Assignment 5 - ML diagnosis

Suppose we train a model and it does well on the training set, after that we have to assess if it can generalize well to unseen data. We can diagnose the quality of ML methods just by comparing training E_t with validation E_v errors. This diagnosis can be further developed if a benchmark (or reference) error level E_{ref} exists. You can read more about it in **Section 6.6** of the course book.

There are several cases of errors comparison:

1. $E_t \approx E_v \approx E_{ref}$: All the errors are on the same level, so the ML method works rather good and there is no need to improve it further.

2. $E_t \ll E_v$: The validation error is larger than the the training error. That means that the model faces overfitting. That can be solved either by reducing the effective size of the hypothesis space or increasing the effective number of training data points. **That is the most common case which is checked in the process of model learning.**

3. $E_t \approx E_v \gg E_{ref}$: Training and validation errors are same, but they are much larger than the reference error. In that case the model does not overfit, but it in general performs bad, i.e. it is underfitting the training data. Two commonly used strategies to conquer underfitting are choosing a larger hypothese space and collecting more relevant features.

4. $E_t \gg E_v$: The situation of having training error significantly larger than validation one is not so common. The increase of training and validation sets sizes by collecting more labeled data points or using data augmentation can increase the performance here. On the contrary, $E_t > E_v$ is not rare, even if the dataset is not small, you are highly likely to see it in your practicing. It usually is due to unlucky splitting combined with underfitting, the former can be mitigated by Kfold CV, the latter may be solved by enlarging hypothesis space and/or collecting more relevant features.

Learning goals

After successfully completing this assignment, you should:

- Understand the basic workflow of training, selecting and testing ML models.
- Be aware of random fluctuations in validation error (due to unlucky splits).
- Know some means for getting a baseline/benchmark/reference value to compare against the obtained average loss on validation/test set.

In [1]:

```
# Import basic libraries needed for this assignment
%config Completer.use_jedi = False  # enable code auto-completion
import numpy as np #import numpy to work with arrays
import pandas as pd #import pandas to manipulate the dataset
from matplotlib import pyplot as plt #import the module matplotlib.pyplot to do
 visulization
from sklearn.model_selection import train_test_split, KFold
from sklearn.preprocessing import PolynomialFeatures    # function to generate p
olynomial and interaction features
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

# Task A5.1:

The data in the correctly organized ML workflow should be divided into three main parts: **training set**, **validation set**, and **test set**. Why do we need to have validation and test parts separately? The validation set is a separate section of dataset that is used to evaluate the model trained on the training set to get a sense of how well the model works. Validation set is heavily used in model developing and selecting. Model can be diagnosed as overfitted or underfitted when we compare training error with the validation error. We use the diagnosis result to instruct model developing, such as enlarge the hypothesis space, weaken/strengthen regularization strength, etc. Anyhow the validation set has affected the model, in other words, the model has also seen the data in validation set, this is why it is important to hold back a completely separate data - the test set to check the final model performance.

In this assignment, you will work with the same data source as in Assignment 4. Here the same steps as in Assignment 4.1 are done to prepare data for model training in Assignment 5.1. But now instead of using just train/validation split, train/validation/test split will be used and examined.

In this task you have to:

- Download and prepare data for future model training.
- Use the function train_test_split() provided by Scikit-learn's model_selection module two times to split the dataset into train, validation and test set. At first data is split into train and remaining groups, then the remaining group is split into validation and test.
- Create Polynomial regression models with different degrees and report train, validation and test errors.

Downloading and processing the dataset

In [2]:

```
# Read in the data stored in the file 'FMIData_Assignment5.csv'
df = pd.read_csv('FMIData_Assignment5.csv')
df.drop(columns=['Time zone','Precipitation amount (mm)','Snow depth (cm)','Air
 temperature (degC)',\
                'Ground minimum temperature (degC)'],inplace=True)  # drop unre
levant columns
df.columns =['year','m','d','time','max temperature','min temperature'] # rename
columns

# Print the first 5 rows of the DataFrame 'df'
df.head(5)
```

Out[2]:

	year	m	d	time	max temperature	min temperature
0	2020	1	1	00:00	3.4	-2.6
1	2020	1	1	06:00	NaN	NaN
2	2020	1	2	00:00	5.1	1.8
3	2020	1	2	06:00	NaN	NaN
4	2020	1	3	00:00	5.7	4.3

In [5]:

```
# Select only weather recordings whose property 'time' is equal to `00:00`
FMIRawData = df[df['time'] == '00:00']

# Create the feature and label vectors
X = FMIRawData['min temperature'].to_numpy().reshape(-1, 1)
y = FMIRawData['max temperature'].to_numpy()
```

Splitting dataset into train/validation/test

In [10]:

```
## Split the dataset into a training set and a remaining set with train_test_spl
it.
## The test_size=0.5 and random_state=42:
X_train, X_rem, y_train, y_rem = train_test_split(X, y, test_size=0.5, random_st
ate=42)
```

In [11]:

```
# Sanity check the percentage of the remainig data up to 1%
assert np.isclose(0.5, X_rem.shape[0] / X.shape[0], atol=0.01)
```

In [12]:

```
# this is the 1st test cell for A5.1, please leave it as it is.
```

In [16]:

```
## Split the remaining dataset into a validation set and test set with train_tes
t_split.
## The test_size=0.2 and random_state=42:
X_val, X_test, y_val, y_test = train_test_split(X_rem, y_rem, test_size=0.2, ran
dom_state=42)
```

In [17]:

```
# Sanity check the percentage of the remainig data up to 1%
assert np.isclose(0.2, X_test.shape[0] / X_rem.shape[0], atol=0.01)
```

In [18]:

```
# this is the 2nd test cell for A5.1
```

After completing A5.1 task three datasets are created:
X_train and y_train
X_val and y_val
X_test and y_test

They will be used in the model training/validation/testing of Assignment A5.2.

Task A5.2:

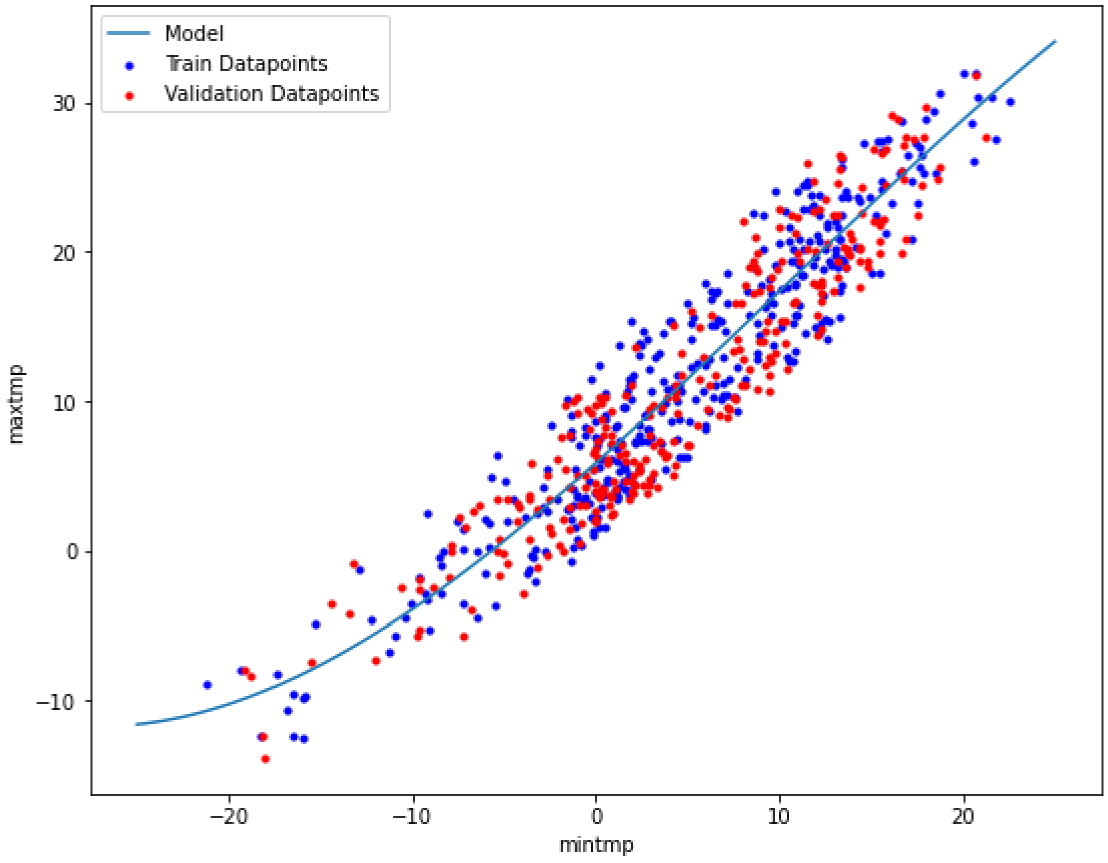
The goal of this task is to train polynomial regression models with different polynomial degrees. The models are trained on training dataset, after that optimal poly degree is chosen with the help of validation errors. Final error is computed on test set using the learnt polynomial regression model with optimal degree. This task is an extension of task A5.1.

Polynomial models' validation errors calculation

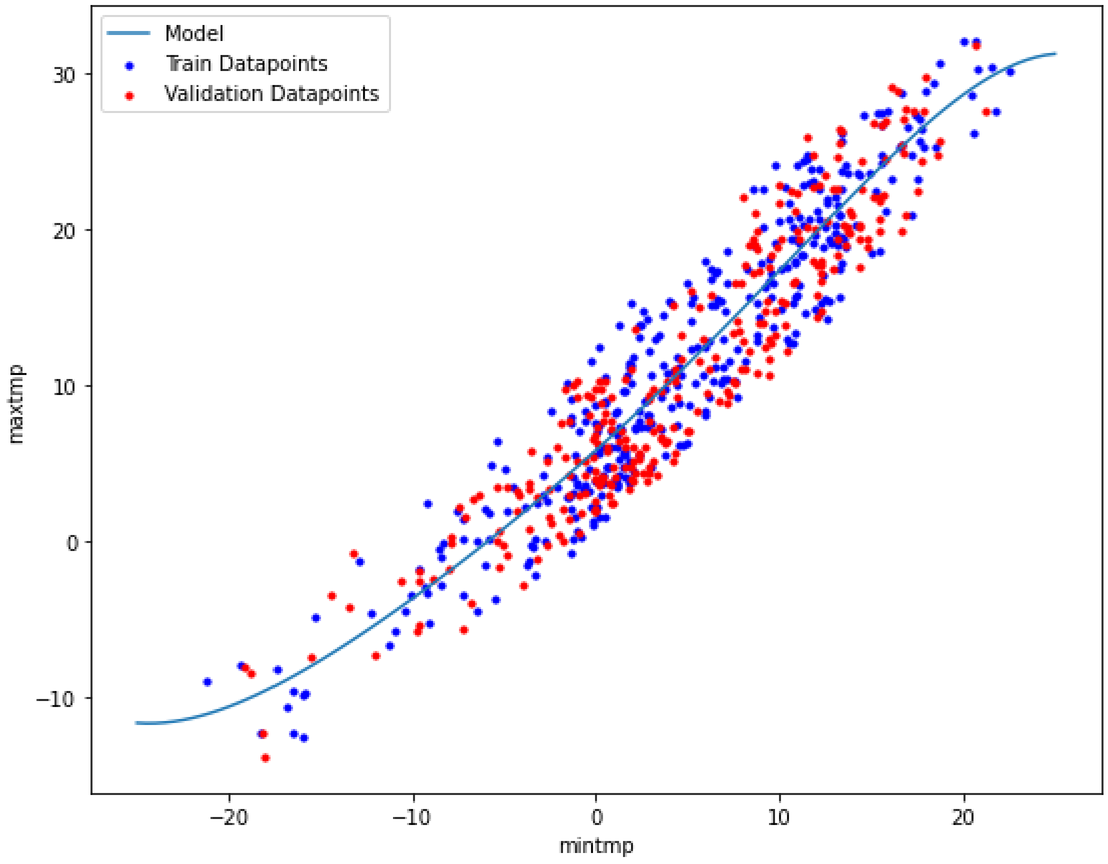
At first, several polynomial regression models are learnt with the help of training data, errors on validation datasets are obtained. By comparison of training errors and validation errors, the best model is chosen.

```
In [19]:
```

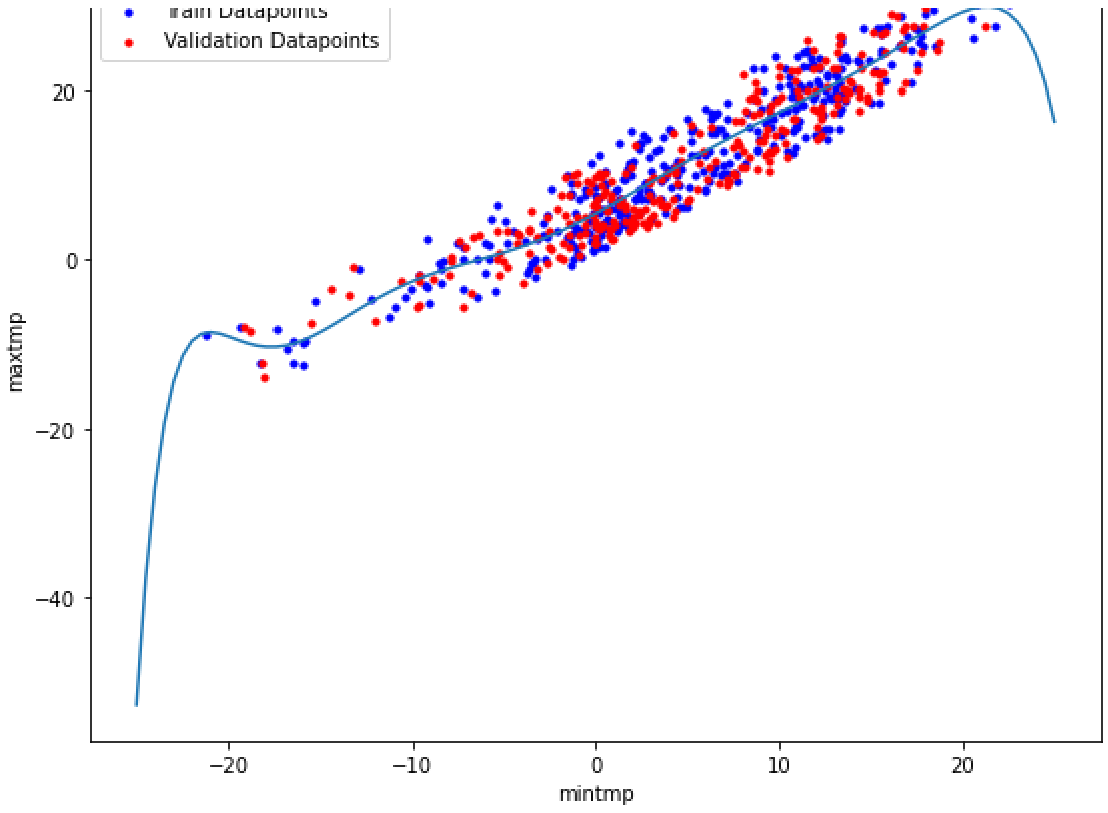
Polynomial degree = 3
Training error = 9.1287
Validation error = 9.3539



Polynomial degree = 5
Training error = 9.1041
Validation error = 9.253



Polynomial degree = 10
Training error = 8.9277
Validation error = 9.4094



In [20]:

In [21]:

```
For polynomial degree 3: the training error is 9.128680532620562, t
he validation error is 9.35393616032128
For polynomial degree 5: the training error is 9.104076276390922, t
he validation error is 9.253001720352678
For polynomial degree 10: the training error is 8.927689630261082,
the validation error is 9.409375966372131
```

Obtaining final test error

As we can see, due to the validation error the best Polynomial model is with degree of 5. After that, the final error on test set is calculated for the best model.

In [26]:

```
The test error is:  11.723735778940444
```

In [27]:

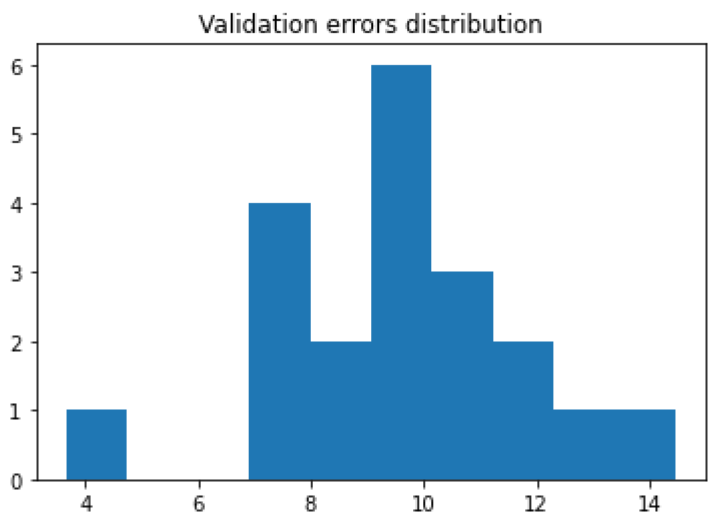
# Task A5.3: This task shows that different division into training and validation data can result in different validation errors for the model with same parameters.
In this task the polynomial regression with the degree of 5 is learnt several times on slightly different subsets of original dataset and validation errors are calculated. Let's see how these validation errors are distributed.
The [KFold](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html) function from sklearn.model_selection is used. It provides train/validation indices to split data in train/validation sets. In our case dataset is shuffled and split into k=20 consecutive folds, each fold is then used once as a validation while the k - 1 remaining folds form the training set.

In [29]:

In [30]:

In [31]:

In [32]:



# A5.4: Answer the following quiz question by setting the corresponding variable to the index of the answer that you consider correct. On what data part does the validation error depend? * Answer 1: Only training data * Answer 2: Only validation data * Answer 3: Both: train and validation data

In [33]:

In [34]:

In [35]:

Reference error

As was stated before the learning goals, it is very good if it's possible to find the reference error E_{ref} to compare against the average loss on validation/test set. That error can be found in several ways.

It can be loss incurred by predictions obtained from humans. For example, with the help of some crowdsourcing platform, like https://www.mturk.com/ (https://www.mturk.com/).

Also one common example is widespread when working with public datasets. For many of them people have already built solutions and shared their results in such competitions, as in Kaggle. So you can just go there, find similar dataset of interest, check the leaderboard scores and assume that the best one is reference mark. For instance, find the competition in Kaggle for Zillow Prize: Zillow's Home Value Prediction (Zestimate). Go the leaderboard and find the smallest MAE score achieved by competitors. That can also be the reference score.

Another variant is connected with probabilistic modelling (Bayes risk), to which A5.5 is devoted.

# Task A5.5:

In this task you will work with probabilistic modelling for Bayes error. The assumption here is our datapoints are from a bivariate Gaussian distribution. It's worth mentioning that we are not fitting a Gaussian to make predictions (which we could do, of course), instead we are modelling Bayes error to explore the intrinsic uncertainty in y given x. This way we can get some idea what to expect from our model, in particular, the least average squared error of the (in a certain sense) optimal hypothesis, i.e. some benchmark for the error. The formula for conditional variance of y, given x is:

$$\sigma^2_{y|x} = (1-\rho^2)\sigma^2_y \ , \text{ where } \ \rho = \frac{cov(x,y)}{\sigma_x\sigma_y}$$

From this formula, we can easily see that the variance of y given x is decreasing the more correlated the two variables are (the larger correlation coeficient ρ is). To make the calculation simpler, we will use an equivalent formula for this task:

$$var(y|x) = var(y) - cov(y,x)var(x)^{-1}cov(x,y)$$

The steps are:

- Fit a (bivariate) Gaussian distribution p(x,y) to the dataset and obtain a covariance matrix. Numpy provides the function [np.cov()] (https://numpy.org/doc/stable/reference/generated/numpy.cov.html) to calculate a covariance matrix. Please set the parameter `ddof=1` (degree of freedom) in order to get unbiased estimators of variances and covariance. the resulted covariance matirx is a symmetric matrix, its main diaganol contains variance of x and y, off-diaganol entries are covariance between x and y.
- Calculate the conditional variance $var(y|x)$ by using unbiased estimates of the variances and covariance you can access from the covariance matrix obtained in last step.

More detailed information about conditional variance of Gaussian distribution could be found in Wikipedia page: https://en.wikipedia.org/wiki/Multivariate_normal_distribution.

In [36]:

```
(713,)
(713,)
(2, 713)
```

In [37]:

```
Covariance Matrix:

 [[65.08407634 69.63357941]
 [69.63357941 84.48492235]]
```

In [38]:

Compute conditional variance of y1, given x1 by using the formula:

$$var(y|x) = var(y) - cov(y, x)var(x)^{-1}cov(x, y)$$

NOTE: For variance of x1, variance of y1 and covariance between x1 and y1, please use the corresponding entry from the covariance matrix you've just got, and $cov(y, x) = cov(x, y)$.

In [43]:

```
The baseline for any other hypothesis is:  [[84.48492235 69.6335794
1]
 [69.63357941 65.08407634]]
```

In [44]:

In [45]:

```
Bayes error is:  [[84.48492235 69.63357941]
 [69.63357941 65.08407634]]
Test error for the best model (degree=5) is:  11.723735778940444

    degrees  train_errors  val_errors
0         3      9.128681    9.353936
1         5      9.104076    9.253002
2        10      8.927690    9.409376
```

The baseline error can be compared with the test error for the best model in Task A5.2. We can see that the difference is not that large, our model performs rather good and some mistakes are because of the noise in data.

# A5.6: Assume you have trained a model resulting in a small train error and a large val. error. Which of the following change will likely reduce the validation error? * Answer 1: Use either more training data points or use a smaller model (e.g. polyreg with smaller degree) * Answer 2: Use more validation data points * Answer 3: Use a larger model (e.g. polyreg with higher degree)

In [51]:

```
In [50]:
```

```
In [49]:
```