



University of Minho  
School of Engineering



# Dados e Aprendizagem Automática

## Data Exploration and Preparation

DAA @ MEI-1º/MiEI-4º – 1º Semestre

Bruno Fernandes, César Analide, Dalila Alves, Filipa Ferraz, Victor Alves

Part III

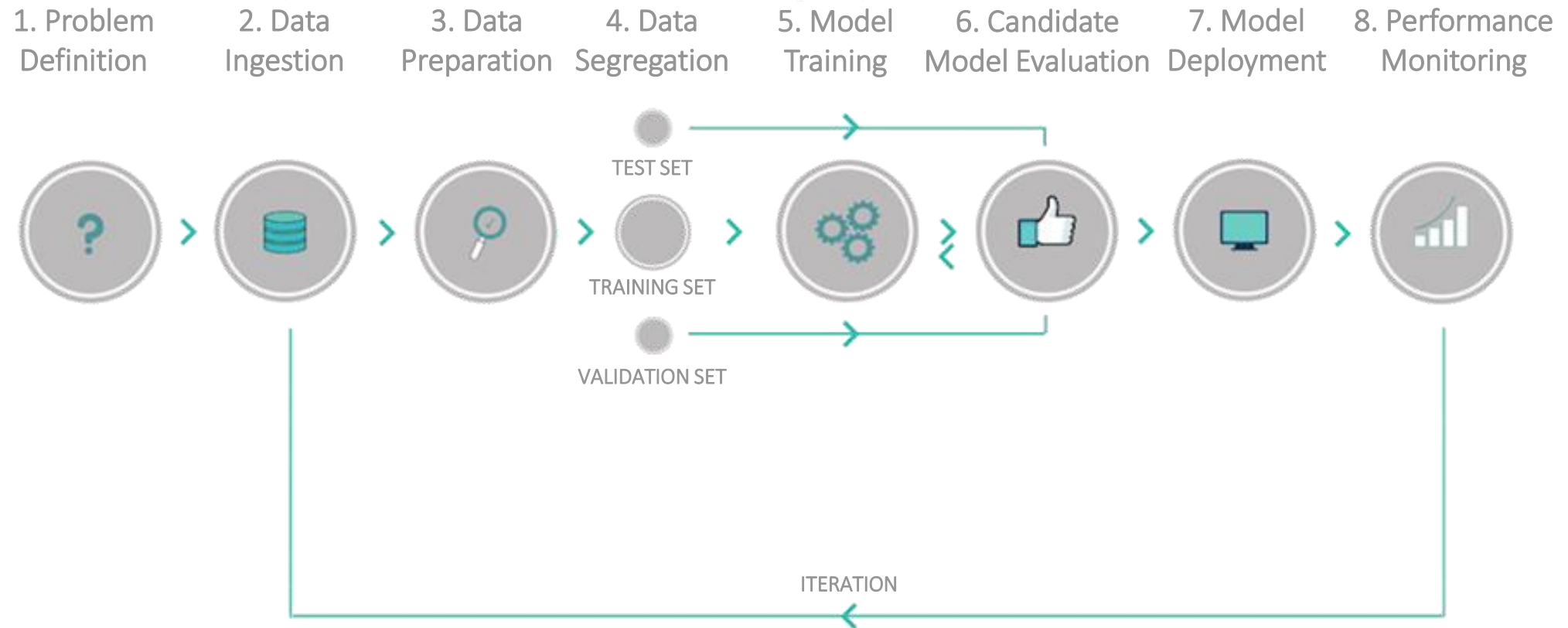
# Contents

2

- Understanding the problem
- Data Exploration
- Data Preparation
- Hands On

# Understanding the problem

3



We must look to our data... We must understand it!

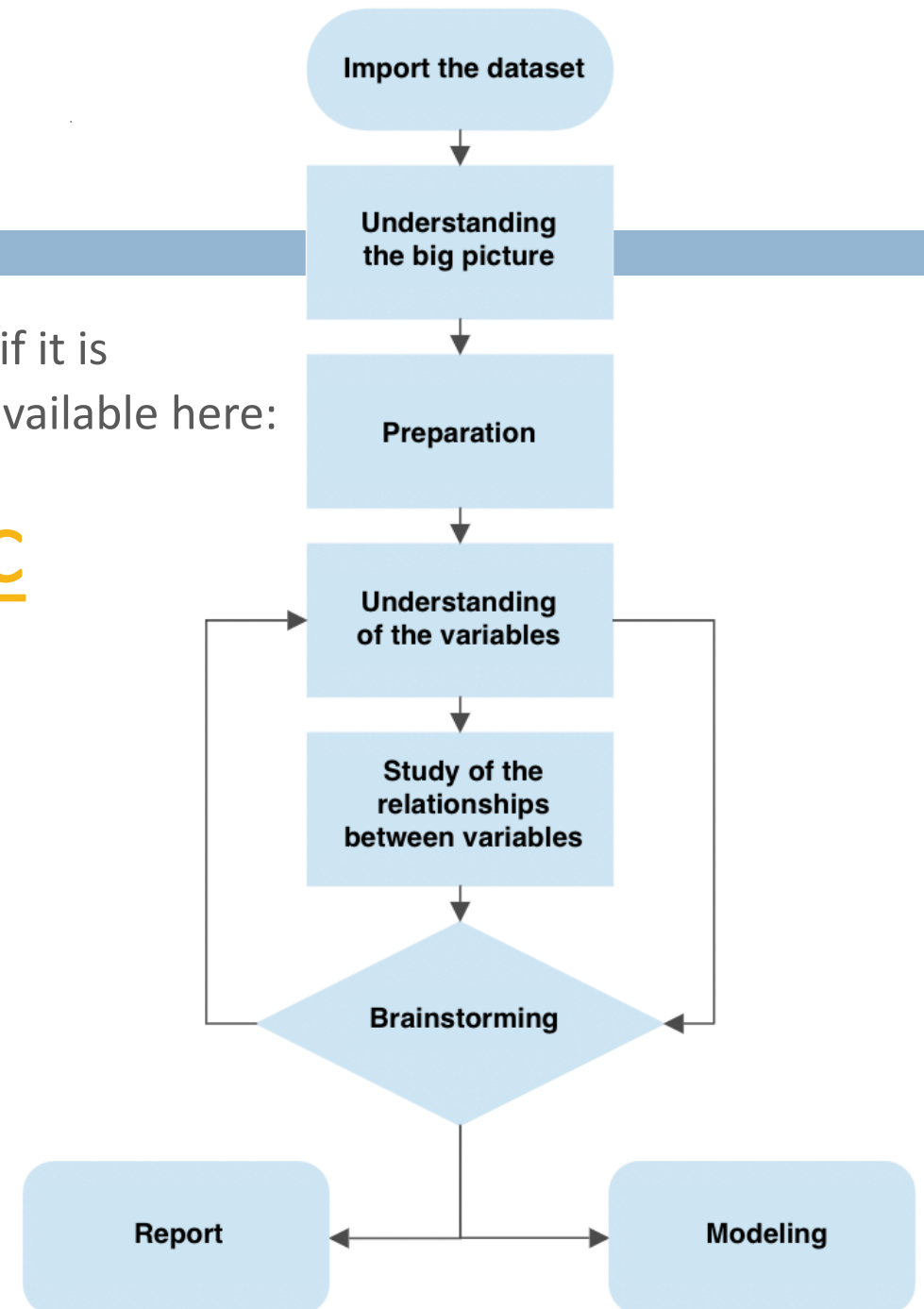
**Data understanding** is a huge step of the process and, as so, it will take its time! Nonetheless, it will give us a flavor of our dataset, at each variable, their meaning, and their relevance to this problem

# Understanding the problem

4

Let's understand the features' type, how important it may be, if it is described in any other feature, ... Let's use the **wine dataset**, available here:

<https://tinyurl.com/4cshpfac>



# Imports

5

Import libraries

```
import sklearn as skl
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
```

Load the dataset and inspect some meta-data

```
'''
Load CSV
'''
df = pd.read_csv('wine.csv')
```

# Data Understanding

6

What about actual data? What can we **see/get/understand** from these data?

```
'''  
Inspect data  
'''  
print(df.columns)
```

```
Index(['Alcohol', 'Malic acid', 'Ash', 'Alcalinity of ash', 'Magnesium',  
      'Total phenols', 'Flavanoids', 'Nonflavanoid phenols',  
      'Proanthocyanins', 'Color intensity', 'Hue',  
      'OD280/OD315 of diluted wines', 'Proline', 'Class'],  
      dtype='object')
```

```
df.head()
```

	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue	OD280/OD315 of diluted wines	Proline	Class
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065	one
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050	one
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185	one
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480	one
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735	one

# Data Understanding

7

```
df.tail()
```

	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue	OD280/OD315 of diluted wines	Proline	Class
<b>173</b>	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	1.06	7.7	0.64	1.74	740	three
<b>174</b>	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	1.41	7.3	0.70	1.56	750	three
<b>175</b>	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	1.35	10.2	0.59	1.56	835	three
<b>176</b>	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	1.46	9.3	0.60	1.62	840	three
<b>177</b>	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	1.35	9.2	0.61	1.60	560	three

```
df.shape
```

```
(178, 14)
```

We can see that we have 178 entries with 14 attributes each.

The Class has 3 classifications: one, two and three that refer to the type of wine.

# Data Understanding

8

```
#df.dtypes  
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 178 entries, 0 to 177  
Data columns (total 14 columns):  
 #   Column                                     Non-Null Count  Dtype    
---  -  
 0   Alcohol                                 178 non-null    float64  
 1   Malic acid                             178 non-null    float64  
 2   Ash                                    178 non-null    float64  
 3   Alcalinity of ash                      178 non-null    float64  
 4   Magnesium                             178 non-null    int64  
 5   Total phenols                          178 non-null    float64  
 6   Flavanoids                             178 non-null    float64  
 7   Nonflavanoid phenols                   178 non-null    float64  
 8   Proanthocyanins                        178 non-null    float64  
 9   Color intensity                        178 non-null    float64  
10   Hue                                    178 non-null    float64  
11   OD280/OD315 of diluted wines          178 non-null    float64  
12   Proline                                178 non-null    int64  
13   Class                                  178 non-null    object  
dtypes: float64(11), int64(2), object(1)  
memory usage: 19.6+ KB
```

There aren't null values and the attributes are all numeric except the Class.



# Data Understanding

9

We can also get some descriptive stats (for the entire numerical data or just the desired ones)...

```
df.describe()
```

	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue	OD280/OD315 of diluted wines	Proline
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000
mean	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.029270	0.361854	1.590899	5.058090	0.957449	2.611685	746.893258
std	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.998859	0.124453	0.572359	2.318286	0.228572	0.709990	314.907474
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.340000	0.130000	0.410000	1.280000	0.480000	1.270000	278.000000
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.205000	0.270000	1.250000	3.220000	0.782500	1.937500	500.500000
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.135000	0.340000	1.555000	4.690000	0.965000	2.780000	673.500000
75%	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.875000	0.437500	1.950000	6.200000	1.120000	3.170000	985.000000
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.080000	0.660000	3.580000	13.000000	1.710000	4.000000	1680.000000

```
df['Color intensity'].describe()
```

```
count    178.000000
mean      5.058090
std       2.318286
min       1.280000
25%       3.220000
50%       4.690000
75%       6.200000
max      13.000000
Name: Color intensity, dtype: float64
```

# Data Understanding

10

What about missing values?

```
'''  
Missing data  
'''  
df.isna().any()
```

Alcohol	False
Malic acid	False
Ash	False
Alcalinity of ash	False
Magnesium	False
Total phenols	False
Flavanoids	False
Nonflavanoid phenols	False
Proanthocyanins	False
Color intensity	False
Hue	False
OD280/OD315 of diluted wines	False
Proline	False
Class	False
dtype:	bool

```
print(df.isna().sum())
```

Alcohol	0
Malic acid	0
Ash	0
Alcalinity of ash	0
Magnesium	0
Total phenols	0
Flavanoids	0
Nonflavanoid phenols	0
Proanthocyanins	0
Color intensity	0
Hue	0
OD280/OD315 of diluted wines	0
Proline	0
Class	0
dtype:	int64

# Data Understanding

11

With our analysis, we can characterize better the dataset:

- It has **178 entries**
- **14 attributes** - 13 are physical-chemical properties of the wine and 1 is the classification
- All **non-null values**
- There **aren't missing values**

The **goal** of working with this dataset can be **identify the type of wine by its properties** - the target is a numeric categorical variable that covers the values of one, two and three.

If used for **modeling**, the features of the wine can be used to **predict its type**.

# Data Preparation

12

It consists of multiple steps... Many times (in reality, a lot of times), you'll need to check the API of the lib you are using... Here are some links you may save for future reference:

- Numpy (<https://numpy.org/doc/stable/>)
- Pandas (<https://pandas.pydata.org/docs/>)
- Matplotlib (<https://matplotlib.org/stable/contents.html>)
- Seaborn (<https://seaborn.pydata.org/api.html>)
- Scikit Learn (<https://scikit-learn.org/stable/modules/classes.html>)

# Data Preparation

13

And here are some basic Pandas functions you'll need (somewhen) in the future for data prep.:

- `pandas.DataFrame.drop`
- `pandas.DataFrame.drop_duplicates`
- `pandas.DataFrame.fillna`
- `pandas.DataFrame.isna`
- `pandas.DataFrame.interpolate`
- `pandas.DataFrame.dropna`
- `pandas.DataFrame.groupby`
- `pandas.DataFrame.loc`
- `pandas.DataFrame.iloc`
- ...

# Data Preparation

14

And here are some basic sklearn functions/classes you'll need (somewhen) in the future for data prep.:

- `sklearn.preprocessing.MinMaxScaler`
- `sklearn.preprocessing.StandardScaler`
- `sklearn.preprocessing.KBinsDiscretizer`
- `sklearn.preprocessing.LabelEncoder`
- `sklearn.feature_selection`
- `sklearn.metrics`
- ...

# Data Preparation and Transformation

15

## Remove duplicate values

```
'''
Drop Duplicates
'''

print(df.duplicated().sum())
print(df.drop_duplicates(inplace=True))
print(df.info())

0
None
<class 'pandas.core.frame.DataFrame'>
Int64Index: 178 entries, 0 to 177
Data columns (total 14 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Alcohol                               178 non-null    float64
1   Malic acid                            178 non-null    float64
2   Ash                                   178 non-null    float64
3   Alcalinity of ash                     178 non-null    float64
4   Magnesium                             178 non-null    int64
5   Total phenols                         178 non-null    float64
6   Flavanoids                            178 non-null    float64
7   Nonflavanoid phenols                  178 non-null    float64
8   Proanthocyanins                       178 non-null    float64
9   Color intensity                       178 non-null    float64
10  Hue                                   178 non-null    float64
11  OD280/OD315 of diluted wines          178 non-null    float64
12  Proline                               178 non-null    int64
13  Class                                 178 non-null    object
dtypes: float64(11), int64(2), object(1)
memory usage: 20.9+ KB
None
```

## Rename attributes

```
'''
Rename complicated columns' names
'''

df.rename(columns={"OD280/OD315 of diluted wines": "Protein Concentration"}, inplace=True)
df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 178 entries, 0 to 177
Data columns (total 14 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Alcohol                               178 non-null    float64
1   Malic acid                            178 non-null    float64
2   Ash                                   178 non-null    float64
3   Alcalinity of ash                     178 non-null    float64
4   Magnesium                             178 non-null    int64
5   Total phenols                         178 non-null    float64
6   Flavanoids                            178 non-null    float64
7   Nonflavanoid phenols                  178 non-null    float64
8   Proanthocyanins                       178 non-null    float64
9   Color intensity                       178 non-null    float64
10  Hue                                   178 non-null    float64
11  Protein Concentration                  178 non-null    float64
12  Proline                               178 non-null    int64
13  Class                                 178 non-null    object
dtypes: float64(11), int64(2), object(1)
memory usage: 20.9+ KB
```

# Data Preparation and Transformation

16

```
...  
Remove values (Ash smaller than 2, Alcalinity bigger than 15)  
...  
df_clean = df.drop(df.loc[(df['Ash']<2) & (df['Alcalinity of ash']>15)].index)  
print(df_clean)  
#df.drop(df.loc[(df['Ash']<2) & (df['Alcalinity of ash']>15)].index, inplace=True) ##alternative
```

	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	\
0	14.23	1.71	2.43	15.6	127	2.80	
1	13.20	1.78	2.14	11.2	100	2.65	
2	13.16	2.36	2.67	18.6	101	2.80	
3	14.37	1.95	2.50	16.8	113	3.85	
4	13.24	2.59	2.87	21.0	118	2.80	
..	...	...	...	...	...	...	
173	13.71	5.65	2.45	20.5	95	1.68	
174	13.40	3.91	2.48	23.0	102	1.80	
175	13.27	4.28	2.26	20.0	120	1.59	
176	13.17	2.59	2.37	20.0	120	1.65	
177	14.13	4.10	2.74	24.5	96	2.05	

	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue	\
0	3.06		0.28	2.29	5.64	1.04
1	2.76		0.26	1.28	4.38	1.05
2	3.24		0.30	2.81	5.68	1.03
3	3.49		0.24	2.18	7.80	0.86
4	2.69		0.39	1.82	4.32	1.04
..	...		...	...	...	...
173	0.61		0.52	1.06	7.70	0.64
174	0.75		0.43	1.41	7.30	0.70
175	0.69		0.43	1.35	10.20	0.59
176	0.68		0.53	1.46	9.30	0.60
...						

Since all the variables appear to be physical-chemical measures, they could all be useful and help define the segmentation of the type of wine. There is no reason to remove columns.



# Univariate Analysis

17

Iterate through each and every relevant variable and get basic information.

```
'''  
Categorical variables  
'''  
df['Alcohol'].value_counts()
```

```
12.37    6  
13.05    6  
12.08    5  
12.29    4  
12.00    3  
..  
13.34    1  
13.69    1  
13.90    1  
13.84    1  
13.75    1  
Name: Alcohol, Length: 126, dtype: int64
```

```
df['Class'].value_counts(normalize=True)
```

```
two      0.398876  
one      0.331461  
three    0.269663  
Name: Class, dtype: float64
```

```
'''  
Numeric variables  
'''  
df['Magnesium'].describe()
```

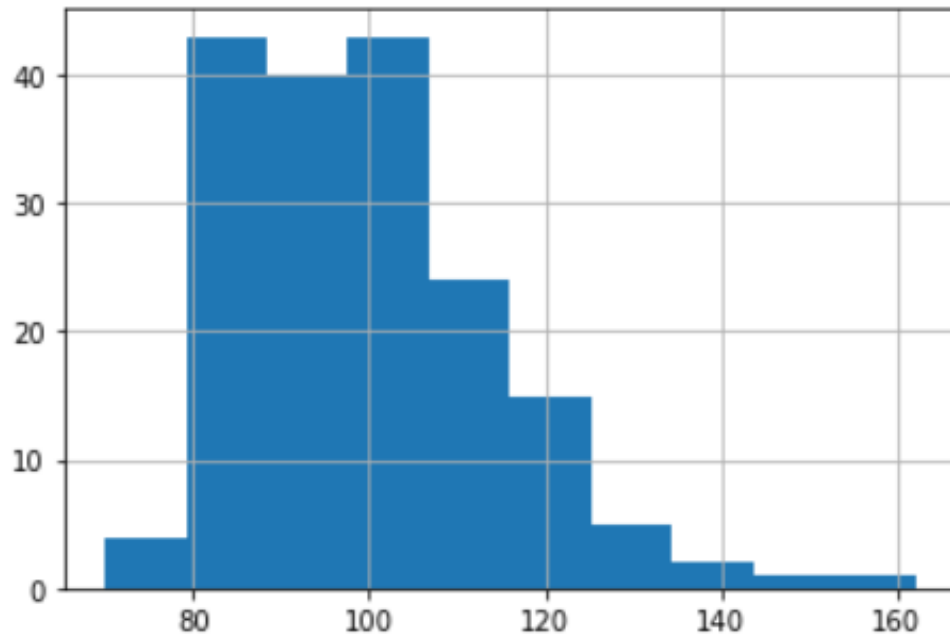
```
count    178.000000  
mean      99.741573  
std       14.282484  
min       70.000000  
25%       88.000000  
50%       98.000000  
75%      107.000000  
max      162.000000  
Name: Magnesium, dtype: float64
```

# Univariate Analysis

18

```
print(f"Histogram: {df['Magnesium'].hist()}")
```

Histogram: AxesSubplot(0.125,0.125;0.775x0.755)



Does not follow a normal curve and has spikes.

```
print(f"Skewness: {df['Magnesium'].skew()}")
```

Skewness: 1.098191054755161

```
print(f"Kurtosis: {df['Magnesium'].kurt()}")
```

Kurtosis: 2.1049913235905557

Kurtosis and asymmetry values are greater than 1.

# Univariate Analysis

19

Now we can summarize the dataset creating a small document with detailed information:

- *Variable*: name of the variable
- *Type*: the type or format of the variable. This can be categorical, numeric, Boolean, and so on
- *Context*: useful information to understand the semantic space of the variable. In the case of our dataset, the context is always the chemical-physical one
- *Expectation*: how relevant is this variable with respect to our task? We can use a scale “High, Medium, Low”.
- *Comments*: whether or not we have any comments to make on the variable

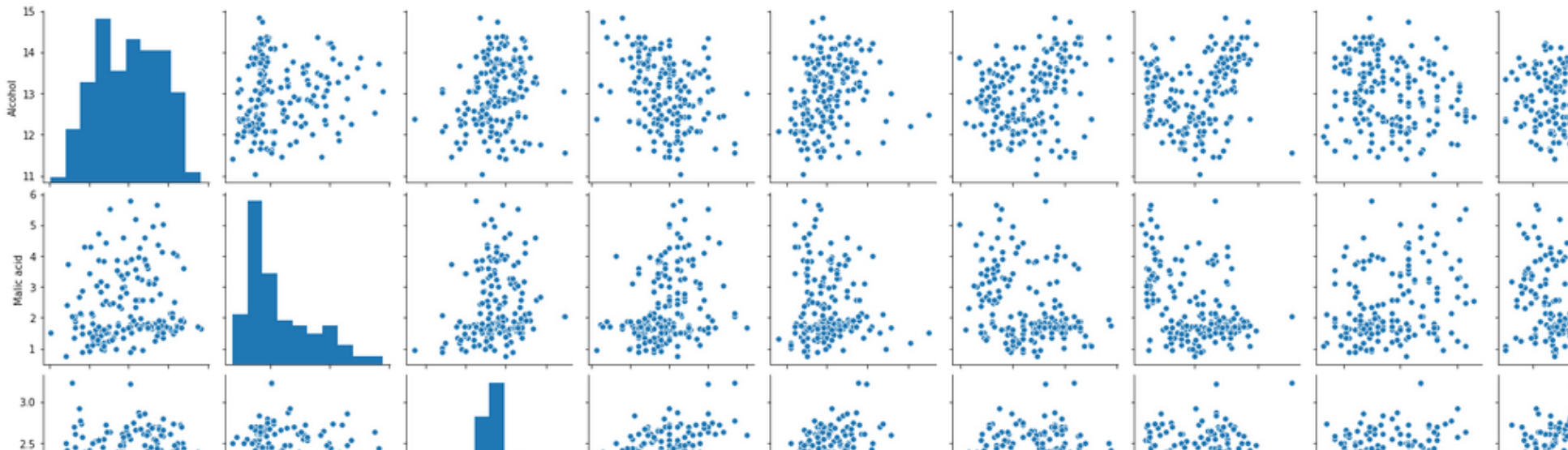
# Multivariate Analysis

20

We can start by seeing the relation between all variables:

```
...  
All variables  
...  
sns.pairplot(df)
```

<seaborn.axisgrid.PairGrid at 0x7f08ccac79d0>



...

# Multivariate Analysis

21

We can group by variables:

```
'''  
Grouping  
'''  
df.groupby(by=['Class']).mean()
```

	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	P	...
Class										
one	13.744746	2.010678	2.455593	17.037288	106.338983	2.840169	2.982373	0.290000		
three	13.153750	3.333750	2.437083	21.416667	99.312500	1.678750	0.781458	0.447500		
two	12.278732	1.932676	2.244789	20.238028	94.549296	2.258873	2.080845	0.363662		



# Multivariate Analysis

23

We can group by variables:

```
print(df.groupby(by=['Alcohol']).agg(pd.Series.mode))
```

Alcohol	Malic acid	Ash	Alcalinity of ash	Total phenols	\
11.03	1.51	2.2	21.5	2.46	
11.41	0.74	2.5	21	2.48	
11.45	2.4	2.42	20	2.9	
11.46	3.74	1.82	19.5	3.18	
11.56	2.05	3.23	28.5	3.18	
...	...	...	...	...	
14.37	1.95	2.5	16.8	3.85	
14.38	[1.87, 3.59]	[2.28, 2.38]	[12.0, 16.0]	[3.25, 3.3]	
14.39	1.87	2.45	14.6	2.5	
14.75	1.73	2.39	11.4	3.1	
14.83	1.64	2.17	14	2.8	

Alcohol	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	\
11.03	2.17	0.52	2.01	1.9	
11.41	2.01	0.42	1.44	3.08	
11.45	2.79	0.32	1.83	3.25	
11.46	2.58	0.24	3.18	3.2	
...	...	...	...	...	

```
print(df.groupby(by=['Alcohol', 'Flavanoids']).mean())
```

Alcohol	Flavanoids	Malic acid	Ash	Alcalinity of ash	Magnesium	\
11.03	2.17	1.51	2.20	21.5	85	
11.41	2.01	0.74	2.50	21.0	88	
11.45	2.79	2.40	2.42	20.0	96	
11.46	2.58	3.74	1.82	19.5	107	
11.56	5.08	2.05	3.23	28.5	119	
...	...	...	...	...	...	
14.38	3.17	3.59	2.28	16.0	102	
	3.64	1.87	2.38	12.0	102	
14.39	2.52	1.87	2.45	14.6	96	
14.75	3.69	1.73	2.39	11.4	91	
14.83	2.98	1.64	2.17	14.0	97	

Alcohol	Flavanoids	Total phenols	Nonflavanoid phenols	Proanthocyanins	\
11.03	2.17	2.46	0.52	2.01	
11.41	2.01	2.48	0.42	1.44	
11.45	2.79	2.90	0.32	1.83	
11.46	2.58	3.18	0.24	3.58	
11.56	5.08	3.18	0.47	1.87	
...	...	...	...	...	
14.38	3.17	3.25	0.27	2.19	
	3.64	3.30	0.29	2.96	
...	...	...	...	...	

# Multivariate Analysis

24

We can create bins.

**Data binning** (or bucketing) groups data in bins (or buckets), in the sense that it replaces values contained into a small interval with a single representative value for that interval. It is a type of data preprocessing, a mechanism which includes also dealing with missing values, formatting, normalization and standardization.

Binning is a technique for data smoothing. Data smoothing is employed to remove noise from data.

```
...
Bins
...

# https://scikit-learn.org/stable/modules/preprocessing.html#discretization
estimator = preprocessing.KBinsDiscretizer(n_bins=3, encode='ordinal', strategy='quantile')
df['alcohol_binned'] = estimator.fit_transform(df[['Alcohol']])
print('Bin Edges')
print(estimator.bin_edges_[0])
print('Alcohol Groups')
print(df.groupby(by=['alcohol_binned']).count())
```

Bin Edges

[11.03 12.52 13.48 14.83]

Alcohol Groups

	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium \
alcohol_binned					
0.0	59	59	59	59	59
1.0	58	58	58	58	58
2.0	61	61	61	61	61

	Total phenols	Flavanoids	Nonflavanoid phenols \
alcohol_binned			
0.0	59	59	59
1.0	58	58	58
2.0	61	61	61

	Proanthocyanins	Color intensity	Hue	Protein Concentration \
alcohol_binned				
0.0	59	59	59	59
1.0	58	58	58	58

...

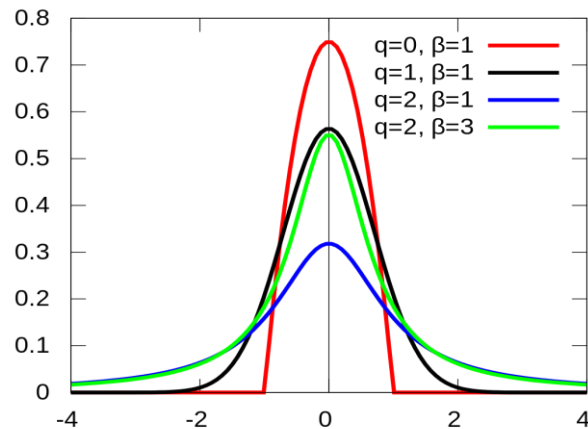


# Multivariate Analysis

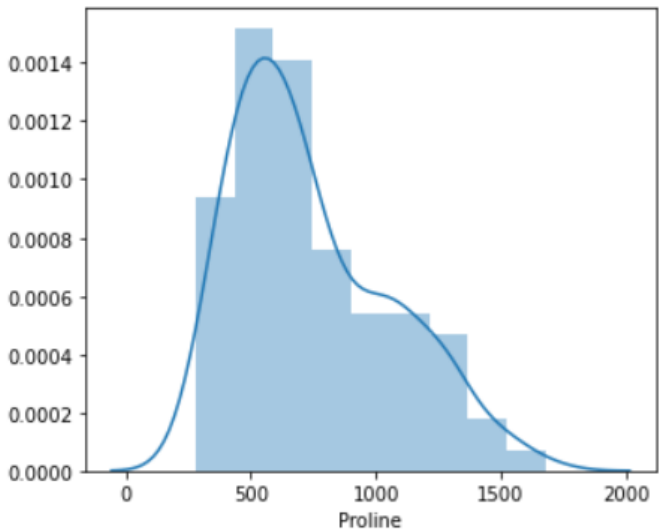
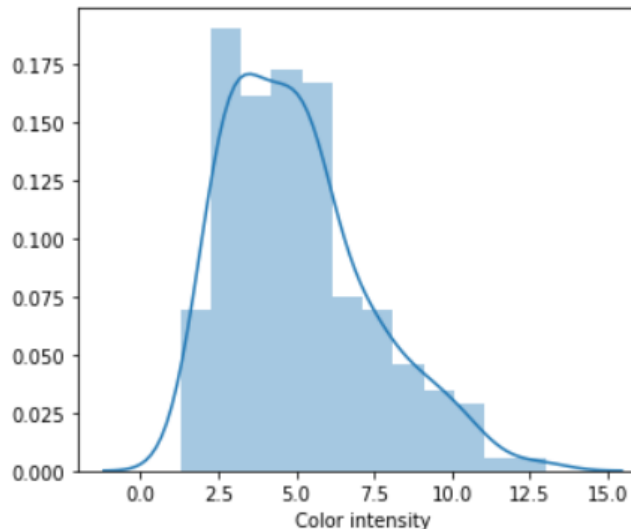
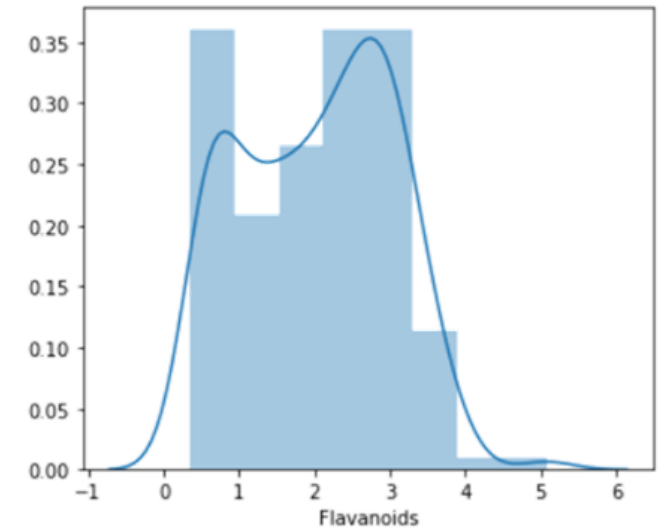
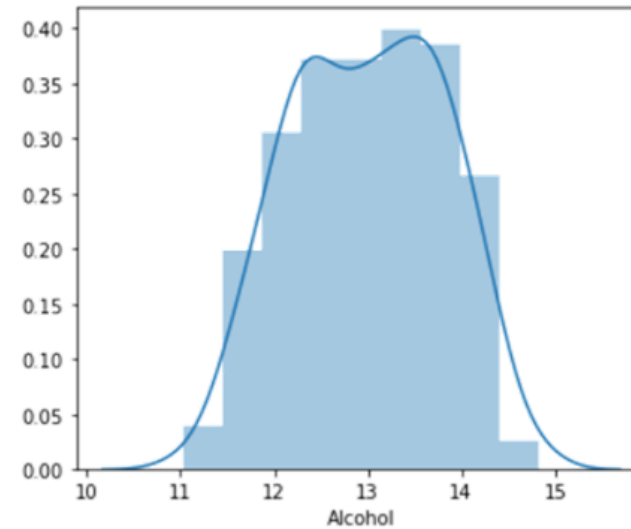
25

Dispersion: does it follow a Gaussian distribution?

```
'''  
Statistical Dispersion  
'''  
fig, axs = plt.subplots(2, 2, figsize=(12, 10))  
fig.suptitle('Histograms')  
  
sns.distplot(df['Alcohol'], ax=axs[0, 0], kde=True)  
sns.distplot(df['Flavanoids'], ax=axs[0, 1], kde=True)  
sns.distplot(df['Color intensity'], ax=axs[1, 0], kde=True)  
sns.distplot(df['Proline'], ax=axs[1, 1], kde=True)
```



[https://en.wikipedia.org/wiki/Q-Gaussian\\_distribution](https://en.wikipedia.org/wiki/Q-Gaussian_distribution)



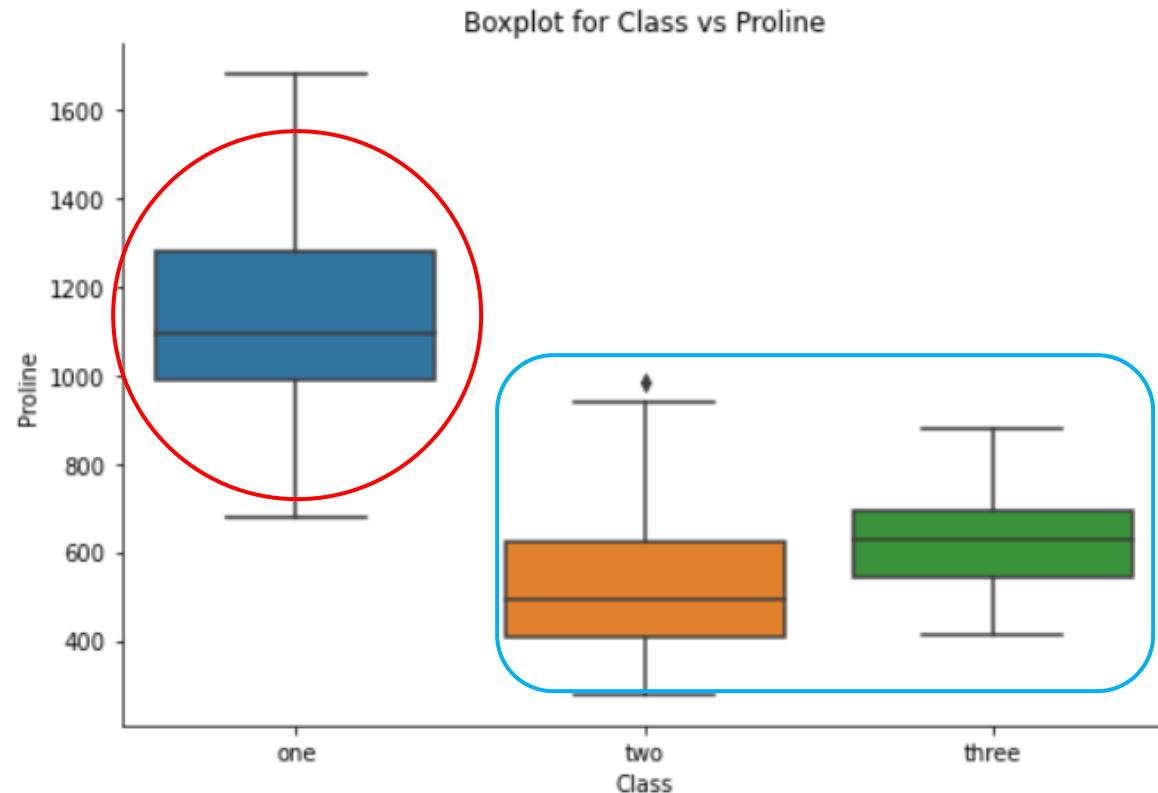
# Multivariate Analysis

26

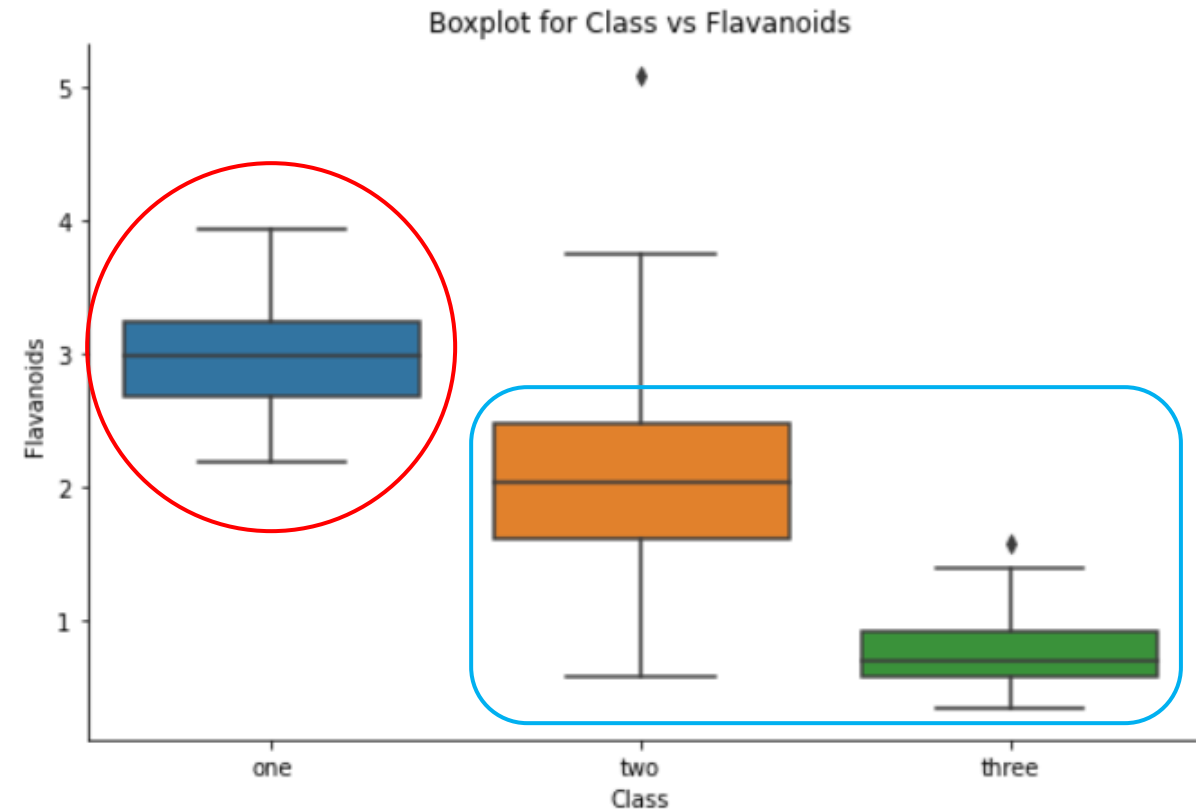
```
...  
Box plots (Outliers)  
...
```

The best way to understand the relationship between a numeric variable and a categorical variable is through a boxplot.

```
sns.catplot(x="Class", y="Proline", data=df, kind="box", aspect=1.5)  
plt.title("Boxplot for Class vs Proline")  
plt.show()
```



```
sns.catplot(x="Class", y="Flavanoids", data=df, kind="box", aspect=1.5)  
plt.title("Boxplot for Class vs Flavanoids")  
plt.show()
```

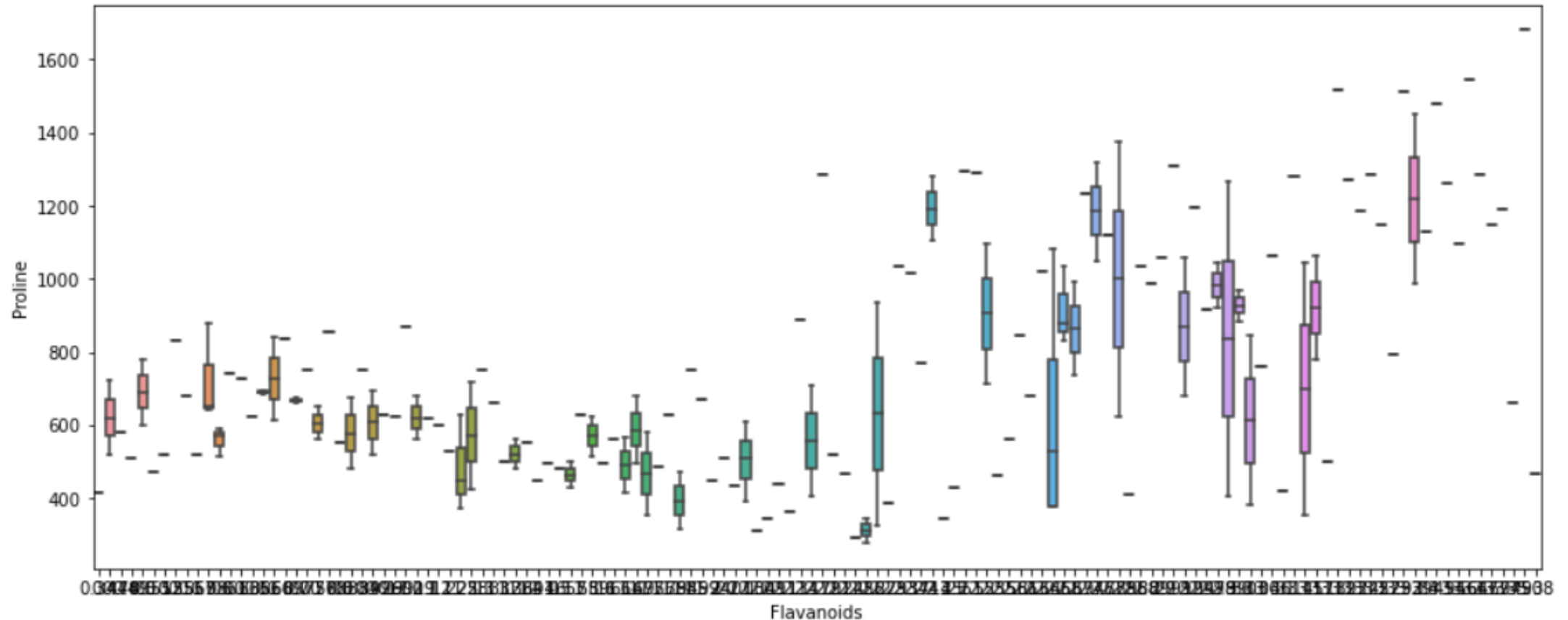


# Multivariate Analysis

27

```
_, ax = plt.subplots(figsize=(15, 6))  
fig.suptitle('Boxplot for Flavanoids vs Proline')  
sns.boxplot(x=df["Flavanoids"], y=df["Proline"])
```

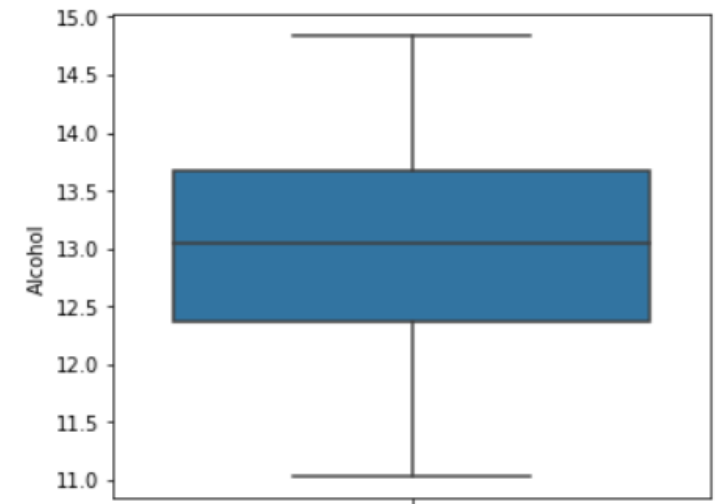
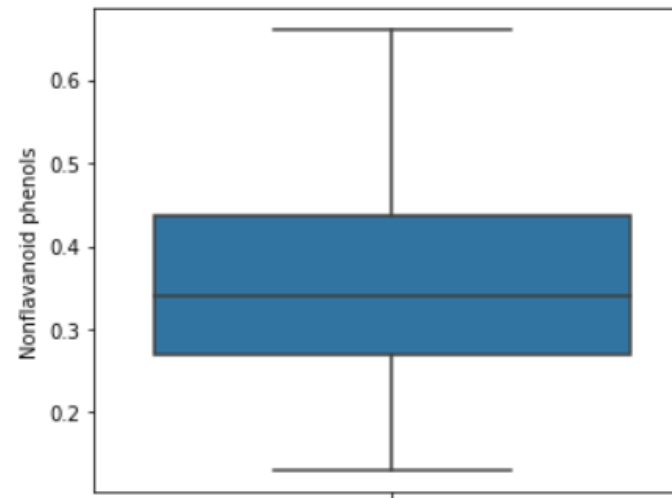
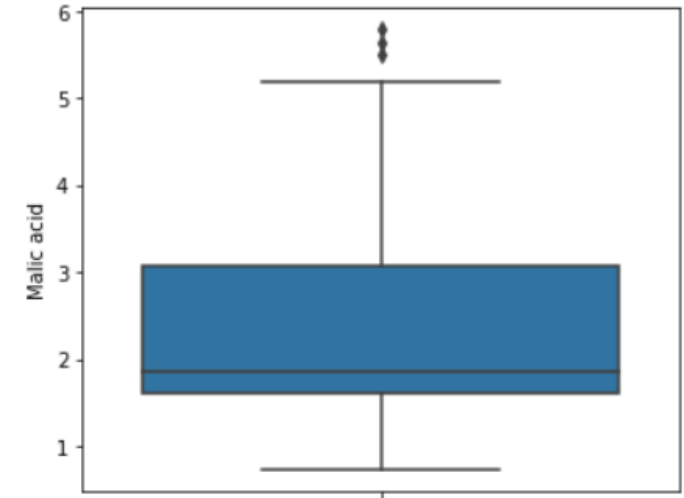
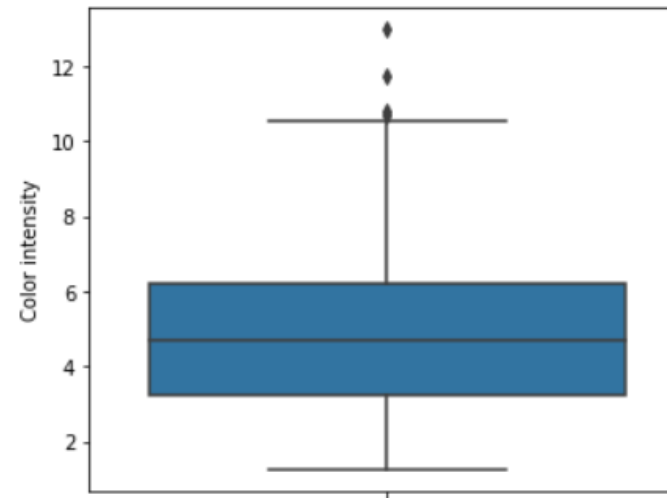
<AxesSubplot:xlabel='Flavanoids', ylabel='Proline'>



# Multivariate Analysis

28

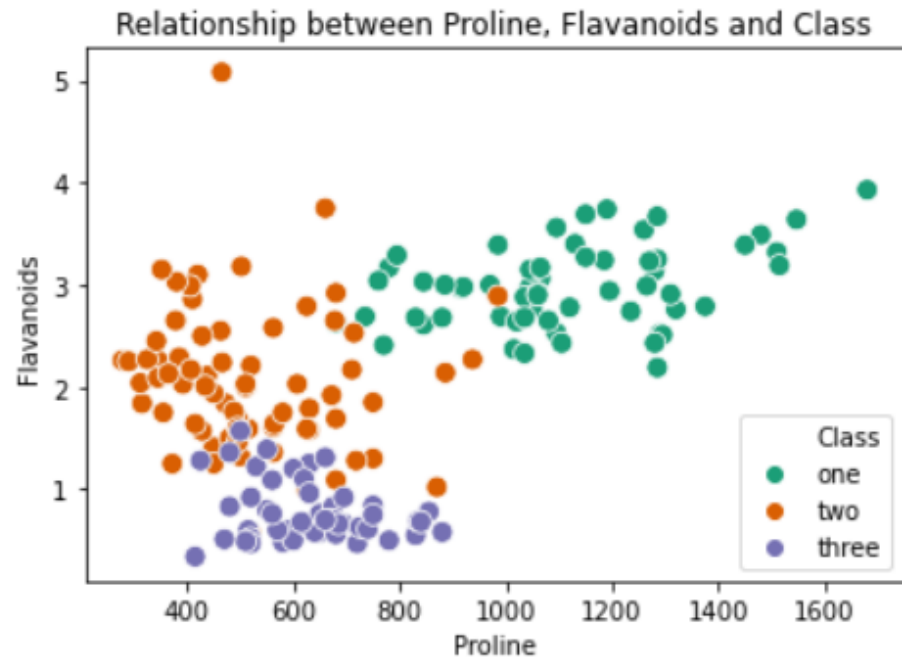
```
fig, axs = plt.subplots(2, 2, figsize=(12, 10))
fig.suptitle('Boxplots for 4 variables')
sns.boxplot(y=df['Color intensity'], ax=axs[0, 0])
sns.boxplot(y=df['Malic acid'], ax=axs[0, 1])
sns.boxplot(y=df['Nonflavanoid phenols'], ax=axs[1, 0])
sns.boxplot(y=df['Alcohol'], ax=axs[1, 1])
```



# Multivariate Analysis

29

```
'''  
Scatter plots  
'''  
sns.scatterplot(x="Proline", y="Flavanoids", hue="Class", data=df, palette="Dark2", s=80)  
plt.title("Relationship between Proline, Flavanoids and Class")  
plt.show()
```

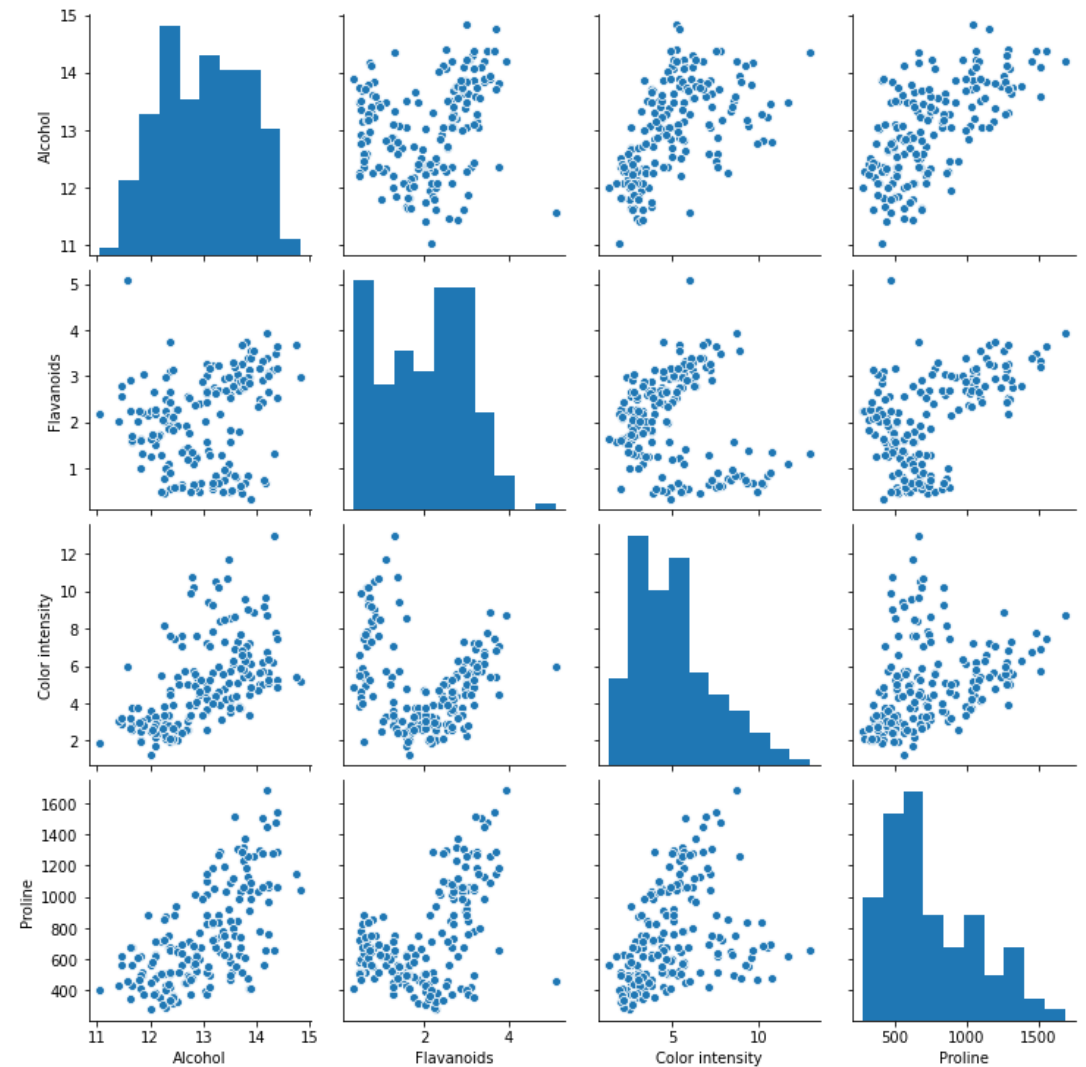


For class one the proline levels are much higher while the flavanoid level is stable around the value of 3.

# Multivariate Analysis

30

```
'''  
Relations  
'''  
cols = ['Alcohol', 'Flavanoids', 'Color intensity', 'Proline']  
_ = sns.pairplot(df[cols], height = 2.5)
```



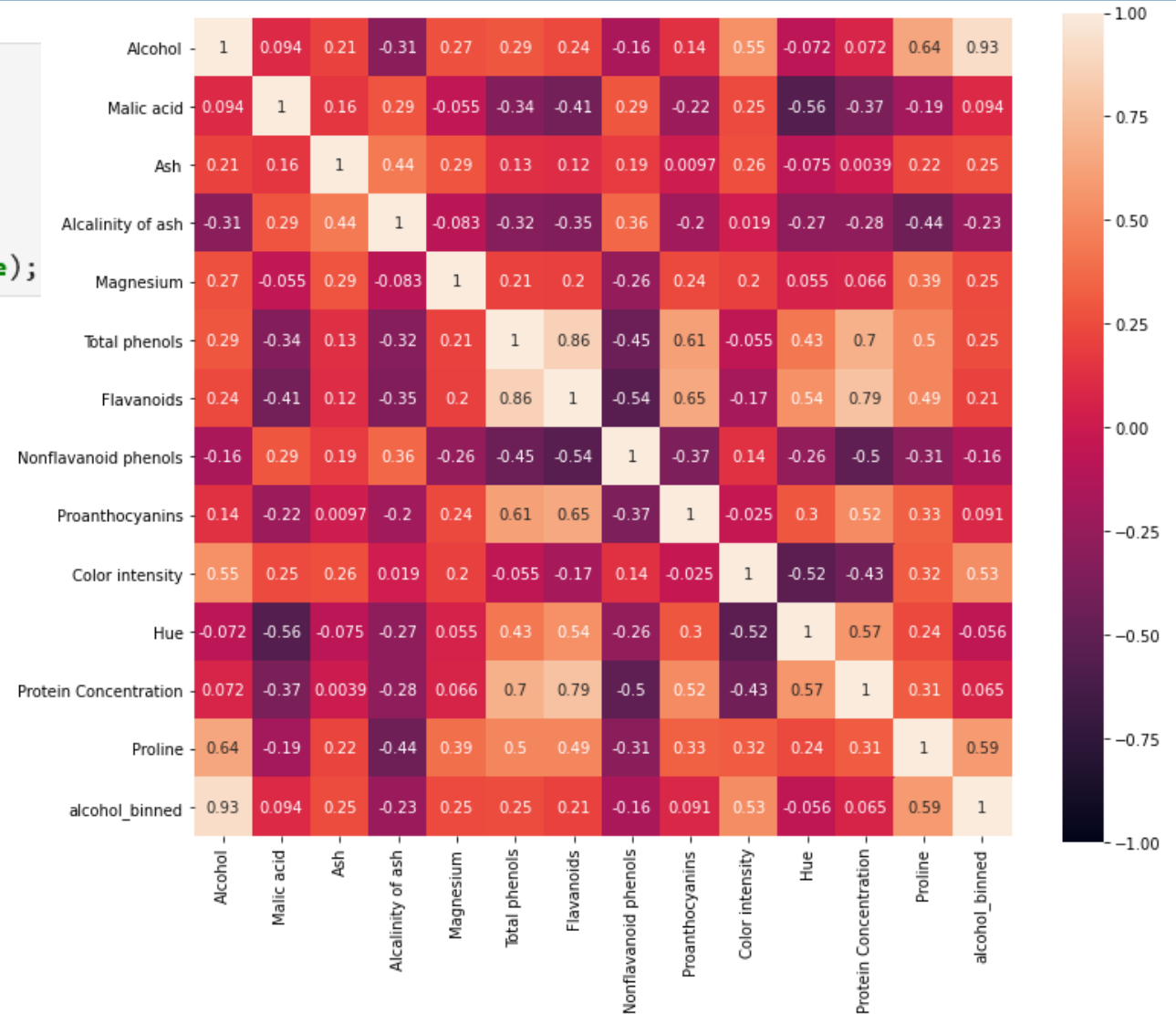
# Multivariate Analysis

31

```
'''
Correlation
'''
corr_matrix = df.corr()
f, ax = plt.subplots(figsize=(12, 10))
sns.heatmap(corr_matrix, vmin=-1, vmax=1, square=True, annot=True);
```

When the Class variable decreases (tendency to go to 0) the flavanoids, total phenols, proline and other proteins tend to increase. And viceversa.

There is a very strong correlation between alcohol and proline. High levels of alcohol correspond to high levels of proline.



# Critical Analysis

32

Which components characterize the various types of wine?

Which component is the most significant?

...



# Hands On

33

**SPYDER (Python 3.6)**

File Edit Search Source Run Debug Consoles Projects Tools View Help

Editor - C:\data\PythonWorkspace\dev\meanshift\_algorithm.py

```
37 class Mean_Shift:
38     def __init__(self, radius=None, radius_normalize_step = 150):
39         self.radius = radius
40         self.radius_normalize_step = radius_normalize_step
41
42     def fit(self, data):
43
44         if self.radius == None:
45             all_data_centroid = np.average(data, axis=0)
46             all_data_norm = np.linalg.norm(all_data_centroid)
47             self.radius = all_data_norm/self.radius_normalize_step
48
49         centroids = {}
50
51         #initialize centroids
52         for i in range(len(data)):
53             centroids[i] = data[i]
54
55         weights = [1 for i in range(self.radius_normalize_step)]
56
57         while True:
58             new_centroids = []
59             for i in centroids:
60                 in_range = []
61                 centroid = centroids[i]
62
63                 for featureset in data:
64                     distance = np.linalg.norm(featureset-centroid)
65                     if distance == 0:
66                         distance = 0.0000000001
67                     weight_index = int(distance/self.radius)
68                     if weight_index > self.radius_normalize_step-1:
69                         weight_index = self.radius_normalize_step-1
70                     to_add = (weights[weight_index]**2)*[featureset]
71                     in_range += to_add
72
73             new_centroid = np.average(in_range, axis=0)
```

**Variable explorer**

Name	Type	Size	Value
batch_size	int	1	100
mnist	contrib.learn.python.learn.datasets.base.Datasets	3	Datasets object of...
n_classes	int	1	10
n_nodes_h1	int	1	500
n_nodes_h2	int	1	500
n_nodes_h3	int	1	500

**IPython console**

See 'tf.nn.softmax\_cross\_entropy\_with\_logits\_v2'.

Epoch 0 completed out of 10 loss: 1666037.4677734375  
Epoch 1 completed out of 10 loss: 377809.3128890991  
Epoch 2 completed out of 10 loss: 201302.4857263565  
Epoch 3 completed out of 10 loss: 119427.91378033161  
Epoch 4 completed out of 10 loss: 72651.25679710507  
Epoch 5 completed out of 10 loss: 45327.621502393486  
Epoch 6 completed out of 10 loss: 31955.17812934518  
Epoch 7 completed out of 10 loss: 23664.35610633137  
Epoch 8 completed out of 10 loss: 18248.740643078025  
Epoch 9 completed out of 10 loss: 19962.00065876091  
Accuracy: 0.9511

In [2]:

IPython console History log

**HANDS ON**