

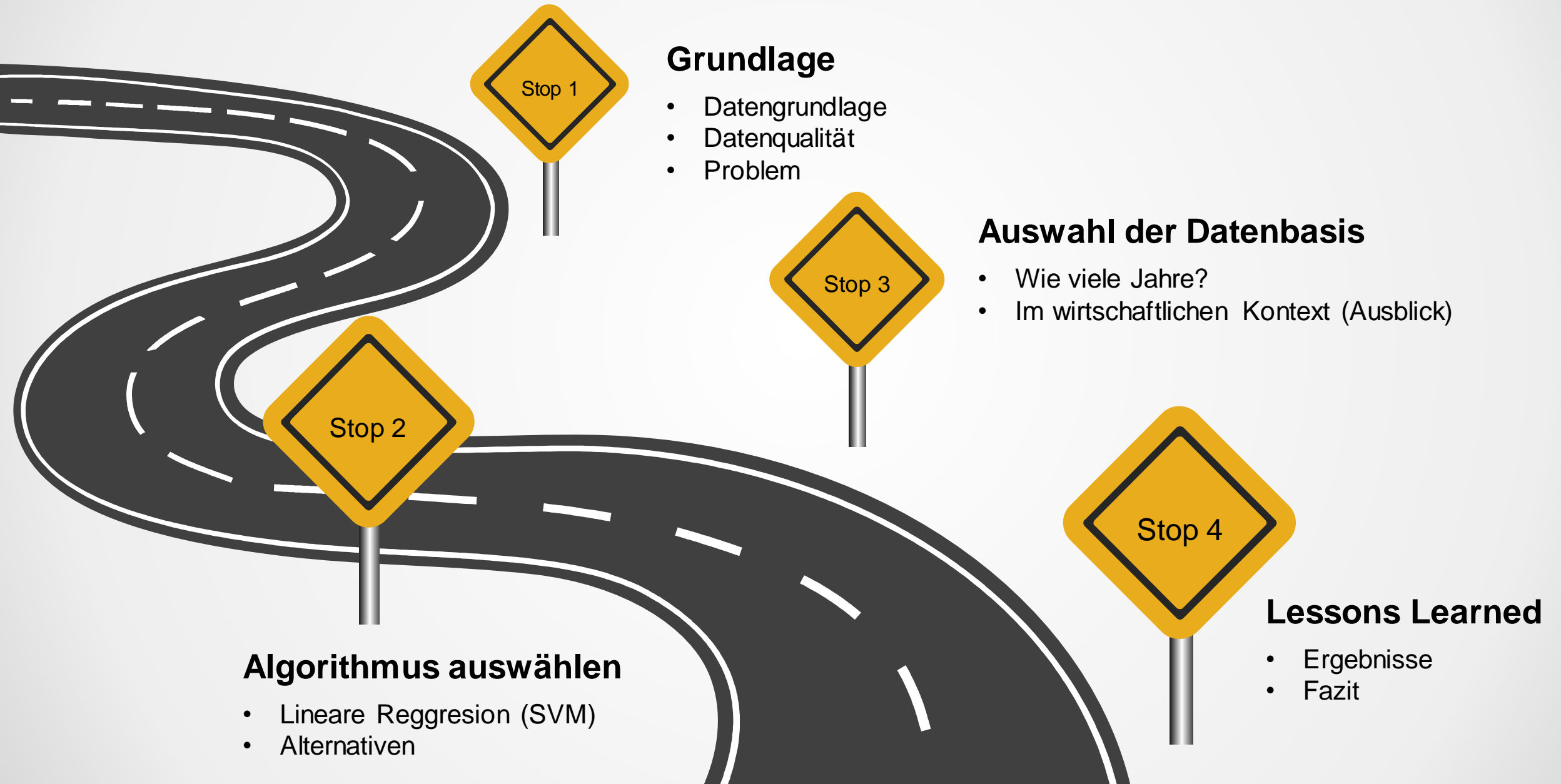


# Wettervorhersage

Gruppe NoData

(Nils-Jannik Klink, Marvin Spurk, Rico  
Joel Siegelin)

# Roadmap





# Datengrundlage

- OpenData-Bereich des DWD bietet Stationsdatenzugriff
- Daten von 01.01.1948 01:00Uhr–  
31.12.2021 23:00Uhr
- Station:5906 -> Wetterstation Mannheim
- Temperaturwerte (C°)
- Luftfeuchtigkeit (%)
- Luftdruck (mBar)



# Datenqualität



Doku der Daten lesen, falls vorhanden



Quelle prüfen



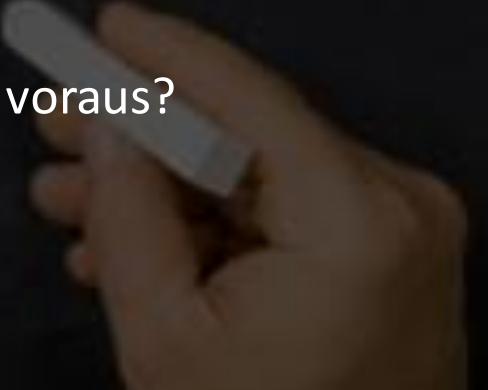
Sind die Daten  
zuverlässig/vollständig/...?

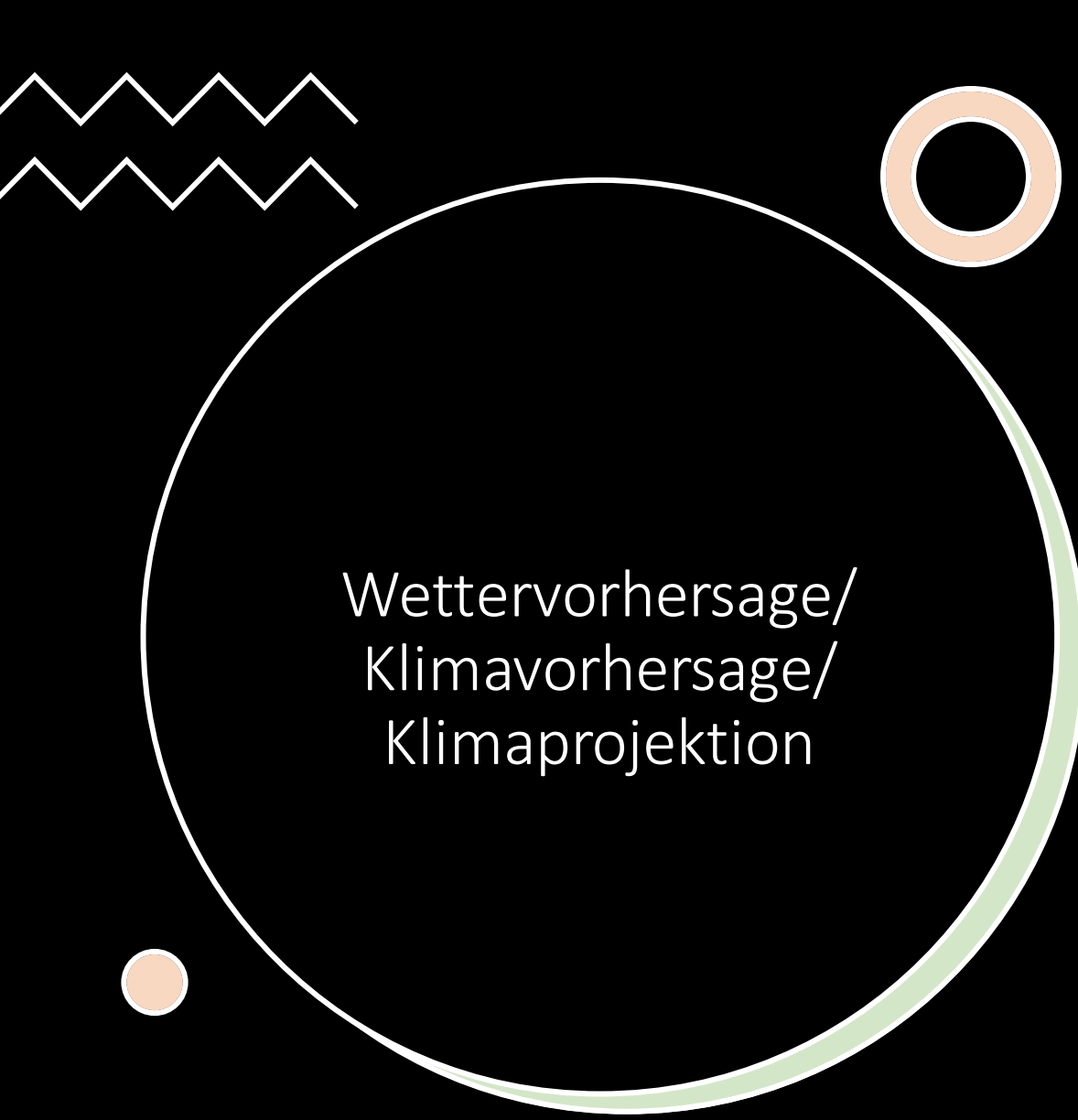


Müssen die Daten noch bereinigt  
werden?



# Problem

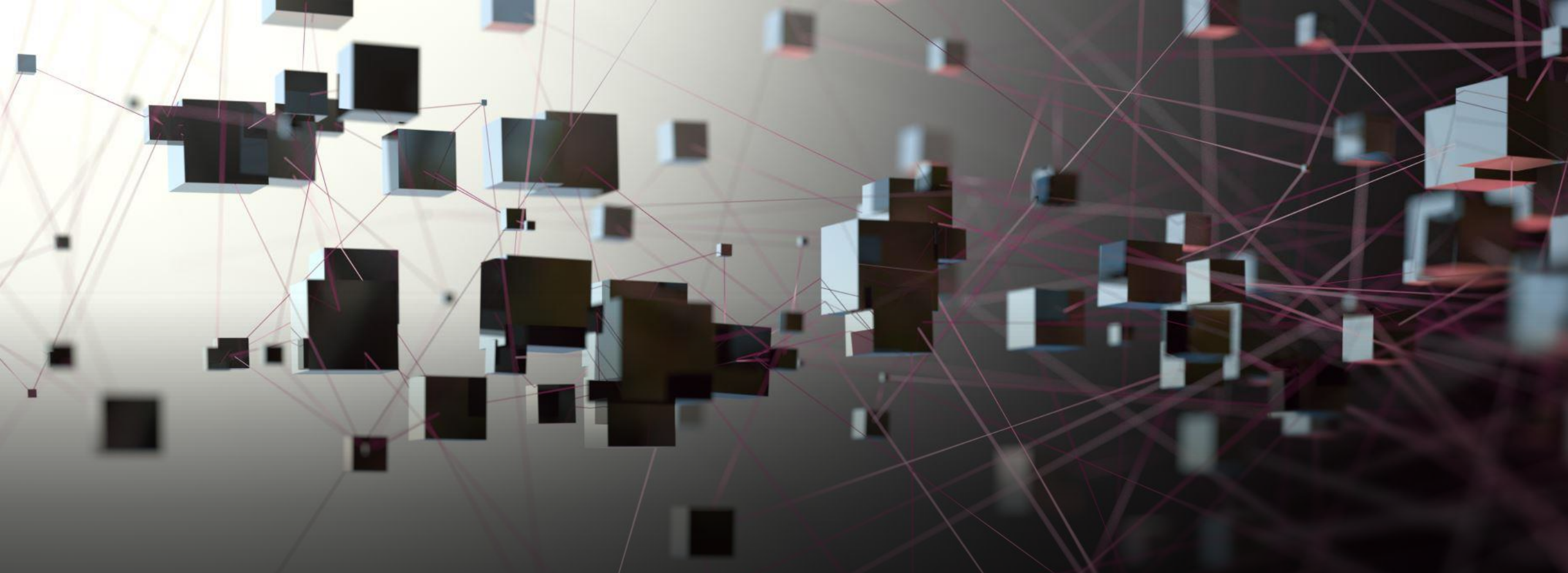
- Algorithmus braucht +10h zum durchlaufen
  - Macht es Sinn Daten von 1948 bis 2021 zu nehmen?
  - Sollte man sich nicht auf Daten der letzten Jahre beschränken (Bez. Klimaveränderung und Temperaturanstieg?)
  - Wenn ja wie viele Jahre?
  - Wie viele Stunden/Tage sagen wir voraus?
- 



## Wettervorhersage/ Klimavorhersage/ Klimaprojektion

- Klassische Wettervorhersage umfasst Zeitraum von 1h bis 14 Tage (kurzfristig)
- Klimavorhersagen umfasst Zeitraum von einigen Wochen bis mehreren Jahren (mittelfristig)
- Klimaprojektion umfasst Zeitraum von 30-100 Jahren (langfristig)
- Klimavorhersagen geben nur grobe Tendenz der Klimaentwicklung über die nächsten Wochen, Monate, Jahre
- Wettervorhersagen geben detaillierte Aussagen über das Wettergeschehen der nächsten Stunde





# Auswahl des Algorithmus



# DLWP



Deep Learning Weather  
Prediction



Vorhersage für 4,5 Tage

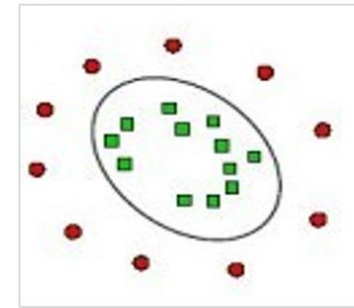


1 –2 Wochen -> sehr schlechte  
Vorhersagen

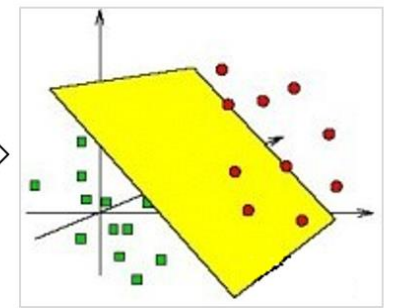
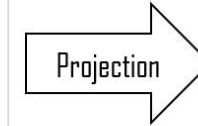


# SVM

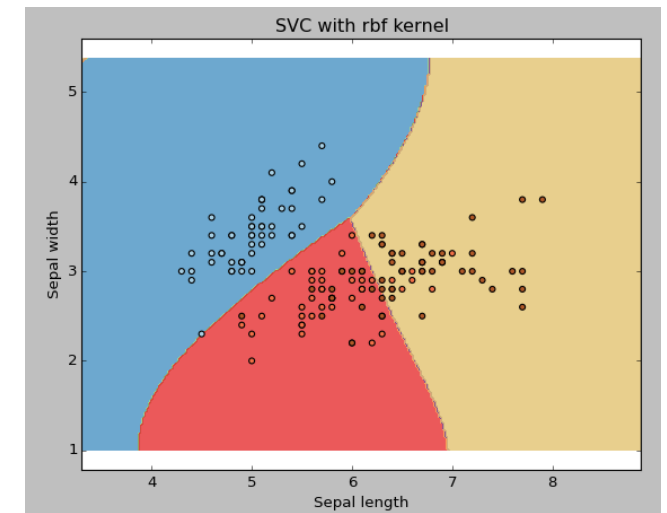
- **Support Vector Machine**
- ermittelten Klassengrenze
- lineare & nicht-lineare
- Mustererkennung
- Supervised Learning



Complex segmentation in low-dimensional space



Easy segmentation in high-dimensional space

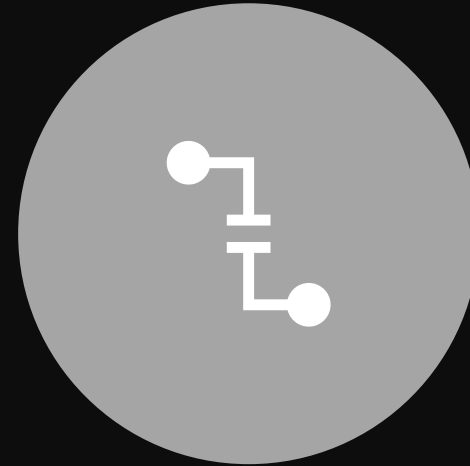


# Lineare Regression -> Support Vektor Machine (rbf)

---



DATEN FÜR EIN JAHR LABELN  
 $24 \cdot 365$



AUFTEILEN IN  
TRAININGS/TESTDATEN

---

Wechsel in VS-Code





# Gamma?

## 4 Jahre als Datenbasis



# Model Scores Daten: 4 Jahren

## $\gamma=0,0001$ vs $\gamma=1$

---

```
Support Vector Regression

#from sklearn.svm import SVR

#Support Vector Regression mit radial basis function

svr_rbf = SVR(kernel="rbf", C=1e3, gamma=0.0001)
svr_rbf.fit(X_train, y_train)

[18] ✓ 22.1s

... SVR(C=1000.0, gamma=0.0001)

#Testen des Modells

model_test = svr_rbf.score(X_test, y_test)
print("Model Score: ", model_test)

[19] ✓ 2.3s

... Model Score: 0.47725488804923033
```

```
Support Vector Regression

#from sklearn.svm import SVR

#Support Vector Regression mit radial basis function

svr_rbf = SVR(kernel="rbf", C=1e3, gamma=1)
svr_rbf.fit(X_train, y_train)

✓ 46.5s

SVR(C=1000.0, gamma=1)

#Testen des Modells

model_test = svr_rbf.score(X_test, y_test)
print("Model Score: ", model_test)

✓ 2.2s

Model Score: 0.4884996369132426
```





Gamma?  
7 Jahre als Datenbasis



# Model Scores Daten: 7 Jahren

## Gamma=0,0001 vs Gamma = 0,1

---

### Support Vector Regression

```
#from sklearn.svm import SVR

#Support Vector Regression mit radial basis function

svr_rbf = SVR(kernel="rbf", C=1e3, gamma=0.0001)
svr_rbf.fit(X_train, y_train)
```

✓ 1m 27.2s

SVR(C=1000.0, gamma=0.0001)

```
#Testen des Models
```

```
model_test = svr_rbf.score(X_test, y_test)
print("Model Score: ", model_test)
```

✓ 8.6s

Model Score: 0.5157642111565306

```
#from sklearn.svm import SVR
```

```
#Support Vector Regression mit radial basis function
```

```
svr_rbf = SVR(kernel="rbf", C=1e3, gamma=0.1)
svr_rbf.fit(X_train, y_train)
```

✓ 1m 47.7s

SVR(C=1000.0, gamma=0.1)

```
#Testen des Models
```

```
model_test = svr_rbf.score(X_test, y_test)
print("Model Score: ", model_test)
```

✓ 8.5s

Model Score: 0.5210969135656605





# Gamma?

## 20 Jahre als Datenbasis



# Model Scores Daten: 20 Jahren

## Gamma = 0,0001 vs Gamma: 0,1

---

```
Support Vector Regression

#from sklearn.svm import SVR

#Support Vector Regression mit radial basis function

svr_rbf = SVR(kernel="rbf", C=1e3, gamma=0.0001)
svr_rbf.fit(X_train, y_train)

[71] ✓ 18m 59.2s

... SVR(C=1000.0, gamma=0.0001)

#Testen des Modells

model_test = svr_rbf.score(X_test, y_test)
print("Model Score: ", model_test)

[72] ✓ 1m 36.9s

... Model Score: 0.5520706659766415
```

```
#from sklearn.svm import SVR

#Support Vector Regression mit radial basis function

svr_rbf = SVR(kernel="rbf", C=1e3, gamma=0.1)
svr_rbf.fit(X_train, y_train)

[57] ✓ 19m 21.4s

... SVR(C=1000.0, gamma=0.1)

#Testen des Modells

model_test = svr_rbf.score(X_test, y_test)
print("Model Score: ", model_test)

[58] ✓ 1m 39.2s

... Model Score: 0.5575878076531018
```





# Forward-Fill 20 Jahre als Datenbasis



# Model Scores Daten: 20 Jahren -999 Werte -> forward fill Gamma= 0,0001 vs Gamma= 0,1

---

```
#from sklearn.svm import SVR

#Support Vector Regression mit radial basis function

svr_rbf = SVR(kernel="rbf", C=1e3, gamma=0.1)
svr_rbf.fit(X_train, y_train)

[217] ✓ 20m 44.9s

... SVR(C=1000.0, gamma=0.1)

#Testen des Modells

model_test = svr_rbf.score(X_test, y_test)
print("Model Score: ", model_test)

[218] ✓ 1m 39.8s

... Model Score: 0.5883114230058979
```

```
#from sklearn.svm import SVR

#Support Vector Regression mit radial basis function

svr_rbf = SVR(kernel="rbf", C=1e3, gamma=0.0001)
svr_rbf.fit(X_train, y_train)

[198] ✓ 19m 45.4s

... SVR(C=1000.0, gamma=0.0001)

#Testen des Modells

model_test = svr_rbf.score(X_test, y_test)
print("Model Score: ", model_test)

[199] ✓ 1m 43.3s

... Model Score: 0.5843055283955371
```

Model Scores Daten: 20 Jahren  
-999 Werte -> forward fill  
Gamma= 0,1  
Vorhersage für: 1 Monat vs 1 Stunde

---

### Support Vector Regression

```
#from sklearn.svm import SVR

#Support Vector Regression mit radial basis function

svr_rbf = SVR(kernel="rbf", C=1e3, gamma=0.1)
svr_rbf.fit(X_train, y_train)
```

16m 56.2s

```
#Testen des Modells

model_test = svr_rbf.score(X_test, y_test)
print("Model Score: ", model_test)
```

Model Score: 0.4709705084625152

### Support Vector Regression

```
#from sklearn.svm import SVR

#Support Vector Regression mit radial basis function

svr_rbf = SVR(kernel="rbf", C=1e3, gamma=0.1)
svr_rbf.fit(X_train, y_train)
```

[45] ✓ 68m 49.7s

SVR(C=1000.0, gamma=0.1)

```
#Testen des Modells

model_test = svr_rbf.score(X_test, y_test)
print("Model Score: ", model_test)
```

[46] ✓ 1m 38.7s

Model Score: 0.9797089468470823



Was bedeutet unser Score?

# Model Score Daten: 6 Jahren

## Gamma = 0,1

### Vorhersage: 7 Tage

---

```
#from sklearn.svm import SVR

#Support Vector Regression mit radial basis function

svr_rbf = SVR(kernel="rbf", C=1e3, gamma=0.1)
svr_rbf.fit(X_train, y_train)
```

✓ 1m 54.4s

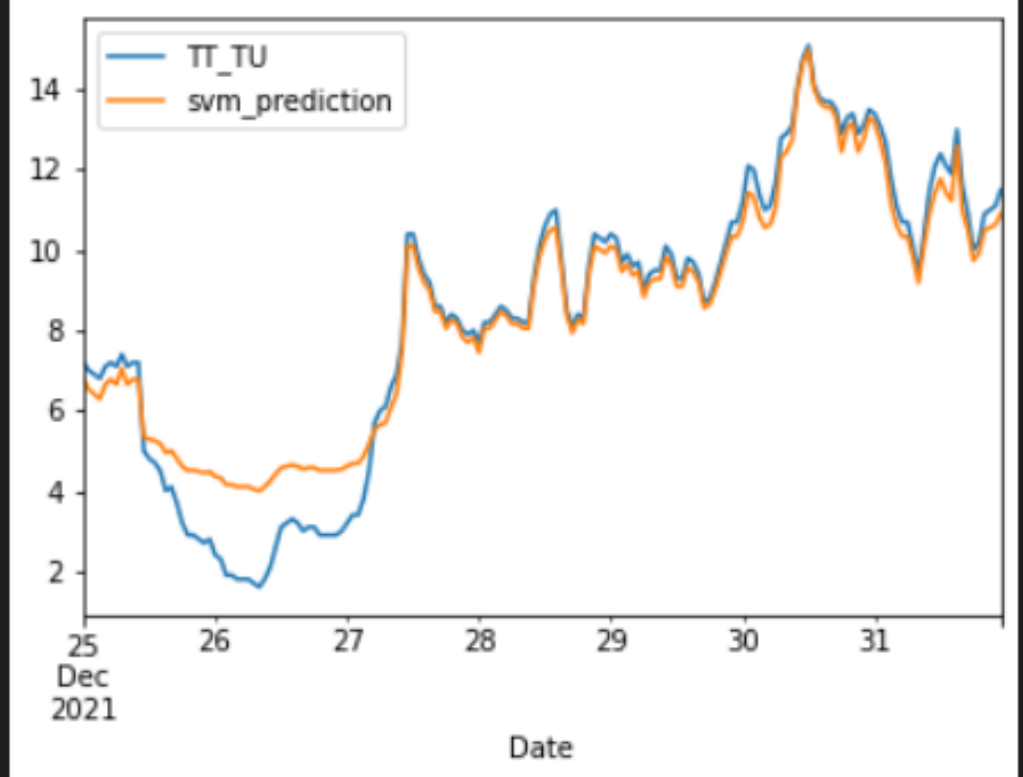
```
SVR(C=1000.0, gamma=0.1)
```

```
#Testen des Modells
```

```
model_test = svr_rbf.score(X_test, y_test)
print("Model Score: ", model_test)
```

✓ 8.3s

Model Score: 0.6287512948527916



# Model Score Daten: 6 Jahren

## Gamma = 0,1

### Vorhersage: 7 Tage

---

```
• #from sklearn.svm import SVR

#Support Vector Regression mit radial basis function

svr_rbf = SVR(kernel="rbf", C=1e3, gamma=0.1)
svr_rbf.fit(X_train, y_train)
```

✓ 0.5s

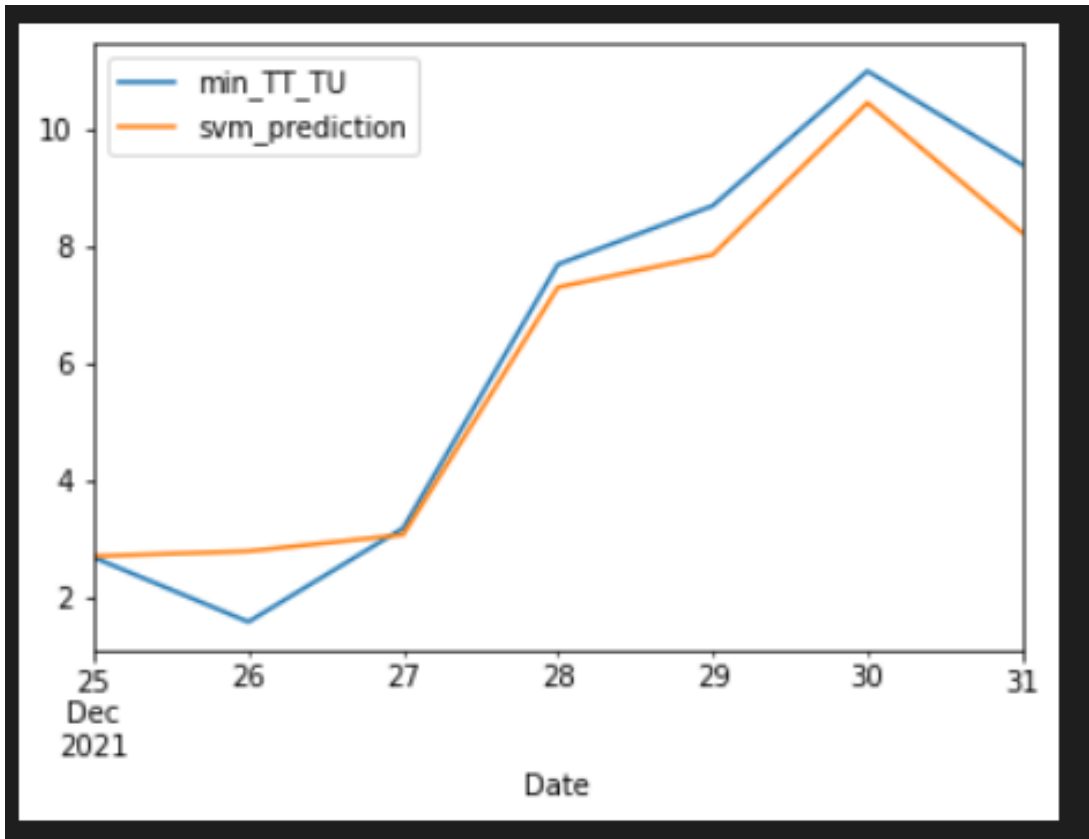
```
SVR(C=1000.0, gamma=0.1)
```

```
#Testen des Modells
```

```
model_test = svr_rbf.score(X_test, y_test)
print("Model Score: ", model_test)
```

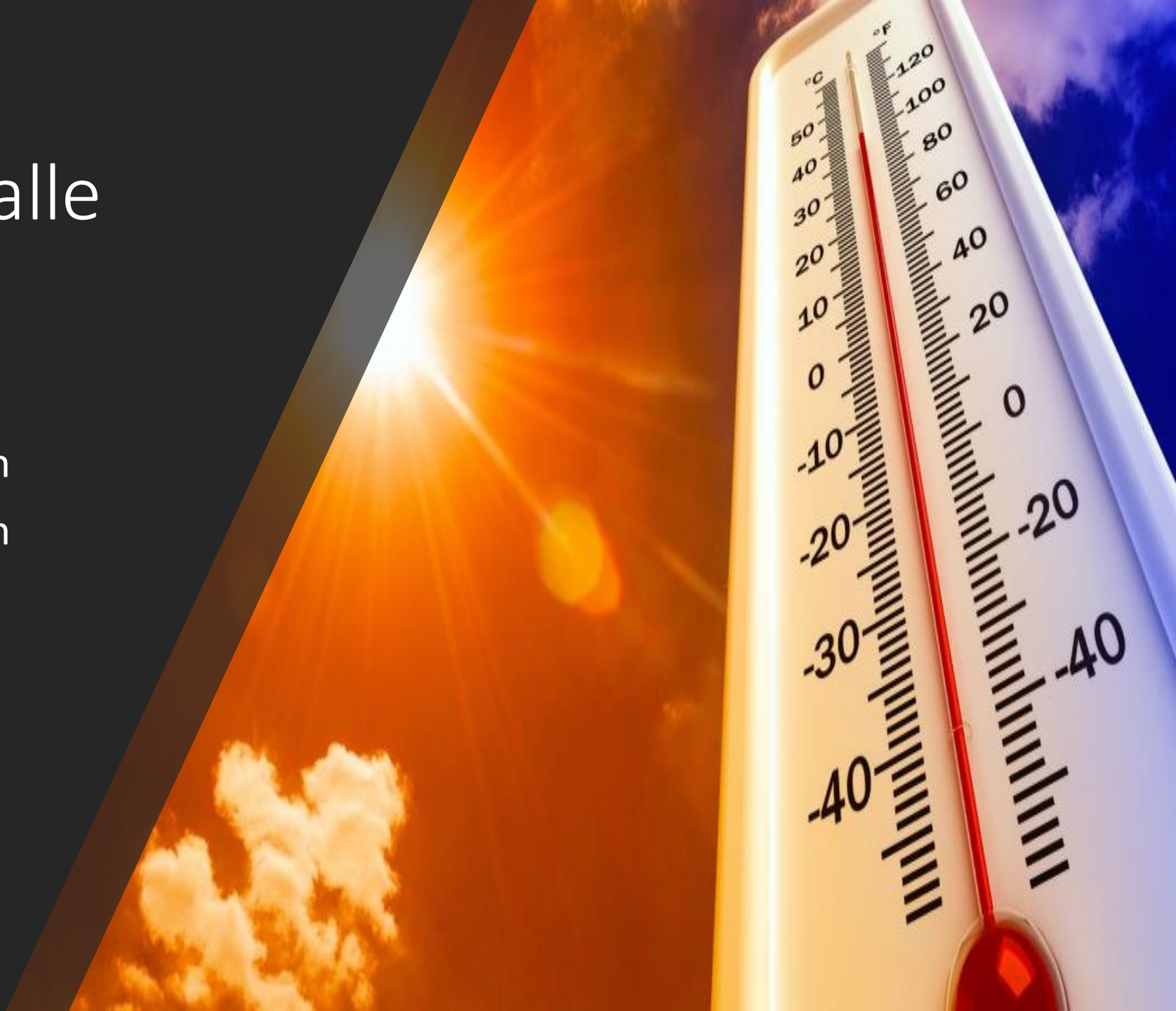
✓ 0.4s

```
Model Score: 0.6218844037539035
```



# Vergleich der Zukunftsintervalle

- 1 Tag
- 7 Tage -> 1 Woche
- 14 Tage -> 2 Wochen
- 28 Tage -> 4 Wochen
- 365 Tage -> 1 Jahr
- 1460 Tage -> 4 Jahre



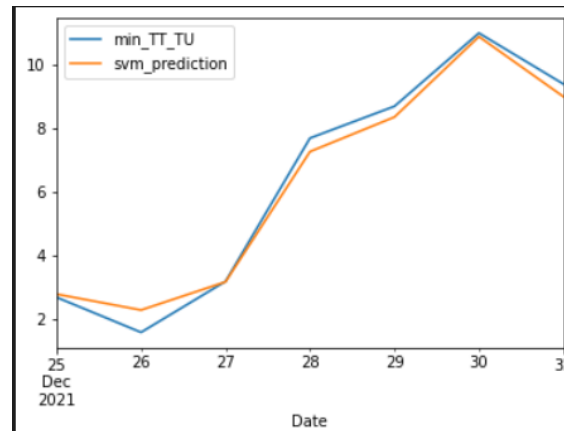


# Vorhersagen mit SVM\_MIN\_MAX

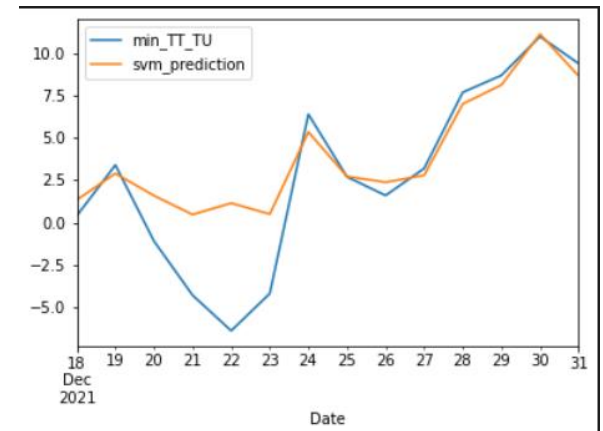
[8.80160753]

7662 9.4

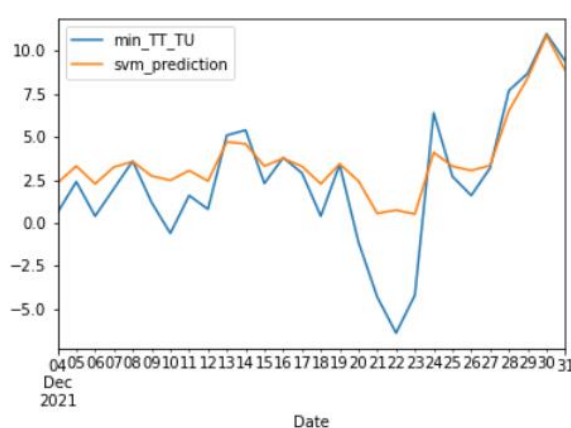
Model Score: 0.8267743055176537



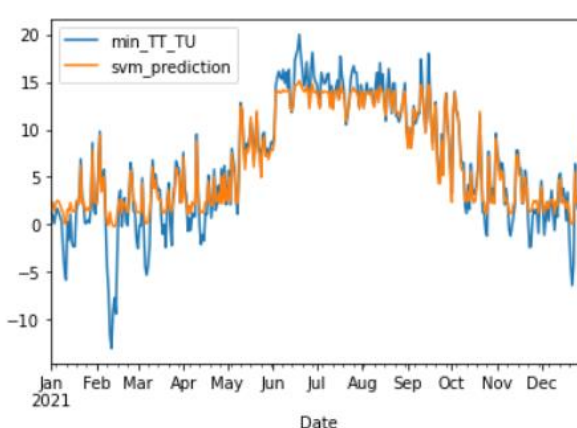
Model Score: 0.5584104829328974



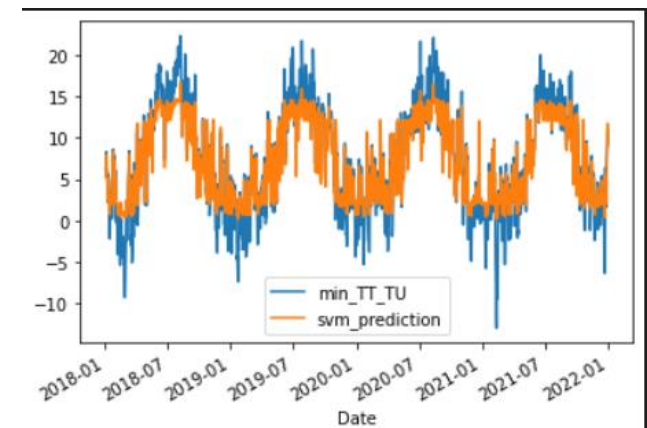
Model Score: 0.518815098186095



Model Score: 0.40307081309863735



Model Score: 0.5413956592011265



Model Score: 0.5311766626870413

# Zusammenfassend

- Mehr Daten -> Score besser  
48% -> 52
- -999 Werte durch vorherige Werte ersetzen  
statt diese zu löschen:  
52.02% -> 58.43%
- Umso geringer die Vorhersage Dauer ->  
Score umso besser
- Schlechter Score -> keine große  
Aussagekraft
- Datenbasis verstehen



**ERFOLGSFAKTOR**

Wirtschaftlicher Kontext  
(Use Case)

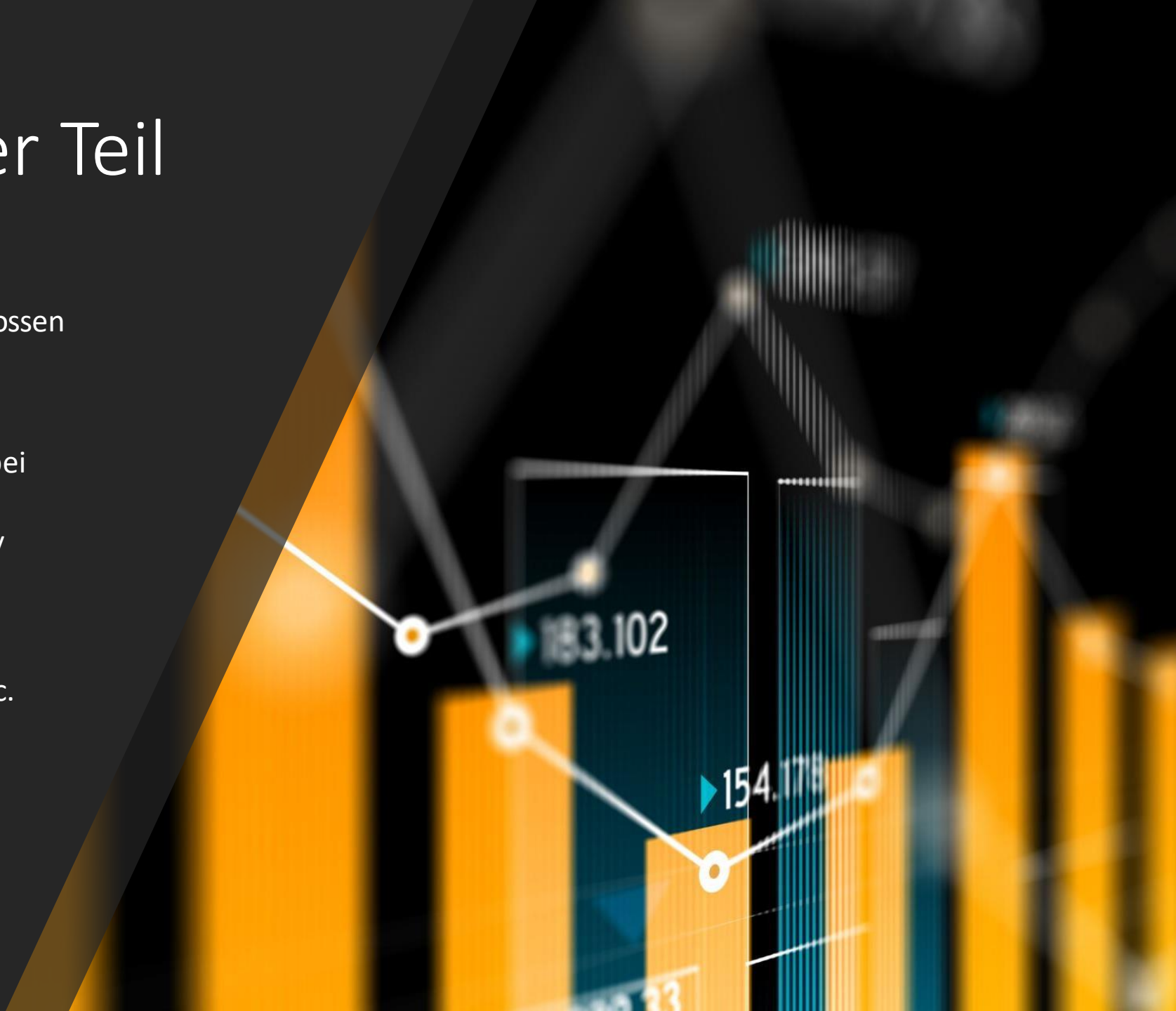
**WETTER**

# Wirtschaftlicher Teil

- Minimum Viable Product abgeschlossen (geringst mögliches Produkt)

## Next Steps:

- Weiterentwicklung des Produktes bei Investment
- Parameter (Feuchtigkeit/Luftdruck/Sonnenstunden/etc.)
- Vermarktung des Produktes an Landwirtschaft, Festival Planung etc.
- Hauptzielgruppen: Erneuerbare Energien / Luftfahrt





# Ausblick: Erneuerbare Energien / Luftfahrt

- Für erneuerbaren Energien: Windstärke / Windrichtung
  - > damit Windräder richtig ausgerichtet werden können
  - > führt zur besseren Versorgung / Strom wird billiger
- Für die Luftfahrt: Windstärke / Windrichtung
  - > Flugzeuge können ihre Tankreserve reduzieren
  - > Große Flugzeugflotte --> hohe Kosteneinsparung






# Wirtschaftlicher Kontext

- **Wettervorhersage wichtig für die Wirtschaft**

## **Hauptziel:**

- **Kosten zu senken von aktuellem Verfahren durch Ressourceneinsparung**
  - **Bessere Ergebnisse?**
- 



**LESSONS LEARNED**



# Lessons Learned

- Datenbasis beeinflusst den Score stark
- Mehr Daten -> Score besser
- Optimale Parameterwahl
- Sinnvollhaftigkeit von längeren Vorhersagen
- Werte ersetzen statt diese zu löschen -> besserer Score
- Daten immer visualisieren / greifbar machen
- Datenbasis verstehen





# Ergebnisse



# Ergebnisse

- Score von **97,9%** bei Vorhersage von **1h**
- Score von **55,8%** bei Vorhersage von **7d**
- Score von **51,9%** bei Vorhersage von **14d**
- Score von **40,3%** bei Vorhersage von **1m**
- Score von **58,4%** bei Vorhersage von **1y**
- Score von **53,1%** bei Vorhersage von **4y**



Danke für eure  
Aufmerksamkeit!

Noch Fragen?