurensing har en positiv skjevhet(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 skjevhet, 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()
plt.legend()
plt.show()
```



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

```
[ ]:
```

	count	mean	std	min \
Max_Temp	2557.0	9.636840	7.696523	-12.00000
Min_Temp	2557.0	3.079937	6.585082	-20.10000
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
Pollution_PM25	2557.0	5.303824	4.427758	-0.39186
Pollution_PM10	2557.0	12.392685	8.732117	1.09158
Max_Power	2557.0	3136.133750	547.332617	1854.00000
Min_Power	2557.0	2710.414548	477.499468	1620.00000

	25%	50%	75%	max
Max_Temp	4.000000	8.800000	15.200000	3.210000e+01
Min_Temp	-1.100000	2.800000	8.300000	2.030000e+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
Pollution_PM25	2.613866	3.984580	6.313460	4.197452e+01

Pollution_PM10	6.593005	10.157918	15.754641	9.125000e+01
Max_Power	2680.000000	3130.000000	3578.000000	4.618000e+03
Min_Power	2357.000000	2671.000000	3109.000000	3.965000e+03

```
[ ]: 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

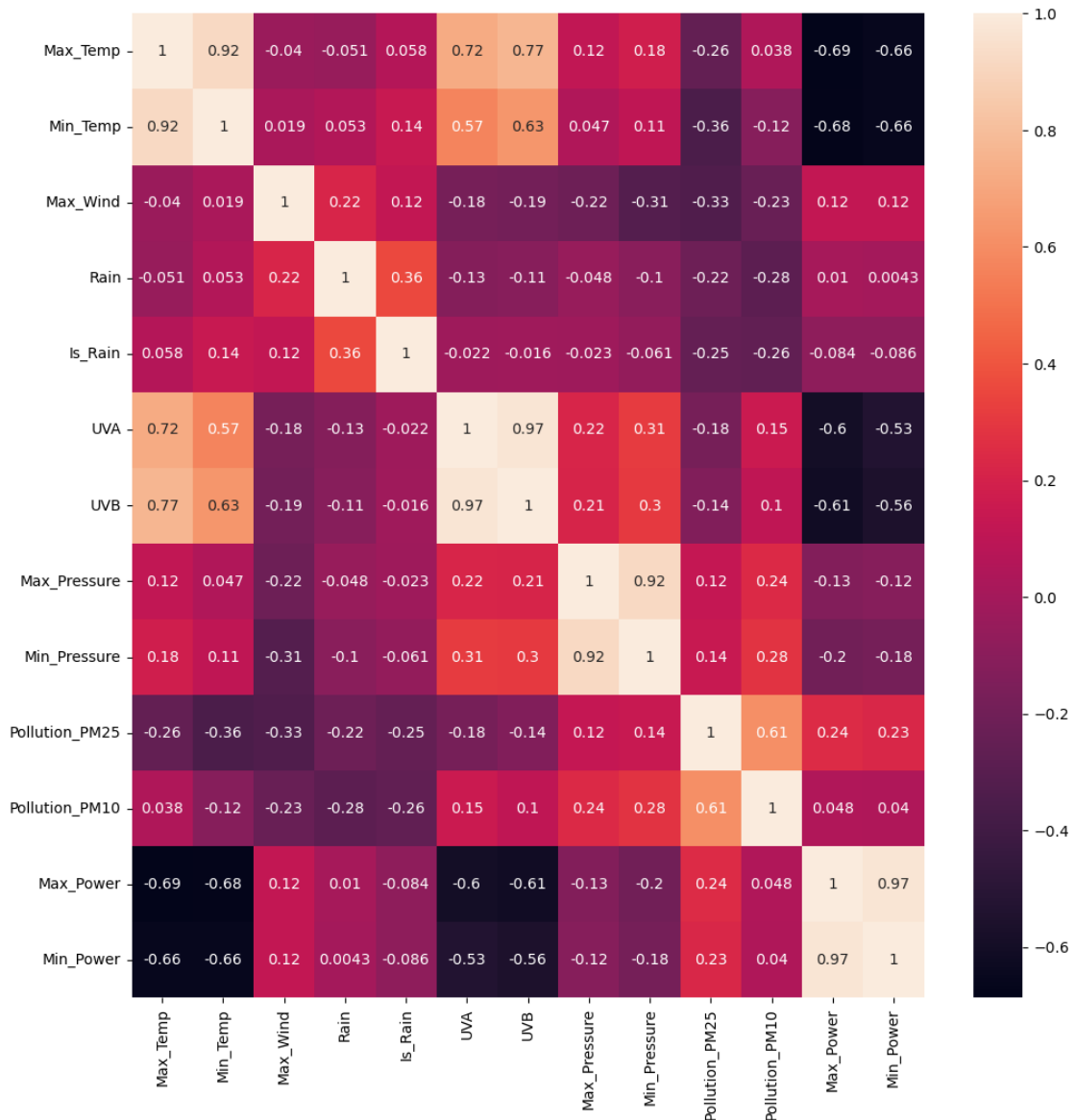
Dette inntraff følgende dager med døgnmiddel:

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

Grenseverdien på 50µg/m3 ble oversteget totalt 16 ganger i løpet av disse årene, dette er langt under grensen på 25 ganger i året. Likevel er det relevant å se hvilke dager dette er og se hvilke faktorer som gjorde at graden av svevestøv var høyere enn normalt.

```
[ ]: plt.figure(figsize=(12,12))
sns.heatmap(weather_power.corr(numeric_only=True), annot=True)
plt.show()
```



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

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

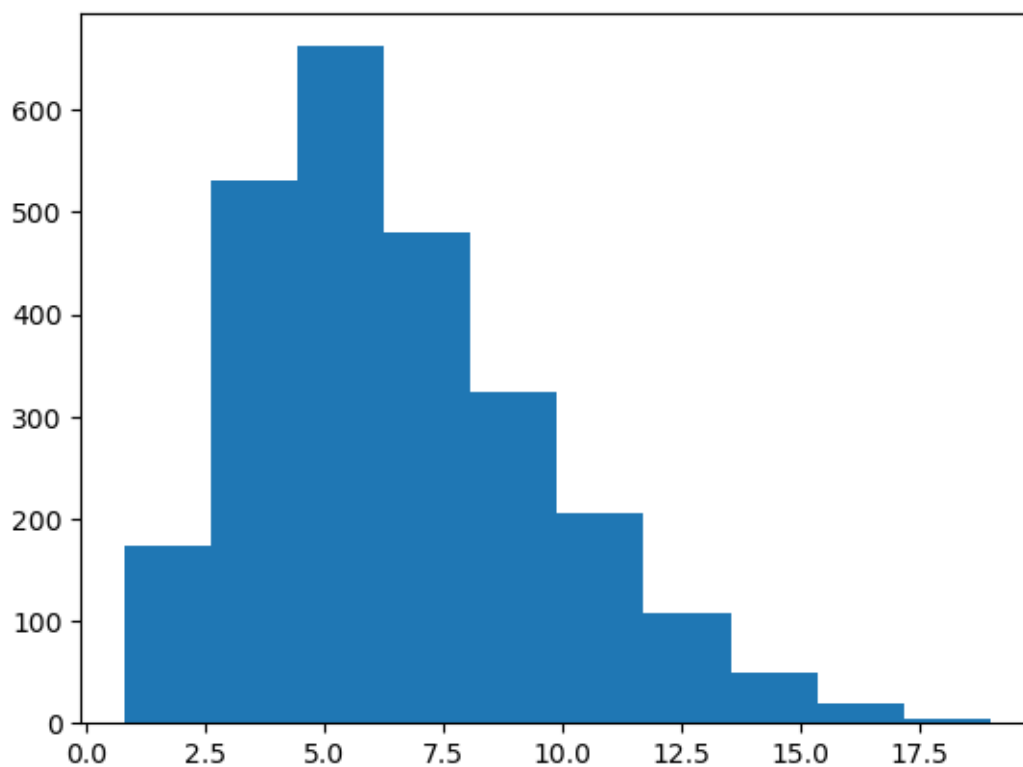
```
[ ]: count    2557.000000
      mean      6.556903
      std       3.091183
      min       0.800000
      25%       4.200000
      50%       5.900000
      75%       8.400000
```

```
max      19.000000
dtype: float64
```

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

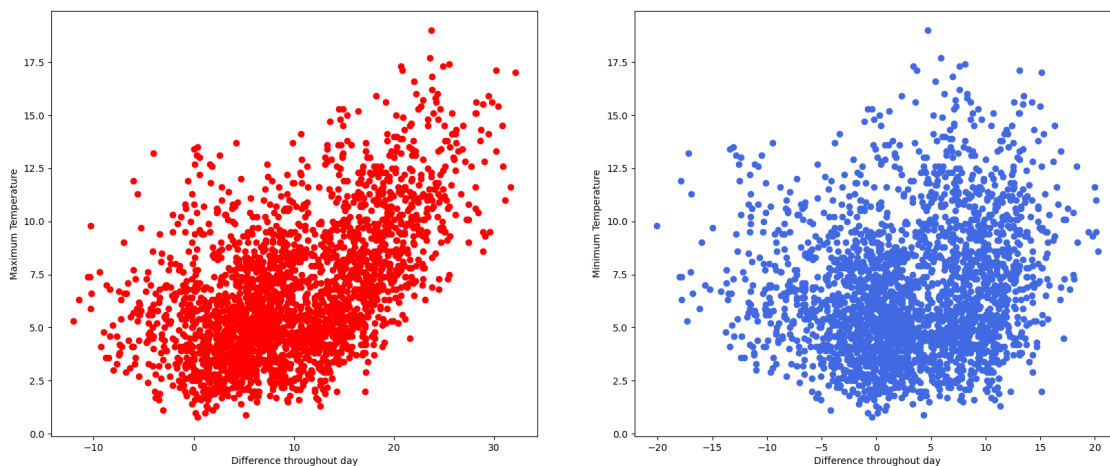
```
[ ]: 2018-05-08    19.0
      2018-05-27    17.7
      2020-06-12    17.4
      2019-04-23    17.3
      2021-06-01    17.3
      2016-06-20    17.1
      2021-07-26    17.1
      2018-07-27    17.0
      2018-05-30    16.8
      2019-04-22    16.6
      2018-07-05    16.2
      2020-07-26    16.0
      2021-08-06    16.0
      2019-04-20    15.9
      2021-07-04    15.9
      dtype: float64
```

```
[ ]: plt.hist(diff_Temp)
      plt.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 skjevhet. Dette innebærer at det vanligvis vil være temperaturforskjell under snittet, men at det inntraff 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

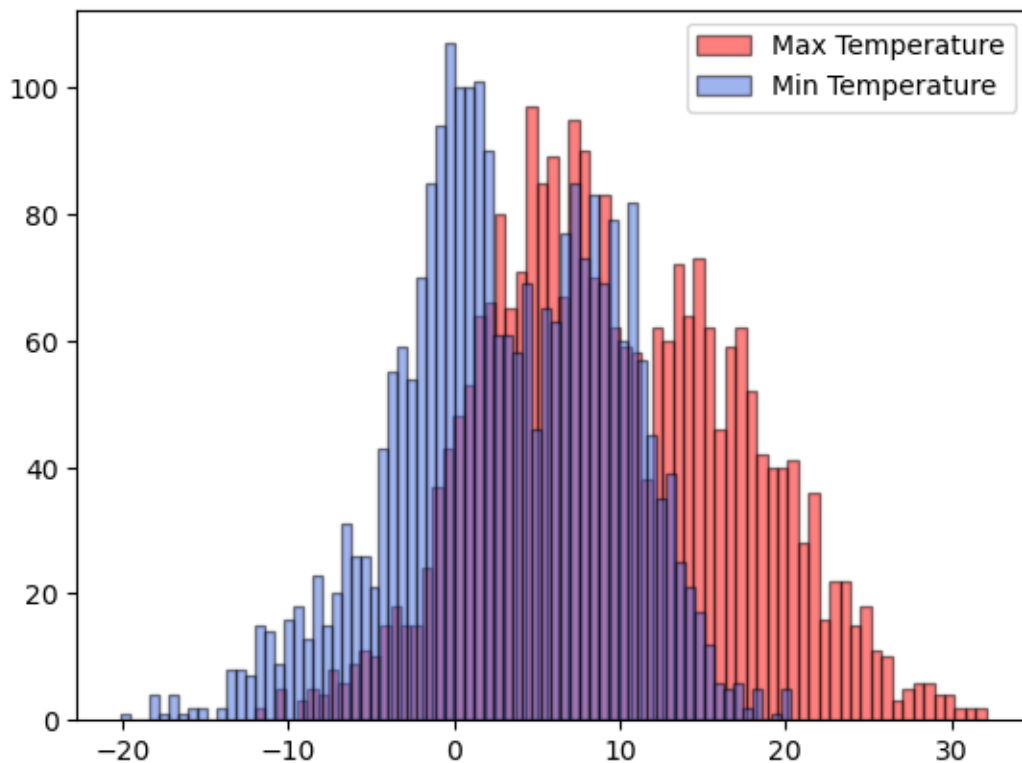
1.5 4. Simulasjon

1.5.1 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.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 ditinkte 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)
```

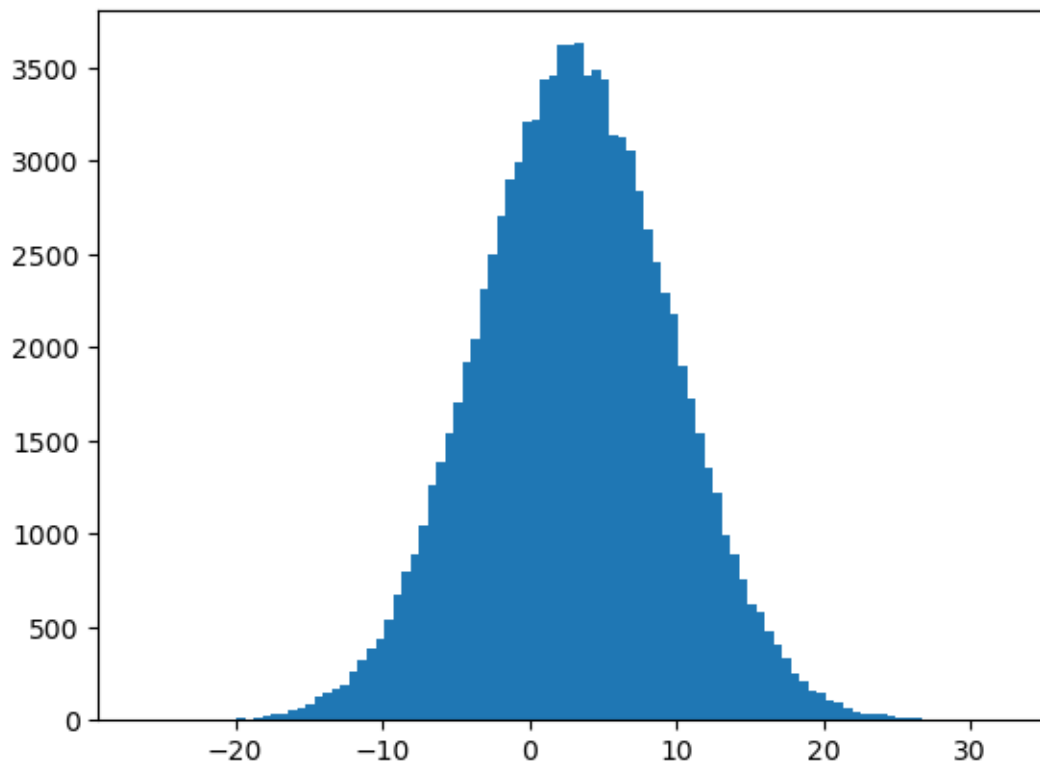

3.079937426671881 6.585082396464526

```
[ ]: 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
```

```
[ ]: 0.14651
```

```
[ ]: realProp
```

```
[ ]: 0.16112631990614001
```

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

```
[ ]: 1.0997632919673743
```

Ser med simulasjonen at antallat tilfeller med minimumstemperatur ved simulasjonen ikke er langt unna det reelle antallet tilfeller. 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 klart at den ikke er normalfordelt.

```
[ ]: stat, p = 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.

1.6 7. 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/>

1.7 8. 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: 874

```
[ ]:
```