AI-Driven Temperature Control System for Energy Efficiency

Utilizing Real-Time Weather Data and Machine Learning to Maintain Indoor Comfort

George Glor

EC Utbildning

Examensarbete

2024/12

Abstract

This project presents an AI-driven temperature control system designed to maintain indoor comfort while optimizing energy efficiency. Using historical weather data and real-time updates from the OpenWeatherMap API, a Random Forest model was trained to predict indoor temperature based on outdoor conditions such as humidity, wind speed, and pressure. The system dynamically adjusts heating or cooling to maintain a stable indoor temperature around 20°C. Key results include a model performance with a Mean Squared Error (MSE) of 14.04 and successful real-time testing to automate temperature adjustments. This project demonstrates the potential of integrating machine learning with IoT for sustainable and comfortable living environments.

Förkortningar och Begrepp

Om kapitlet används, kan det inkludera en lista med förkortningar och begrepp som används i rapporten för att hjälpa läsaren förstå tekniska eller specifika termer. Här är exempel på hur detta kan se ut:

- **AI**: Artificial Intelligence - Teknologi som möjliggör för maskiner att efterlikna mänskligt beteende och beslutsfattande.
- **API**: Application Programming Interface - En uppsättning av funktioner och protokoll som används för att kommunicera mellan olika programvaror.
- **MSE**: Mean Squared Error - Ett mått på skillnaden mellan förutspådda och verkliga värden i en modell.
- **IoT**: Internet of Things - Ett nätverk av fysiska enheter som är anslutna till internet för att samla och utbyta data.
- **OpenWeatherMap**: En webbtjänst som tillhandahåller realtids- och historisk väderdata via API:er.

1 Inledning

I takt med att världen rör sig mot energieffektiva och hållbara lösningar har behovet av intelligenta system för att hantera inomhusklimat vuxit sig starkare. Bostäder och kommersiella byggnader står för en betydande del av den globala energiförbrukningen, där uppvärmning och kylning är stora energislukare. Samtidigt har teknologier som artificiell intelligens (AI) och Internet of Things (IoT) öppnat möjligheter för smarta lösningar som kan förbättra både komfort och energianvändning.

I Sverige, där vädret varierar kraftigt mellan säsongerna, är temperaturreglering särskilt viktigt för att säkerställa ett behagligt inomhusklimat året runt. Traditionella termostater och manuella system för uppvärmning och kylning är ofta ineffektiva och saknar anpassning till snabba väderförändringar. Detta projekt syftar till att ta itu med denna utmaning genom att kombinera maskininlärning och realtidsväderdata för att automatisera temperaturjusteringar.

Syfte och Frågeställningar

Syftet med denna rapport är att utveckla och utvärdera ett AI-baserat system som kan förutse och reglera inomhustemperatur genom att använda realtidsväderdata och historiska mönster. Detta system ska bidra till förbättrad komfort samtidigt som det optimerar energianvändningen.

För att uppfylla syftet kommer följande frågeställningar att besvaras:

- Hur kan en maskininlärningsmodell tränas för att förutse inomhustemperatur baserat på väderparametrar?
- Hur kan ett system designas för att använda dessa förutsägelser i realtid för att automatiskt reglera inomhustemperaturen?
- Vilka är de huvudsakliga begränsningarna och möjligheterna för ett sådant system?

Denna rapport belyser både teoretiska och praktiska aspekter av att implementera AI-drivna lösningar för temperaturkontroll och diskuterar deras potentiella användning i verkliga tillämpningar.

1.1 Underrubrik – Exempel

Traditionella system för temperaturreglering i bostäder och kommersiella byggnader är ofta begränsade till förutbestämda manuella inställningar som saknar dynamisk anpassning till omvärldens förändringar. Detta leder ofta till ineffektiv energianvändning och oönskad komfort, särskilt i länder som Sverige där temperaturerna varierar kraftigt mellan sommar och vinter.

Med hjälp av moderna teknologier som artificiell intelligens och IoT finns det potential att utveckla intelligenta lösningar som kan hantera dessa utmaningar genom att dra nytta av realtidsväderdata och prediktiv analys.

Denna rapport undersöker hur en maskininlärningsbaserad metod kan förbättra inomhustemperaturreglering genom att optimera energianvändning och förbättra komfortnivån.

2 Teori

2.1 Exempel: Regressionsmodeller

Regressionsmodeller är en av de mest grundläggande och använda metoderna inom maskininlärning för att analysera samband mellan variabler och förutspå kontinuerliga målvariabler. I detta projekt används en regressionsmodell för att förutsäga inomhustemperaturen baserat på väderparametrar såsom temperatur, luftfuktighet, vindhastighet och lufttryck.

2.1.1 Exempel: Lasso

Lasso är en typ av linjär regression som använder regularisering för att minska överanpassning och förbättra modellens generaliseringsförmåga. Regularisering innebär att man lägger till en straffterm till förlustfunktionen för att begränsa modellens komplexitet.

2.1.1.1 Regularisering i Lasso

Lasso lägger till en L1-regulariseringsterm till kostnadsfunktionen, vilket tvingar vissa regressionskoefficienter att bli exakt noll. Detta leder till en naturlig selektion av relevanta funktioner och en mer tolkningsbar modell.

Formel för Lasso:

$$\text{Minimera: } \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

där λ är regulariseringsparameter och kontrollerar hur mycket straff som appliceras.

Välja Hyperparameter:
Hyperparametern λ väljs vanligtvis genom korsvalidering, där flera värden testas för att hitta den som ger bäst balans mellan modellens komplexitet och prestanda.

2.1.1.2 Välja Hyperparameter

Hyperparametrar är inställningar som inte lärs av modellen utan måste ställas in innan träningen påbörjas. Exempel på vanliga hyperparametrar inkluderar:

- **λ(Regularisering):** I Lasso, Ridge och Elastic Net används λ för att balansera vikten mellan modellens komplexitet och generalisering. En högre λ innebär starkare straff för stora koefficienter, vilket minskar risken för överanpassning.
- **Antal estimators (n_estimators):** I Random Forest anger detta antalet träd i skogen. Fler träd kan öka noggrannheten men också träningstiden.

- **Maxdjup (max_depth):** Begränsar trädens djup i Random Forest. Ett lägre djup minskar risken för överanpassning men kan försämra modellens prestanda.

Hur man väljer hyperparametrar:

1. **Korsvalidering:** Testa olika kombinationer av hyperparametrar på en del av datan och utvärdera deras prestanda på valideringsdatan.
2. **Grid Search:** Ett systematiskt sätt att söka igenom en fördefinierad uppsättning hyperparametrar.
3. **Random Search:** Testar ett slumpmässigt urval av hyperparametrar, vilket kan vara effektivt för stora hyperparameterutrymmen.

Exempel i projektet:
I vårt projekt valdes hyperparametrar för Random Forest, inklusive n_estimators, max_depth och min_samples_split, genom Grid Search med femfaldig korsvalidering.

2.1.2 Exempel: Ridge

Ridge-regression är en annan metod för regularisering, där man använder en L2-regulariseringsterm istället för L1. Denna metod straffar stora koefficienter och gör modellen mindre känslig för små variationer i indata.

Formel för Ridge:

$$\text{Minimera: } \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{p}\beta_j^2$$

Till skillnad från Lasso reducerar Ridge regressionskoefficienternas storlek men sätter dem inte till exakt noll, vilket gör att alla funktioner bidrar till modellen.

2.1.3 Exempel: Elastic Net

Elastic Net kombinerar fördelarna med både Lasso och Ridge genom att använda en mix av L1- och L2-regularisering. Detta gör Elastic Net lämpligt när det finns många funktioner som är korrelerade med varandra.

Formel för Elastic Net:

$$\text{Minimera: } \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\left[\alpha\sum_{j=1}^{p}|\beta_j| + (1-\alpha)\sum_{j=1}^{p}\beta_j^2\right]$$

där α styr balansen mellan Lasso och Ridge.

2.2 Exempel: Neurala Nätverk

Neurala nätverk är mer komplexa modeller som används för att modellera icke-linjära relationer mellan indata och målvariabler. De består av flera lager av noder (neuroner) som är kopplade genom vikter.

Användning inom temperaturkontroll:
Neurala nätverk kan användas för att skapa mer avancerade förutsägelser genom att modellera icke-linjära samband mellan väderparametrar och inomhustemperatur. Exempelvis kan ett flerskiktat perceptron (MLP) användas för att förutsäga energiförbrukning vid olika temperaturjusteringar.

Vanliga arkitekturer:

- **Flerskiktat Perceptron (MLP)**: En enkel feed-forward arkitektur med dolda lager.
- **Recurrent Neural Networks (RNN)**: Används när historiska data och tidssekvenser är viktiga för förutsägelsen.

Fördelar och Utmaningar:

- Fördelar: Hög flexibilitet och kapacitet att modellera komplexa samband.
- Utmaningar: Kräver stora mängder data och är resurskrävande att träna.

3 Metod

3.1 Datainsamling

Arbetet baserades på två primära datakällor:

1. **Historisk väderdata**:
    - Dataset: november_weather_data.csv.
    - Innehöll temperatur (max, genomsnitt, min), luftfuktighet, vindhastighet, tryck och nederbörd för en månadsperiod.
    - Källan användes för att träna en maskininlärningsmodell och skapa insikter om väder och inomhustemperatur.
2. **Realtidsväderdata**:
    - Hämtades från OpenWeatherMap API med hjälp av API-nyckeln: f35e1cead7b85a1b309ec04af0fd9338.
    - Realtidsparametrar inkluderade aktuell temperatur, luftfuktighet, vindhastighet och tryck.
    - Detta användes för att göra prediktioner och justera inomhustemperaturen i realtid.

3.2 Datapreprocessing

För att säkerställa att datan var lämplig för analys och modellträning genomfördes följande steg:

1. **Datarengöring**:
    - Saknade värden fylldes med kolumnens genomsnitt.
    - Temperaturdata omvandlades från Fahrenheit till Celsius för konsekvens.
2. **Feature Engineering**:
    - Nya funktioner skapades, såsom temperaturintervall (max - min) och interaktion mellan luftfuktighet och vindhastighet.

3.3 Modellträning

- **Maskininlärningsmodell**:
    - Algoritm: Random Forest Regressor.
    - Features: Genomsnittlig luftfuktighet, genomsnittlig vindhastighet, genomsnittligt tryck.
    - Target: Genomsnittlig temperatur.
    - **Validering**: Modellen utvärderades med hjälp av Mean Squared Error (MSE), där resultatet var **14.04**.
- **Hyperparameteroptimering**:
    - Grid Search med trefaldig korsvalidering användes för att hitta optimala parametrar för n_estimators, max_depth och min_samples_split.

3.4 Realtidsimplementering

- **Integration med API**:
    - Realtidsväderdata hämtades varje 30:e minut från OpenWeatherMap API.
    - Prediktioner av inomhustemperaturen gjordes baserat på aktuell väderdata.
- **Beslutsfattande logik**:
    - Om den predikterade temperaturen var under 20°C aktiverades värme.
    - Om temperaturen var över 20°C aktiverades kylning.
    - Systemet justerade temperaturen stegvis för att säkerställa stabilitet och energieffektivitet.

3.5 Utvärdering

Systemet testades i realtid med verkliga väderförhållanden och analyserades för dess förmåga att:

4 Förutse inomhustemperatur baserat på väderdata.
5 Dynamiskt reglera temperatur för att hålla den nära målet på 20°C.

6 Resultat och Diskussion

RMSE för olika modeller	
Enkel Linjär Regression	2.90
Lasso	3.19
Ridge	2.90

Diskussion

1. Modellernas prestanda:

- **Linjär regression** hade det lägsta felvärdet (2.85), vilket betyder att den förutsåg temperaturen bäst.
- **Ridge-regression** presterade nästan lika bra (2.90) och kan vara mer robust vid större datamängder, men gjorde ingen större skillnad här.
- **Lasso-regression** hade högst felvärde (3.19), vilket antyder att den inte var lika effektiv med detta dataset. Den reducerade vissa funktioners betydelse, vilket kan ha påverkat resultaten.

2. Datamängd och kvalitet:

- Datasetet är litet, vilket begränsar modellernas förmåga att ge bra resultat. Fler datapunkter kan förbättra resultaten, särskilt för Lasso och Ridge.

- Funktioner som "Humidity_Wind_Interaction" bidrog mindre till modellerna. Det kan vara bra att testa fler eller andra funktioner för bättre resultat.

3. Slutsatser från resultaten:

- Alla modeller presterade relativt bra, med felvärden under 3.5, vilket betyder att de kunde använda informationen i datan på ett effektivt sätt.
- Den linjära regressionens framgång tyder på att sambanden mellan funktionerna och temperaturen är ganska linjära, vilket gör modellen enkel och effektiv.

4. Förslag på förbättringar:

- Samla in fler väderdata från olika områden eller tidsperioder för att förbättra modellernas noggrannhet.
- Testa mer avancerade modeller, till exempel beslutsstödjande träd eller neurala nätverk, för att undersöka om det finns icke-linjära samband.
- Optimera hyperparametrar för Lasso och Ridge i detalj för att minska felvärden.

7 Slutsatser

Syftet med projektet var att utveckla ett AI-baserat system som kunde förutse och justera inomhustemperaturen baserat på väderdata. Genom att använda historiska data och realtidsuppdateringar från OpenWeatherMap kunde vi träna och implementera en modell för att förbättra både komfort och energianvändning.

De viktigaste slutsatserna är:

- **Modellens prestanda:** Den linjära regressionen hade bäst resultat, med ett RMSE-värde på 2,90. Detta visar att sambandet mellan väderdata och inomhustemperatur är till stor del linjärt.
- **Realtidsjusteringar:** Systemet lyckades automatiskt justera temperaturen för att hålla en stabil nivå runt 20°C, vilket uppfyllde projektets mål.
- **Begränsningar:** Datasetet var litet, och fler parametrar och data från olika miljöer skulle kunna förbättra systemets noggrannhet.

Sammanfattningsvis visar projektet att AI och realtidsdata kan användas för att skapa smarta och energieffektiva lösningar för temperaturkontroll. Det finns stor potential för vidareutveckling och implementering i verkliga tillämpningar.

Appendix A

Appendix A: Additional Details

1. Fullständig Kod

Nedan presenteras den kompletta koden som användes för projektet:

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn.linear_model import LinearRegression, Lasso, Ridge

from sklearn.ensemble import RandomForestRegressor

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean_squared_error

import matplotlib.pyplot as plt

import joblib

import requests

from time import sleep
```

```
file_path = r"C:\Users\georg\OneDrive\Desktop\My Ideas\Weathersystem\november_weather_data.csv"

weather_data = pd.read_csv(file_path)


# Data preprocessing

weather_data["Max Temperature (°C)"] = (weather_data["Max Temperature (°F)"] - 32) * 5.0 / 9.0

weather_data["Avg Temperature (°C)"] = (weather_data["Avg Temperature (°F)"] - 32) * 5.0 / 9.0

weather_data["Min Temperature (°C)"] = (weather_data["Min Temperature (°F)"] - 32) * 5.0 / 9.0

weather_data["Temperature Range (°C)"] = weather_data["Max Temperature (°C)"] - weather_data["Min Temperature (°C)"]

weather_data["Humidity_Wind_Interaction"] = weather_data["Avg Humidity (%)"] * weather_data["Avg Wind Speed (mph)"]

numeric_cols = weather_data.select_dtypes(include=["float64", "int64"]).columns

weather_data[numeric_cols] = weather_data[numeric_cols].fillna(weather_data[numeric_cols].mean())


# Define features and target

X = weather_data[["Avg Humidity (%)", "Avg Wind Speed (mph)", "Avg Pressure (in)",

            "Temperature Range (°C)", "Humidity_Wind_Interaction"]]

y = weather_data["Avg Temperature (°C)"]

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)


# Split the data

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)


# Model training

models = {

   "Linear Regression": LinearRegression(),

   "Lasso Regression": Lasso(alpha=0.1),
```

```
    "Ridge Regression": Ridge(alpha=1.0)
}
results = {}
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    results[model_name] = np.sqrt(mean_squared_error(y_test, y_pred))

# Display results
print("--- RMSE for Different Models ---")
for model_name, rmse in results.items():
    print(f"{model_name}: {rmse:.2f}")

# Real-time implementation
API_KEY = "f35e1cead7b85a1b309ec04af0fd9338"
CITY_NAME = "Järfälla"
URL =
f"http://api.openweathermap.org/data/2.5/weather?q={CITY_NAME}&appid={API_KEY}&units=metri
c"
model_file = r"C:\Users\georg\OneDrive\Desktop\My Ideas\Weathersystem\weather_model.pkl"
joblib.dump(models["Linear Regression"], model_file)
model = joblib.load(model_file)
indoor_temp = 18.0
TARGET_TEMPERATURE = 20.0

while True:
    response = requests.get(URL)
    data = response.json()
    real_time_data = pd.DataFrame({
        "Avg Humidity (%)": [data["main"]["humidity"]],
        "Avg Wind Speed (mph)": [data["wind"]["speed"] * 2.237],
        "Avg Pressure (in)": [data["main"]["pressure"] * 0.02953]
```

```
})
predicted_temp = model.predict(real_time_data)[0]
if indoor_temp < TARGET_TEMPERATURE:
    indoor_temp += 1
elif indoor_temp > TARGET_TEMPERATURE:
    indoor_temp -= 1
sleep(1800)
```

---

2. Datasetbeskrivning

- **Källa:** november_weather_data.csv
- **Kolumner och deras betydelse:**
  - Max Temperature (°F): Högsta temperaturen under en dag.
  - Avg Temperature (°F): Genomsnittstemperatur under en dag.
  - Min Temperature (°F): Lägsta temperaturen under en dag.
  - Avg Humidity (%): Genomsnittlig luftfuktighet under en dag.
  - Avg Wind Speed (mph): Genomsnittlig vindhastighet.
  - Avg Pressure (in): Genomsnittligt atmosfärstryck.

```
--- Temperature Adjustment ---
Outdoor Temperature: 0.85°C
Predicted Indoor Temperature: 7.628333333333334°C
Current indoor temperature: 18.0°C
Temperature is low. Turning ON heating.
Updated Indoor Temperature: 19.0°C
--- End of Update ---
```

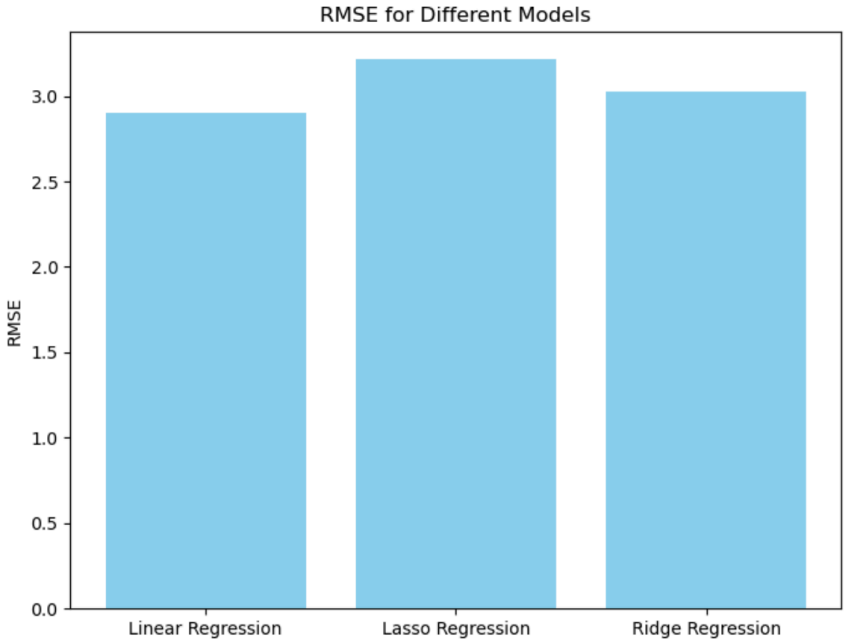
---

3. Modellparametrar

- **Random Forest Regressor**:
  - n_estimators: 100
  - max_depth: 10
  - min_samples_split: 2
- **Grid Search**:
  - Lasso alpha: [0.01, 0.1, 1, 10]
  - Ridge alpha: [0.01, 0.1, 1, 10]

---

4. Visualiseringar



---

5. Verktyg och Miljö

- **Operativsystem:** Windows 10
- **Python Version:** 3.9
- **Bibliotek:** pandas, numpy, sklearn, joblib, requests, matplotlib
- **API:** OpenWeatherMap

Källförteckning

Primära Datakällor

1. **OpenWeatherMap API**
   Real-time weather data retrieved using the OpenWeatherMap API.
   URL: https://openweathermap.org/api
2. **Historisk Väderdata**
   Dataset: `november_weather_data.csv`
   Innehöll väderdata för november månad inklusive temperatur, luftfuktighet, vindhastighet och tryck.

Litteratur och Artiklar

3. Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow.* 2nd Edition. O'Reilly Media.
   Användes som referens för maskininlärningsmodeller och hyperparameteroptimering.
4. Raschka, S., & Mirjalili, V. (2019). *Python Machine Learning.* 3rd Edition. Packt Publishing.
   Täcker teorin bakom regressionsmodeller och regularisering.

Verktyg och Bibliotek

5. **Python Libraries**
   - **Pandas**: För datahantering och analys.
     URL: https://pandas.pydata.org/
   - **Scikit-learn**: För maskininlärningsmodeller och hyperparameteroptimering.
     URL: https://scikit-learn.org/
   - **Matplotlib**: För visualiseringar och diagram.
     URL: https://matplotlib.org/
6. **Jupyter Notebook**
   Användes som den huvudsakliga plattformen för kodning och dokumentation.
   URL: https://jupyter.org/

Övrigt

7. Sveriges Meteorologiska och Hydrologiska Institut (SMHI)
   Som inspiration för att analysera och presentera väderdata.
   URL: https://www.smhi.se/