```
!pip install scikit-learn
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import Ridge
```

```
Requirement already satisfied: scikit-learn in /opt/conda/lib/python3.12/site-packages (1.7.2)
Requirement already satisfied: numpy>=1.22.0 in /opt/conda/lib/python3.12/site-packages (from scikit-learn) (2.3.5)
Requirement already satisfied: scipy>=1.8.0 in /opt/conda/lib/python3.12/site-packages (from scikit-learn) (1.16.3)
Requirement already satisfied: joblib>=1.2.0 in /opt/conda/lib/python3.12/site-packages (from scikit-learn) (1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /opt/conda/lib/python3.12/site-packages (from scikit-learn) (3.6.0)
```

```
import os
os.getcwd()
```

```
'/work/bda2/notebooks'
```

```
df_merged = pd.read_csv("../datasets/df_merged.csv")
```

```
'''
Purpose of the model planning phase:
- Identify which input features are relevant for the predictive task
- Decide which models to experiment with and justify the choices
- Define the hyperparameter search strategy
- Determine how the model will be evaluated (metrics, validation approach)
'''
```

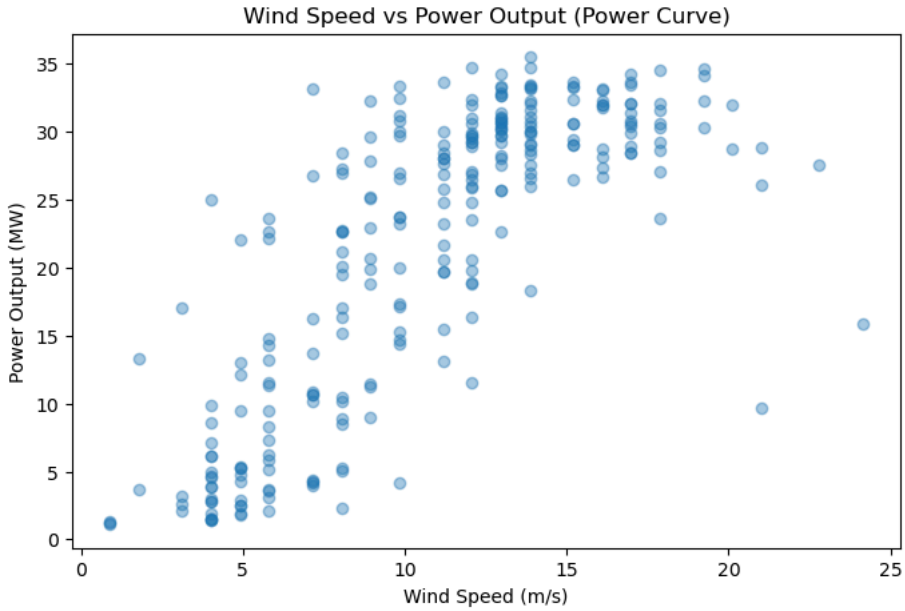
```
'\nPurpose of the model planning phase:\n- Identify which input features are relevant for the predictive task \n- Decide which models to experiment with and justify the choices\n- Define the hyperparameter search strategy\n- Determine how the model will be evaluated (metrics, validation approach)\n'
```

```
'''
Identify which input features are relevant for the predictive task
 --> given in the assignment with wind speed and wind direction
'''
```

```
'\nIdentify which input features are relevant for the predictive task \n --> given in the assignment with wind speed and wind direction\n'
```

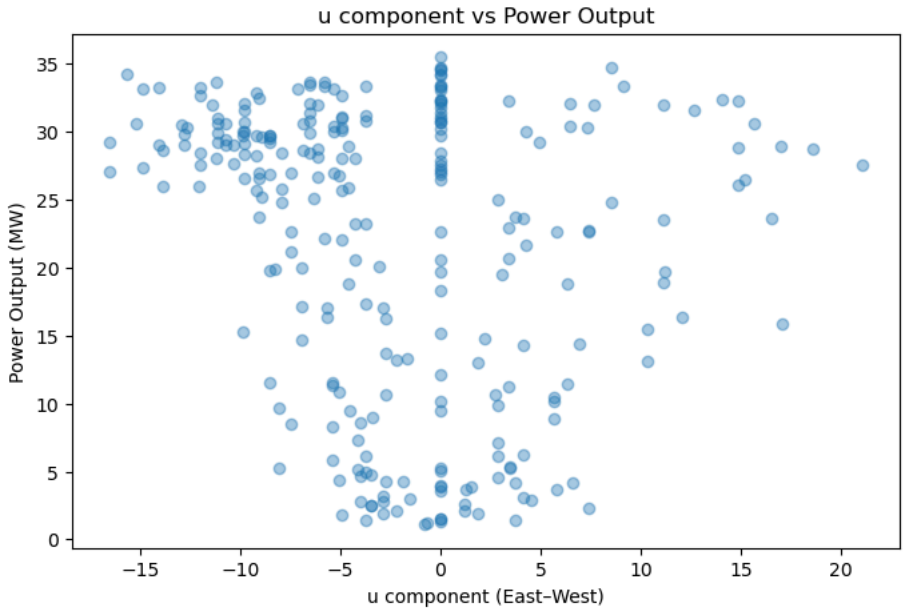
```
# Decide which models to experiment with and justify the choices
```

```
#plotting wind speed vs power generation

plt.figure(figsize=(8,5))
plt.scatter(df_merged["Speed"], df_merged["Total"], alpha=0.4)
plt.xlabel("Wind Speed (m/s)")
plt.ylabel("Power Output (MW)")
plt.title("Wind Speed vs Power Output (Power Curve)")
plt.show()
```



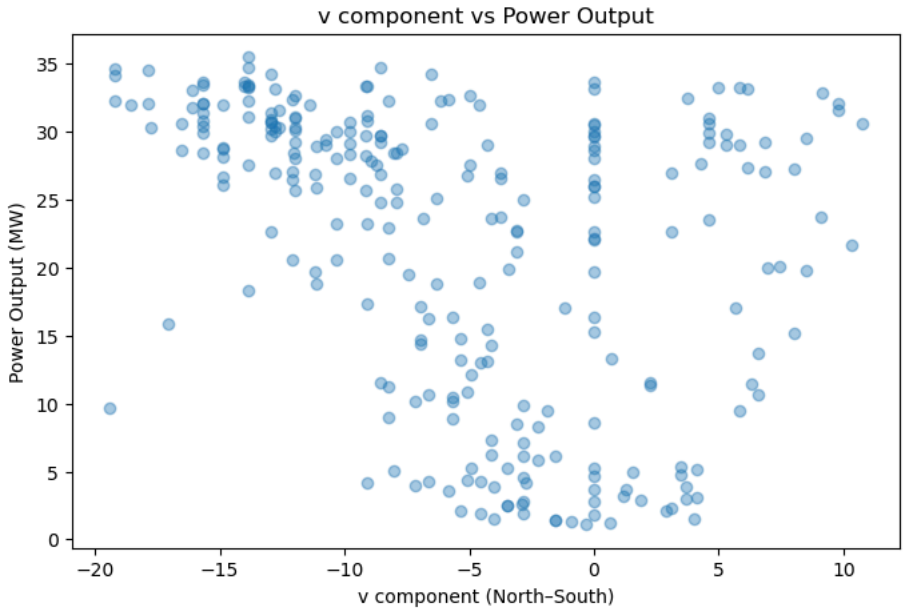

```
# sigmoud shape (like in assignment description), i.e., not linear
```

```
#plotting u vs power generation
# u is x axis and refers to x axis (east/west)
plt.figure(figsize=(8,5))
plt.scatter(df_merged["u"], df_merged["Total"], alpha=0.4)
plt.xlabel("u component (East-West)")
plt.ylabel("Power Output (MW)")
plt.title("u component vs Power Output")
plt.show()
```



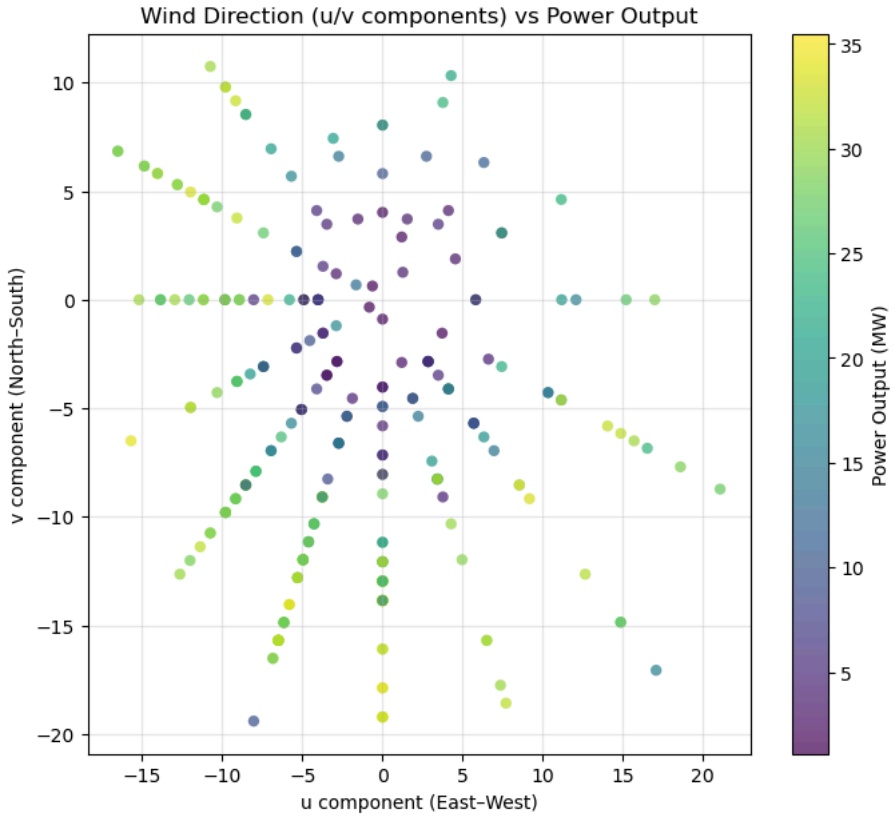

```
#--> not a "clear" trend visible
```

```
#plotting v vs power
# v refers to y axis (south/north)
plt.figure(figsize=(8,5))
plt.scatter(df_merged["v"], df_merged["Total"], alpha=0.4)
plt.xlabel("v component (North-South)")
plt.ylabel("Power Output (MW)")
plt.title("v component vs Power Output")
plt.show()
```




```
#--> not "really" a relationship between v component and power output visible
```

```
#another approach is to combine both u and v as a scatterplot with the power output as color

plt.figure(figsize=(8, 7))

scatter = plt.scatter(
    df_merged["u"],
    df_merged["v"],
    c=df_merged["Total"],
    cmap="viridis",
    alpha=0.7,
    edgecolor="none"
)

# Farbskala
cbar = plt.colorbar(scatter)
cbar.set_label("Power Output (MW)")

# Achsenbeschriftungen
plt.xlabel("u component (East-West)")
plt.ylabel("v component (North-South)")
plt.title("Wind Direction (u/v components) vs Power Output")

# Gitter zur Orientierung
plt.grid(alpha=0.3)

plt.show()
```

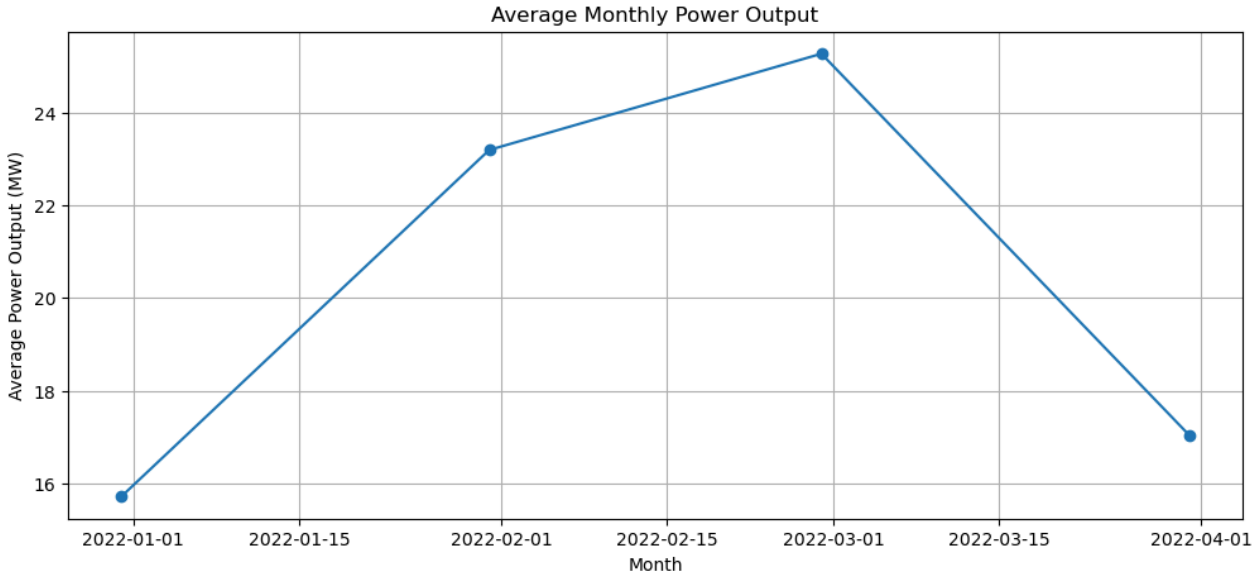



```
'''
The 2D scatter of u and v (wind vector components) colored by power output shows that the highest power values occur for la
This confirms that wind speed is the dominant feature, while wind direction exhibits only a secondary influence.
Certain directional quadrants show slightly higher average output, likely due to terrain or turbulence effects,
but no strong directional dependency is visible overall.

'''
```

```
'\nThe 2D scatter of u and v (wind vector components) colored by power output shows that the highest power values occur for
large absolute vector magnitudes, regardless of direction.\nThis confirms that wind speed is the dominant feature, while
wind direction exhibits only a secondary influence.\nCertain directional quadrants show slightly higher average output,
likely due to terrain or turbulence effects,\nbut no strong directional dependency is visible overall.\n\n'
```

```
#plotting power output against time to check, if there are any recurrent trends
#changing dtype of target_time to dateime
df_merged["target_time"] = pd.to_datetime(df_merged["target_time"])
#since I want to see the plot on a monthly basis, I merge the time values for each month and use the mean
df_temp = df_merged.set_index("target_time")
df_monthly = df_temp.resample("M")["Total"].mean()

plt.figure(figsize=(12,5))
plt.plot(df_monthly.index, df_monthly.values, marker='o')
plt.xlabel("Month")
plt.ylabel("Average Power Output (MW)")
plt.title("Average Monthly Power Output")
plt.grid(True)
plt.show()
```

```
/tmp/ipykernel_758/2549479513.py:6: FutureWarning: 'M' is deprecated and will be removed in a future version, please use 'ME
  df_monthly = df_temp.resample("M")["Total"].mean()
```




```
#checking if time/ x-axis covers all months
df_merged.iloc[[0, -1]]

'''
The monthly average plot aligns with the temporal range of the dataset,
but it does not reveal any meaningful recurring pattern or seasonal structure.
Power output fluctuates month-to-month without showing a consistent upward
or downward trend.

This indicates that the time dimension does not contain predictive temporal
patterns (such as seasonality or autocorrelation) that time-series models
like ARIMA or RNNs would rely on.

Therefore, classical time-series forecasting models are not appropriate
for this task. Instead, we focus on supervised learning models that predict
power output directly from the weather features (wind speed and direction).
'''
```

```
'\nThe monthly average plot aligns with the temporal range of the dataset, \nbut it does not reveal any meaningful
recurring pattern or seasonal structure.\nPower output fluctuates month-to-month without showing a consistent upward \nor
downward trend.\n\nThis indicates that the time dimension does not contain predictive temporal \npatterns (such as
seasonality or autocorrelation) that time-series models \nlike ARIMA or RNNs would rely on. \n\nTherefore, classical time-
series forecasting models are not appropriate \nfor this task. Instead, we focus on supervised learning models that predict
\npower output directly from the weather features (wind speed and direction).\n'
```

```
'''
Conclusion regarding the type of relationship:
The strongest and most meaningful relationship exists between wind conditions (especially wind speed)
and power generation. As mentioned in the assignment, the power curve follows a non-linear,
sigmoid-like behaviour. Therefore, linear models such as Linear Regression are not appropriate.

For this reason, the assignment explicitly recommends using a tree-based model,
specifically a Random Forest Regressor, because it naturally captures non-linear relationships
and handles continuous data effectively.

As a second model, a Gradient Boosting Regressor is employed.Both Random Forest and Gradient Boosting belong to the family
of tree-based ensemble methods, but they differ fundamentally in how they construct their ensembles.
Random Forest trains many decision trees in parallel, each independently contributing to the final prediction.
In contrast, Gradient Boosting builds trees sequentially, where each new tree is trained to correct the residual errors
of the previous ones. This sequential error-refinement often leads to higher predictive accuracy, especially for complex,
non-linear relationships.
'''
```

```
'\nConclusion regarding the type of relationship:\nThe strongest and most meaningful relationship exists between wind
conditions (especially wind speed) \nand power generation. As mentioned in the assignment, the power curve follows a non-
linear, \nsigmoid-like behaviour. Therefore, linear models such as Linear Regression are not appropriate.\n\nFor this
reason, the assignment explicitly recommends using a tree-based model, \nspecifically a Random Forest Regressor, because it
naturally captures non-linear relationships \nand handles continuous data effectively.\n\n(As a second model, gradient
boosting regressor will be used.)\n'
```

```
df_merged.head()
```

	target_time	Direction	Lead_hours	Source_time	Speed	Total	dir_deg	dir_rad	u	v
0	2021-12-11 15:00:00+00:00	SSE	1	1639227600	11.17600	27.621048	157.5	2.748894	-10.325278	4.276870e+00
1	2021-12-11 18:00:00+00:00	SSW	1	1639238400	8.04672	21.135542	202.5	3.534292	-7.434200	-3.079346e+00
2	2021-12-11 21:00:00+00:00	WSW	1	1639249200	11.17600	20.616209	247.5	4.319690	-4.276870	-1.032528e+01

```
# Define the hyperparameter search strategy
'''
For defining the hyperparameter strategy for the random forest regressor,
I used this website as inspiration: https://scikit-learn.org/0.17/modules/generated/sklearn.ensemble.RandomForestRegressor.
n_estimators(number of trees): since df_merged is rather small, I use 100 as baseline, 300 and 500. Note: the model just ge
and does not overfit, when there are "too" many trees.
criterion (loss function to minimise): since I want to minimze the MSE (see below), I use squared_error.
random_state is on to make the model reusable with same results.
bootstrapping is on different trees have different samples of the data. bootstrapping = false would make sense, if the data
This is because by sequential data the order matters.
max_depth controls  how deep each tree is allowed to grow.I test None, 10, and 20 to compare unrestricted trees with
moderately regularised ones.
min_samples_leaf defines the minimum number of samples required in a leaf node.I use 1, 3, and 5 to evaluate
how leaf size influences smoothness and variance reduction.
```