Ishika kesarwani **roll no. 92** **SAP ID 500076373**

Pattern and Anomaly Detection Lab 1

Using pandas and numpy

PANDAS - KAGGLE

(https://www.kaggle.com/learn/pandas)

Importing the required library

In [2]:

```
import pandas as pd
```

Reading the data

In [4]:

```
df=pd.read_csv("winemag-data_first150k.csv",index_col=0)
df.head()
```

Out[4]:

	country	description	designation	points	price	province	region_1	region_2	variety	winery
0	US	This tremendous 100% varietal wine hails from ...	Martha's Vineyard	96	235	California	Napa Valley	Napa	Cabernet Sauvignon	Heitz
1	Spain	Ripe aromas of fig, blackberry and cassis are ...	Carodorum Selección Especial Reserva	96	110	Northern Spain	Toro	NaN	Tinta de Toro	Bodega Carmen Rodríguez
2	US	Mac Watson honors the memory of a wine once ma...	Special Selected Late Harvest	96	90.0	California	Knights Valley	Sonoma	Sauvignon Blanc	Macauley
3	US	This spent 20 months in 30% new French oak, an...	Reserve	96	65.0	Oregon	Willamette Valley	Willamette Valley	Pinot Noir	Ponzi

4	France	This is the top wine from La Bégude, named aft...	La Brûlade	95	66.0	Provence	Bandol	NaN	Provence red blend	Domaine de la Bégude

In [5]:

```
df.shape
```

Out[5]:

```
(150930, 10)
```

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150930 entries, 0 to 150929
Data columns (total 10 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   country      150925 non-null  object 
 1   description  150930 non-null  object 
 2   designation  105195 non-null  object 
 3   points       150930 non-null  int64  
 4   price        137235 non-null  float64
 5   province     150925 non-null  object 
 6   region_1     125870 non-null  object 
 7   region_2     60953 non-null   object 
 8   variety      150930 non-null  object 
 9   winery       150930 non-null  object 
dtypes: float64(1), int64(1), object(8)
memory usage: 12.7+ MB
```

In [8]:

```
df.describe()
```

Out[8]:

	points	price
count	150930.000000	137235.000000
mean	87.888418	33.131482
std	3.222392	36.322536

min	80.000000	4.000000
25%	86.000000	16.000000
50%	88.000000	24.000000
75%	90.000000	40.000000
max	100.000000	2300.000000

Select the description column from df and assign the result to the variable desc.

In [11]:

```
desc = df.description
desc
```

Out[11]:

```
0         This tremendous 100% varietal wine hails from ...
1         Ripe aromas of fig, blackberry and cassis are ...
2         Mac Watson honors the memory of a wine once ma...
3         This spent 20 months in 30% new French oak, an...
4         This is the top wine from La Bégude, named aft...
                              ...
150925    Many people feel Fiano represents southern Ita...
150926    Offers an intriguing nose with ginger, lime an...
150927    This classic example comes from a cru vineyard...
150928    A perfect salmon shade, with scents of peaches...
150929    More Pinot Grigios should taste like this. A r...
Name: description, Length: 150930, dtype: object
```

Select the first value from the description column of df, assigning it to variable first_description.

In [12]:

```
first_description = df.description.iloc[0]
first_description
```

Out[12]:

```
'This tremendous 100% varietal wine hails from Oakville and was
aged over three years in oak. Juicy red-cherry fruit and a
compelling hint of caramel greet the palate, framed by elegant,
fine tannins and a subtle minty tone in the background. Balanced
and rewarding from start to finish, it has years ahead of it to
develop further nuance. Enjoy 2022—2030.'
```

Select the first row of data (the first record) from df, assigning it to the variable first_row.

In [13]:

```
first_row = df.iloc[0]
first_row
```

Out[13]:

```
country                                                     US
description    This tremendous 100% varietal wine hails from ...
designation                                  Martha's Vineyard
points                                                      96
price                                                    235.0
province                                            California
region_1                                           Napa Valley
region_2                                                  Napa
variety                                      Cabernet Sauvignon
winery                                                   Heitz
Name: 0, dtype: object
```

Select the first 10 values from the description column in df, assigning the result to variable first_descriptions.

Hint: format your output as a pandas Series.

In [14]:

```
first_descriptions = df.description.iloc[:10]
first_descriptions
```

Out[14]:

```
0    This tremendous 100% varietal wine hails from ...
1    Ripe aromas of fig, blackberry and cassis are ...
2    Mac Watson honors the memory of a wine once ma...
3    This spent 20 months in 30% new French oak, an...
4    This is the top wine from La Bégude, named aft...
5    Deep, dense and pure from the opening bell, th...
6    Slightly gritty black-fruit aromas include a s...
7    Lush cedary black-fruit aromas are luxe and of...
8    This re-named vineyard was formerly bottled as...
9    The producer sources from two blocks of the vi...
Name: description, dtype: object
```

Select the records with index labels 1, 2, 3, 5, and 8, assigning the result to the variable sample_reviews.

In other words, generate the following DataFrame:

In [16]:

```
indices = [1,2,3,5,8]
sample_reviews=df.iloc[indices]
```

```
sample_reviews
```

Out[16]:

	country	description	designation	points	price	province	region_1	region_2	variety	winery
1	Spain	Ripe aromas of fig, blackberry and cassis are ...	Carodorum Selección Especial Reserva	96	110	Northern Spain	Toro	NaN	Tinta de Toro	Bodega Carmen Rodríguez
2	US	Mac Watson honors the memory of a wine once ma...	Special Selected Late Harvest	96	90.0	California	Knights Valley	Sonoma	Sauvignon Blanc	Macauley
3	US	This spent 20 months in 30% new French oak, an...	Reserve	96	65.0	Oregon	Willamette Valley	Willamette Valley	Pinot Noir	Ponzi
5	Spain	Deep, dense and pure from the opening bell, th...	Numanthia	95	73.0	Northern Spain	Toro	NaN	Tinta de Toro	Numanthia
8	US	This re-named vineyard was formerly bottled as...	Silice	95	65.0	Oregon	Chehalem Mountains	Willamette Valley	Pinot Noir	Bergström

Create a variable dfs containing the country, province, region_1, and region_2 columns of the records with the index labels 0, 1, 10, and 100. In other words, generate the following DataFrame:

In [17]:

```
cols=['country', 'province', 'region_1', 'region_2']
indices=[0,1,10,100]
dfs=df.loc[indices,cols]
dfs
```

Out[17]:

	country	province	region_1	region_2
0	US	California	Napa Valley	Napa

1	Spain	Northern Spain	Toro	NaN
10	Italy	Northeastern Italy	Collio	NaN
100	US	California	South Coast	South Coast

Create a variable dfs containing the country and variety columns of the first 100 records.

Hint: you may use loc or iloc. When working on the answer this question and the several of the ones that follow, keep the following "gotcha" described in the tutorial:

iloc uses the Python stdlib indexing scheme, where the first element of the range is included and the last one excluded. loc, meanwhile, indexes inclusively.

This is particularly confusing when the DataFrame index is a simple numerical list, e.g. 0,...,1000. In this case df.iloc[0:1000] will return 1000 entries, while df.loc[0:1000] return 1001 of them! To get 1000 elements using loc, you will need to go one lower and ask for df.iloc[0:999].

In [19]:

```
cols = ['country', 'variety']
dfs = df.loc[:99, cols]
#or iloc[:100,cols]
dfs
```

Out[19]:

	country	variety
0	US	Cabernet Sauvignon
1	Spain	Tinta de Toro
2	US	Sauvignon Blanc
3	US	Pinot Noir
4	France	Provence red blend
...	...	...
95	France	Malbec-Merlot
96	US	Chardonnay
97	US	Cabernet Sauvignon
98	France	Merlot-Malbec

99	France	Ugni Blanc-Colombard

100 rows × 2 columns

Create a DataFrame italian_wines containing reviews of wines made in Italy.
Hint: reviews.country equals what?

In [21]:

```
italian_wines = df[df.country == 'Italy']
italian_wines.head()
```

Out[21]:

	country	description	designation	points	price	province	region_1	region_2	variety	winery
10	Italy	Elegance, complexity and structure come togeth...	Ronco della Chiesa	95	80.0	Northeastern Italy	Collio	NaN	Friulano	Borgo del Tiglio
32	Italy	Underbrush, scorched earth, menthol and plum s...	Vigna Piaggia	90	NaN	Tuscany	Brunello di Montalcino	NaN	Sangiovese	Abbadia Ardenga
35	Italy	Forest floor, tilled soil, mature berry and a ...	Riserva	90	135	Tuscany	Brunello di Montalcino	NaN	Sangiovese	Carillon
37	Italy	Aromas of forest floor, violet, red berry and ...	NaN	90	29.0	Tuscany	Vino Nobile di Montepulciano	NaN	Sangiovese	Avignonesi
38	Italy	This has a charming nose that boasts rose, vio...	NaN	90	23.0	Tuscany	Chianti Classico	NaN	Sangiovese	Casina di Cornia

Create a DataFrame top_oceania_wines containing all reviews with at least 95 points (out of 100) for wines from Australia or New Zealand.

In [24]:

```
top_oceania_wines = df.loc[
    (df.country.isin(['Australia', 'New Zealand']))
    & (df.points >= 95)
]
```

```
top_oceania_wines.head()
```

Out[24]:

	country	description	designation	points	price	province	region_1	region_2	variety	winery
214	Australia	Full-bodied and plush yet vibrant and imbued w...	The Factor	98	125	South Australia	Barossa Valley	NaN	Shiraz	Torbreck
245	Australia	This is a top example of the classic Australia...	The Peake	96	150	South Australia	McLaren Vale	NaN	Cabernet-Shiraz	Hickinbotham
303	Australia	This Cabernet equivalent to Grange has explode...	Bin 707	95	500	South Australia	South Australia	NaN	Cabernet Sauvignon	Penfolds
304	Australia	From vines planted in 1912, this has been an i...	Mount Edelstone Vineyard	95	200	South Australia	Eden Valley	NaN	Shiraz	Henschke
304	Australia	This is a throwback to those brash, flavor-exu...	One	95	95.0	South Australia	Langhorne Creek	NaN	Red Blend	Heartland

NUMPY

Numpy arryas are the main way we will use numpy

They come in two favors: Vectors & Matrices

Vectors are 1D and Matrix are 2D(can still have one row one col)

In [31]:

```
list=[1,2,3]
list
```

Out[31]:

```
[1, 2, 3]
```

In [32]:

```
import numpy as np
```

In [33]:

```
#converting a list to numpy arrays
arr=np.array(list)
arr
```

Out[33]:

```
array([1, 2, 3])
```

In [34]:

```
#making a 2D array
arr2=np.array([[1,2,3],[4,5,6]])
arr2
```

Out[34]:

```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [35]:

```
# Arange is one of the most useful function for quicly
generating an array
np.arange(0,10)
```

Out[35]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [36]:

```
np.arange(0,11,2)
```

Out[36]:

```
array([ 0,  2,  4,  6,  8, 10])
```

Creating Identity Matrix

2D square matrix, having ones in diagonal

In [37]:

```
#linspace is used to generate a linearly spaced array
```

```
np.linspace(0,10,5)
```

Out[37]:

```
array([ 0. ,  2.5,  5. ,  7.5, 10. ])
```

In [ ]:

In [38]:

```
np.random.rand(3,3)
```

Out[38]:

```
array([[0.0045401 , 0.65103743, 0.9836216 ],
       [0.32558264, 0.59378234, 0.0221181 ],
       [0.44835784, 0.47559874, 0.34880876]])
```

In [39]:

```
np.random.rand(3)
```

Out[39]:

```
array([0.078572  , 0.54046068, 0.53293922])
```

In [40]:

```
np.random.randint(1,100)
```

Out[40]:

```
6
```

In [41]:

```
np.random.randint(1,100,10)
```

Out[41]:

```
array([90, 16, 95, 80, 47, 40, 57, 23, 14, 12])
```

In [42]:

```
np.arange(0,10,2)
```

Out[42]:

```
array([0, 2, 4, 6, 8])
```

In [43]:

```
random_arr=np.random.randint(0,50,10)
```

```
random_arr
```

Out[43]:

```
array([30, 24, 38, 38, 22, 41, 48, 30, 38, 16])
```

In [44]:

```
random_arr.max()
```

Out[44]:

```
48
```

In [45]:

```
random_arr.min()
```

Out[45]:

```
16
```

In [27]:

```
random_arr.mean()
```

Out[27]:

```
25.9
```

In [48]:

```
np.random.randint(0,100,10)
```

Out[48]:

```
array([40, 47, 61, 34, 63, 90, 12, 70, 11, 73])
```

In [49]:

```
arr.dtype
```

Out[49]:

```
dtype('int64')
```

In [51]:

```
arr=np.arange(0,11)
arr
```

Out[51]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

In [52]:

```
np.sqrt(arr)
```

Out[52]:

```
array([0.        , 1.        , 1.41421356, 1.73205081, 2.
,
       2.23606798, 2.44948974, 2.64575131, 2.82842712, 3.
,
       3.16227766])
```

In [53]:

```
arr+arr
```

Out[53]:

```
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18, 20])
```