TensorFlow

TECH TALENT
ACADEMY

WOMEN IN DATA
ACADEMY

TECH TALENT ACADEMY

Session Content

An introduction into deep learning in Python
Learning to pre-process your data, model, evaluate and optimize neural networks.

WHAT IS TENSORFLOW?

HISTORY OF TENSORFLOW

DEEP LEARNING

ARTIFICIAL NEURAL NETWORKS

WOMEN IN DATA ACADEMY

What is TensorFlow?

- TensorFlow is an open-source library used in Machine Learning.
- It is designed to enable fast experimentation with the deep Neural Network.
- TensorFlow is user-friendly, making it easy to create machine learning models.
- High and low-level neural network API.

History of TensorFlow

TensorFlow was released in 2015 and was developed by The Google Brains Team. It came off the back of the DistBelief machine learning system that was built in 2011.

TensorFlow has a flexible architecture allowing for easy implementation of computation across a variety of platforms.

TensorFlow is a Python-friendly open source library for numerical computation that makes machine learning faster and easier.

It can be used to train & serve models in live mode to real customers.
Therefore industrial researchers can apply their ideas to products faster.

TensorFlow allows you to make the most of your available hardware with its advanced support.

Deep Learning

Machine learning, is the study of computer algorithms that improve automatically through experience.

Deep learning is part of a broader family of machine learning methods based on artificial neural networks with representation learning

Artificial neural networks (ANNs) were inspired by information processing and distributed communication nodes in biological systems (brain)

Deep learning is one of the hottest fields in data science with many case studies that have astonishing results in robotics, image recognition and Artificial Intelligence (AI).

Artificial Neural Networks

A neural network is created by connecting neurons. The human brain is then an example of such a neural network, which is composed of billions of neurons.

The brain is proficient in performing quite complex calculations, and this is where the inspiration for Artificial Neural Networks comes from.

The network on a whole is a powerful modelling tool.

The steps for Machine Learning

- Load dataset
- Explore dataset
- Process data
- Visualise data
- Pick algorithms
- Train and test sets
- Standardise data (scale)
- Compile and fit
- Learning from the model and evaluations

Session content

Predicting Wine Types Red or White?

LOADING DATASETS

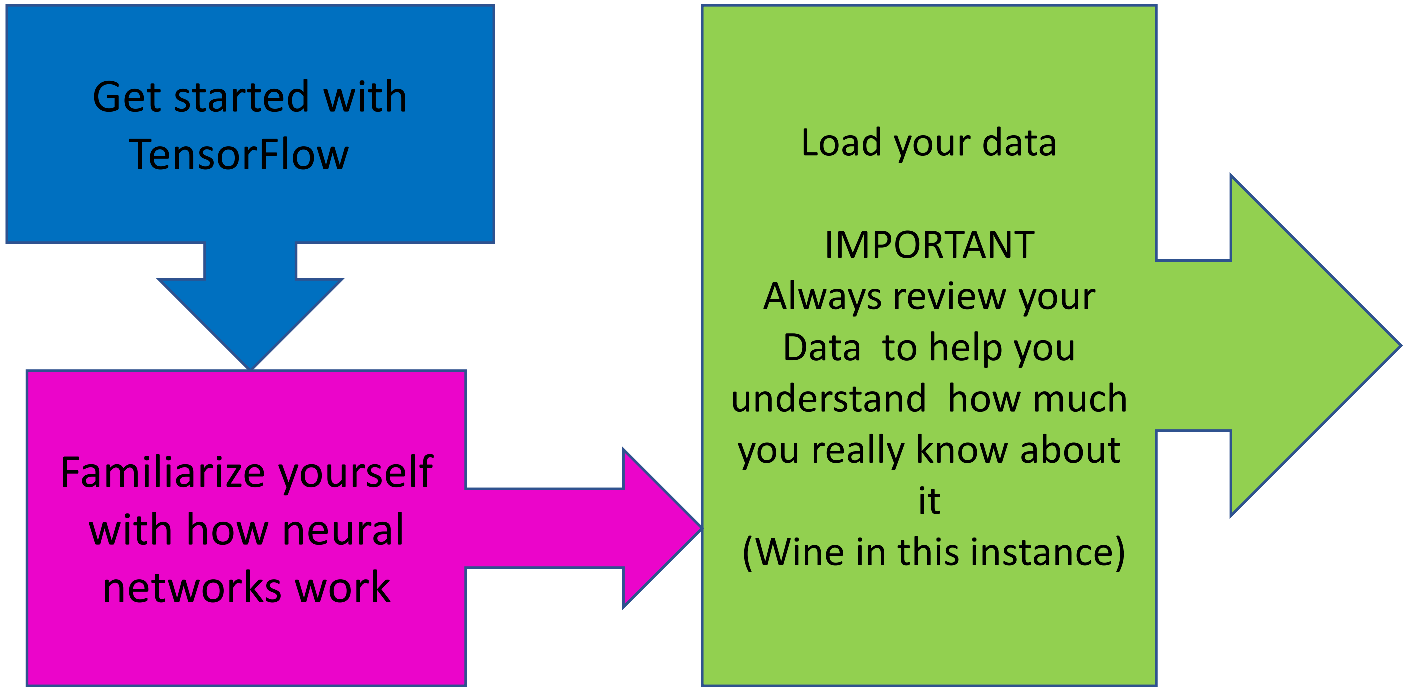
EXPLORING THE DATA

Datasets

We will use the wine quality data from the UCI Machine Learning Repository.

In the real world bigger data sets are used, but for initial learning purposes we are using a smaller dataset.

The aim here is:



In general, there are two very popular types of wine
Red and White.

By looking at the data you will realise that there are consideration factors like:

- Acids
- Sugar Sulphates
- Wine Quality

TECH TALENT
ACADEMY

Loading your Dataset

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

```
white = pd.read_csv("http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv", sep=";")
```

```
white.head()
```

```
white.tail(25)
```

```
red = pd.read_csv("http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv", sep=";")
```

```
red.head()
```

Run a simple .head() .tail() to ensure your import was successful

WOMEN IN DATA
ACADEMY

Data Exploration

```python
# Print info on white wine
print(white.info())

# Print info on red wine
print(red.info())
```

Console 2/A

```
memory usage: 459.2 KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

TECH TALENT ACADEMY

Data Exploration

In [4]:
```
# First rows of `red`
red.head()

# Last rows of `white`
white.tail()

# Take a sample of 5 rows of `red`
red.sample(5)

# Describe `white`
white.describe()

# Double check for null values in `red`
pd.isnull(red)
```

Out[4]:

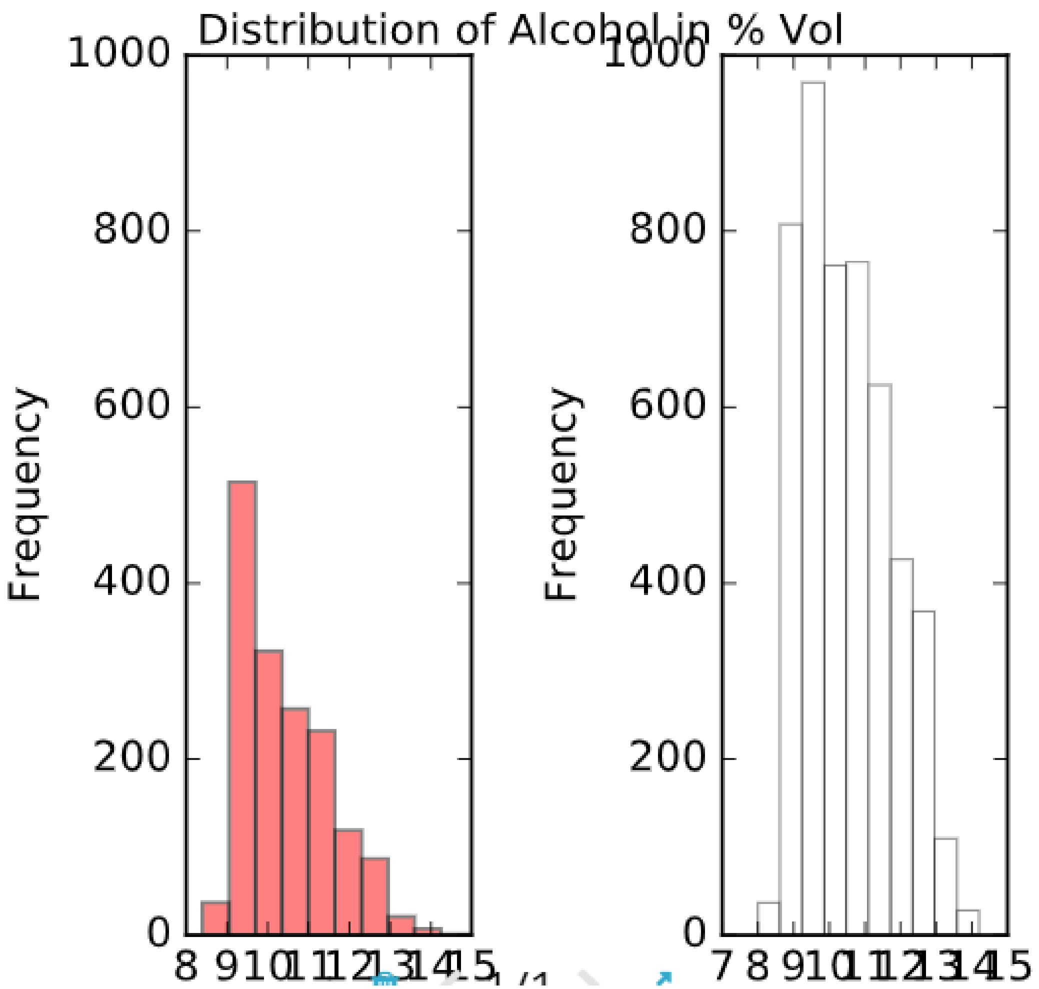
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	False	False	False	False	False	False	False	False	False	False	False	False
1595	False	False	False	False	False	False	False	False	False	False	False	False
1596	False	False	False	False	False	False	False	False	False	False	False	False
1597	False	False	False	False	False	False	False	False	False	False	False	False
1598	False	False	False	False	False	False	False	False	False	False	False	False

1599 rows × 12 columns

WOMEN IN DATA ACADEMY

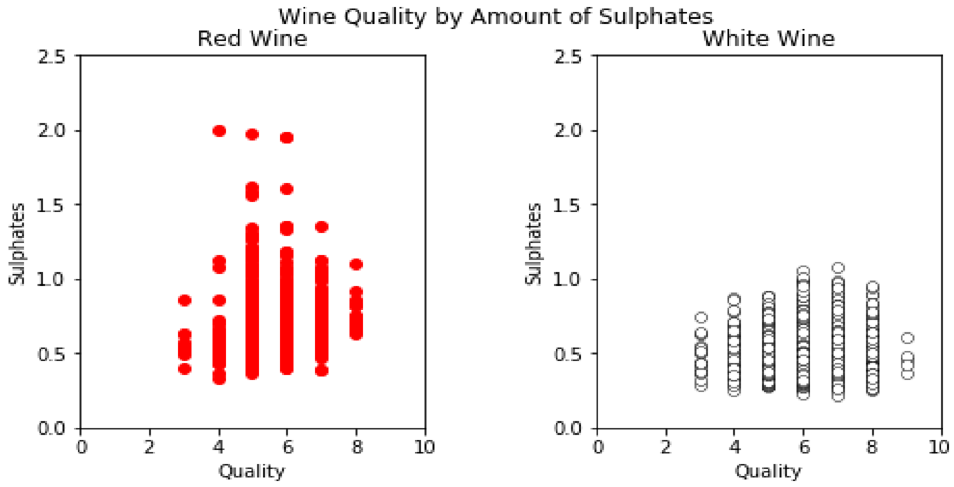
Visualising the Data –levels of alcohol

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(1, 2)

ax[0].hist(red.alcohol, 10, facecolor='red', alpha=0.5,
label="Red wine")
ax[1].hist(white.alcohol, 10, facecolor='white', ec
="black", lw=0.5, alpha=0.5, label="White wine")

fig.subplots_adjust(left=0, right=1, bottom=0, top=0.5,
hspace=0.05, wspace=1)
ax[0].set_ylim([0, 1000])
ax[0].set_xlabel("Alcohol in % Vol")
ax[0].set_ylabel("Frequency")
ax[1].set_xlabel("Alcohol in % Vol")
ax[1].set_ylabel("Frequency")
#ax[0].legend(loc='best')
#ax[1].legend(loc='best')
fig.suptitle("Distribution of Alcohol in % Vol")

plt.show()
```



Relation between the sulphates and the quality of the wine

In [7]:
```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(1, 2, figsize=(8, 4))

ax[0].scatter(red['quality'], red["sulphates"], color="red")
ax[1].scatter(white['quality'], white['sulphates'], color="white", edgecolors="black", lw=0.5)

ax[0].set_title("Red Wine")
ax[1].set_title("White Wine")
ax[0].set_xlabel("Quality")
ax[1].set_xlabel("Quality")
ax[0].set_ylabel("Sulphates")
ax[1].set_ylabel("Sulphates")
ax[0].set_xlim([0,10])
ax[1].set_xlim([0,10])
ax[0].set_ylim([0,2.5])
ax[1].set_ylim([0,2.5])
fig.subplots_adjust(wspace=0.5)
fig.suptitle("Wine Quality by Amount of Sulphates")

plt.show()
```



Wrapping Up The Exploratory Data Analysis (EDA)

Recap of what has been seen during your EDA that could be important :

- Some of the variables of your data sets have values that are significantly far apart.
- You have an ideal scenario: there are no null values in the data sets.
- Most wines included in the data set have around 9% of alcohol.
- Red wine seems to contain more sulphates than the white wine, which has less sulphates above 1 g/dm3.

Session content

Predicting Wine Types - Red or White?

PRE-PROCESS/TEST

INTERMEZZO: CORRELATION MATRIX

Pre-process Data

We have explored the data, lets now look at the insights we have gained!

First we need to pre-process the data in order to start building our own neural network!

In [11]:
```
# Add `type` column to `red` with value 1
red['type'] = 1

# Add `type` column to `white` with value 0
white['type'] = 0

# Append `white` to `red`
wines = red.append(white, ignore_index=True)
print (wines)
```

```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0               7.4              0.70         0.00             1.9      0.076
1               7.8              0.88         0.00             2.6      0.098
2               7.8              0.76         0.04             2.3      0.092
3              11.2              0.28         0.56             1.9      0.075
4               7.4              0.70         0.00             1.9      0.076
...             ...               ...          ...             ...        ...
6492            6.2              0.21         0.29             1.6      0.039
6493            6.6              0.32         0.36             8.0      0.047
6494            6.5              0.24         0.19             1.2      0.041
6495            5.5              0.29         0.30             1.1      0.022
6496            6.0              0.21         0.38             0.8      0.020

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                    11.0                  34.0  0.99780  3.51       0.56
1                    25.0                  67.0  0.99680  3.20       0.68
```

Train and Test Sets

```
]: from sklearn.model_selection import train_test_split
```

```
]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
   print(X_train.shape)
   print(X_test.shape)
```

Standardise The Data

Standardisation is a way to deal with the values that lie so far apart.

The scikit-learn package offers you a great and fast way of getting your data standardised:

Import the Standard Scaler module from *sklearn.preprocessing* and you're ready to scale your train and test data!

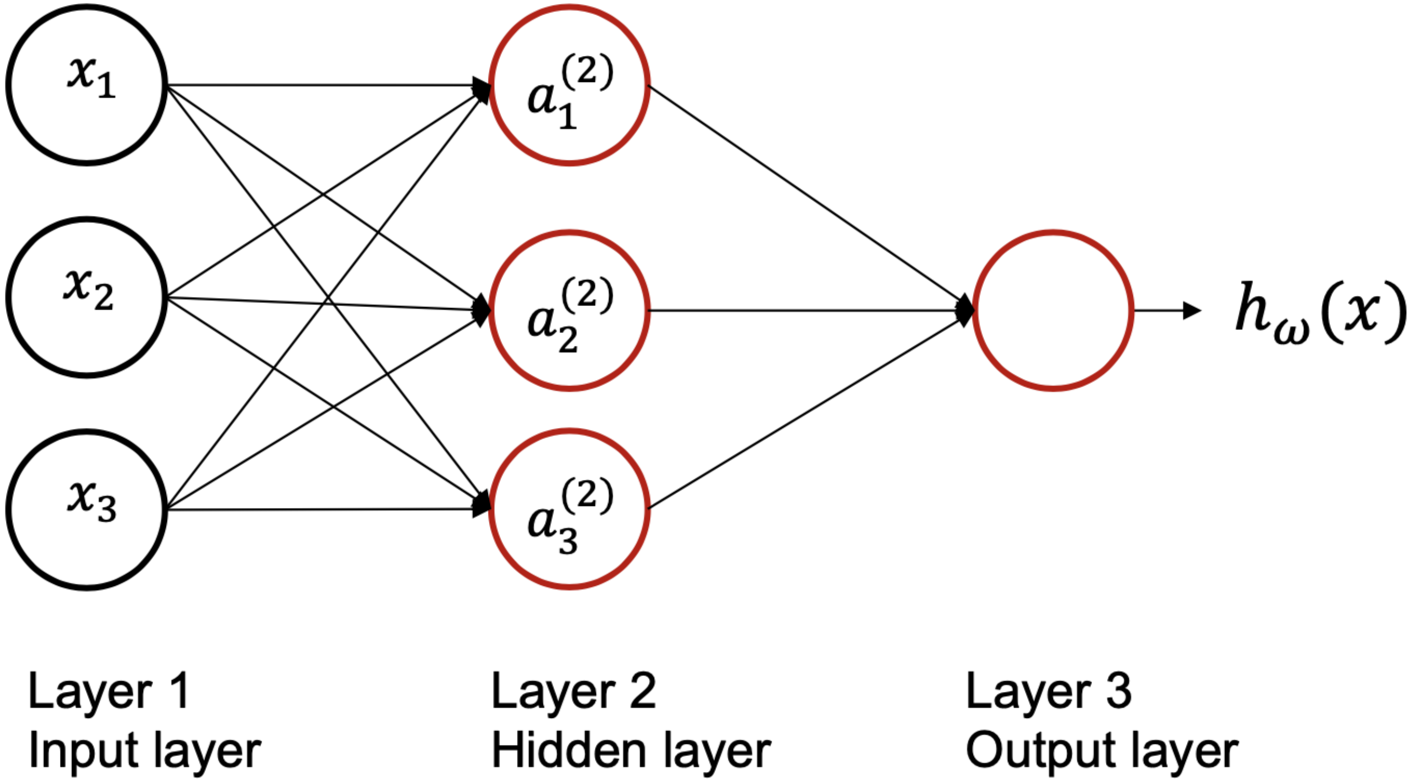
```
[ ]: from sklearn.preprocessing import StandardScaler
```

```
[ ]: scalar = StandardScaler().fit(X_train)
     X_train = scalar.transform(X_train)
     X_test = scalar.transform(X_test)
```

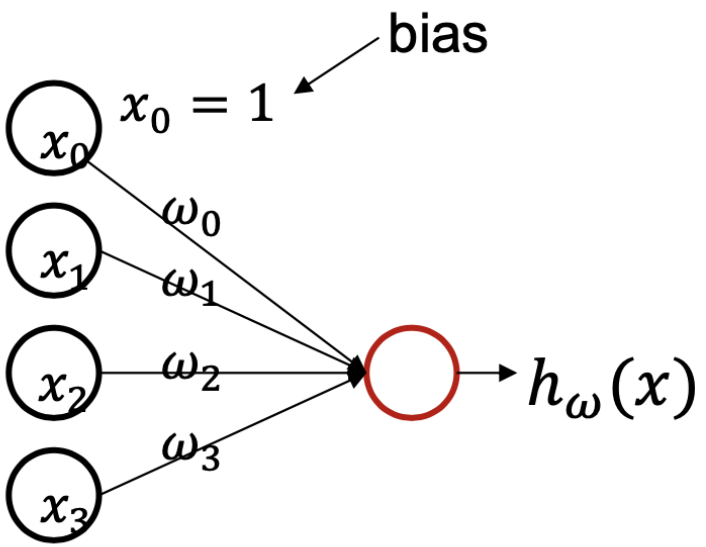
```
[ ]: import tensorflow as tf

     from tensorflow              import keras
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense
```

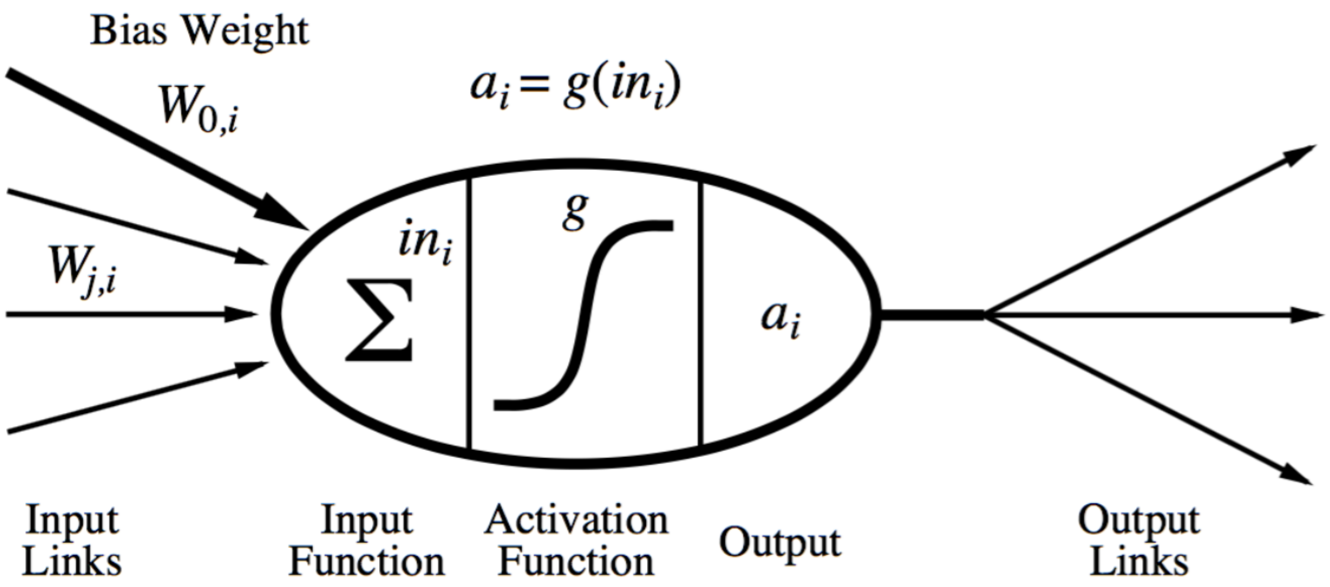
Now that we've pre-processed the data, we can move on to the real work: building our own neural network to classify wines.

Overview of a Neural Network



Inside each Neuron





Forward and Back Propagation

- Forward:
  - At first Neurons weights are set at random.
  - The training ANN will then make predictions with the training data.
  - Once it gets to the output, it will check how well it has done.
  - With the results it will then start to do back propagation
- Back:
  - The NN will go back through the ANN and update the weights to a value it thinks might be better to help it predict its label.

- This is repeat multiple times depending on the epochs value that has been set.

Multi-layer Perceptron

A quick way to start building your multi-layer perceptron is to use the Keras Sequential model.

This is a linear stack of layers. You can easily create the model by passing a list of layer instances to the constructor, you set this up by running

model = Sequential().

```
[ ]: model = Sequential()
```

creating the model's layers:

```
[ ]: # Input layer
model.add(Dense(12, activation='relu', input_shape=(12,)))

#Hidden layer
model.add(Dense(8, activation='relu'))

# Output label
model.add(Dense(1, activation='sigmoid'))
```

Information from the Model

```
[29]: print("Output shape:", model.output_shape)

Output shape: (None, 1)

[30]: model.summary()

Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 12)                156
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 104
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 9
=================================================================
Total params: 269
Trainable params: 269
Non-trainable params: 0
```

```
[31]: print("Model weights:", model.get_weights())

Model weights: [array([[ 0.03435898, -0.24827
          0.20286858, -0.32607782,  0.29336548
         -0.14217973,  0.4732268 ],
        [ 0.32511783,  0.4507426 ,  0.4869498
          0.03820753, -0.11768901, -0.4701327
          0.1668328 ,  0.28152716],
        [ 0.36597955, -0.21855116,  0.41177273
          0.10882175,  0.27422035, -0.47514927
         -0.29566026,  0.39479446],
        [ 0.41026127, -0.34389877, -0.24463332
         -0.39401162, -0.4610442 ,  0.28947282
```

The closer the value is to 1 means it is likely to be a red wine, the closer to 0 it is likely to be a white wine!

Compile and Fit

To compile your model and fit the model to the data, use:
compile() and fit()

```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.fit(X_train, y_train,epochs=20, batch_size=1, verbose=1)
```

Test Data Predictions

```
y_pred = model.predict(X_test)
y_pred[:5]
```

```
array([[1.3831258e-04],
       [9.9752289e-01],
       [8.0875907e-06],
       [2.6916542e-08],
       [3.7994080e-10]], dtype=float32)
```

```
predictions = [1 if p > 0.5 else 0 for p in y_pred]
```

```
predictions[:15]
```

```
[0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1]
```

```
y_test[:15]
```

```
array([0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1], dtype=int64)
```

Now we can use this data to predict which is a red/white wine?

Evaluate Model

```
score = model.evaluate(X_test, y_test, verbose=1)
score
```

```
68/68 [==============================] - 0s 932us/step - loss: 0.0272 - accuracy: 0.9944
[0.027239905670285225, 0.9944055676460266]
```

This is using the model just built using the test data to make a prediction using data the model hasn't used.
This evaluates the performance of the model,
eg: how good the model is.

Evaluation Metrics

Before you start re-arranging the data and putting it together in a different way, it's always a good idea to try out different evaluation metrics.

For this, you can rely on scikit-learn (>> import as sklearn) for this. You will test out some basic classification evaluation techniques like:

- The confusion matrix – This is a breakdown of predictions into a table showing correct predictions and the types of incorrect predictions that have been made.
- You should only see numbers in the diagonal, which means that all your predictions were correct!
- Precision is a measure of a classifier's exactness. The higher the precision, the more accurate the classifier.

```
from sklearn.metrics import confusion_matrix, precision_score
```

```
confusion_matrix(y_test, predictions)
```

```
array([[1585,    3],
       [   9,  548]], dtype=int64)
```

```
precision_score(y_test, y_pred.round())
```

```
0.9945553539019963
```

The Confusion Matrix Explained

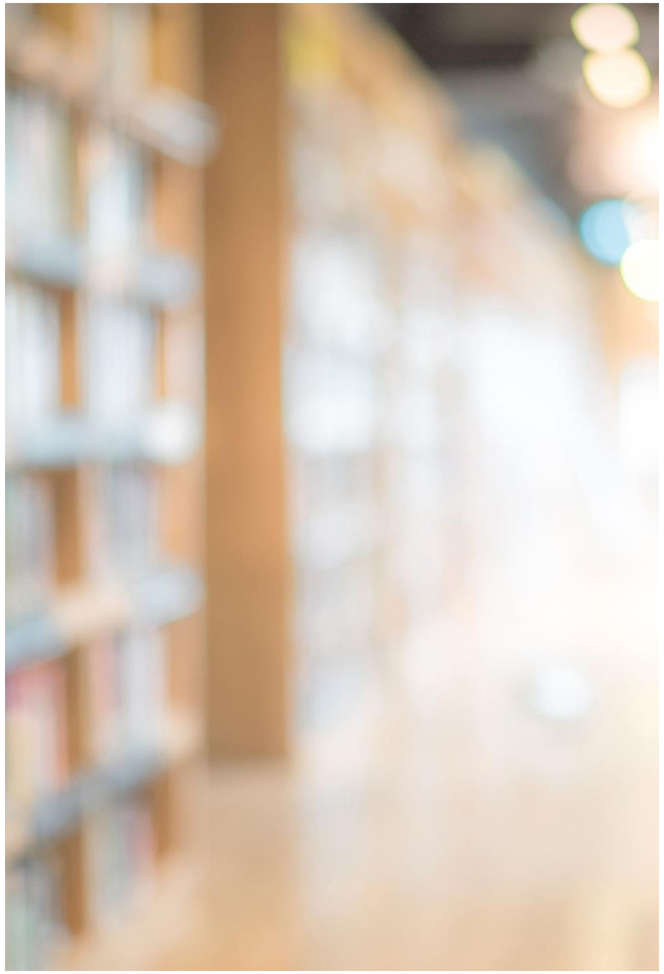
Predicted Values	Actual Values: Positive (1)	Actual Values: Negative (0)
Positive (1)	TP	FP
Negative (0)	FN	TN

Use the below website to practice. This uses the online.

https://machinelearningmastery.com/tensorflow-tutorial-deep-learning-with-tf-keras/

The explanation and code is in there and you can use the copy function to help.

The steps and code are very similar – you should start to notice the similarities yourselves.



Practical: Artificial Intelligence (AI)

In Data Science we process a lot data through AI. With the GDPR, it is becoming increasingly important to understand the ethics behind the data that is collected, stored, processed and evaluated.

Your task is to:

- Find out what Responsible AI is?
- Find instances where AI has failed? Or been used maliciously or incorrectly.
- Implications of when AI fails. There is a specific article in the GDPR Law that covers this, especially with automated decision making. (opt in and out options).
- What should organisations do to ensure that they are being responsible with AI and the wider use of data in general?

WOMEN IN DATA ACADEMY

Extension

Investigate the 3 challenges in AI:

- Time
- Talent
- Trust

WOMEN IN DATA
ACADEMY

TECH TALENT
ACADEMY