IT UNIVERSITY OF COPENHAGEN
BIG DATA MANAGEMENT 2021

Assignment 1: Data Pipelines

Forecasting the Wind Power Production in Orkney

September 21, 2021

1 INTRODUCTION

In this assignment you will design and implement a data preprocessing pipeline for a small wind energy forecasting system using `sklearn`. You'll be using data from Orkney, an archipelago off the north eastern coast of Scotland. Orkney has around 20,000 inhabitants and they produce around ~ 120% of their annual energy consumption in wind energy.

This document will give you an extremely brief intro to wind power forecasting, describe the data you will be working with, list the requirements for the assignment and provide suggested reading and useful links. This document also includes the `template.py` file to will help you get started.

1.1 Wind power forecasting

As you might expect, the estimated wind speed is the primary input to any model that seeks to estimate wind power production. If we make a scatter plot of wind speed against power output, it might look like Figure 1.1. If you squint at the data, you'll see a sigmoid-ish curve, called the "power curve". The power curve is essential for wind power forecasting, as it describes the relationship between wind speeds and power output.

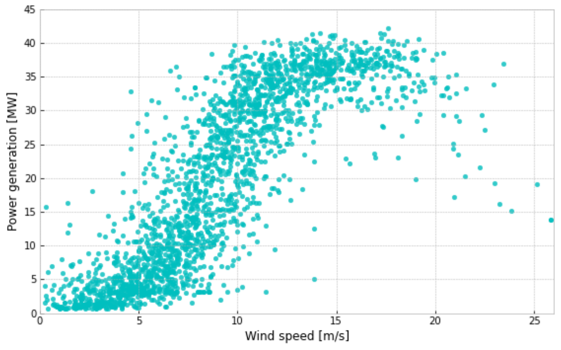

Figure 1.1: Wind speed plotted against power generation, showing a "power curve" for Orkney

But the wind might meet different environments, depending on which direction the wind is coming from, affecting the wind speed or creating turbulence. So we should probably include the the wind direction as a feature in our model, increasing the dimensionality of the input.

The wind direction is naturally circular and is currently encoded as a string in our data. However, our models only accept numerical input, so we need to process the data to extract useful features. We can either map the text string to degrees or radians, or encode them as a categorical (or "one-hot") vector. This is up to you to decide.

2 The Data

You will be working with data from two sources:

1. Orkney's renewable power generation - Sourced from Scottish and Souther Electricity Networks (SSEN)
2. Weather forecasts for Orkney- Sourced from the UK MetOffice

The data is stored in an InfluxDB, which is a non-relational time-series database. InfluxDB can be queried using InfluxQL, a "SQL-like" query language for time-series data. InfluxDB does not have tables with rows and columns, instead data is stored in *measurements* with has *fields* and *tags*[1].

2.1 Renewable energy generation

The power generation data is sampled every minute from a public website and is stored in the measurement `Generation` with the following "schema":

```
name: Generation
Key       Type
--------  ---------
time      float     (Time of measurement)
ANM       float     (Not relevant for this assignment)
Non-ANM   float     (Not relevant for this assignment)
Total     float     (Renewable power generation in MegaWatts)
```

2.2 Weather forecasts

The weather forecasts are sourced from the UK governmental weather service the MetOffice. For forecasts we talk about "Source time", the time at which the forecast was generated, "target time", the time that is forecasted, and "lead time" or "forecast horizon", the difference between the source time and the target time. There are 3 hours between each timestamp.

The forecasts are stored in the measurement `MetForecasts` with the following "schema":

```
name: MetForecasts
Key          Type
--------     ---------
time         float     (Target time of forecasts)
Speed        float     (Wind speed in M/S)
Direction    string    (Wind direction, e.g. "S" or "NW")
Source_time  integer   (Time of forecast generation)
Lead_hours   string    (Forecast horizon in hours, actually a tag)
```

[1]Tags are always strings and are indexed, while fields are not indexed

3 Requirements and Hand-in

3.1 System requirements

For this assignment you should create the data preproccesing pipeline using `sklearn` in a system that:

- Reads the latest data from the InfluxDB
- Prepares the data for model training, including
  - Aligning the timestamps of the two data sources (e.g. through resampling)
  - Handling missing data
  - Altering the wind direction to be a usable feature (by mapping to radians or encoding as categorical)
  - Scaling the data to be within a set range
- Trains a linear regression model
- Uses the model to forecast wind power production

Much of the scaffolding for this is provided in `template.py`, so you primarily need to focus on the pipeline.

3.2 Hand-in

You should hand in a report describing your solution, including which design choices and trade-offs you made. Which steps does your pipeline include? What is the format of the data once it reaches the model? How could the pipeline/system be improved?

3.3 Suggested reading and useful links

- Page 66-76 of Hands-On Machine Learning, especially the difference between estimators, transformers and predictors.[2]
- User guide for sklearn pipelines
- sklearns Pipeline API
- sklearns LinearRegression model

3.4 Possible extensions (Very optional)

If you think wind power forecasting is extremely interesting and soo much fun (which you of course do, i mean, why wouldn't you?), here's some ideas for extending your model and pipeline. These aren't especially connected to the course, so there is no expectation for you to do any of these.

- The power curve in Figure 1.1 does not look especially linear, does it? So maybe the linear regression is not a great fit. Add polynomial features to the wind speed input to handle the non-linearity.
- How accurate is your model? Evaluate your model using non-shuffled K-fold cross validation
- What works better, converting the wind direction to radians or to a categorical encoding?
- The current model is oblivious to the time sequence of the data. Exchange the linear regression model with a Auto-Regressive Moving-Average eXogenous input (ARMAX) model or a Recurrent Neural Network with Long Short-Term Memory cells

[2] The whole book is highly recommended, if you would like to get started with ML

4 Code Template

```python
## Getting the data

from influxdb import InfluxDBClient # install via "pip install influxdb"
import pandas as pd

client = InfluxDBClient(host='influxus.itu.dk', port=8086, username='lsda',
     password='icanonlyread')
client.switch_database('orkney')

def get_df(results):
    values = results.raw["series"][0]["values"]
    columns = results.raw["series"][0]["columns"]
    df = pd.DataFrame(values, columns=columns).set_index("time")
    df.index = pd.to_datetime(df.index) # Convert to datetime-index
    return df

# Get the last 90 days of power generation data
generation = client.query(
    "SELECT * FROM Generation where time > now()-90d"
    ) # Query written in InfluxQL

# Get the last 90 days of weather forecasts with the shortest lead time
wind  = client.query(
    "SELECT * FROM MetForecasts where time > now()-90d and time <= now()
    and Lead_hours = '1'"
    ) # Query written in InfluxQL

gen_df = get_df(generation)
wind_df = get_df(wind)


###################################################
## Preprocess the data

from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression

# Align the data frames

pipeline = Pipeline([
    # This is your job - make a pipeline.
    # This is also where you add your model
])

# Fit the pipeline


###################################################
## Do forecasting

# Get all future forecasts regardless of lead time
forecasts  = client.query(
    "SELECT * FROM MetForecasts where time > now()"
    ) # Query written in InfluxQL
for_df = get_df(forecasts)

# Limit to only the newest source time
newest_source_time = for_df["Source_time"].max()
newest_forecasts = for_df.loc[for_df["Source_time"] == newest_source_time].
    copy()

# Preprocess the forecasts and do predictions in one fell swoop
# using your pipeline
pipeline.predict(newest_forecasts)
```