Delrapport 1 - Kod i python

Alva Andersson, Elin Björklund, Elis Ekeberg, Marcus Linder, Josefin Lindström, Daniella Shima, Teo Trulsson, Alfred Ädel Kronberg

1 Pythonkod för Ysby kyrka

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# ====================================
# 1. LÄR MODELL (C och U)
# ====================================
df_train = pd.read_csv("Ysby.csv", sep=";", encoding='cp1252')
df_train.columns = df_train.columns.str.strip()
df_train["Value"] =
pd.to_numeric(df_train["Value"].astype(str).str.replace(",", "."),
errors="coerce")
df_train["Time"] = pd.to_datetime(df_train["Time"])

sig_col = [c for c in df_train.columns if "Signal_Id" in c][0]
valda_signaler = [244027, 244019, 244103]
df_filtered = df_train[df_train[sig_col].isin(valda_signaler)]
df_pivot = df_filtered.pivot(index="Time", columns=sig_col, values="Value")

df_pivot["T_smooth"] = df_pivot[244027].rolling(9, center=True).mean()
dt_series = df_pivot.index.to_series().diff().dt.total_seconds()
df_pivot["dT_dt"] = df_pivot["T_smooth"].diff() / dt_series

df_selected = df_pivot.loc["2026-02-16 15:15:00":"2026-04-08
05:00:00"].dropna()
df_clean = df_selected.rename(columns={244027: "T_in", 244019: "T_ut", 244103:
"Effekt"})

y = df_clean["dT_dt"].values
X = np.column_stack([df_clean["Effekt"].values, (df_clean["T_in"] -
df_clean["T_ut"]).values])
model = LinearRegression().fit(X, y)
```

```python
C = 1 / model.coef_[0]
U = -model.coef_[1] * C

# ====================================
# 2. HJÄLPFUNKTIONER FÖR MOLLIER
# ====================================
def get_absolute_humidity(temp, rh_percent):
    """Beräknar g vatten per m3 luft"""
    if pd.isna(temp) or pd.isna(rh_percent): return 0
    t = float(temp)
    rh = float(rh_percent)
    es = 6.112 * np.exp((17.67 * t) / (t + 243.5)) * 100
    v_sat = (es) / (461.5 * (t + 273.15)) * 1000
    return v_sat * (rh / 100)

def find_optimal_temp_for_humidity(abs_fukt):
    """Hittar lägsta temp (10-18C) som håller RH under 62%"""
    f = float(abs_fukt)
    for t in np.arange(10.0, 18.1, 0.1):
        es = 6.112 * np.exp((17.67 * t) / (t + 243.5)) * 100
        v_sat_t = (es) / (461.5 * (t + 273.15)) * 1000
        rh_t = (f / v_sat_t) * 100
        if 30.0 <= rh_t <= 62.0:
            return round(t, 1)
    return 14.0

# ====================================
# 3. LÄS 2025 DATA & BERÄKNA BÖRVÄRDEN
# ====================================
print("--- Steg 2: Behandlar 2025-data ---")
df = pd.read_csv("Ysby_2025.csv", sep=";", encoding='cp1252', engine='python')
df = df.iloc[:, :5]
df.columns = ["Time", "Fukt_A", "Utetemp", "Bör_temp", "SEK_per_kWh"]

for col in ["Fukt_A", "Utetemp", "Bör_temp", "SEK_per_kWh"]:
    df[col] = pd.to_numeric(df[col].astype(str).str.replace(",", "."),
errors="coerce")

df["Time"] = pd.to_datetime(df["Time"], errors='coerce')
df = df.dropna(subset=["Utetemp", "Time", "Fukt_A"])
```

```
df['medel_19h'] = df['SEK_per_kWh'].rolling(19, center=True,
min_periods=1).mean()
df['avvikelse_procent'] = ((df['SEK_per_kWh'] - df['medel_19h']) /
df['medel_19h']) * 100

df['Abs_Fukt_Ute'] = df.apply(lambda x: get_absolute_humidity(x['Utetemp'],
x['Fukt_A']), axis=1)

def bestam_borvarde(row):
    if row['avvikelse_procent'] <= -10:
        return row["Bör_temp"]
    return row["Bör_temp"]

df['Bör_temp'] = df.apply(bestam_borvarde, axis=1)

# ======================================
# 4. SIMULERING
# ======================================
print("--- Steg 3: Kör simulering ---")
T = 14.0
T_min, T_max, P_max = 10, 22, 30000
T_list, P_list, energy_list = [], [], []
baslast = df_clean["Effekt"].quantile(0.2)

for i in range(len(df)):
    row = df.iloc[i]
    bör_temp = row["Bör_temp"]
    utetemp = row["Utetemp"]

    if T < bör_temp:
        ny_temp_target = T + 0.5
    elif T > bör_temp:
        ny_temp_target = T - 0.5
    else:
        ny_temp_target = T

    dT_target = (ny_temp_target - T) / 3600
    P_teoretisk = C * dT_target + U * (T - utetemp)

    if utetemp > 17:
        P = 0
        p_bas = baslast * 0.1
```

```
    else:
        P = max(0, min(P_teoretisk, P_max))
        p_bas = baslast

    dT_dt = (P - U * (T - utetemp)) / C
    T = T + dT_dt * 3600
    T = max(T_min, min(T_max, T))

    T_list.append(T)
    P_list.append(round(P))
    energy_list.append((P + p_bas) / 1000)

df["Temp_sim"] = T_list
df["Energy_kWh"] = energy_list
df["Hourly_Cost_SEK"] = df["Energy_kWh"] * df["SEK_per_kWh"]
df["Effekt_kW"] = np.array(P_list) / 1000

# ====================================
# 5. RESULTAT & PLOT
# ====================================
EFFEKT_PRIS_PER_KW = 81.25

print("\n--- KONTROLL PER MÅNAD (INKL. EFFEKTTARIFF) ---")

manads_koll = df.groupby(df['Time'].dt.month).agg({
    'Utetemp': 'mean',
    'Bör_temp': 'mean',
    'Temp_sim': 'mean',
    'Energy_kWh': 'sum',
    'Hourly_Cost_SEK': 'sum',
    'Effekt_kW': 'max'
}).rename(columns={
    'Energy_kWh': 'Total_kWh',
    'Hourly_Cost_SEK': 'Energikostnad_SEK',
    'Effekt_kW': 'Max_Effekt_kW'
})

manads_koll['Effekttariff_SEK'] = manads_koll['Max_Effekt_kW'] *
EFFEKT_PRIS_PER_KW
manads_koll['Total_Kostnad_SEK'] = manads_koll['Energikostnad_SEK'] +
manads_koll['Effekttariff_SEK']
```

```
print(manads_koll.round(2))

total_energi_kostnad = df['Hourly_Cost_SEK'].sum()
total_effekt_kostnad = manads_koll['Effekttariff_SEK'].sum()

print(f"\nTotal årsenergi: {df['Energy_kWh'].sum():.0f} kWh")
print(f"Total energikostnad: {total_energi_kostnad:.0f} SEK")
print(f"Total effektkostnad (tariffer): {total_effekt_kostnad:.0f} SEK")
print(f"TOTAL SUMMA: {total_energi_kostnad + total_effekt_kostnad:.0f} SEK")
print(f"U: {U:.2f}, C: {C:.2f}")

plt.figure(figsize=(12, 5))
plt.xlabel('Månad')
plt.ylabel('Grader Celsius')
plt.plot(df["Time"], df["Temp_sim"], label="Innetemp (Sim)")
plt.plot(df["Time"], df["Bör_temp"], label="Börvärde (Mollier)", alpha=0.5)
plt.legend()
plt.title("Temperaturutveckling under året")
plt.grid(True)
plt.show()

# ====================================
# 6. NY GRAF: ENERGI OCH PRIS PER MÅNAD
# ====================================
df_monthly = df.groupby(df['Time'].dt.to_period('M')).agg({
    'Energy_kWh': 'sum',
    'SEK_per_kWh': 'mean'
})

df_monthly.index = df_monthly.index.astype(str)

fig, ax1 = plt.subplots(figsize=(12, 7))

color_energy = 'tab:blue'
ax1.set_xlabel('Månad')
ax1.set_ylabel('Total energiförbrukning (kWh)', color=color_energy,
fontweight='bold')
ax1.bar(df_monthly.index, df_monthly['Energy_kWh'], color=color_energy,
alpha=0.6, label='Energi (kWh)')
ax1.tick_params(axis='y', labelcolor=color_energy)
ax1.grid(axis='y', linestyle='--', alpha=0.7)
```

```
ax2 = ax1.twinx()
color_price = 'tab:red'
ax2.set_ylabel('Medelpris (SEK/kWh)', color=color_price, fontweight='bold')
ax2.plot(df_monthly.index, df_monthly['SEK_per_kWh'], color=color_price,
marker='o', linewidth=2, label='Medelpris (SEK)')
ax2.tick_params(axis='y', labelcolor=color_price)

plt.title('Månadsvis energiförbrukning vs Medelpris (2025)', fontsize=14)
fig.tight_layout()
plt.show()
```