README

June 1, 2020

[1]:
```
# Let me provide the context first!
# This is a exercise that simulates the real thing, the interaction between␣
 ↪Data Science and
# Machine Learning teams, in a small scale of course, but you will get the␣
 ↪point :))

# These are the roles:
# 1) I'm Lisa, the Machine Learning Team Leader, and I'm a facilitator.
# 2) Brunno is a Data Scientist and it is working on a very clever model that's␣
 ↪going to change the world and
# make the Business Unit very happy. This model is a classifier which uses 4␣
 ↪features (input variables) to define
# an output.
# 3) You are our Machine Learning Engineer and has a mixed hard-skills such as␣
 ↪coding software, machine learning
# and DevOps.

# Data Science cycle is commonly divided into 4 steps:
# 1. Business understanding
# 2. Data acquisition and understanding (includes capture, preparation,␣
 ↪wrangling, exploration and cleaning)
# 3. Modeling (includes Feature engineering, model training and evaluation)
# 4. Model deployment into production (includes scoring, performance monitoring␣
 ↪etc)
# More info available on:
# https://docs.microsoft.com/en-us/azure/machine-learning/
 ↪team-data-science-process/overview

# As a Machine Learning Engineer you are committed to productize ML models and␣
 ↪make life easier for DS team.
# Challenge is going to prepare you to make the right decisions so that the␣
 ↪transition from step 3 to step 4
# occurs smoothly.
# It means you need to pick up the model made at step 3 and put it into␣
 ↪production on step 4.
```

[1]:
```
# Beginning of the DS journey... bear in mind you don't need touch anything␣
↪here, but it is a good thing
# to understand this snippet of code
```

[1]:
```
# import all important stuff
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

[2]:
```
# loading the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target
```

[3]:
```
# more about this dataset: https://en.wikipedia.org/wiki/Iris_flower_data_set

# visualize what'is inside
pd_iris = pd.DataFrame(data= np.c_[iris['data'], iris['target']],
                       columns= iris['feature_names'] + ['target'])
pd_iris
```

[3]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	\
0	5.1	3.5	1.4	0.2	
1	4.9	3.0	1.4	0.2	
2	4.7	3.2	1.3	0.2	
3	4.6	3.1	1.5	0.2	
4	5.0	3.6	1.4	0.2	
5	5.4	3.9	1.7	0.4	
6	4.6	3.4	1.4	0.3	
7	5.0	3.4	1.5	0.2	
8	4.4	2.9	1.4	0.2	
9	4.9	3.1	1.5	0.1	
10	5.4	3.7	1.5	0.2	
11	4.8	3.4	1.6	0.2	
12	4.8	3.0	1.4	0.1	
13	4.3	3.0	1.1	0.1	
14	5.8	4.0	1.2	0.2	
15	5.7	4.4	1.5	0.4	
16	5.4	3.9	1.3	0.4	
17	5.1	3.5	1.4	0.3	
18	5.7	3.8	1.7	0.3	

19	5.1	3.8	1.5	0.3
20	5.4	3.4	1.7	0.2
21	5.1	3.7	1.5	0.4
22	4.6	3.6	1.0	0.2
23	5.1	3.3	1.7	0.5
24	4.8	3.4	1.9	0.2
25	5.0	3.0	1.6	0.2
26	5.0	3.4	1.6	0.4
27	5.2	3.5	1.5	0.2
28	5.2	3.4	1.4	0.2
29	4.7	3.2	1.6	0.2
..	…	…	…	…
120	6.9	3.2	5.7	2.3
121	5.6	2.8	4.9	2.0
122	7.7	2.8	6.7	2.0
123	6.3	2.7	4.9	1.8
124	6.7	3.3	5.7	2.1
125	7.2	3.2	6.0	1.8
126	6.2	2.8	4.8	1.8
127	6.1	3.0	4.9	1.8
128	6.4	2.8	5.6	2.1
129	7.2	3.0	5.8	1.6
130	7.4	2.8	6.1	1.9
131	7.9	3.8	6.4	2.0
132	6.4	2.8	5.6	2.2
133	6.3	2.8	5.1	1.5
134	6.1	2.6	5.6	1.4
135	7.7	3.0	6.1	2.3
136	6.3	3.4	5.6	2.4
137	6.4	3.1	5.5	1.8
138	6.0	3.0	4.8	1.8
139	6.9	3.1	5.4	2.1
140	6.7	3.1	5.6	2.4
141	6.9	3.1	5.1	2.3
142	5.8	2.7	5.1	1.9
143	6.8	3.2	5.9	2.3
144	6.7	3.3	5.7	2.5
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

	target
0	0.0
1	0.0
2	0.0

```
3       0.0
4       0.0
5       0.0
6       0.0
7       0.0
8       0.0
9       0.0
10      0.0
11      0.0
12      0.0
13      0.0
14      0.0
15      0.0
16      0.0
17      0.0
18      0.0
19      0.0
20      0.0
21      0.0
22      0.0
23      0.0
24      0.0
25      0.0
26      0.0
27      0.0
28      0.0
29      0.0
..      …
120     2.0
121     2.0
122     2.0
123     2.0
124     2.0
125     2.0
126     2.0
127     2.0
128     2.0
129     2.0
130     2.0
131     2.0
132     2.0
133     2.0
134     2.0
135     2.0
136     2.0
137     2.0
138     2.0
```

```
139     2.0
140     2.0
141     2.0
142     2.0
143     2.0
144     2.0
145     2.0
146     2.0
147     2.0
148     2.0
149     2.0

[150 rows x 5 columns]
```

[4]:
```
# replace target's number by labels in pd_iris
def get_label(x):
    if x == 0:
        return 'setosa'
    elif x == 1:
        return 'versicolor'
    else:
        return 'virginica'

pd_iris['target'] = pd_iris['target'].apply(get_label)
pd_iris
```

[4]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	\
0	5.1	3.5	1.4	0.2	
1	4.9	3.0	1.4	0.2	
2	4.7	3.2	1.3	0.2	
3	4.6	3.1	1.5	0.2	
4	5.0	3.6	1.4	0.2	
5	5.4	3.9	1.7	0.4	
6	4.6	3.4	1.4	0.3	
7	5.0	3.4	1.5	0.2	
8	4.4	2.9	1.4	0.2	
9	4.9	3.1	1.5	0.1	
10	5.4	3.7	1.5	0.2	
11	4.8	3.4	1.6	0.2	
12	4.8	3.0	1.4	0.1	
13	4.3	3.0	1.1	0.1	
14	5.8	4.0	1.2	0.2	
15	5.7	4.4	1.5	0.4	
16	5.4	3.9	1.3	0.4	
17	5.1	3.5	1.4	0.3	
18	5.7	3.8	1.7	0.3	
19	5.1	3.8	1.5	0.3	

20	5.4	3.4	1.7	0.2
21	5.1	3.7	1.5	0.4
22	4.6	3.6	1.0	0.2
23	5.1	3.3	1.7	0.5
24	4.8	3.4	1.9	0.2
25	5.0	3.0	1.6	0.2
26	5.0	3.4	1.6	0.4
27	5.2	3.5	1.5	0.2
28	5.2	3.4	1.4	0.2
29	4.7	3.2	1.6	0.2
..	…	…	…	…
120	6.9	3.2	5.7	2.3
121	5.6	2.8	4.9	2.0
122	7.7	2.8	6.7	2.0
123	6.3	2.7	4.9	1.8
124	6.7	3.3	5.7	2.1
125	7.2	3.2	6.0	1.8
126	6.2	2.8	4.8	1.8
127	6.1	3.0	4.9	1.8
128	6.4	2.8	5.6	2.1
129	7.2	3.0	5.8	1.6
130	7.4	2.8	6.1	1.9
131	7.9	3.8	6.4	2.0
132	6.4	2.8	5.6	2.2
133	6.3	2.8	5.1	1.5
134	6.1	2.6	5.6	1.4
135	7.7	3.0	6.1	2.3
136	6.3	3.4	5.6	2.4
137	6.4	3.1	5.5	1.8
138	6.0	3.0	4.8	1.8
139	6.9	3.1	5.4	2.1
140	6.7	3.1	5.6	2.4
141	6.9	3.1	5.1	2.3
142	5.8	2.7	5.1	1.9
143	6.8	3.2	5.9	2.3
144	6.7	3.3	5.7	2.5
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

	target
0	setosa
1	setosa
2	setosa
3	setosa

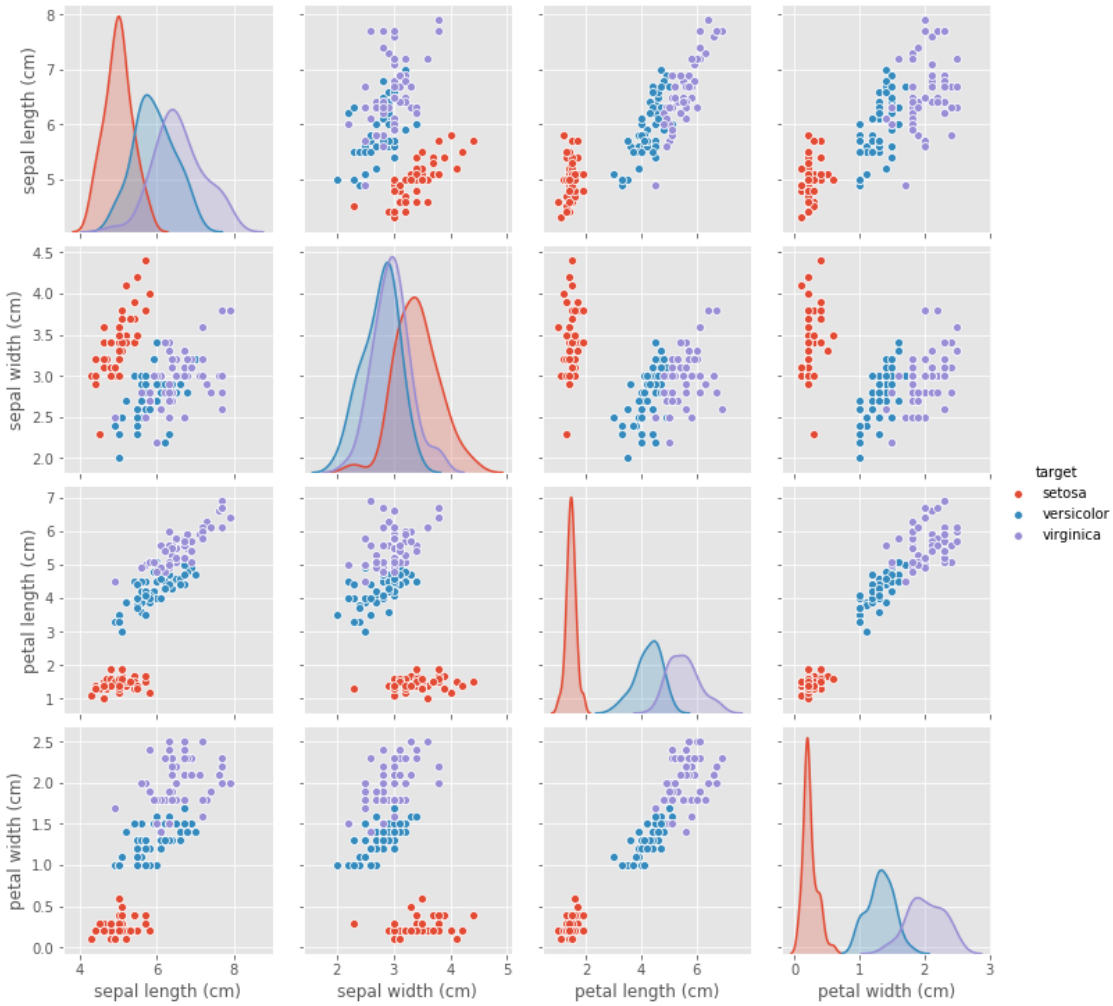
```
4       setosa
5       setosa
6       setosa
7       setosa
8       setosa
9       setosa
10      setosa
11      setosa
12      setosa
13      setosa
14      setosa
15      setosa
16      setosa
17      setosa
18      setosa
19      setosa
20      setosa
21      setosa
22      setosa
23      setosa
24      setosa
25      setosa
26      setosa
27      setosa
28      setosa
29      setosa
..         …
120  virginica
121  virginica
122  virginica
123  virginica
124  virginica
125  virginica
126  virginica
127  virginica
128  virginica
129  virginica
130  virginica
131  virginica
132  virginica
133  virginica
134  virginica
135  virginica
136  virginica
137  virginica
138  virginica
139  virginica
```

```
140  virginica
141  virginica
142  virginica
143  virginica
144  virginica
145  virginica
146  virginica
147  virginica
148  virginica
149  virginica

[150 rows x 5 columns]
```

[5]:
```
# see the scatter plot of the data set
plt.style.use('ggplot')
sns.pairplot(pd_iris, hue= 'target')
```

[5]:
```
<seaborn.axisgrid.PairGrid at 0x7f23463d19b0>
```



[7]:
```
# split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42,␣
↪test_size=0.5)
```

[8]:
```
# build the model
clf = RandomForestClassifier(n_estimators=10)
```

[9]:
```
# train the classifier
clf.fit(X_train, y_train)
```

[9]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

[14]:
```
# make some predictions
predicted = clf.predict(X_test)
```

[17]:
```
# check accuracy of this model
print(accuracy_score(predicted, y_test))
```

```
0.9733333333333334
```

[18]:
```
# End of the DS journey!!!
# clf model seems okay and Brunno has just said: "I need it into production␣
↪pleeeeeease"
# From now on, you, as machine learning engineer, needs to make it available␣
↪for business unit and
# make Brunno happy
```

[19]:
```
# But before you go ahead,let me give you some instructions to make your life␣
↪easier:

# 1. In general, the trained model should be saved in a file and restored in␣
↪order to reuse it: The saving of data
# is called serialization, while restoring the data is called deserialization

# 2. clf model needs to be available to anyone on Internet (and locally for␣
↪development tests)

# 3. clf model shall be triggered through a synchronous HTTP request
```

```
# 4. User shall be able to access this clf model through the Browser, Postman␣
 ↪or any another tool like that

# 5. Post method shall have the four mandatory features, i.e. sepal length,␣
 ↪sepal width, petal lentgh, petal width
# and model's answer shall be 0, 1 or 2 (i.e. setosa, virginica or versicolor)
#
# request is something like that:
# features_input = [[5.8, 2.7, 5.1, 1.9]]
# model's output should be like that:
# output_prediction = clf.predict_proba(features_input)

# 6. Solution needs to be scalable and run everywhere

# 7. Containerization is a key point

# 8. Python coding shall simplify a little bit how things should be done in the␣
 ↪backend

# 9. It might be good to include a simple health check: we can easily check if␣
 ↪model is on or not
```

[20]:
```
# What we expect you can deliver at the end of this challenge:

# 1. Present an architectural diagram of the solution: list all the components␣
 ↪required to make clf model
# available in the production environment

# 2. Put clf model to work locally or in any cloud but don't forget the steps 6/
 ↪7 above (run everywhere) and
# make it very simple to test, i.e. using this POST request: {"sepal_l": 5,␣
 ↪"sepal_w": 2, "petal_l": 3, "petal_w": 4}
```

[21]:
```
# I hope you can make your best and I wish you a good luck!!
```

[22]:
```
# End of challenge
```