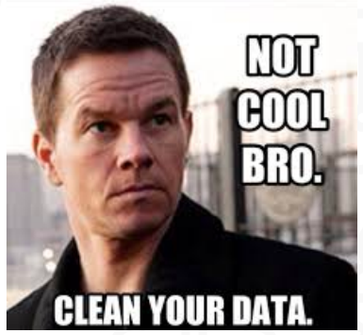
DataJam

BEGINNER

Data Cleaning/ Pandas

Concepts & Introduction

Context : Pandas



- Pandas → Manipulate data
- Transform inputs into outputs
- Data cleaning
- Data exploration

DataJam

Pandas 101 (load a .csv file)

In [234]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

- Please go to : https://datahub.io/machine-learning/iris#data
- Download the the iris dataset

In [8]:
```
store.datahub.io/machine-learning/iris/iris_csv/data/8bce8766530bf404228ea3fc026dfee3/iris_csv.csv -o ./iris_dataset.csv
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  4753  100  4753    0     0   4946      0 --:--:-- --:--:-- --:--:--  4945
```

In [9]:
```
!ls *.csv
```

```
iris_dataset.csv                reading.csv
my_healthcare_download_file.csv
```

In [260]:
```
# read a .csv file
my_dataframe = pd.read_csv("./iris_dataset.csv", sep=',')
my_dataframe.head(2)
```

Out[260]:

	sepallength	sepalwidth	petallength	petalwidth	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa

In [119]:
```
# create a random dataframe from a dictionary (simplest and most intuitive)
a_random_dataframe = pd.DataFrame({"value1": [2,3,4],"value2":["a","b","c"]})
a_random_dataframe.head()
```

Out[119]:

	value1	value2
0	2	a
1	3	b
2	4	c

DataJam

Pandas 101 (first lines)

In [13]:
```
# print the k first lines
k =10
my_dataframe.head(k)
```

Out[13]:

	sepallength	sepalwidth	petallength	petalwidth	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

DataJam

Pandas 101 (data types)

Change the data type

In [208]:
```
# let's change the data type of the dataframes columns
my_dataframe_copy["sepallength_int"] = my_dataframe_copy["sepallength"].astype("int")
my_dataframe_copy["sepalwidth_int"] = my_dataframe_copy["sepalwidth"].astype("int")
my_dataframe_copy["petallength_int"] = my_dataframe_copy["petallength"].astype("int")
my_dataframe_copy["petalwidth_int"] = my_dataframe_copy["petalwidth"].astype("int")
```

In [211]:
```
# display of selected columns
my_dataframe_copy[["sepallength_int","sepalwidth_int","petallength_int","petalwidth_int"]].head()
```

Out[211]:

	sepallength_int	sepalwidth_int	petallength_int	petalwidth_int
0	5	3	1	0
1	4	3	1	0
2	4	3	1	0
3	4	3	1	0
4	5	3	1	0

iloc style of referencing

In [89]:
```
# return the first row
my_dataframe_copy.iloc[[1],:]
```

Out[89]:

	sepallength	sepalwidth	petallength	petalwidth	class
1	4.9	3.0	1.4	0.2	Iris-setosa

DataJam

Pandas 101: Iloc vs loc

In [92]:
```
# return the first column
my_dataframe_copy.iloc[:,[1]]
```

Out[92]:

	sepalwidth
0	3.5
1	3.0
2	3.2
3	3.1
4	3.6
...	...
145	3.0
146	2.5
147	3.0
148	3.4
149	3.0

150 rows × 1 columns

DataJam

Pandas 101: Iloc vs loc

WARNING pandas can be a bit tricky at first : do not forget the double brackets for the columns selection

In [96]:
```
# return the sepal length column and petal length
my_dataframe_copy.loc[:]["sepallength","petallength"]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~/Downloads/Lambda/Lambda-env/lib/python3.7/site-packages/pandas/core/indexes/base.py in get_loc(self, key, method, t
olerance)
   2896             try:
-> 2897                 return self._engine.get_loc(key)
   2898             except KeyError:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: ('sepallength', 'petallength')

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-96-a6d769826315> in <module>
      1 # return the sepal length column and petal length
----> 2 my_dataframe_copy.loc[:]["sepallength","petallength"]

~/Downloads/Lambda/Lambda-env/lib/python3.7/site-packages/pandas/core/frame.py in __getitem__(self, key)
   2978             if self.columns.nlevels > 1:
   2979                 return self._getitem_multilevel(key)
-> 2980             indexer = self.columns.get_loc(key)
   2981             if is_integer(indexer):
   2982                 indexer = [indexer]

~/Downloads/Lambda/Lambda-env/lib/python3.7/site-packages/pandas/core/indexes/base.py in get_loc(self, key, method, t
olerance)
   2897                 return self._engine.get_loc(key)
   2898             except KeyError:
-> 2899                 return self._engine.get_loc(self._maybe_cast_indexer(key))
   2900         indexer = self.get_indexer([key], method=method, tolerance=tolerance)
```

DataJam

Pandas 101: Iloc vs loc

In [261]:
```
# return the sepal length column and petal length
my_dataframe_copy.loc[:,["sepallength","petallength"]]
```

Out[261]:

	sepallength	petallength
0	5.1	1.4
1	4.9	1.4
2	4.7	1.3
3	4.6	1.5
4	5.0	1.4
...	...	...
145	6.7	5.2
146	6.3	5.0
147	6.5	5.2
148	6.2	5.4
149	5.9	5.1

150 rows × 2 columns

DataJam

Pandas 101:
SELECT * WHERE Blabla

In [101]:
```
# filtering based on condition

# goold ole sql :)
# SUPPOSE the .csv file has been loaded into a TABLE called iris
# SELECT AVG(petallength) as AVG, petallength, sepallength, petalwidth, sepalwidth FROM iris
# WHERE petallength > AVG
```

In [102]:
```
# the condition on which the filtering is done is a list of True and False
my_dataframe_copy["petallength"]> my_dataframe_copy["petallength"].mean()
```

Out[102]:
```
0      False
1      False
2      False
3      False
4      False
       ...
145     True
146     True
147     True
148     True
149     True
Name: petallength, Length: 150, dtype: bool
```

In [103]:
```
conditional_filtering = my_dataframe[my_dataframe_copy["petallength"]> my_dataframe_copy["petallength"].mean()]
conditional_filtering.head(5)
```

Out[103]:

	sepallength	sepalwidth	petallength	petalwidth	class
50	7.0	3.2	4.7	1.4	Iris-versicolor
51	6.4	3.2	4.5	1.5	Iris-versicolor
52	6.9	3.1	4.9	1.5	Iris-versicolor
53	5.5	2.3	4.0	1.3	Iris-versicolor
54	6.5	2.8	4.6	1.5	Iris-versicolor

DataJam

Pandas 101: Text operations

In [104]:
```
# replace the hyphen/dash with and underscore
my_dataframe_copy["rewritten_class"] = my_dataframe_copy["class"].str.replace("-","_")
```

In [105]:
```
my_dataframe_copy.tail(5)
```

Out[105]:

	sepallength	sepalwidth	petallength	petalwidth	class	rewritten_class
145	6.7	3.0	5.2	2.3	Iris-virginica	Iris_virginica
146	6.3	2.5	5.0	1.9	Iris-virginica	Iris_virginica
147	6.5	3.0	5.2	2.0	Iris-virginica	Iris_virginica
148	6.2	3.4	5.4	2.3	Iris-virginica	Iris_virginica
149	5.9	3.0	5.1	1.8	Iris-virginica	Iris_virginica

In [106]:
```
# replace the hyphen/dash with and underscore
my_dataframe_copy["simple_class"] = my_dataframe_copy["class"].str.replace("Iris-","")
```

In [107]:
```
my_dataframe_copy.head(5)
```

Out[107]:

	sepallength	sepalwidth	petallength	petalwidth	class	rewritten_class	simple_class
0	5.1	3.5	1.4	0.2	Iris-setosa	Iris_setosa	setosa
1	4.9	3.0	1.4	0.2	Iris-setosa	Iris_setosa	setosa
2	4.7	3.2	1.3	0.2	Iris-setosa	Iris_setosa	setosa
3	4.6	3.1	1.5	0.2	Iris-setosa	Iris_setosa	setosa
4	5.0	3.6	1.4	0.2	Iris-setosa	Iris_setosa	setosa

DataJam

Pandas 101: built-in operations (mean)

In [113]:
```
# this will not work because among all of the columns only the 4 first columns are numeric!
numeric_cols = columns_list[:-1]
for col in numeric_cols:
    mean_values = [my_dataframe[col].mean() for col in numeric_cols]
```

In [114]:
```
mean_values
```

Out[114]:
```
[5.843333333333334, 3.0540000000000003, 3.758666666666666, 1.1986666666666668]
```

In [115]:
```
# dataframe have built-in operations (such as mean calculation)
# if axis=0 the mean will be calculated for each column (which is numeric)
my_dataframe.mean(axis=0)
```

Out[115]:
```
sepallength    5.843333
sepalwidth     3.054000
petallength    3.758667
petalwidth     1.198667
dtype: float64
```

In [116]:
```
# dataframe have built-in operations (such as mean calculation)
# if axis=0 the mean will be calculated for each row  (in our case it does not make any sense)
my_dataframe.mean(axis=1)
```

Out[116]:
```
0      2.550
1      2.375
2      2.350
3      2.350
4      2.550
       ...
145    4.300
146    3.925
147    4.175
148    4.325
149    3.950
Length: 150, dtype: float64
```

DataJam

Concatenate or Join

CONCATENATE

In [272]:
```
# concatenation columns side by side (columns)
# as u can see the indexes are not matching

concatenate_me = pd.concat([df_left,df_right],axis=1)
concatenate_me.head(5)
```

Out[272]:

	description	industry	level	index_left	size	line_code	value	index_right
0	At the end of the last financial year, which o...	total	0	5153	total	D1504	3672	5627
1	How has not receiving the full amount of debt ...	Motion picture	2	4651	total	D1705	0	654
2	For your most recent debt finance, how did you...	Non-metallic mineral product	2	4614	total	D2101	279	412
3	For your most recent request, who provided the...	total	0	2061	total	D0700.06	9	341
4	Over the last financial year, how have this bu...	Agriculture, forestry, & fishing support services	2	43	total	D1300.02	633	2329

JOIN

In [273]:
```
# JOIN 2 dataframes
# as u can see the indexes are matching !
join_me = df_left.merge(df_right, left_on= "index_left",right_on="index_right",how="inner")
join_me.head(5)
```

Out[273]:

	description	industry	level	index_left	size	line_code	value	index_right
0	At the end of the last financial year, which o...	total	0	5153	total	D0404.02	162	5153
1	How has not receiving the full amount of debt ...	Motion picture	2	4651	total	D0209	2871	4651
2	For your most recent debt finance, how did you...	Non-metallic mineral product	2	4614	total	D1808	0	4614
3	For your most recent request, who provided the...	total	0	2061	total	D1600.01	0	2061
4	Over the last financial year, how have this bu...	Agriculture, forestry, & fishing support services	2	43	total	D1108	6	43

DataJam

Lambda functions (1)

LAMBDA FUNCTION

oneliner

In [190]:
```
# precise the encoding to make sure to read the files
dataframe1 = pd.read_csv("./biz.csv",sep=',',encoding='latin1')
dataframe1.head()
```

Out[190]:

	description	industry	level	size	line_code	value
0	At the end of the last financial year, which o...	total	0	619 employees	D0201	14604
1	At the end of the last financial year, which o...	total	0	2049 employees	D0201	3792
2	At the end of the last financial year, which o...	total	0	5099 employees	D0201	1071
3	At the end of the last financial year, which o...	total	0	100+ employees	D0201	777
4	At the end of the last financial year, which o...	Agriculture, forestry, & fishing	1	total	D0201	2268

In [279]:
```
# one liner lambda function to get  the number of sentences
dataframe1["description_length"]= dataframe1["description"].apply(lambda x:len([ sentence for sentence in x.split(",")]
dataframe1.head()
```

Out[279]:

	description	industry	level	size	line_code	value	description_length
0	At the end of the last financial year, which o...	total	0	619 employees	D0201	14604	2
1	At the end of the last financial year, which o...	total	0	2049 employees	D0201	3792	2
2	At the end of the last financial year, which o...	total	0	5099 employees	D0201	1071	2
3	At the end of the last financial year, which o...	total	0	100+ employees	D0201	777	2
4	At the end of the last financial year, which o...	Agriculture, forestry, & fishing	1	total	D0201	2268	2

DataJam

Lambda functions (2)

UDF

In [287]:
```
# user defined function to get the words number in a description
def get_number_of_words(x):
    words_list = []
    sentences = x.split(",")
    for sentence in sentences:
        raw_list = sentence.split(" ")
        words_list+=raw_list
    return(len(words_list))
```

In [288]:
```
dataframe1["description_word_count"] = dataframe1["description"].apply(lambda x:get_number_of_words(x))
dataframe1.head()
```

Out[288]:

	description	industry	level	size	line_code	value	description_length	description_word_count
0	At the end of the last financial year, which o...	total	0	619 employees	D0201	14604	2	23
1	At the end of the last financial year, which o...	total	0	2049 employees	D0201	3792	2	23
2	At the end of the last financial year, which o...	total	0	5099 employees	D0201	1071	2	23
3	At the end of the last financial year, which o...	total	0	100+ employees	D0201	777	2	23
4	At the end of the last financial year, which o...	Agriculture, forestry, & fishing	1	total	D0201	2268	2	23

DataJam

Lambda functions (3)

In [289]:
```
# let's check that the udf really works and does not retrun the same value
dataframe1["description_word_count"].unique()
```

Out[289]:
```
array([23, 33, 29, 25, 28, 31, 20, 22, 21, 18, 19, 17, 26, 24, 14, 16, 13,
       12, 27, 39, 10, 15, 11])
```

In [295]:
```
# data vizualisation of the data (kinda looks normally distributed)
plt.title('Words length distribution')
my_words_frequency = dataframe1["description_word_count"].hist()
```