teleuniv
innovative interactive immersive

Linear Regression

Inferential Statistics

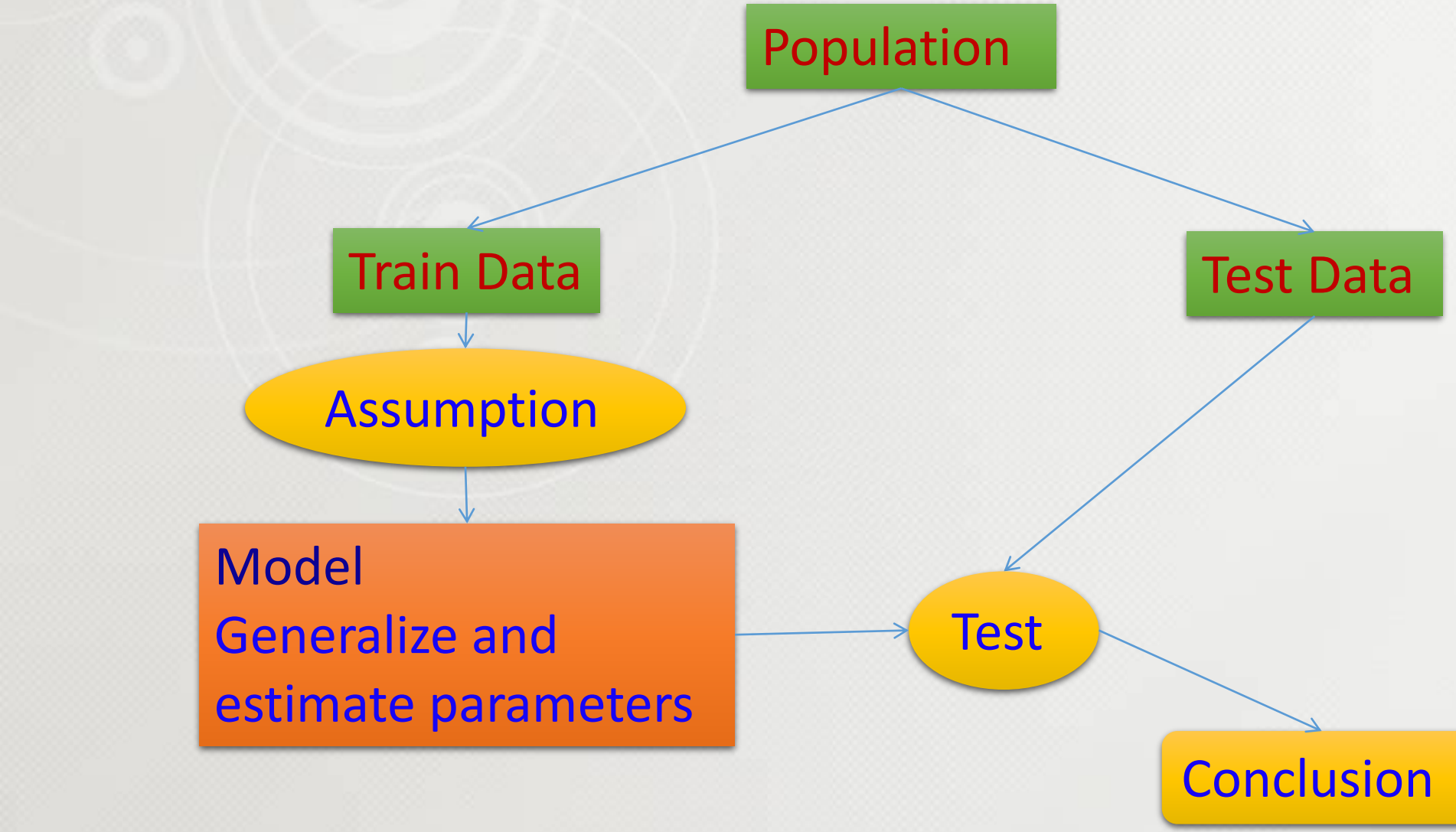
Inferential statistics use a random sample of data taken from a population to describe and make inferences about the population.

Population : Any group of data, which includes all the data interested in,

Sample: A smaller set of data, which are used to represent the larger population

The methods of inferential statistics
(1) the estimation of parameter(s)
(2) testing of statistical hypotheses

Inferential Statistics



Statistical Learning

Supervised

Input Variable

Predictors /
Independent Variables /
Features

Output Variable

Response /
Dependent Variables

Fit a model that relates to response to the predictors, for predicting the response for future observations.

Linear Regression

Logistic Regression

Etc…

Unsupervised

Predictors

No response variable to supervise so is called unsupervised learning.

Cluster Analysis…

Analyzing Data

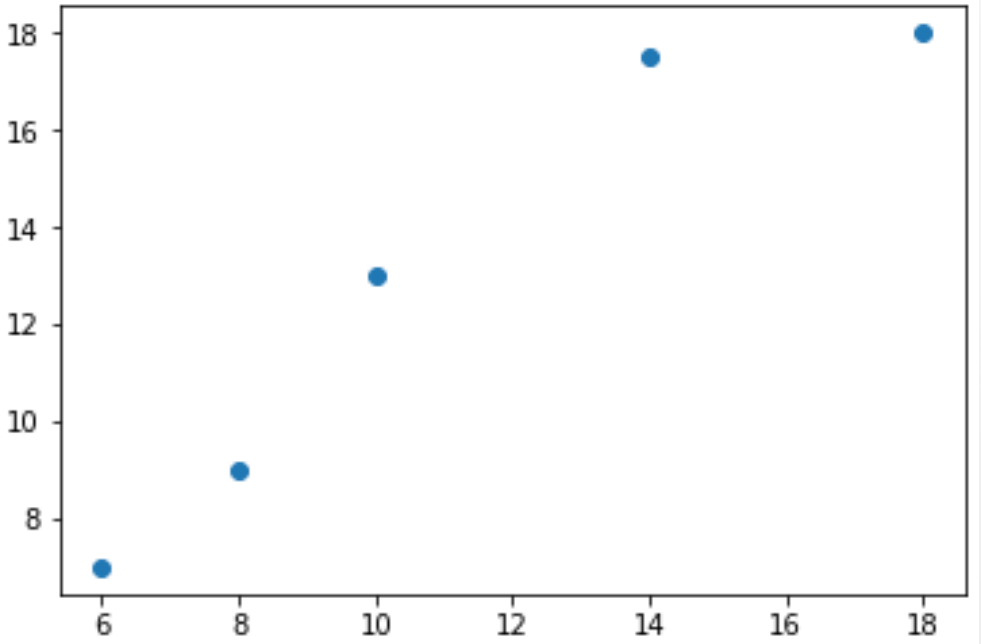
Lets take 2 arrays with 5 values each

x : Pizza size

y : Cost in Dollars

```
import matplotlib.pyplot as plt
import numpy as np
x = np.array([6,8,10,14,18])
print(x)
y = np.array([7,9,13,17.5,18])
print(y)

plt.scatter(x,y)
```

```
[ 6  8 10 14 18]
[  7.    9.   13.   17.5  18. ]
Out[1]: <matplotlib.collections.PathCollection at 0x
```



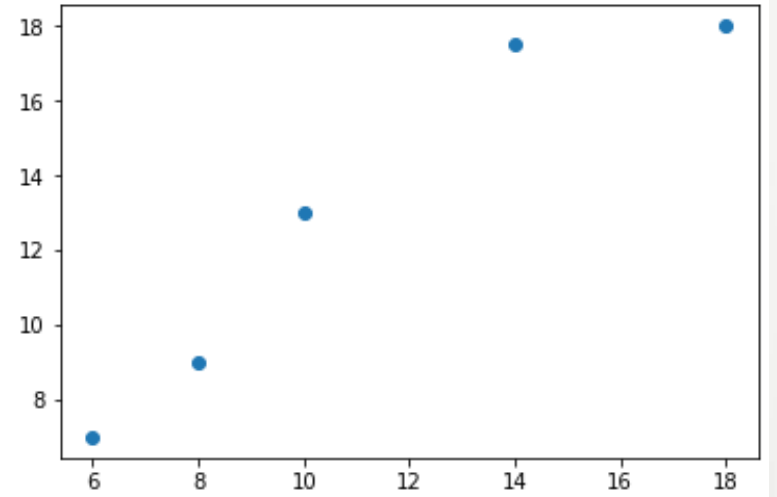
Analyzing Data

Observe the various statistics

Mean, variance

```
xm=x.mean()
ym=y.mean()
print("mean of x : ",xm,"\nmean of y :",ym)

xv=np.var(x)
print("Variance of X:",xv)

xyc = np.cov(x,y)
print("Covariance of x and y\n",xyc)
print("Vovariance of x and y : ", xyc[0,1])
```

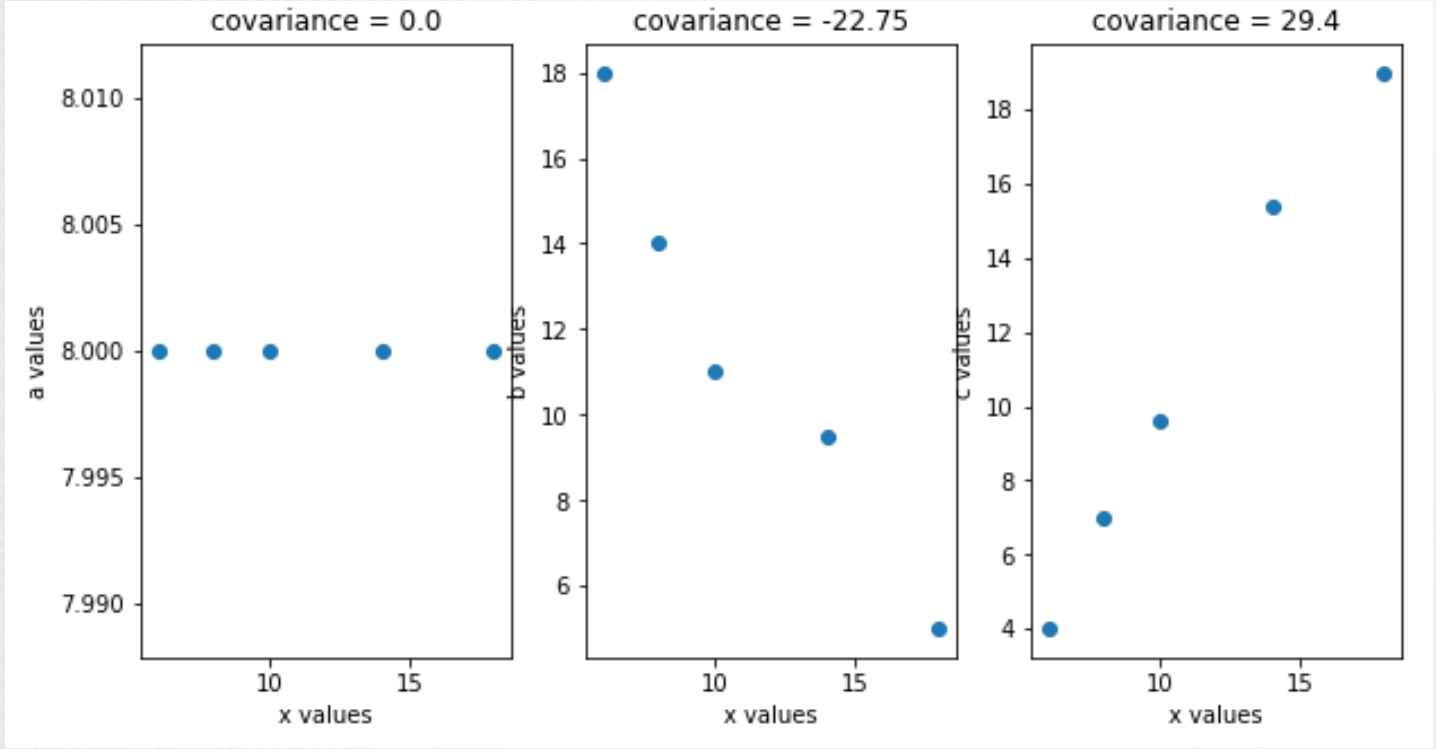
```
[ 6  8 10 14 18]
[  7.    9.   13.   17.5  18. ]
Out[1]: <matplotlib.collections.PathCollection at 0x
```




```
mean of x :  11.2
mean of y : 12.9
Variance of X: 18.56
Covariance of x and y
 [[ 23.2   22.65]
 [ 22.65  24.3 ]]
Vovariance of x and y :  22.65
```

Analyzing Data

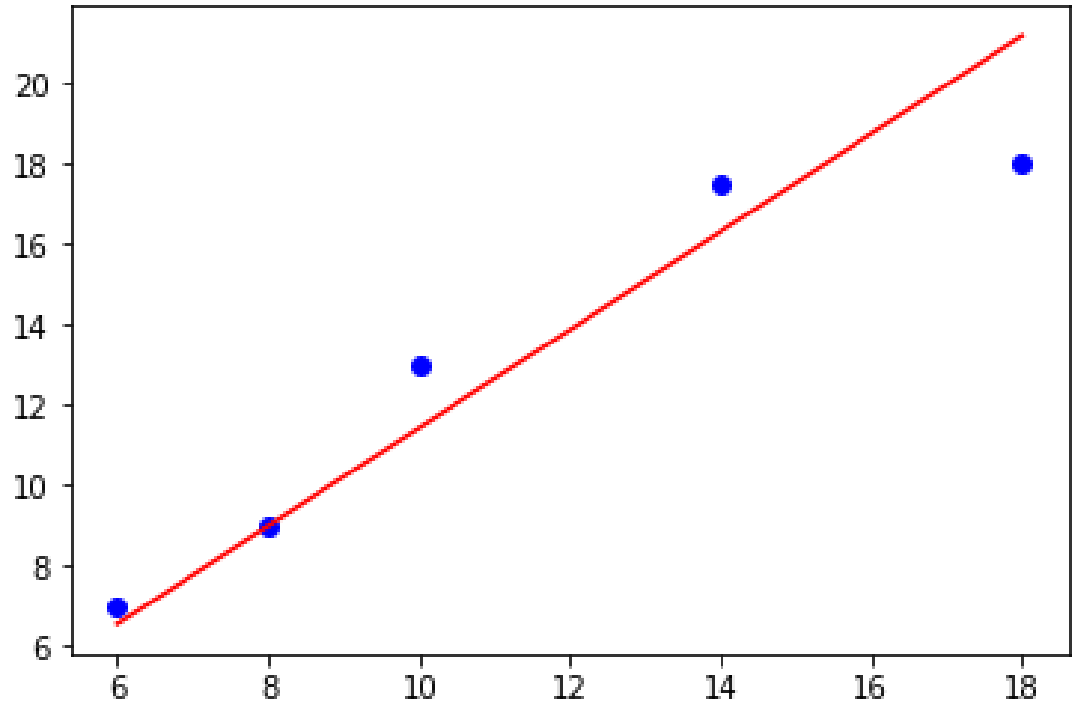
```
import matplotlib.pyplot as plt
import numpy as np
x = np.array([6,8,10,14,18])
print(x)
a = np.array([8,8,8,8,8])
b = np.array([18,14,11,9.5,5])
c = np.array([4,7,9.6,15.4,19])

#print(y)
plt.figure(figsize=(10,5))
plt.subplot(131)
plt.scatter(x,a)
plt.xlabel("x values")
plt.ylabel("a values")
plt.title("covariance = "+str(np.cov(x,a)[0][1]))

plt.subplot(132)
plt.scatter(x,b)
plt.xlabel("x values")
plt.ylabel("b values")
plt.title("covariance = "+str(np.cov(x,b)[0][1]))

plt.subplot(133)
plt.scatter(x,c)
plt.xlabel("x values")
plt.ylabel("c values")
plt.title("covariance = "+str(np.cov(x,c)[0][1]))
```



Analyzing Data

```
import matplotlib.pyplot as plt
import numpy as np
x = np.array([6,8,10,14,18])
y = np.array([7,9,13,17.5,18])

xyc = np.cov(x,y)
xv=np.var(x)

beta1=xyc[0,1]/xv
beta0=y.mean()-(beta1*x.mean())

print(beta0,beta1)

y_pred=beta0+(beta1*x)
plt.scatter(x,y,color='b')
plt.plot(x,y_pred,'r')

p12=beta0+(beta1*12)
print(p12)
```

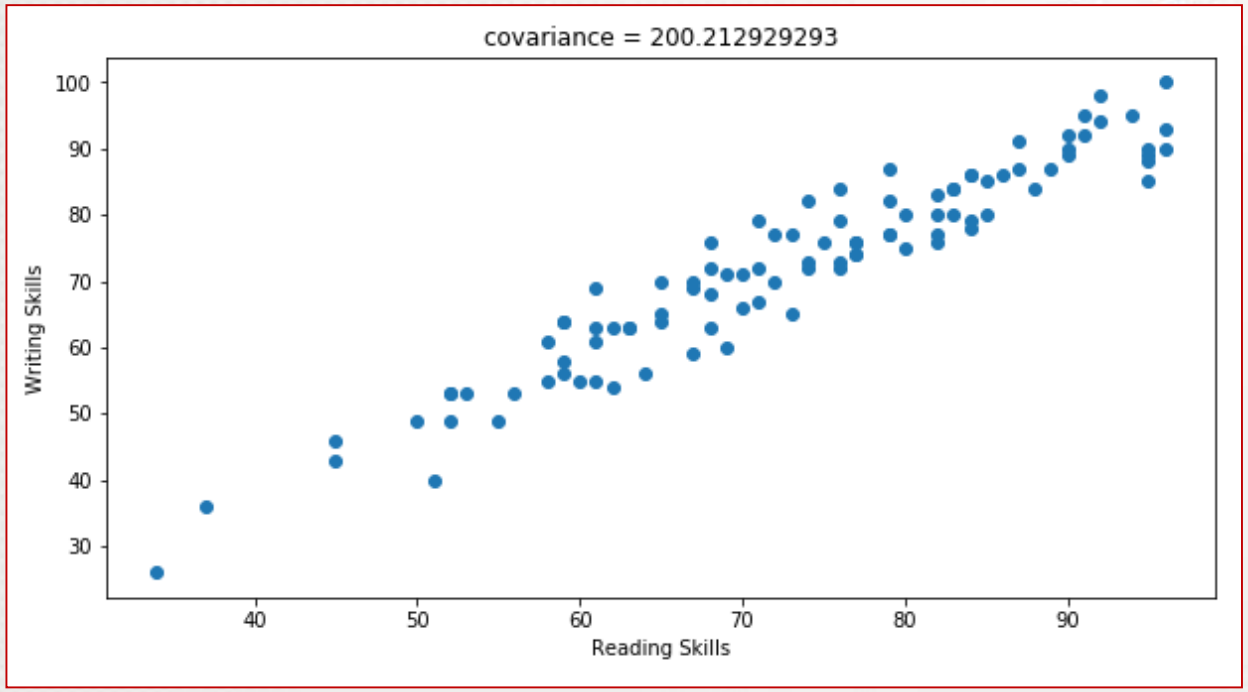
```
-0.768103448276 1.22036637931
Out[17]: [<matplotlib.lines.Line2D at 0x1c67041a2e8>]
```




```
In [18]: p12=beta0+(beta1*12)
    ...: print(p12)
13.8762931034
```

Analyzing Data

```
import matplotlib.pyplot as plt
import numpy as np
r = np.array([61,77,68,69,75,67,79,60,87,67,82,70,71,71,90
print(x)
w = np.array([69,74,68,71,76,69,77,55,91,59,80,66,79,72,90

#print(y)
plt.figure(figsize=(10,5))
#plt.subplot(131)
plt.scatter(r,w)
plt.xlabel("Reading Skills")
plt.ylabel("Writing Skills")
plt.title("covariance = "+str(np.cov(r,w)[0][1]))
```

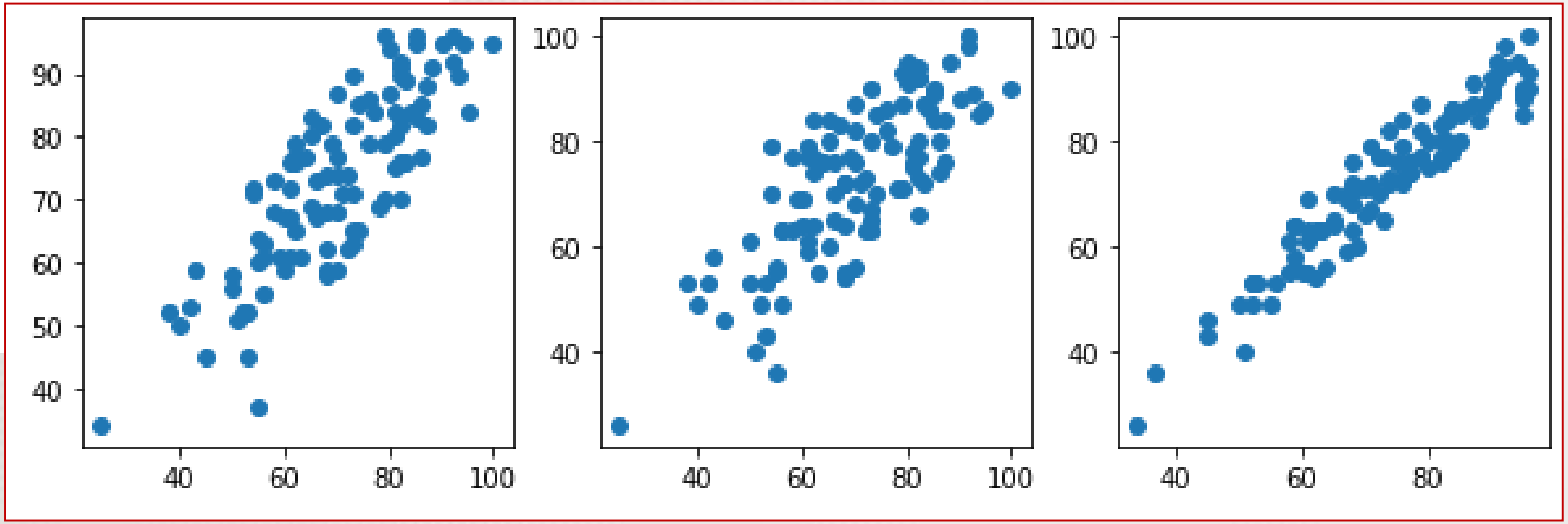



```
rv=r.var()
wv=w.var()
print("Reading Variance",rv,"Writing Variance",wv)
rwc = np.cov(r,w)
print("Covariance between Read and Writing\n",rwc)
print(rwc[0][1])
cor=np.correlate(r,w)

print("Correlation between Read and Write",cor)
```

```
Reading Variance 196.8784 Writing Variance 218.0771
Covariance between Read and Writing
 [[ 198.86707071  200.21292929]
 [ 200.21292929  220.27989899]]
200.212929293
Correlation between Read and Write [547389]
```

Analyzing Data

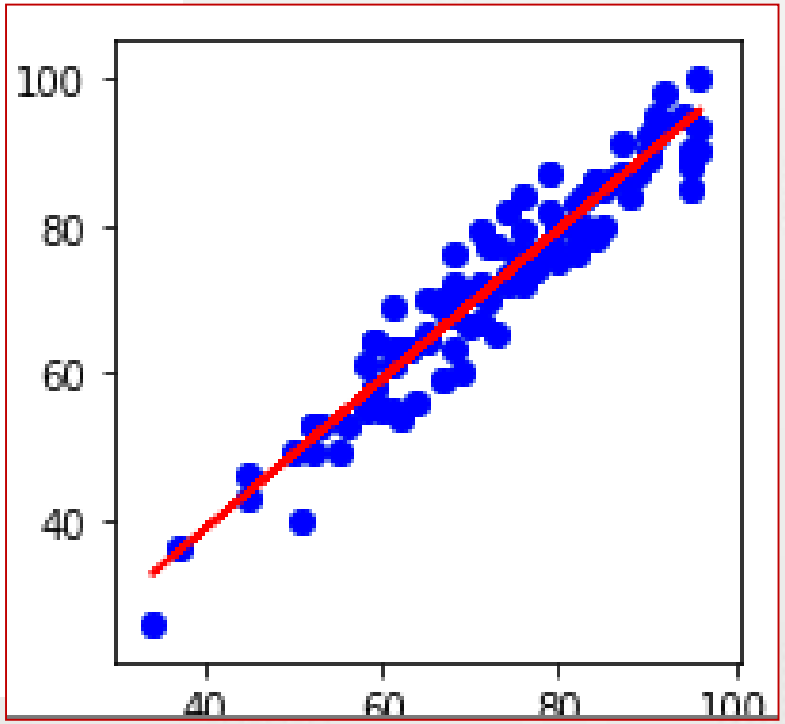
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data=pd.read_csv("C://Users/kmit/Desktop/exams.csv")
#print(data)
print(data.columns)
m=data['math score']

r=data['reading score']
w=data['writing score']

plt.figure(figsize=[10,3])
plt.subplot(131)
plt.scatter(m,r)
plt.subplot(132)
plt.scatter(m,w)
plt.subplot(133)
plt.scatter(r,w)
```



Analyzing Data

```
rv= np.var(r)
wv=np.var(w)
mv=np.var(m)
print('variance for read...', rv)
print('variance for write...',wv)
print('variance for maths...',mv)

crw= np.cov(r,w)
cmr=np.cov(m,r)
cmw=np.cov(m,w)
print('covariance for read & write...\n', crw)
print('covariance for math & read...\n',cmr)
print('covariance for math & write...\n',cmw)

ccrw= np.corrcoef(r,w)
ccmr=np.corrcoef(m,r)
ccmw=np.corrcoef(m,w)
print('correlation for read & write...\n', ccrw)
print('correlation for math & read...\n',ccmr)
print('correlation for math & write...\n',ccmw)

#line   y=f(w)=alpha+beta.r

beta=(crw/rv)[0][1]
alpha=w.mean()-(beta*r.mean())
print(beta, alpha)

wp=alpha+(beta*r)

plt.scatter(r,w,color='b')
plt.plot(r,wp,'r-')
```

```
variance for read... 196.8784
variance for write... 218.07710000000003
variance for maths... 200.41999999999993
covariance for read & write...
 [[ 198.86707071  200.21292929]
 [ 200.21292929  220.27989899]]
covariance for math & read...
 [[ 202.44444444  165.42020202]
 [ 165.42020202  198.86707071]]
covariance for math & write...
 [[ 202.44444444  163.91111111]
 [ 163.91111111  220.27989899]]
correlation for read & write...
 [[ 1.          0.9565843]
 [ 0.9565843  1.       ]]
correlation for math & read...
 [[ 1.          0.82443074]
 [ 0.82443074  1.       ]]
correlation for math & write...
 [[ 1.          0.77618997]
 [ 0.77618997  1.       ]]
1.01693699915 -2.04707841772
```

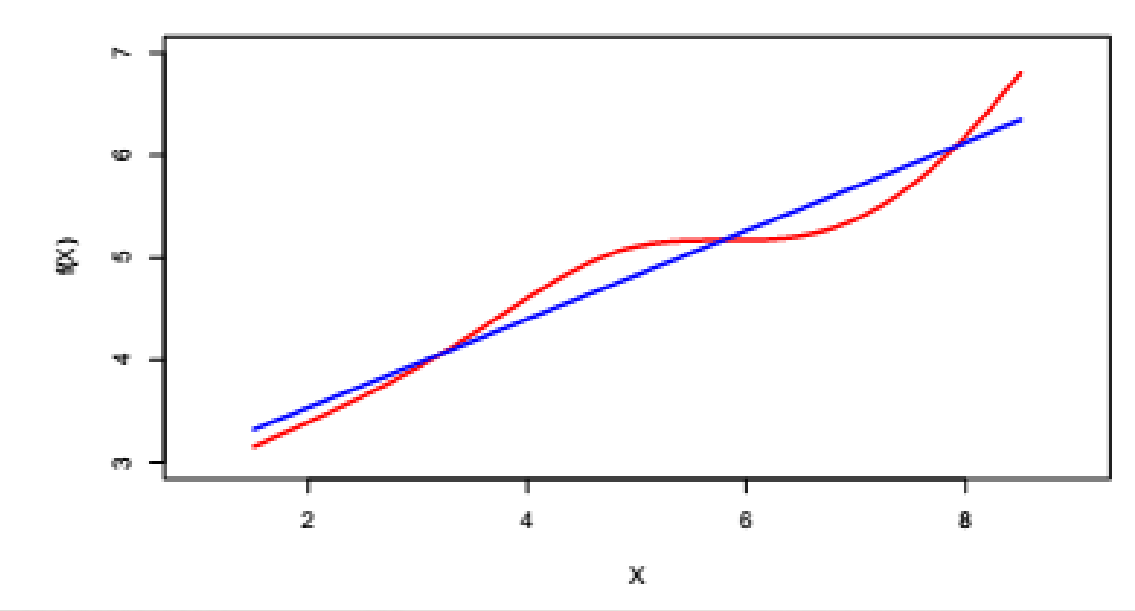


Linear Regression

Tool for predicting an unknown value based on existing data.

Linear Regression is Supervised Learning

Predictors X1,X2,X3,… Response Y=f(x)

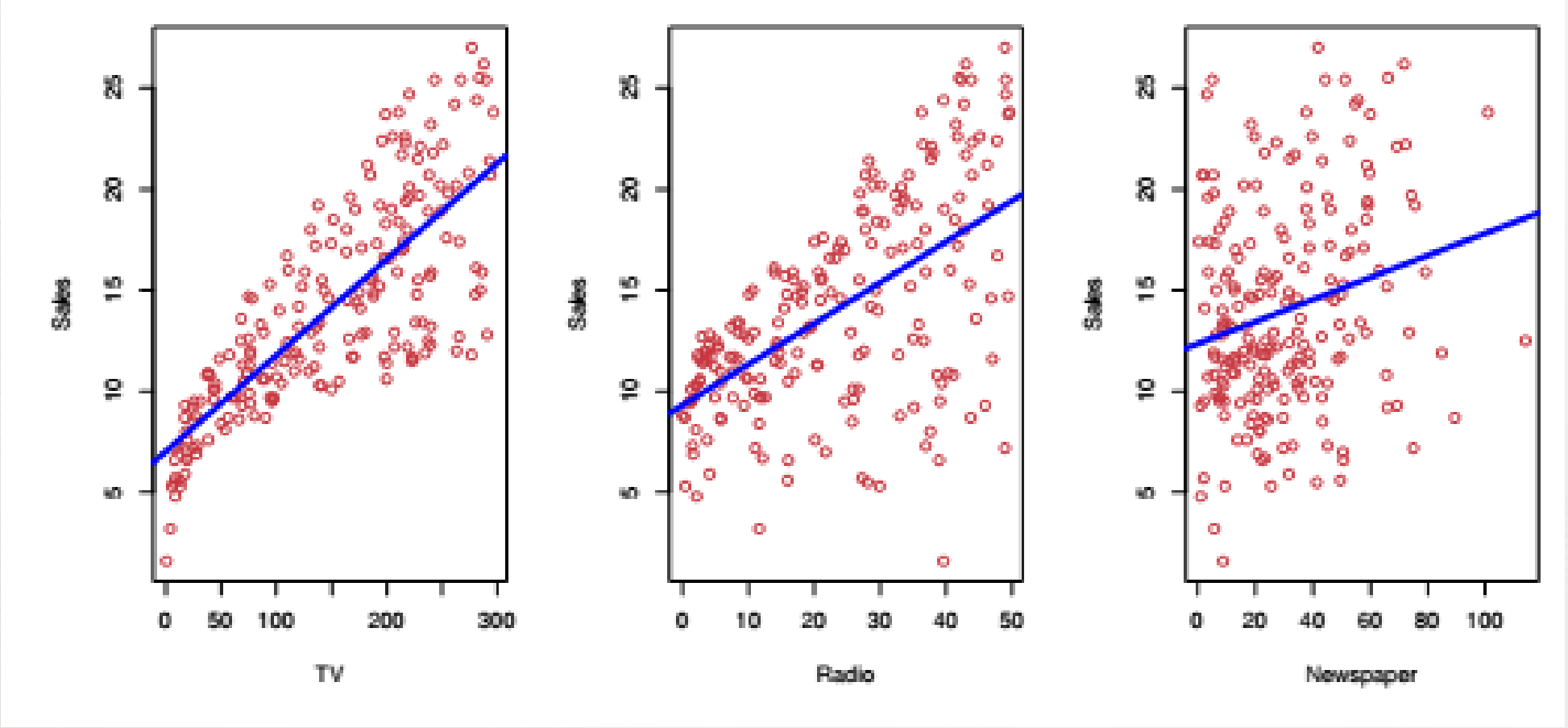


Linear Regression

Actual Regression

Linear Regression

In Advertising data,
Predictors: Budget for TV, Radio, News Paper
Response: Sales



Linear Regression

Linear regression answers…

- Is there a relationship between advertising budget and sales?
- How strong is the relationship between advertising budget and sales?
- Which media contribute to sales?
- How accurately can we predict future sales?
- Is the relationship linear?
- Is there synergy among the advertising media?

Simple Linear Regression

Predicting a quantitative response Y on the basis of single predictor variable X.

Assumption: there is a linear relationship between x and y

Model:

$$Y = \beta_0 + \beta_1 X + \epsilon$$

Sales

intercept

slope

Coefficients / parameters

TV

Error term

Estimate: $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$

^ represents estimated term

teleuniv
innovative interactive immersive

Train the model

Machine Learning

Linear Regression
Model-fit

In [62]:
```
from sklearn import linear_model
regr=linear_model.LinearRegression()

regr.fit(train_set_x, train_set_y)
```

Out[62]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

Train the model

Linear Regression

Prediction:

```
In [113]: house_y_predict=regr.predict(test_set_y)
          house_y_predict

Out[113]: array([[ 1.98],
                 [ 1.9 ],
                 [ 0.75],
                 ...,
                 [ 0.73],
                 [ 1.26],
                 [ 1.45]])
```

teleuniv
innovative interactive immersive

Fine-tune the model.

Machine Learning

Linear Regression evaluating

In [114]:
```
from sklearn.metrics import mean_squared_error, r2_score
# np.set_printoptions(precision=2)
np.set_printoptions(suppress=True)
print(test_set_y.dtype,house_y_predict.dtype)
print("Actual Data..",test_set_y)
print("Predicted Values..",house_y_predict)
print(mean_squared_error(test_set_y, house_y_predict))
```

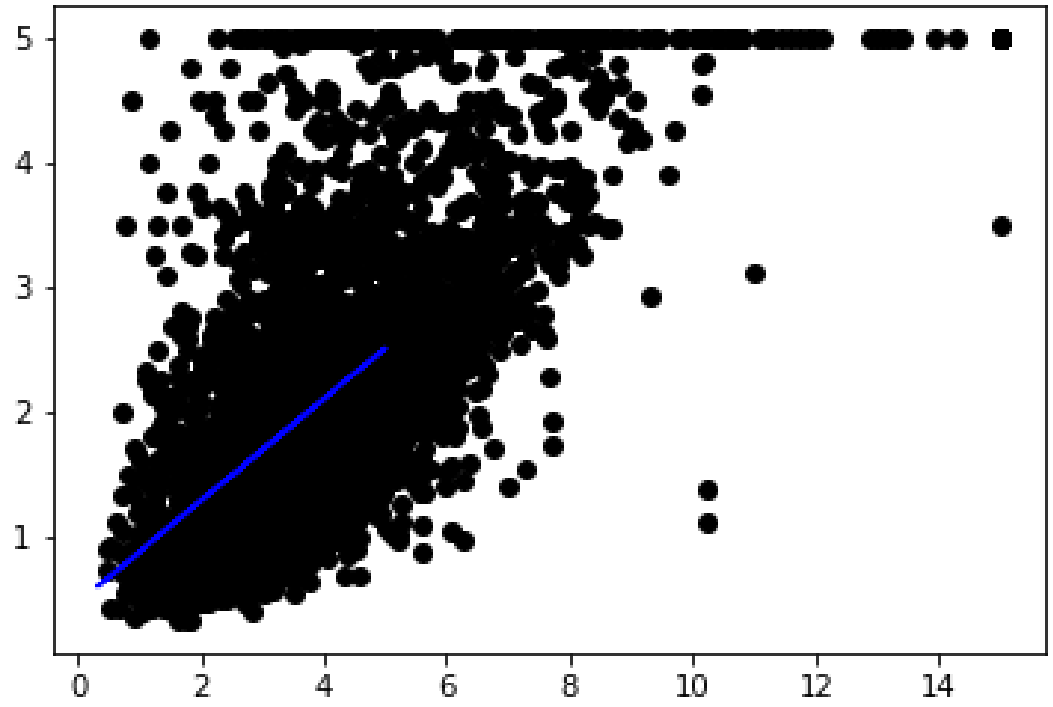
```
float64 float64
Actual Data.. [[ 3.71]
 [ 3.5 ]
 [ 0.69]
 ...,
 [ 0.64]
 [ 1.94]
 [ 2.4 ]]
Predicted Values.. [[ 1.98]
 [ 1.9 ]
 [ 0.75]
 ...,
 [ 0.73]
 [ 1.26]
 [ 1.45]]
1.05221512487
```

Fine-tune the model.

Linear Regression

Visualize the model

In [115]:
```
plt.scatter(test_set_x, test_set_y, color='black')
plt.plot(test_set_y,house_y_predict, color='blue')

# plt.xticks(())
# plt.yticks(())
plt.show()
```



Conclusion

Discussed about ...

- Files – Reading – Writing

Next Session .....

Data Pre-Processing


THANK
YOU