selection techniques

In [2]:
```
import numpy as np
sample_array=np.arange(1,20)
```

In [3]:
```
sample_array
```

Out[3]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19])
```

In [4]:
```
sample_array+sample_array
```

Out[4]:
```
array([ 2,  4,  6,  8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34,
       36, 38])
```

In [5]:
```
np.exp(sample_array) #exponential
```

Out[5]:
```
array([2.71828183e+00, 7.38905610e+00, 2.00855369e+01, 5.45981500e+01,
       1.48413159e+02, 4.03428793e+02, 1.09663316e+03, 2.98095799e+03,
       8.10308393e+03, 2.20264658e+04, 5.98741417e+04, 1.62754791e+05,
       4.42413392e+05, 1.20260428e+06, 3.26901737e+06, 8.88611052e+06,
       2.41549528e+07, 6.56599691e+07, 1.78482301e+08])
```

In [6]:
```
np.sqrt(sample_array) #square root
```

Out[6]:
```
array([1.        , 1.41421356, 1.73205081, 2.        , 2.23606798,
       2.44948974, 2.64575131, 2.82842712, 3.        , 3.16227766,
       3.31662479, 3.46410162, 3.60555128, 3.74165739, 3.87298335,
       4.        , 4.12310563, 4.24264069, 4.35889894])
```

In [7]:
```
np.log(sample_array)
```

Out[7]:
```
array([0.        , 0.69314718, 1.09861229, 1.38629436, 1.60943791,
       1.79175947, 1.94591015, 2.07944154, 2.19722458, 2.30258509,
       2.39789527, 2.48490665, 2.56494936, 2.63905733, 2.7080502 ,
       2.77258872, 2.83321334, 2.89037176, 2.94443898])
```

In [8]:
```
np.max(sample_array)
```

Out[8]:
```
19
```

In [9]:
```
np.min(sample_array)
```

Out[9]:
```
1
```

In [10]:
```
np.argmax(sample_array) #index of maxm
```

Out[10]:
```
18
```

In [11]:
```
np.argmin(sample_array)#index of minm
```

Out[11]:
```
0
```

In [12]:
```
np.square(sample_array)
```

```
Out[12]: array([  1,   4,   9,  16,  25,  36,  49,  64,  81, 100, 121, 144, 169,
                196, 225, 256, 289, 324, 361])
```

```
In [13]: np.log(sample_array)
```

```
Out[13]: 5.477225575051661
```

```
In [14]: np.mean(sample_array) #returns mean of array
```

```
Out[14]: 10.0
```

```
In [15]: np.var(sample_array)# variance
```

```
Out[15]: 30.0
```

```
In [16]: np.std(sample_array) #standard deviation
```

```
Out[16]: 5.477225575051661
```

```
In [17]: array=np.random.rand(3,4)
         array
```

```
Out[17]: array([[0.85699409, 0.41723811, 0.79782157, 0.49338392],
                [0.59429334, 0.10691167, 0.73680294, 0.98483325],
                [0.85416898, 0.23319265, 0.43440017, 0.98281745]])
```

```
In [19]: np.round(array,decimals=2)
```

```
Out[19]: array([[0.86, 0.42, 0.8 , 0.49],
                [0.59, 0.11, 0.74, 0.98],
                [0.85, 0.23, 0.43, 0.98]])
```

```
In [20]: sports=np.array(['golf','cricket','fball','cricket'])
         np.unique(sports)
```

```
Out[20]: array(['cricket', 'fball', 'golf'], dtype='<U7')
```

pandas

```
In [1]: import pandas as pd
        import numpy as np
```

```
In [3]: sports1=pd.Series([1,2,3,4],index=['cricket','football','basketball','golf'])#use capi
```

```
In [4]: sports1
```

```
Out[4]: cricket       1
        football      2
        basketball    3
        golf          4
        dtype: int64
```

```
In [6]: sports1['football']
```

```
Out[6]: 2
```

```
In [7]: sports1['golf']
```

```
Out[7]: 4
```

```
In [8]: sports2=pd.Series([11,2,3,4],index=['cricket','football','baseball','golf'])
        sports2
```

```
Out[8]: cricket     11
        football     2
        baseball     3
        golf         4
        dtype: int64
```

```
In [9]: sports1+sports2
```

```
Out[9]: baseball       NaN
        basketball     NaN
        cricket       12.0
        football       4.0
        golf           8.0
        dtype: float64
```

```
In [10]: df1=pd.DataFrame(np.random.rand(8,5),index='A B C D E F G H'.split(),columns='score1 s
         df1
```

Out[10]:

	score1	score2	score3	score4	score5
A	0.781753	0.522637	0.953453	0.731209	0.431534
B	0.897936	0.406701	0.518689	0.571039	0.780865
C	0.015094	0.854960	0.693746	0.932325	0.517336
D	0.939106	0.756608	0.926863	0.197073	0.695055
E	0.634229	0.599603	0.166709	0.826154	0.140410
F	0.722289	0.608178	0.170286	0.649481	0.895841
G	0.069579	0.160183	0.262033	0.561119	0.492496
H	0.924558	0.155573	0.063697	0.442866	0.505243

```
In [12]: df1["score1"]
```

```
Out[12]: A    0.781753
         B    0.897936
         C    0.015094
         D    0.939106
         E    0.634229
         F    0.722289
         G    0.069579
         H    0.924558
         Name: score1, dtype: float64
```

```
In [15]: df1[["score1","score2","score3"]]
```

Out[15]:

	score1	score2	score3
A	0.781753	0.522637	0.953453
B	0.897936	0.406701	0.518689
C	0.015094	0.854960	0.693746
D	0.939106	0.756608	0.926863
E	0.634229	0.599603	0.166709
F	0.722289	0.608178	0.170286
G	0.069579	0.160183	0.262033
H	0.924558	0.155573	0.063697

In [16]:

```
df1["score6"]=df1["score1"]+df1['score2']
df1
```

Out[16]:

	score1	score2	score3	score4	score5	score6
A	0.781753	0.522637	0.953453	0.731209	0.431534	1.304390
B	0.897936	0.406701	0.518689	0.571039	0.780865	1.304637
C	0.015094	0.854960	0.693746	0.932325	0.517336	0.870054
D	0.939106	0.756608	0.926863	0.197073	0.695055	1.695714
E	0.634229	0.599603	0.166709	0.826154	0.140410	1.233831
F	0.722289	0.608178	0.170286	0.649481	0.895841	1.330467
G	0.069579	0.160183	0.262033	0.561119	0.492496	0.229763
H	0.924558	0.155573	0.063697	0.442866	0.505243	1.080132

In [17]:

```
df2={'ID': ["101","102","103","107","176"],'Name':["john",'mercy','akash','kavin','la
df=pd.DataFrame(df2)
df
```

Out[17]:

	ID	Name	profit
0	101	john	20
1	102	mercy	54
2	103	akash	56
3	107	kavin	87
4	176	lally	123

In [18]:

```
df["ID"]
```

```
Out[18]: 0    101
         1    102
         2    103
         3    107
         4    176
         Name: ID, dtype: object
```

```
In [19]: df[["ID","Name","profit"]]
```

Out[19]:

	ID	Name	profit
0	101	john	20
1	102	mercy	54
2	103	akash	56
3	107	kavin	87
4	176	lally	123

```
In [21]: df=df.drop("ID",axis=1)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[21], line 1
----> 1 df=df.drop("ID",axis=1)

File ~\anaconda3\lib\site-packages\pandas\util\_decorators.py:331, in deprecate_nonke
yword_arguments.<locals>.decorate.<locals>.wrapper(*args, **kwargs)
    325 if len(args) > num_allow_args:
    326     warnings.warn(
    327         msg.format(arguments=_format_argument_list(allow_args)),
    328         FutureWarning,
    329         stacklevel=find_stack_level(),
    330     )
--> 331 return func(*args, **kwargs)

File ~\anaconda3\lib\site-packages\pandas\core\frame.py:5399, in DataFrame.drop(self,
labels, axis, index, columns, level, inplace, errors)
   5251 @deprecate_nonkeyword_arguments(version=None, allowed_args=["self", "label
s"])
   5252 def drop(  # type: ignore[override]
   5253     self,
   (...)
   5260     errors: IgnoreRaise = "raise",
   5261 ) -> DataFrame | None:
   5262     """
   5263     Drop specified labels from rows or columns.
   5264 
   (...)
   5397             weight  1.0     0.8
   5398     """
-> 5399     return super().drop(
   5400         labels=labels,
   5401         axis=axis,
   5402         index=index,
   5403         columns=columns,
   5404         level=level,
   5405         inplace=inplace,
   5406         errors=errors,
   5407     )

File ~\anaconda3\lib\site-packages\pandas\util\_decorators.py:331, in deprecate_nonke
yword_arguments.<locals>.decorate.<locals>.wrapper(*args, **kwargs)
    325 if len(args) > num_allow_args:
    326     warnings.warn(
    327         msg.format(arguments=_format_argument_list(allow_args)),
    328         FutureWarning,
    329         stacklevel=find_stack_level(),
    330     )
--> 331 return func(*args, **kwargs)

File ~\anaconda3\lib\site-packages\pandas\core\generic.py:4505, in NDFrame.drop(self,
labels, axis, index, columns, level, inplace, errors)
   4503 for axis, labels in axes.items():
   4504     if labels is not None:
-> 4505         obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4507 if inplace:
   4508     self._update_inplace(obj)

File ~\anaconda3\lib\site-packages\pandas\core\generic.py:4546, in NDFrame._drop_axis
(self, labels, axis, level, errors, only_slice)
```

```
   4544         new_axis = axis.drop(labels, level=level, errors=errors)
   4545     else:
-> 4546         new_axis = axis.drop(labels, errors=errors)
   4547     indexer = axis.get_indexer(new_axis)
   4549 # Case for non-unique axis
   4550 else:

File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:6934, in Index.drop(se
lf, labels, errors)
   6932 if mask.any():
   6933     if errors != "ignore":
-> 6934         raise KeyError(f"{list(labels[mask])} not found in axis")
   6935     indexer = indexer[~mask]
   6936 return self.delete(indexer)

KeyError: "['ID'] not found in axis"
```

In [22]:
```
df
```

Out[22]:

	Name	profit
0	john	20
1	mercy	54
2	akash	56
3	kavin	87
4	lally	123

In [23]:
```
df.drop(3)
```

Out[23]:

	Name	profit
0	john	20
1	mercy	54
2	akash	56
4	lally	123

In [ ]:
```

```