Essential Python for Data Analyst Part2

sunsun-datateathyme\

```
import numpy as np
import pandas as pd
```

```
# numpy: numerical python
friends = ["jay", "john", "jane"]
```

```
type(friends)
```

```
list
```

```
arr_friends = np.array(friends)
```

```
arr_friends[0]
```

```
np.str_('jay')
```

```
arr_friends[0:4]
```

```
array(['jay', 'john', 'jane'], dtype='<U4')
```

```
# exam scores
scores = [90, 80, 95, 100, 50]
```

```
arr_scores = np.array(scores)
```

```
arr_scores
```

```
array([ 90,  80,  95, 100,  50])
```

```
np.mean(arr_scores)
```

```
np.float64(83.0)
```

```
np.sum(arr_scores)
```

```
np.int64(415)
```

```
np.median(arr_scores)
```

```
np.float64(90.0)
```

```
np.std(arr_scores)
```

```
np.float64(17.776388834631177)
```

```
## array 2d,3d
arr_3d = np.array([[1,2,3], [4,5,6], [7,8,9]])
print(arr_3d)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

```
arr_3d[1][2]
```

```
np.int64(6)
```

```
arr_3d[0][0]
```

```
np.int64(1)
```

```
arr_3d[2][2]
```

```
np.int64(9)
```

```
# matrix multiplication (dot)
mat1 = np.array([[1,2], [3,4]])
mat2 = np.array([[3,3], [4,5]])
```

```
mat1
```

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

```
mat2
```

```
array([[3, 3],
       [4, 5]])
```

```
mat1.dot(mat2)
```

```
array([[11, 13],
       [25, 29]])
```

```
np.dot(mat1, mat2)
```

```
array([[11, 13],
       [25, 29]])
```

```
import pandas as pd
import numpy as np

# Create a sample DataFrame with 5 columns and 10 records
data = {
    'StudentID': np.arange(1, 11),
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Emily', 'Frank', 'Grace', 'Henry', 'Ivy', 'Jack'],
    'Age': np.random.randint(18, 22, size=10),
    'Gender': np.random.choice(['Male', 'Female'], size=10),
    'Grade': np.random.randint(70, 101, size=10)
}

student_df = pd.DataFrame(data)
print(student_df)
```

```
   StudentID     Name  Age  Gender  Grade
0          1    Alice   19  Female     77
1          2      Bob   20    Male     90
2          3  Charlie   21  Female     97
3          4    David   19  Female     97
4          5    Emily   18    Male     91
5          6    Frank   18    Male     72
6          7    Grace   20  Female     91
7          8    Henry   21  Female     74
8          9      Ivy   19    Male     93
9         10     Jack   18  Female     85
```

```
student_df["Age"].mean()
```

```
np.float64(19.3)
```

```
np.mean(student_df["Age"])
```

```
np.float64(19.3)
```

```
student_df.columns
```

```
Index(['StudentID', 'Name', 'Age', 'Gender', 'Grade'], dtype='object')
```

```
student_df[["StudentID", "Name"]].head()
```

	StudentID	Name
0	1	Alice
1	2	Bob
2	3	Charlie
3	4	David
4	5	Emily

```
# filter row
student_df.query("Age > 20")
```

	StudentID	Name	Age	Gender	Grade
2	3	Charlie	21	Female	97
7	8	Henry	21	Female	74

```
student_df.query("Gender == 'Female'")
```

	StudentID	Name	Age	Gender	Grade
0	1	Alice	19	Female	77
2	3	Charlie	21	Female	97
3	4	David	19	Female	97
6	7	Grace	20	Female	91
7	8	Henry	21	Female	74
9	10	Jack	18	Female	85

```
student_df.query("Gender == 'Female' & Grade > 90")[["Name", "Gender", "Age"]]
```

	Name	Gender	Age
2	Charlie	Female	21
3	David	Female	19
6	Grace	Female	20

sklearn => machine learning model

```
## sklearn => machine learning model
```

```
## Python ML AI > R
```

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

```
mtcars = pd.read_csv("https://gist.githubusercontent.com/seankross/a412dfbd88b3db70b74b/raw/5f23f993cd87c283ce766e7ac6b
```

```
mtcars.head()
```

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

```
## prepare data
X = mtcars.drop(["model", "mpg"], axis = 1) # axis1 == column
y = mtcars["mpg"]
```

```
X.head()
```

	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	8	360.0	175	3.15	3.440	17.02	0	0	3	2

```
y.head()
```

	mpg
0	21.0
1	21.0
2	22.8
3	21.4
4	18.7

dtype: float64

🔥 ML workflow

1. split data
2. train
3. score
4. evaluate

```
## ML workflow
# 1. split data
# 2. train
# 3. score
# 4. evaluate
```

```
# split data
# sed.seed(42)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size = 0.25, random_state = 42
)
```

🎯 LinearRegression

```
# train model
model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression()
```

```
# prediction
p = model.predict(X_test)
```

```
p
```

```
array([19.816545  , 10.98232893, 16.31616932, 27.16613904, 28.59706508,
       18.29855129, 14.85758111, 27.41057736])
```

```
# evaluate R-squared
model.score(X_test, y_test)
```

```
0.7856209608689562
```

```
model.score(X_train, y_train)
```

```
0.8667068951242609
```

DecisionTreeRegressor

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
```

```
mtcars = pd.read_csv("https://gist.githubusercontent.com/seankross/a412dfbd88b3db70b74b/raw/5f23f993cd87c283ce766e7ac6b
```

```
mtcars.head()
```

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

Next steps: Generate code with mtcars | View recommended plots | New interactive sheet

```
## prepare data
X = mtcars.drop(["model", "mpg"], axis = 1) # axis1 == column
y = mtcars["mpg"]
```

```
# split data
# sed.seed(42)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size = 0.25, random_state = 42
)
```

```
# train model
model = DecisionTreeRegressor()
model.fit(X_train, y_train)
```

```
DecisionTreeRegressor
DecisionTreeRegressor()
```

```
# prediction
p = model.predict(X_test)
```

```
p
```

```
array([17.8, 10.4, 18.7, 33.9, 24.4, 17.8, 15.8, 33.9])
```

```
# evaluate R-squared
model.score(X_test, y_test)
```

```
0.833302930803843
```

```
model.score(X_train, y_train)
```

```
1.0
```

KNeighborsRegressor

```
from sklearn.neighbors import KNeighborsRegressor
```

```
# train model
model = KNeighborsRegressor()
model.fit(X_train, y_train)
```

```
▾ KNeighborsRegressor
KNeighborsRegressor()
```

```
# prediction
p = model.predict(X_test)
```

```
p
```

```
array([22.32, 14.28, 14.72, 28.7 , 22.18, 20.54, 15.42, 28.7 ])
```

```
# evaluate R-squared
model.score(X_test, y_test)
```

```
0.8175982001702541
```

```
model.score(X_train, y_train)
```

```
0.7906748511415767
```