Typical structure of Notebook

Neural Networks for Machine Learning Applications 2023

Sakari Lukkarinen

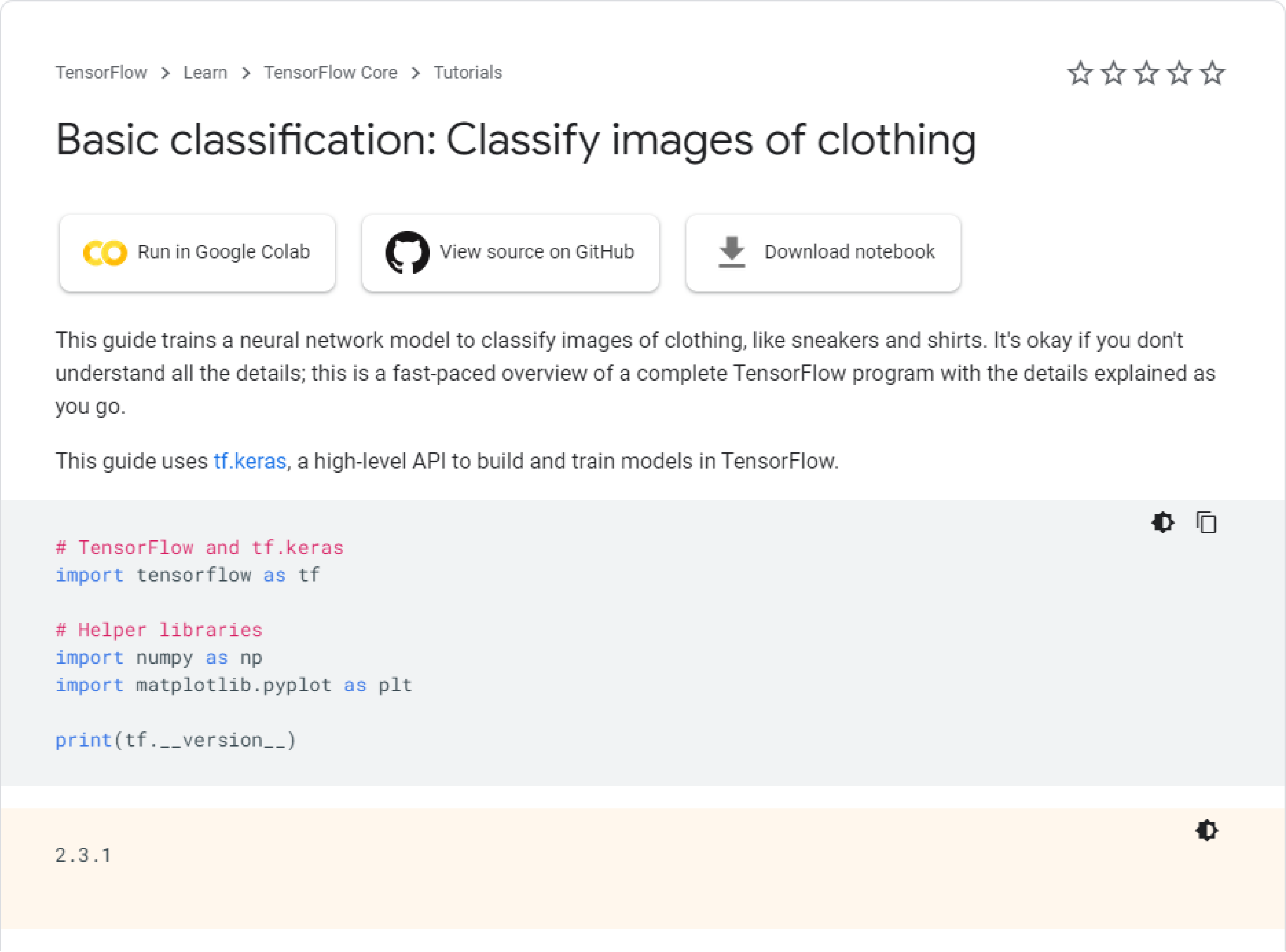
Metropolia University of Applied Sciences

Contents

- Examples of documented Notebooks
  - Basic multiclassification
  - Basic binary classification
  - Basic regression
  - Comparison of the examples
- Table of contents – Template
  - Example
  - Exercise

What are the contents of a typical neural networks (Tensorflow) Notebook?

Example: **multi**classification



TensorFlow > Learn > TensorFlow Core > Tutorials

☆☆☆☆☆

Basic classification: Classify images of clothing

This guide trains a neural network model to classify images of clothing, like sneakers and shirts. It's okay if you don't understand all the details; this is a fast-paced overview of a complete TensorFlow program with the details explained as you go.

This guide uses tf.keras, a high-level API to build and train models in TensorFlow.

```
# TensorFlow and tf.keras
import tensorflow as tf

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)
```

```
2.3.1
```

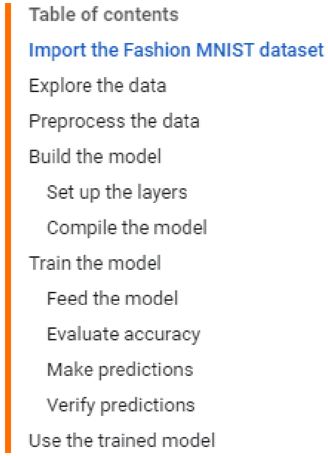
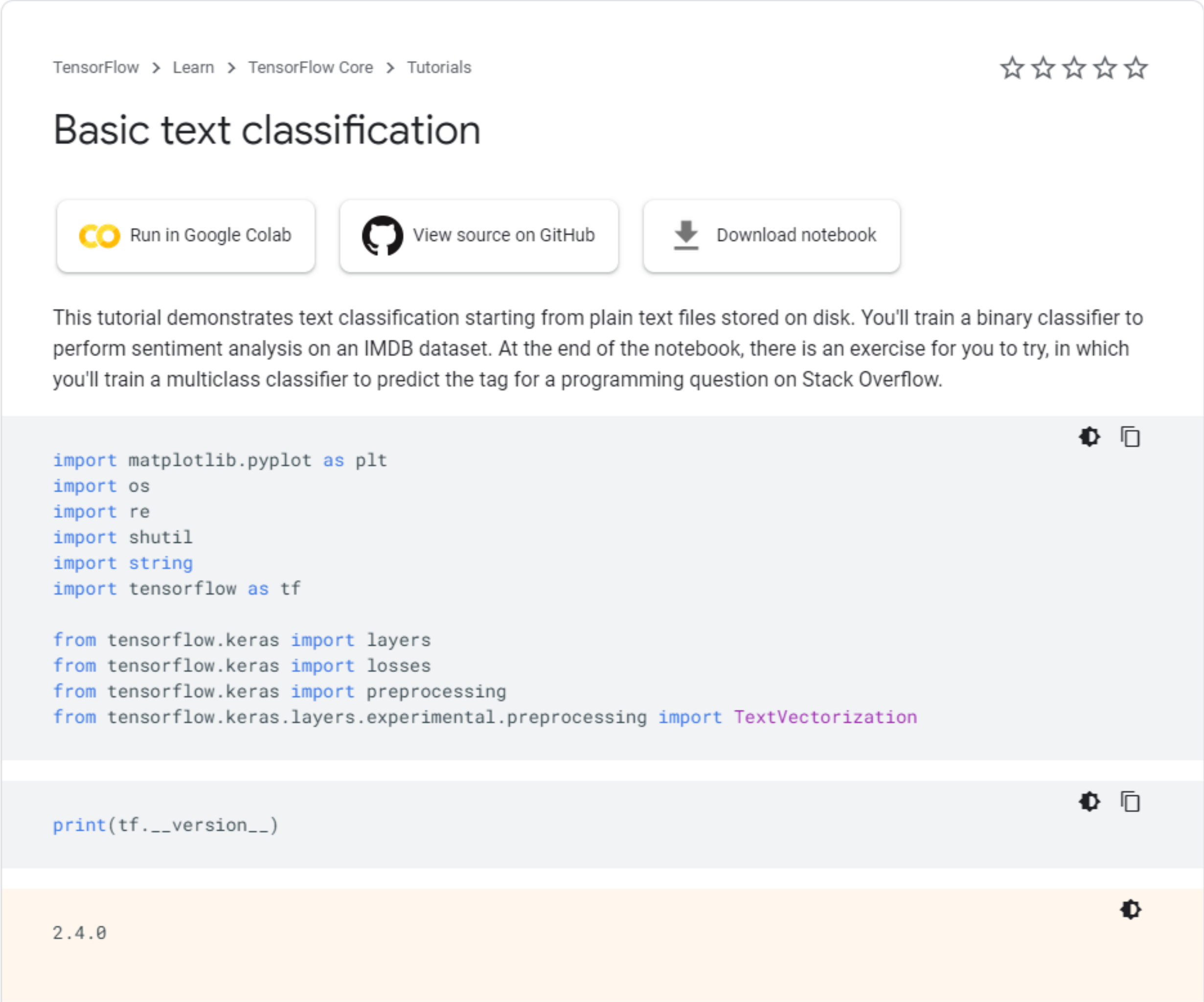


Table of contents

[Basic classification: Classify images of clothing | TensorFlow Core]

Example: **binary** classification



TensorFlow > Learn > TensorFlow Core > Tutorials ☆☆☆☆☆

Basic text classification

This tutorial demonstrates text classification starting from plain text files stored on disk. You'll train a binary classifier to perform sentiment analysis on an IMDB dataset. At the end of the notebook, there is an exercise for you to try, in which you'll train a multiclass classifier to predict the tag for a programming question on Stack Overflow.

```
import matplotlib.pyplot as plt
import os
import re
import shutil
import string
import tensorflow as tf

from tensorflow.keras import layers
from tensorflow.keras import losses
from tensorflow.keras import preprocessing
from tensorflow.keras.layers.experimental.preprocessing import TextVectorization
```

```
print(tf.__version__)
```

```
2.4.0
```

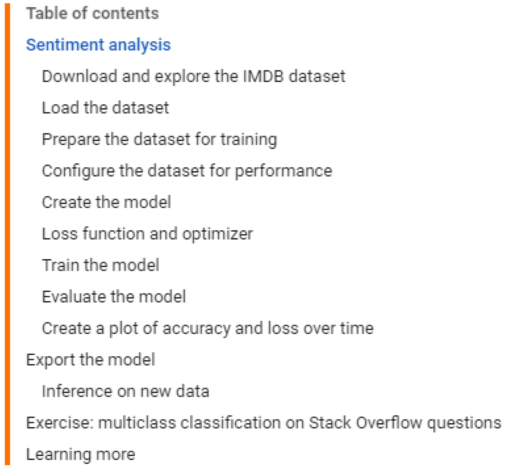
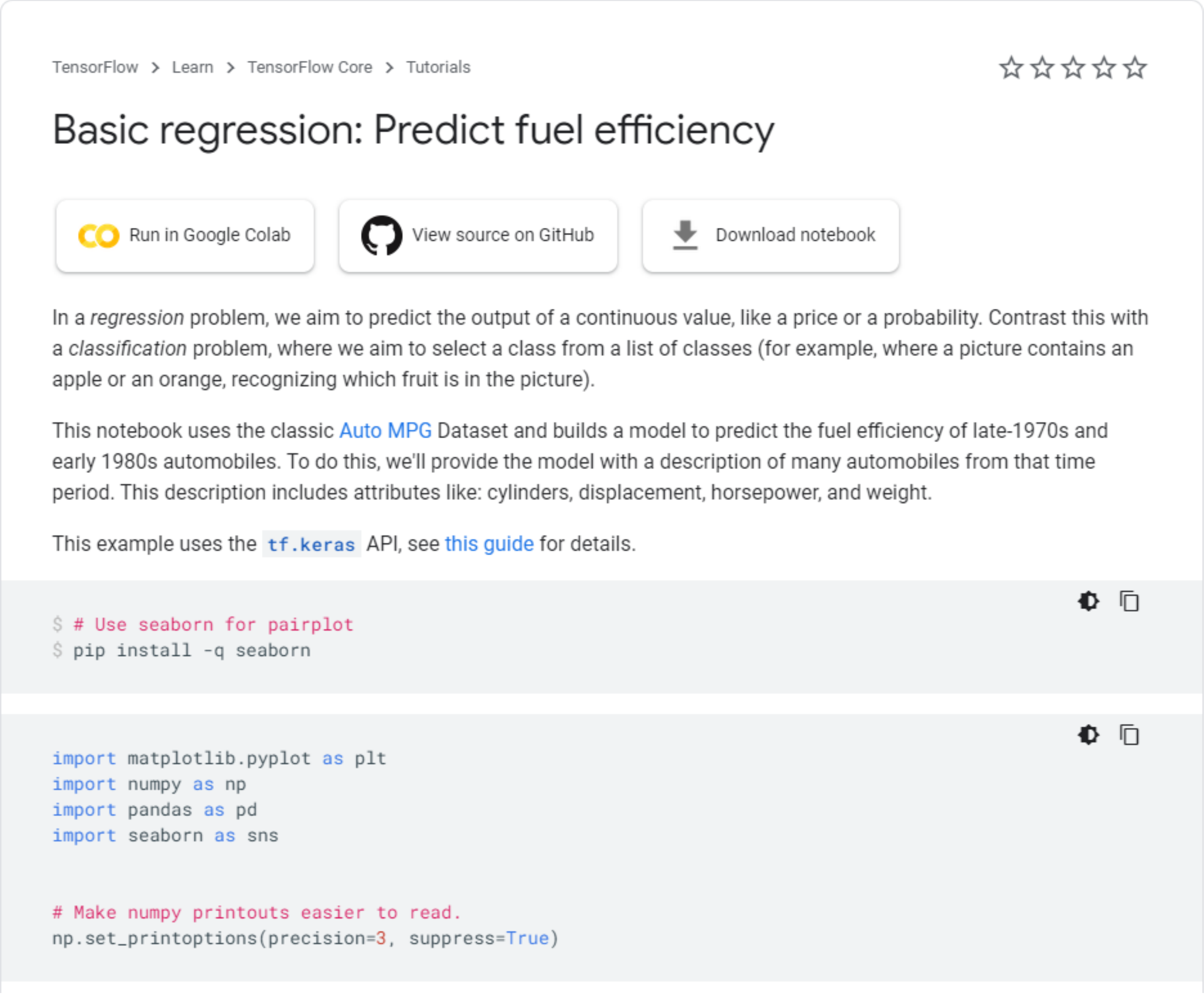


Table of contents

Basic text classification | TensorFlow Core

Example: **regression**



Basic regression: Predict fuel efficiency

In a *regression* problem, we aim to predict the output of a continuous value, like a price or a probability. Contrast this with a *classification* problem, where we aim to select a class from a list of classes (for example, where a picture contains an apple or an orange, recognizing which fruit is in the picture).

This notebook uses the classic Auto MPG Dataset and builds a model to predict the fuel efficiency of late-1970s and early 1980s automobiles. To do this, we'll provide the model with a description of many automobiles from that time period. This description includes attributes like: cylinders, displacement, horsepower, and weight.

This example uses the `tf.keras` API, see this guide for details.

```
$ # Use seaborn for pairplot
$ pip install -q seaborn
```

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns


# Make numpy printouts easier to read.
np.set_printoptions(precision=3, suppress=True)
```

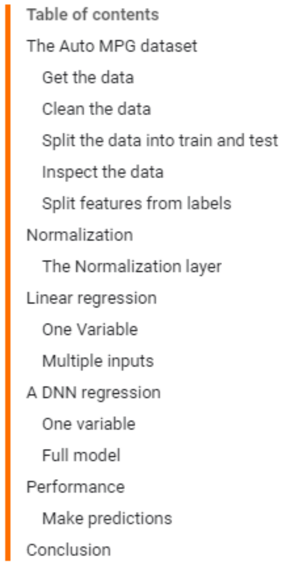


Table of contents

Basic regression: Predict fuel efficiency | TensorFlow Core

Table of Contents comparison

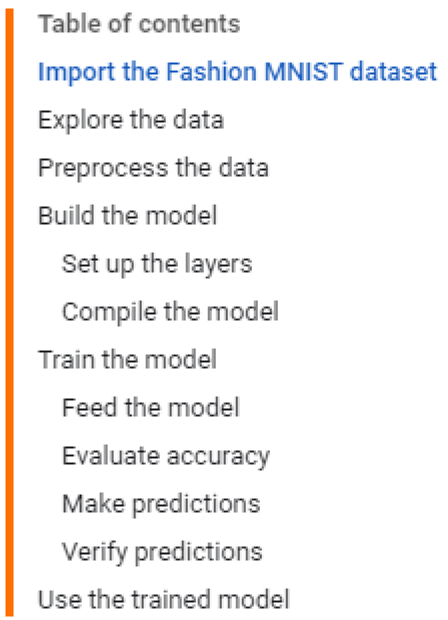


Table of contents

- Import the Fashion MNIST dataset
- Explore the data
- Preprocess the data
- Build the model
  - Set up the layers
  - Compile the model
- Train the model
  - Feed the model
  - Evaluate accuracy
  - Make predictions
  - Verify predictions
- Use the trained model

[Basic classification]

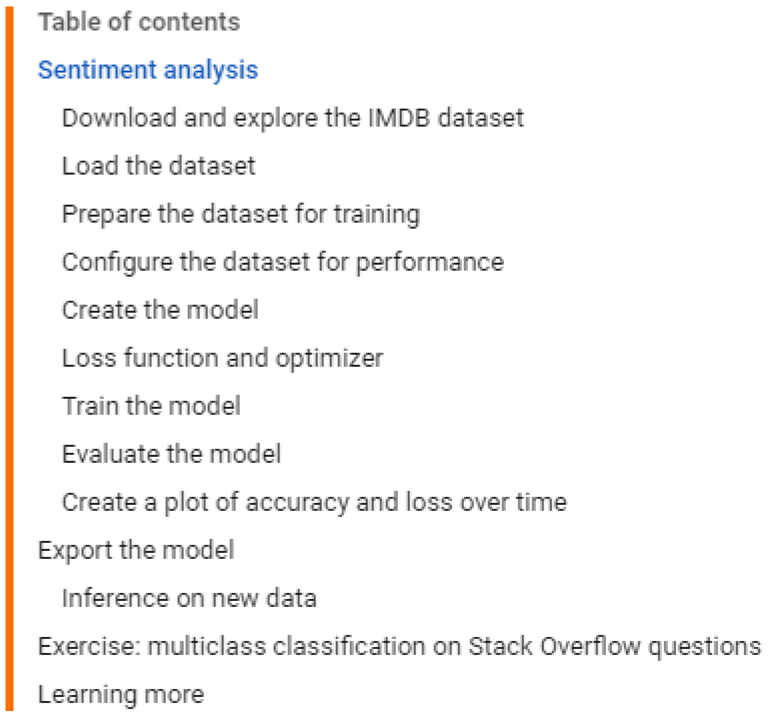


Table of contents

- Sentiment analysis
  - Download and explore the IMDB dataset
  - Load the dataset
  - Prepare the dataset for training
  - Configure the dataset for performance
  - Create the model
  - Loss function and optimizer
  - Train the model
  - Evaluate the model
  - Create a plot of accuracy and loss over time
- Export the model
  - Inference on new data
- Exercise: multiclass classification on Stack Overflow questions
- Learning more

[Basic text classification]

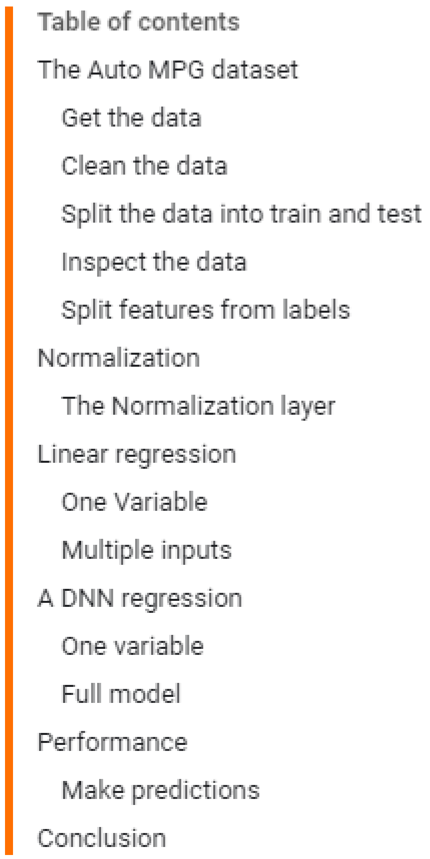


Table of contents

- The Auto MPG dataset
  - Get the data
  - Clean the data
  - Split the data into train and test
  - Inspect the data
  - Split features from labels
- Normalization
  - The Normalization layer
- Linear regression
  - One Variable
  - Multiple inputs
- A DNN regression
  - One variable
  - Full model
- Performance
  - Make predictions
- Conclusion

[Basic regression]

Table of Contents - Template

1. Introduction
   - Library Setup
2. Dataset
   - Reading the data
   - Exploratory or descriptive statistics
3. Preprocessing
   - Handling missing values
   - Convert data to numbers
   - Separate input features and output labels
   - Normalize features
   - Split data into training and test sets
4. Modeling
   - Model architecture (input, hidden layers, output)
5. Training
   - Epochs, batch size, validation dataset size
   - Callbacks and stop criteria
6. Evaluation of performance
   - Loss (error) and metrics analysis
7. Conclusions

Introduction explains shortly what this experiment (Notebook) is about. **Setup** lists all necessary *imported libraries* and their *version numbers*.

Dataset *explains* what the dataset is about, shows the steps to *read the data* in to the memory and usually *explores or describes statistically* the data.

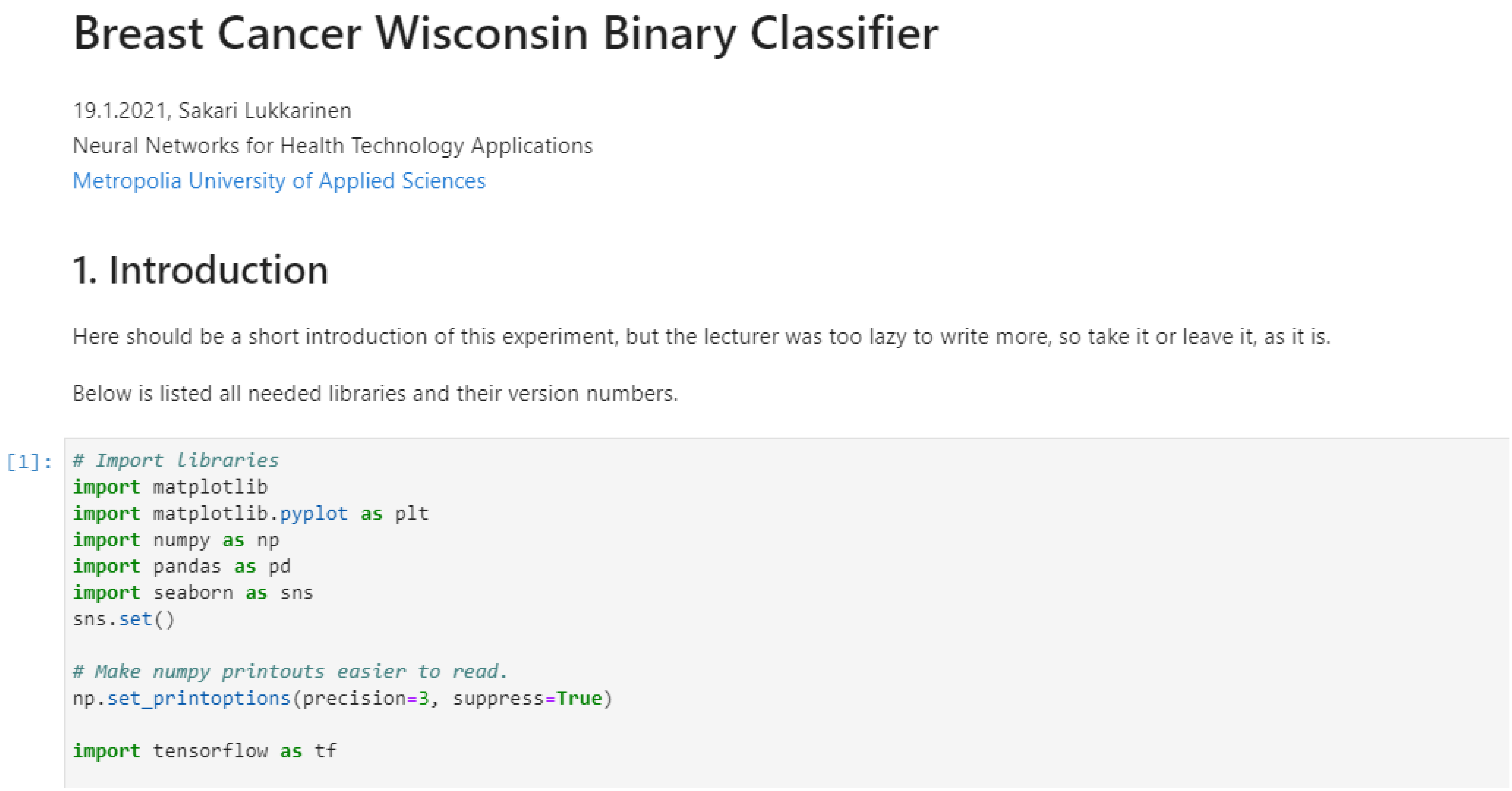
Preprocessing prepares the data for training. Typical steps are *handling the missing values*, separate the input *features* and output *labels*, *normalize* the features and split the data into *training and test sets*.

Modeling describes the model architecture (*input, hidden, and output layers, activation functions*, …) and *compiles* the model with *loss function*, *optimizer* and *metrics*.

Training runs the model (*fit*) with the training dataset several times (*epochs*) in small *batches*.

In **Evaluation and performance** step the model's *loss* (error) is evaluated against the *test dataset* and the performance is analysed by reviewing the *metrics (accuracy, …)*.

Conclusions summarize what results were achieved and which settings and model architecture were used.



Breast Cancer Wisconsin Binary Classifier

19.1.2021, Sakari Lukkarinen

Neural Networks for Health Technology Applications

Metropolia University of Applied Sciences

1. Introduction

Here should be a short introduction of this experiment, but the lecturer was too lazy to write more, so take it or leave it, as it is.

Below is listed all needed libraries and their version numbers.

[1]:

```
# Import libraries
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
sns.set()

# Make numpy printouts easier to read.
np.set_printoptions(precision=3, suppress=True)

import tensorflow as tf
```

See: OMA > Documents > Notebooks for more details.

Exercise

Study the Example Notebook – Breast Cancer Wisconsin Binary Classifier

- How the contents was structured?
- How informative was the introduction?
- What libraries were used and why?
    - Were there some libraries that were NOT used in the Notebook?
- What analysis tools were used to make the explorative and descriptive statistical analysis?
- How the data was preprocessed?
- How the model was build?
- What were the settings for model training?
- How the results and the performance were evaluated?
- What was written in the conclusions?