Pandas

Dr. Ram Prasad K
VisionCog R&D

ram.krish@visioncog.com
https://www.visioncog.com

Pandas Fundamentals

Pandas

Easy to use data analysis Python library

Contains three types of **data structures**:

- Series
- DataFrames
- Panels (Deprecated in latest versions)

Pandas Fundamentals

Series:

1-dimensional labeled array capable of holding any data type.

DataFrame:

2-dimensional labeled data structure with columns of potentially different types.

Panel:

3-dimensional labeled data structure.

Less commonly used, so it is replaced by multi-index extension of DataFrame.

Pandas Fundamentals

Series

```
import numpy as np
import pandas as pd

# Create Series
s = pd.Series([2, -1, 3, 5])

print(s)

# 0    2
# 1   -1
# 2    3
# 3    5
# dtype: int64
```

```
s1 = np.exp(s)
print(s1)

# 0      7.389056
# 1      0.367879
# 2     20.085537
# 3    148.413159
# dtype: float64
```

Pandas Fundamentals

Series

```
s_idx = pd.Series([12, 15, 3, 60],
                  index=["a", 2, "c", "d"])

print(s_idx)

# a    12
# 2    15
# c     3
# d    60
# dtype: int64

print(s_idx.loc["c"])
# 3

# There is also a default int index
print(s_idx.iloc[3])
# 60
```

```
print(s_idx["c"])
# 3

print(s_idx[3])
# 60
```

- Not safe to access the series like this.
- Always use 'loc' or 'iloc'

Pandas Fundamentals

Series

```
s2 = pd.Series([1000, 1001, 1002, 1003, 1004])

print(s2)

# 0    1000
# 1    1001
# 2    1002
# 3    1003
# 4    1004
# dtype: int64
```

```
s2_slice = s2[2:]

print(s2_slice)

# 2    1002
# 3    1003
# 4    1004
# dtype: int64
```

Pandas Fundamentals

Series

```
s2_slice = s2[2:]

print(s2_slice)

# 2    1002
# 3    1003
# 4    1004
# dtype: int64
```

```
print(s2_slice[0])

# ---------------------------------------
# KeyError
# Traceback (most recent call last)
# <ipython-input-14-e4a7d3f28305> in <module>
# ----> 1 s2_slice[0]
```

```
print(s2_slice.iloc[0])

# 1002
```

Pandas Fundamentals

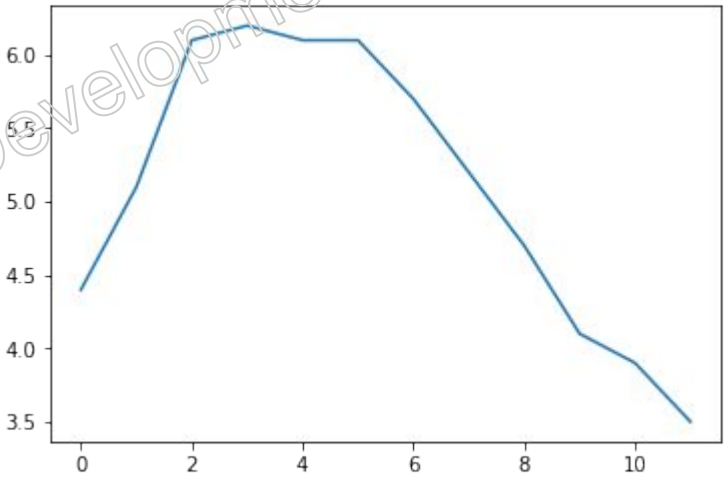
Series

```
%matplotlib inline
import matplotlib.pyplot as plt

temperatures = [4.4,5.1,6.1,6.2,6.1,6.1,
                5.7,5.2,4.7,4.1,3.9,3.5]

temp = pd.Series(temperatures,
                 name="Temperature")

temp.plot()
plt.show()
```

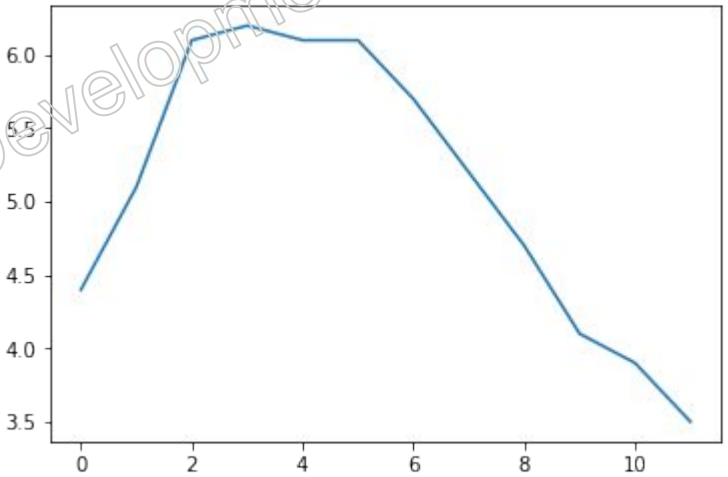


Pandas Fundamentals

Series

```
print(temp)
# Series name is also displayed

# 0     4.4
# 1     5.1
# 2     6.1
# 3     6.2
# 4     6.1
# 5     6.1
# 6     5.7
# 7     5.2
# 8     4.7
# 9     4.1
# 10    3.9
# 11    3.5
# Name: Temperature, dtype: float64
```



Pandas Fundamentals

DataFrames

```
people_dict = {
    "weight": pd.Series([68, 83, 112],
                        index=["alice", "bob", "charles"]),

    "birthyear": pd.Series([1984, 1985, 1992],
                           index=["bob", "alice", "charles"],
                           name="year"),

    "children": pd.Series([0, 3],
                          index=["charles", "bob"]),

    "hobby": pd.Series(["Biking", "Dancing"],
                       index=["alice", "bob"]),
}
```

```
# Converting dict to DataFrame
people = pd.DataFrame(people_dict)
print(people)
```

	weight	birthyear	children	hobby
alice	68	1985	NaN	Biking
bob	83	1984	3.0	Dancing
charles	112	1992	0.0	NaN

A few things to note:

1. The Series were automatically aligned based on their index

2. Missing values are represented as NaN

3. Series names are ignored (the name "year" was dropped),

Pandas Fundamentals

DataFrames

	weight	birthyear	children	hobby
alice	68	1985	NaN	Biking
bob	83	1984	3.0	Dancing
charles	112	1992	0.0	NaN

```
print(people["birthyear"])
```

```
alice      1985
bob        1984
charles    1992
Name: birthyear, dtype: int64
```

```
print(people[["birthyear", "hobby"]])
```

	birthyear	hobby
alice	1985	Biking
bob	1984	Dancing
charles	1992	NaN

Pandas Fundamentals

DataFrames

	weight	birthyear	children	hobby
alice	68	1985	NaN	Biking
bob	83	1984	3.0	Dancing
charles	112	1992	0.0	NaN

```
print(people.loc["charles"])
```

```
weight        112
birthyear    1992
children        0
hobby         NaN
Name: charles, dtype: object
```

```
print(people.iloc[2])
```

```
weight        112
birthyear    1992
children        0
hobby         NaN
Name: charles, dtype: object
```

Pandas Fundamentals

DataFrames

	weight	birthyear	children	hobby
alice	68	1985	NaN	Biking
bob	83	1984	3.0	Dancing
charles	112	1992	0.0	NaN

```
print(people.iloc[1:3])
```

	weight	birthyear	children	hobby
bob	83	1984	3.0	Dancing
charles	112	1992	0.0	NaN

```
print(people.iloc[1:3, 2:])
```

	children	hobby
bob	3.0	Dancing
charles	0.0	NaN

Pandas Fundamentals

DataFrames

	weight	birthyear	children	hobby
alice	68	1985	NaN	Biking
bob	83	1984	3.0	Dancing
charles	112	1992	0.0	NaN

```
people["age"] = 2019 - people["birthyear"]

print(people)
```

	weight	birthyear	children	hobby	age
alice	68	1985	NaN	Biking	34
bob	83	1984	3.0	Dancing	35
charles	112	1992	0.0	NaN	27

Handling CSV Files

Pandas Fundamentals

```
# Download csv file from ISL website
# http://www-bcf.usc.edu/~gareth/ISL/data.html

advData = pd.read_csv("data/Advertising.csv")

print(advData.head())
```

	Unnamed: 0	TV	radio	newspaper	sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

Pandas Fundamentals

```
print(advData.tail())
```

	Unnamed: 0	TV	radio	newspaper	sales
195	196	38.2	3.7	13.8	7.6
196	197	94.2	4.9	8.1	9.7
197	198	177.0	9.3	6.4	12.8
198	199	283.6	42.0	66.2	25.5
199	200	232.1	8.6	8.7	13.4

Pandas Fundamentals

```
print(advData.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
Unnamed: 0    200 non-null int64
TV            200 non-null float64
radio         200 non-null float64
newspaper     200 non-null float64
sales         200 non-null float64
dtypes: float64(4), int64(1)
memory usage: 7.9 KB
None
```

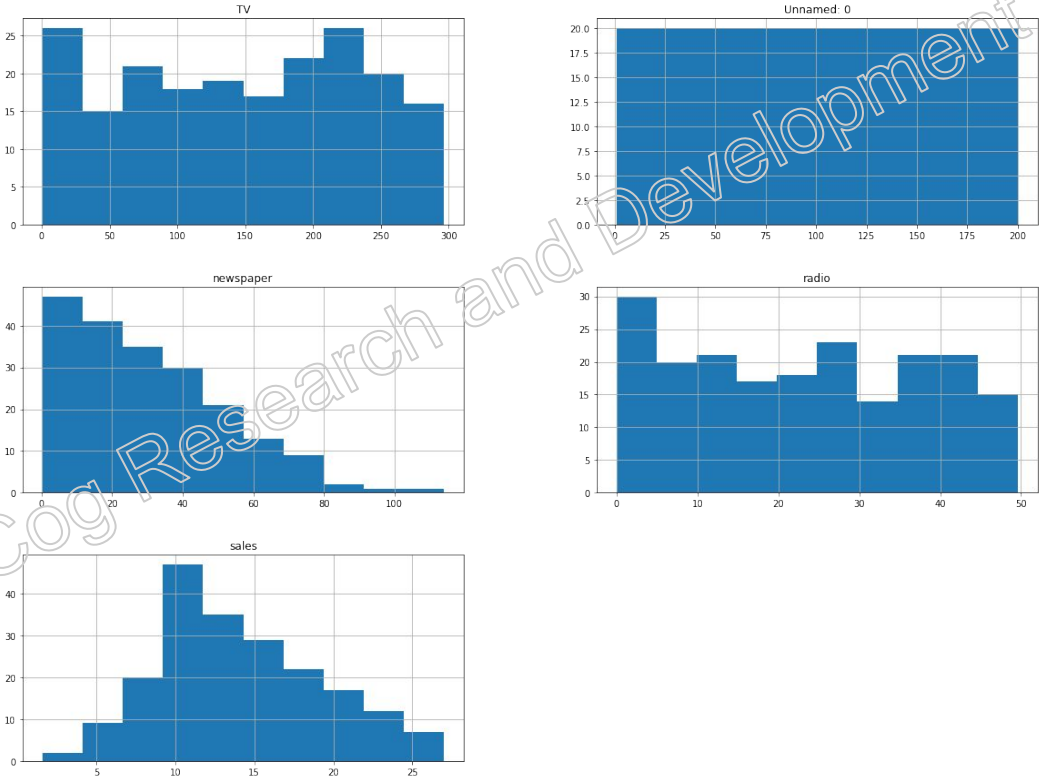
Pandas Fundamentals

```
print(advData.describe())
```

	Unnamed: 0	TV	radio	newspaper	sales
count	200.000000	200.000000	200.000000	200.000000	200.000000
mean	100.500000	147.042500	23.264000	30.554000	14.022500
std	57.879185	85.854236	14.846809	21.778621	5.217457
min	1.000000	0.700000	0.000000	0.300000	1.600000
25%	50.750000	74.375000	9.975000	12.750000	10.375000
50%	100.500000	149.750000	22.900000	25.750000	12.900000
75%	150.250000	218.825000	36.525000	45.100000	17.400000
max	200.000000	296.400000	49.600000	114.000000	27.000000

Pandas Fundamentals

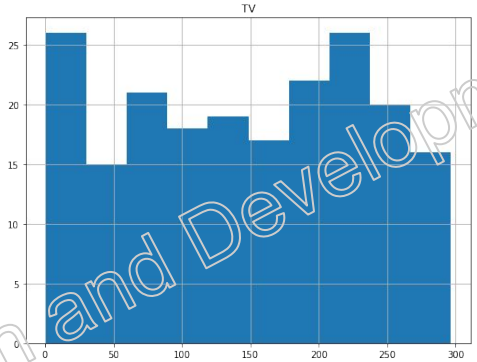
```
advData.hist(bins=10, figsize=(20,15))
```

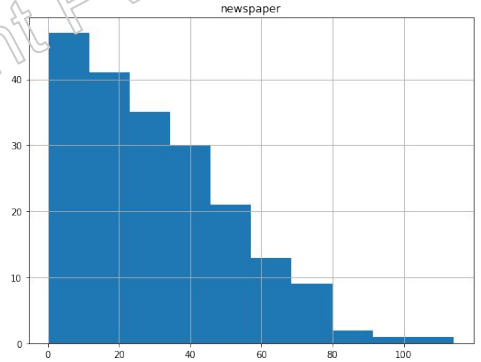


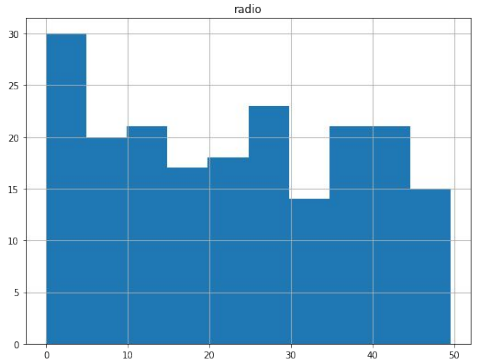
Pandas Fundamentals

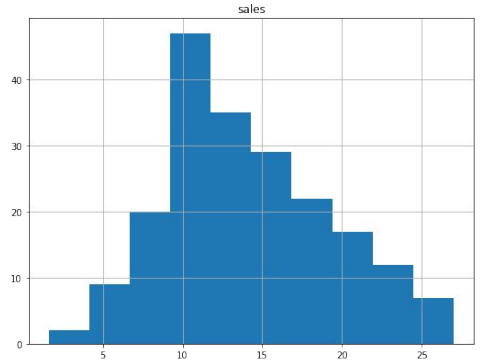
```
del advData["Unnamed: 0"]

advData.hist(bins=10, figsize=(20,15))
```









Pandas Fundamentals

```
X = advData.drop("sales", axis=1)

print(X.head())
```

	TV	radio	newspaper
0	230.1	37.8	69.2
1	44.5	39.3	45.1
2	17.2	45.9	69.3
3	151.5	41.3	58.5
4	180.8	10.8	58.4

Pandas Fundamentals

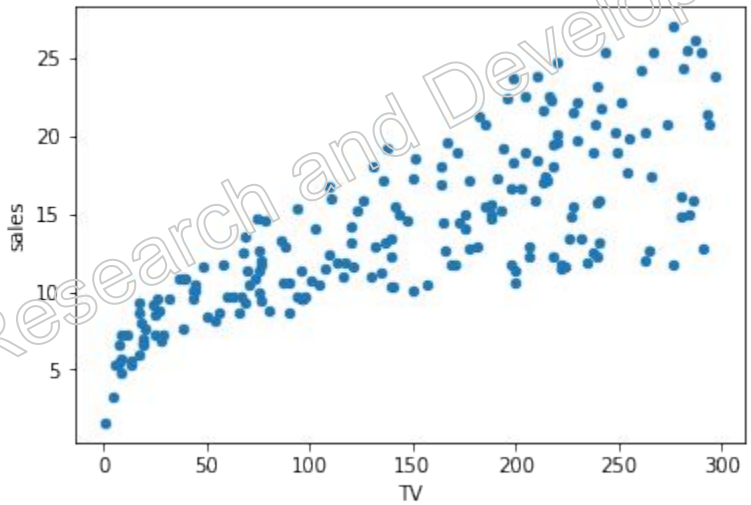
```
y = advData["sales"]

print(y.head())
```

```
0    22.1
1    10.4
2     9.3
3    18.5
4    12.9
Name: sales, dtype: float64
```

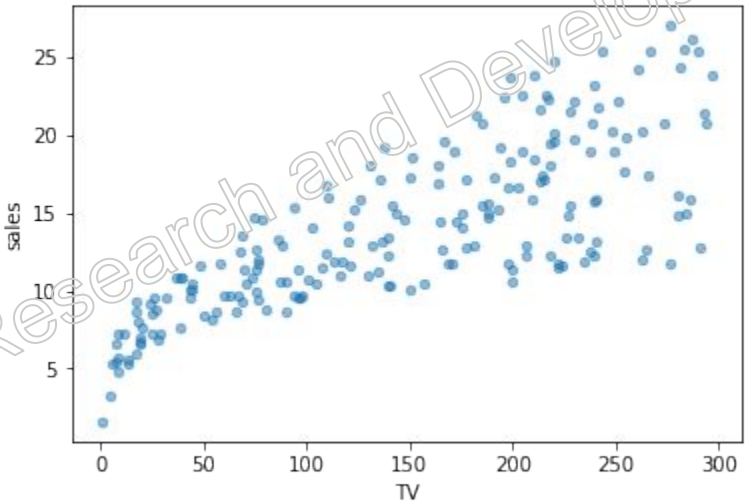
Pandas Fundamentals

```
advData.plot(kind="scatter", x = 'TV', y = 'sales')
```



Pandas Fundamentals

```
advData.plot(kind="scatter", x = "TV", y = "sales", alpha=0.5)
```



Pandas Fundamentals

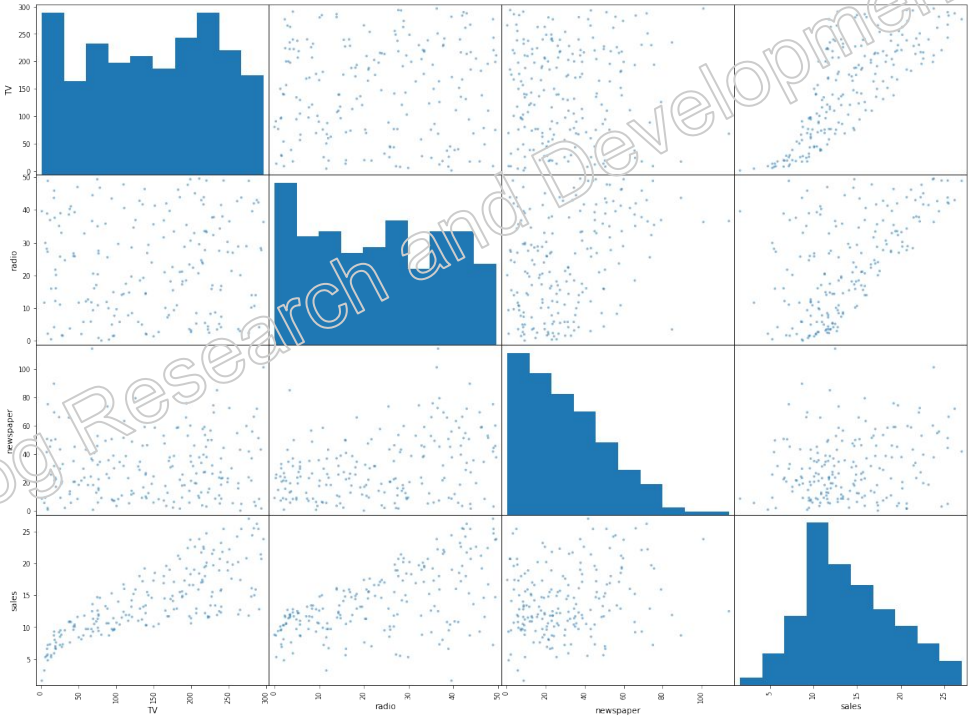
```
correlationMatrix = advData.corr()

print(correlationMatrix)
```

	TV	radio	newspaper	sales
TV	1.000000	0.054809	0.056648	0.782224
radio	0.054809	1.000000	0.354104	0.576223
newspaper	0.056648	0.354104	1.000000	0.228299
sales	0.782224	0.576223	0.228299	1.000000

Pandas Fundamentals

```
from pandas.tools.plotting import scatter_matrix
scatter_matrix(advData, figsize=(20,15))
```



Pandas Fundamentals

Exercise

Open “data/wdbc.csv” and analyze various attributes.

(https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic))

Note:

```
df = pd.read_csv('data/wdbc.csv', header=None)
df.head()
```