Machine learning

Machine learning / MachinelearningIntroduction

It is the scientific study of algorithms and statistical models that computer systems use to perform a specific. Its algorithms build a mathematical model based on sample / training data in order to make predictions / decisions without being explicitly programmed to perform the task.

Data Set

It a data set is any collection of data from list / set / array / file / data base e.t.c . It can be anything from an array to a complete database.

Data Types

It is an attribute of data which tells the compiler / interpreter how programmer intends to use the data.

Data Type	**Details**
Numerical	These are numbers.They are of 2 types. Discrete Data -These are numbers that are limited to integers. Continuous Data -These are numbers that are of infinite value.
Categorical	It represents characteristics (persons gender, marital status, e.t.c) and can take on numerical values also but don’t have mathematical meaning.
Ordinal	It with the property that its values can be counted.

Machine learning Using Numpy / Python Related Installations

Numpy, scipy and matplotlib Installations

1. Install python latest version.
2. Type “pip install numpy” in the command prompt for mean and mode calculation.
3. Type “python -m pip install scipy” in the command prompt for mode calculation.
4. Type “python -m pip install matplotlib” in the command prompt for data visualization(Histograms).
5. Type “pip install pandas” in the command prompt for pandas installation to read data from CSV(comma separated value) file.
6. Type "pip install -U scikit-learn" to install scikit-learn.
7. Type pip install pydotplus to install pydotplus

Machine learning / Mean Calculation Using Numpy

It is the average value of any data set.

Mean = Sum of all data / Number of Items Count in Data Set.

Example: Let us take a data set of some numbers

Ds= [40,10,20,25,24,40,40,14,16]

Number of points in the data set = 9

Mean = (40+10+20+25+24+40+40+14+16) / 9= 229/9= 25.4

Statistical Calculation	**Python Program**	**Output**

Mean	import numpy
	ds = [40,10,20,25,24,40,40,14,16]
	meancal= numpy.mean(ds)
	print(meancal)
		25.44444

Machine learning / Median Calculation Using Numpy
It is the middle number in a sorted list of numbers.

Mean = Sum of all data / Number of Items Count in Data Set.

Example: Let us take a data set of some numbers
Ds= [40,10,20,25,24,40,40,14,16]
Number of points in the data set = 9
median value after sorting the data set = 24

Statistical Calculation	**Python Program**	**Output**

Median	import numpy
	ds = [40,10,20,25,24,40,40,14,16]
	mediancal = numpy.median(ds)
	print(mediancal)
		24.0

Machine learning / Mode Calculation Using Numpy
It is the most repeated value in the data set.

Mode = most repeated value in the Data Set.

Example: Let us take a data set of some numbers
Ds= [40,10,20,25,24,40,40,14,16]
Number of points in the data set = 9
Mode value = 40

Statistical Calculation	**Python Program**	**Output**

Mode	from scipy import stats
	ds = [40,10,20,25,24,40,40,14,16]
	modecal = stats.mode(ds)
	print(modecal)
		40

Machine learning / Variance Calculation Using Numpy
It is the expectation of the squared deviation of a random variable from its mean in the data set.

Variance =[(x1 - mean)2 + (x2 - mean)2 + (x3 - mean)2 + ... + (xn - mean)2] / [n - 1]

Example: Let us take a data set of some numbers
Ds= [40,10,20,25,24,40,40,14,16]
Number of points in the data set = 9
Variance value =[(x1 - mean)2 + (x2 - mean)2 + (x3 - mean)2 + ... + (xn - mean)2] / [n - 1] = 125.135802

Statistical Calculation	**Python Program**	**Output**

Variance	import numpy
	ds = [40,10,20,25,24,40,40,14,16]
	variancecal = numpy.var(ds)
	print(variancecal)
		125.135802

Machine learning / Standard Deviation Calculation Using Numpy
It is the square root of the variance in the data set.

Standard Deviation = Square root of the variance

Example: Let us take a data set of some numbers
Ds= [40,10,20,25,24,40,40,14,16]
Number of points in the data set = 9
Standard Deviation value =square root of the variance = 11.186411

Statistical Calculation	**Python Program**	**Output**

Standard Deviation	import numpy
	ds = [40,10,20,25,24,40,40,14,16]
	stdcal = numpy.std(ds)
	print(stdcal)
		125.135802

Machine learning / Percentile / Centile Calculation
It is a measure used in statistics indicating the value below which a given percentage of observations in a group of observations falls.

Example: Let us take a data set of some numbers
Ds= [40,10,20,25,24,40,40,14,16]
Number of points in the data set = 9
Percentiles value = 40.0

Statistical Calculation	**Python Program**	**Output**

Percentiles	import numpy
	ds = [40,10,20,25,24,40,40,14,16]
	Percentilecal = numpy.percentile(ds, 90)
		40.0

	print(Percentilecal)	

Machine learning / Data Distribution
It is a function / a list which shows all the possible values / intervals of the data.

Example: Let us take a data set of some numbers
Ds= [40,10,20,25,24,40,40,14,16]
Number of points in the data set = 9

Statistical Calculation	**Python Program**	**Output**

Data Distribution	Example: Array contains 60 random floats between 1 and 4:
	
	import numpy
	dd =
	numpy.random.uniform(1.0, 4.0, 60)
	print(dd)
		[3.21258214 1.28331078
		3.51185008 3.07476343
		1.28686508 1.33148431
		3.31476411 2.76986549
		1.87498267 2.75833363
		1.37309329 2.12108411
		2.34681988 2.53021237
		1.97291071 1.62166667
		1.78760123 1.26174762
		2.04166142 3.4823043
		1.57037567 2.06301906
		1.51432998 3.14319527
		1.87475162 1.29929306
		2.34300642 2.35638228
		1.6050797 3.96598089
		3.87540477 2.72004967
		1.16310317 2.21566377
		2.82368836 3.8425559
		3.07532334 2.56885442
		1.8456798 2.29819962
		1.16998725 3.71949883
		1.46764767 1.56129962
		2.183093 2.55620378
		2.6980193 1.62944213
		3.49027593 2.92464324
		2.10505275 3.44077608
		3.22249584 1.13513995
		2.66612039 3.59905547
		3.54972128 1.70253925
		3.79539849 3.85218412]

Machine learning / Histograms in python
It is an accurate representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable
and was first introduced by Karl Pearson.

Example: Let us take a data set of some numbers
Ds= [40,10,20,25,24,40,40,14,16]
Number of points in the data set = 9

Statistical Calculation	**Python Program**	**Output**

Data Distribution	import numpy
	import matplotlib.pyplot as mplobject
	dd = numpy.random.uniform(1.0, 4.0, 60)
	mplobject.hist(dd, 9)
	mplobject.show()
		(array([5., 7., 4., 5., 6., 10., 8., 11., 4.]), array([1.04607125, 1.3694
		1105, 1.69275085, 2.01609064, 2.33943044,2.66277024, 2.98611003,
		3.30944983, 3.63278963, 3.95612942]),
		)

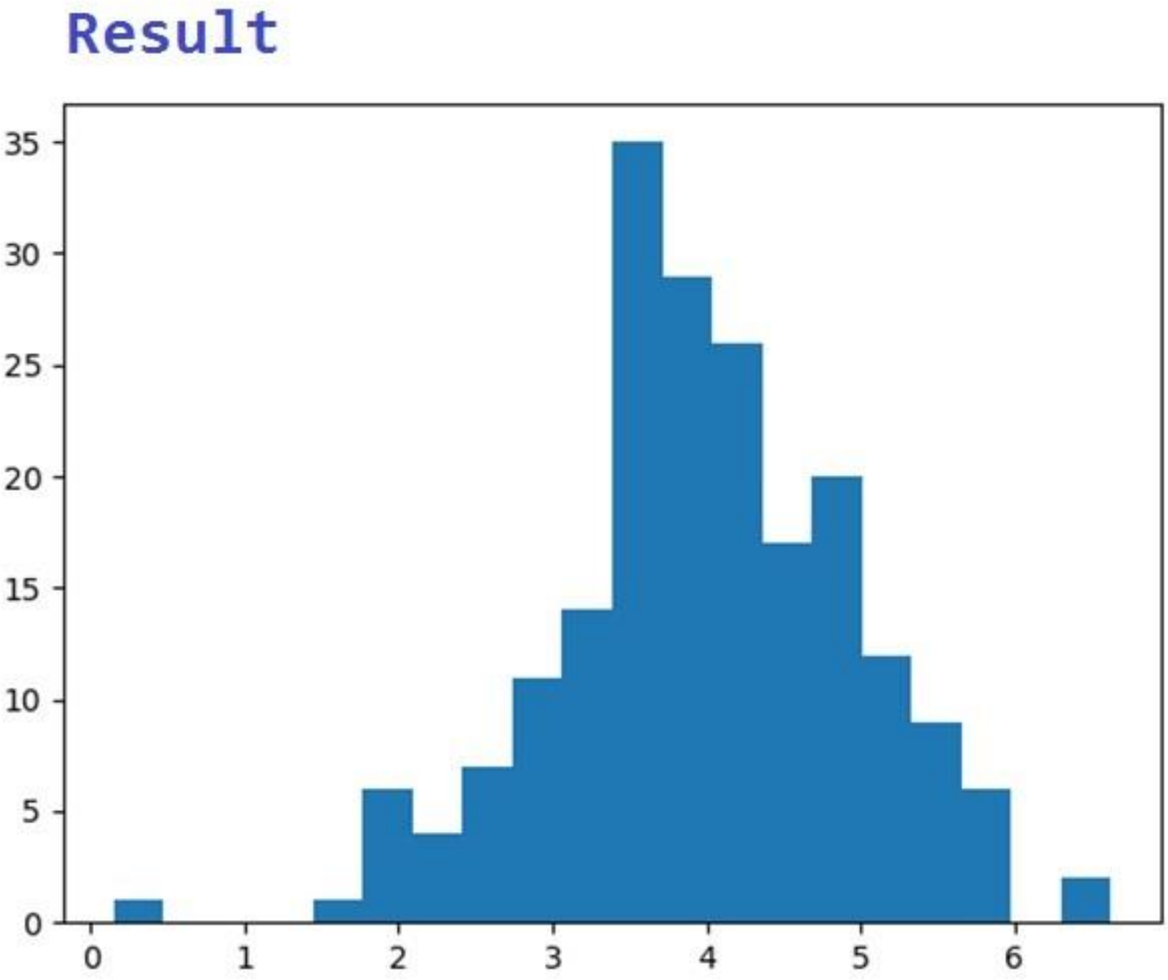
Machine learning / Normal Data Distribution

It is a type of continuous probability distribution for a real-valued random variable.
Other Names: Normal / Gaussian/ Gauss / Laplace Data Distribution

Example: Let us take a data set of some numbers
Ds= [40,10,20,25,24,40,40,14,16]
Number of points in the data set = 9

Statistical Calculation	**Python Program**	**Output**

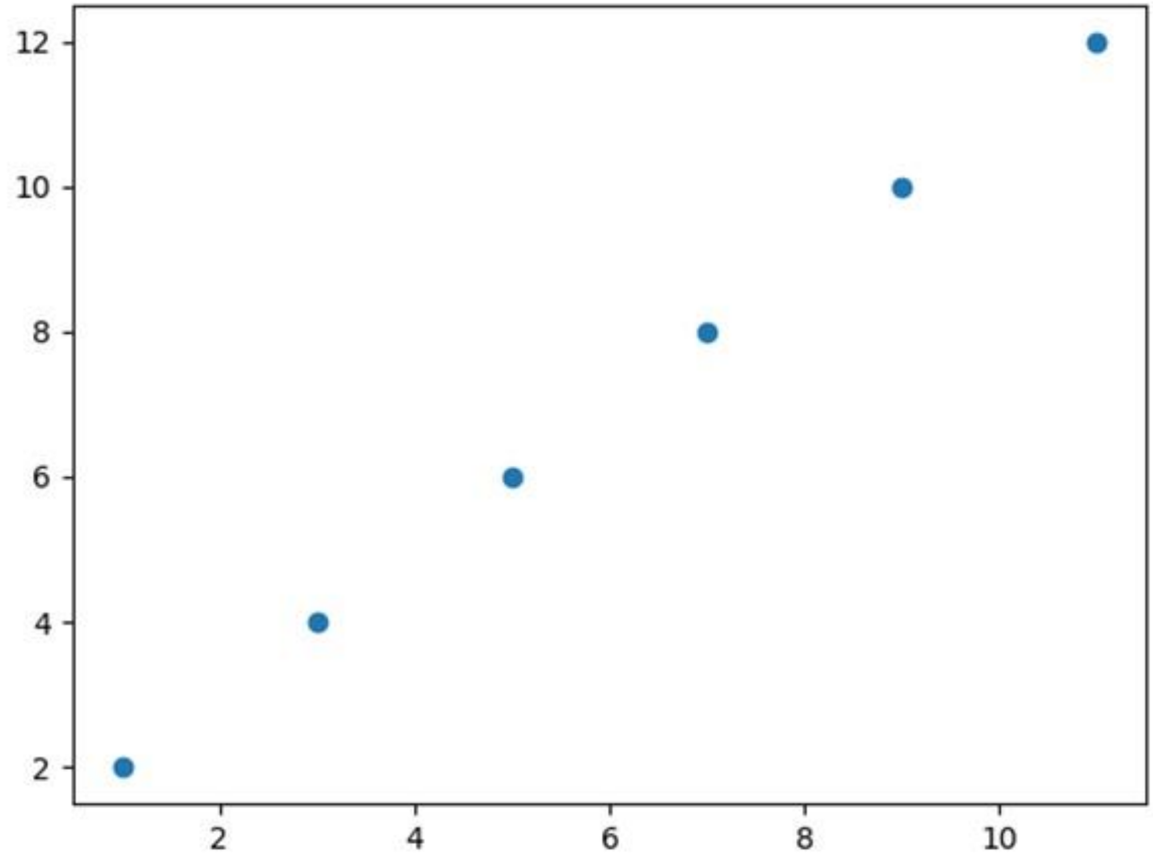
Normal Data Distribution	import numpy
	import matplotlib.pyplot as mplobject
	dd = numpy.random.normal(4.0, 1.0, 200)
	mplobject.hist(dd, 20)
	mplobject.show()
	
	Explanation
	mean = 4.0
	Standard Deviation = 1.0
		(array([1., 0., 0., 0., 1., 6., 4., 7., 11., 14., 35., 29., 26.,17., 20., 12., 9., 6., 0., 2.]), array([0.15032205, 0.47405083, 0.79777961, 1.1215084 , 1.44523718,1.76896596, 2.09269475,2.41642353, 2.74015231, 3.0638811 ,3.38760988, 3.71133866, 4.03506745, 4.35879623, 4.68252501, 5.0062538 , 5.32998258, 5.65371136, 5.97744015, 6.30116893,6.62489771]), <a list of 20 Patch objects>)



Machine learning / Scatter Plot
It is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data.
Other Names: Scatter Plot / graph / diagram / scattergram / mathematical diagram.

Statistical Calculation	**Python Program**	**Output**

Scatter Plot	import matplotlib.pyplot as mplobject
	x = [1,3,5,7,9,11]
	y = [2,4,6,8,10,12]
	mplobject.scatter(x, y)
	mplobject.show()
		matplotlib.collections.PathCollection object at 0x08ACDE20>



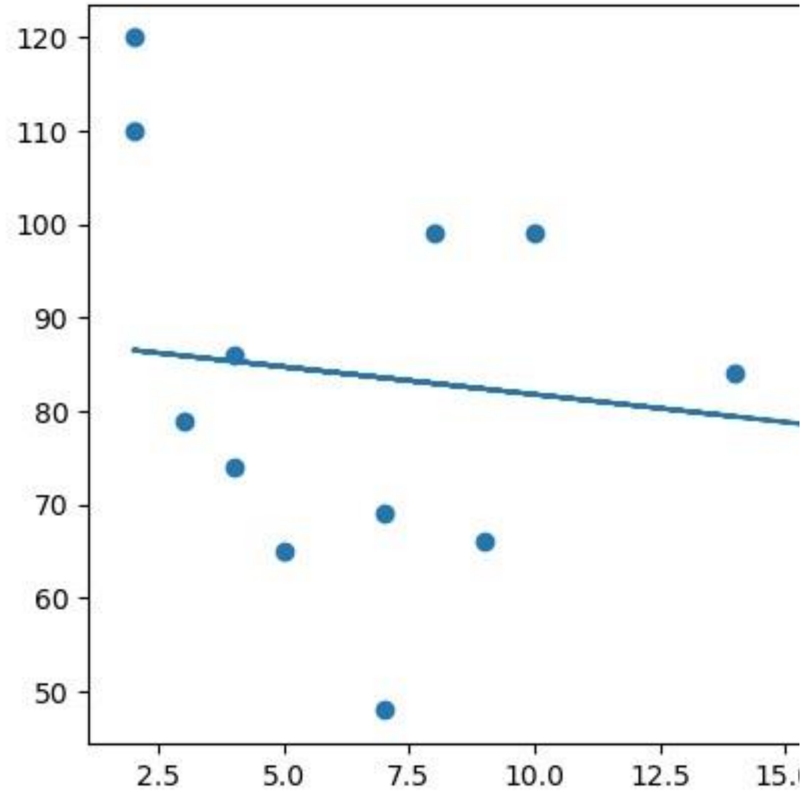
Machine learning / Regression
It attempts to determine the strength of the relationship between 1 dependent variable and a series of other changing variables (independent variables).

Statistical Calculation	Python Program

Regression	import matplotlib.pyplot as mplobject
	from scipy import stats
	xpts = [3,5,7,9,2,20,2,8,4,14,10,7,4]
	ypts = [79,65,48,66,120,85,110,99,86,8
	slope, intercept, r, p, std_err = stats.linre
	ypts)
	def ourfunc(xpts):
	return slope * xpts + intercept
	ourmodel = list(map(ourfunc, xpts))
	mplobject.scatter(xpts, ypts)
	mplobject.plot(xpts, ourmodel)

	mplobject.show()

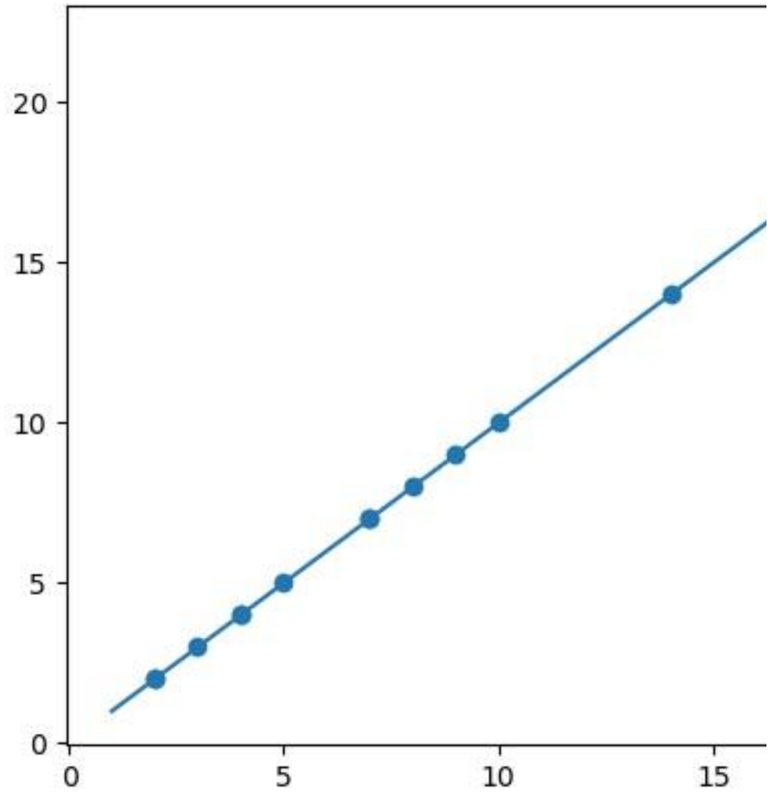
Result



Polynomial Regression Using Numpy It uses the relationship between the variables x and y to find the best way to draw a line through the data points.

Statistical Calculation	Python Program

Polynomial Regression	import numpy
	import matplotlib.pyplot as mplobject
	xpts = [3,5,7,9,2,20,2,8,4,14,10,7,4]
	ypts = [79,65,48,66,120,85,110,99,86,8
	ourmodel = numpy.poly1d(numpy.poly
	3))
	ourline = numpy.linspace(1, 22, 100)
	mplobject.scatter(xpts, xpts)
	mplobject.plot(ourline, ourmodel(ourli
	mplobject.show()



Multiple Regression Using Numpy

It is used to predict a value depends on 2 / more variables. i.e. it is similar to linear regression.

Consider the below data set("products.csv").

Product	price	quantity	profit	category	productcondi
A1	200	190	49	x	YES
A2	400	560	45	x	YES
A3	200	329	45	x	YES
A4	100	265	40	x	YES
A5	700	540	55	x	YES
A6	200	329	55	x	YES
A7	600	509	40	x	YES
A8	700	765	42	x	YES
A9	700	512	48	x	YES
A10	800	550	49	x	YES
A11	300	380	49	x	YES

A12	500	390	51	x	YES
A13	200	512	49	x	YES
A14	800	652	44	x	YES
A15	800	726	47	x	YES
A16	800	730	47	y	YES
A17	800	765	49	y	YES
A18	1400	680	54	y	YES
A19	800	519	54	y	YES
A20	1200	728	55	y	YES
A21	800	984	44	y	NO
A22	1200	828	49	y	NO
A23	1300	765	49	y	NO
A24	800	815	49	y	NO
A25	1200	815	49	y	NO
A26	700	865	52	y	NO
A27	1200	890	54	y	NO
A28	1200	1125	64	y	NO
A29	800	923	59	z	NO
A30	1200	1105	64	z	NO
A31	1300	1005	65	z	NO
A32	1200	1146	67	z	NO
A33	800	635	54	z	NO
A34	800	790	58	z	NO
A35	800	805	59	z	NO
A36	1700	795	70	z	NO

Predict Profit of Products Based On Quantity and Price

Statistical Calculation	**Python Program**

Polynomial Regression	import pandas
	from sklearn import linear_model
	fp = pandas.read_csv("products.csv")
	Xpoints = fp[['price', 'quantity']]
	ypoints = fp['profit']
	regr = linear_model.LinearRegression()
	regr.fit(Xpoints, ypoints)
	#predict profit of a products where price=230(
	and quantity = 1200
	predictprofit = regr.predict([[300, 1200]])
	print(predictprofit)

Machine learning / Scaling of Features

It is used to compare data with different values / units we do scaling.

Scaling Standard method use formula:
$z = (x - u) / s$

where

z	new value
x	original value
u	mean
s	standard deviation

Consider the below data set("products.csv").

Product	**price**	**quantity**	**profit**	**category**	**productconditiongooc**
A1	200	190	49	x	YES
A2	400	560	45	x	YES
A3	200	329	45	x	YES
A4	100	265	40	x	YES
A5	700	540	55	x	YES
A6	200	329	55	x	YES
A7	600	509	40	x	YES
A8	700	765	42	x	YES
A9	700	512	48	x	YES
A10	800	550	49	x	YES
A11	300	380	49	x	YES
A12	500	390	51	x	YES
A13	200	512	49	x	YES
A14	800	652	44	x	YES
A15	800	726	47	x	YES
A16	800	730	47	y	YES
A17	800	765	49	y	YES
A18	1400	680	54	y	YES
A19	800	519	54	y	YES
A20	1200	728	55	y	YES
A21	800	984	44	y	NO
A22	1200	828	49	y	NO
A23	1300	765	49	y	NO
A24	800	815	49	y	NO
A25	1200	815	49	y	NO
A26	700	865	52	y	NO

A27	1200	890	54	y	NO
A28	1200	1125	64	y	NO
A29	800	923	59	z	NO
A30	1200	1105	64	z	NO
A31	1300	1005	65	z	NO
A32	1200	1146	67	z	NO
A33	800	635	54	z	NO
A34	800	790	58	z	NO
A35	800	805	59	z	NO
A36	1700	795	70	z	NO

Statistical Calculation	**Python Program**	**Output**

Scaling of Features	import pandas
	from sklearn import linear_model
	from sklearn.preprocessing import StandardScaler
	scale = StandardScaler() fp = pandas.read_csv("products.csv")
	Xpoints = fp[['price', 'quantity']]
	scaledXpoints = scale.fit_transform(Xpoints)
	print(scaledXpoints)
		[[-1.59336644 -2
		1.07190106 -0.55
		1.59336644 -1.52
		1.85409913 -1.78
		0.28970299 -0.63
		1.59336644 -1.52
		0.55043568 -0.76
		0.28970299 0.304
		0.28970299 -0.75
		0.0289703 -0.595
		1.33263375 -1.30
		0.81116837 -1.26
		1.59336644 -0.75
		0.0289703 -0.168
		0.0289703 0.1412
		0.0289703 0.1580
		0.0289703 0.3046
		1.53542584 -0.05
		0.0289703 -0.725
		1.01396046 0.149
		0.0289703 1.2219
		1.01396046 0.568
		1.27469315 0.304
		0.0289703 0.5140
		1.01396046 0.514
		0.28970299 0.723
		1.01396046 0.828
		1.01396046 1.812
		0.0289703 0.9664
		1.01396046 1.728

		1.27469315 1.30990057] [1.01396046 1.90050772] [-0.0289703 -0.23991961] [-0.0289703 0.40932938] [-0.0289703 0.47215993] [2.31762392 0.4302729]]

Explanation

First record	200	190
Second record	-1.59336644	-2.10389

Comparing first row is not easy but Comparing second row is easy because values are small. That’s why we use scaling.

Machine learning / Data Sets for Testing and Training
Model
It is a representation of real world process and is used to predict on the test data.

There are 3 data sets used in different stages of the creation of the model. They were

Data set Name	**Used to**
Training Data Set	Fits the model
Test Data Set	Tests Model
Validation Data set	Predicts the responses for the observations

Training dataset
It is used to fit the parameters of the model.

Validation dataset
The validation dataset provides an unbiased evaluation of a model fit on the training dataset while tuning the models hyperparameters.It is used to stop training when the error on the validation dataset increases i.e it’s a sign of over fitting to the training dataset.

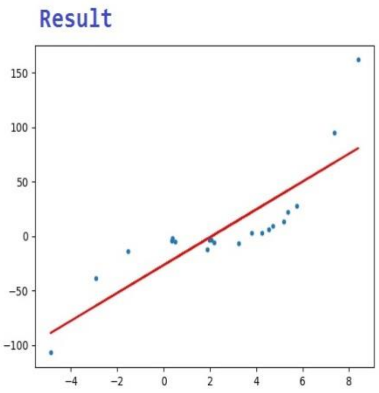
Test / holdout dataset
It is used to provide an unbiased evaluation of a final model fit on the training dataset.

Split Into Train or Test Data Set
Intial data set = Train Data Set + Test Data Set.

Example
Train Data Set = 70 % + Test Data Set = 30 % = Initial Data set (Total Data Set)

Apply a linear regression model to this dataset

Python Program	**Output**

import numpy as npobj
import matplotlib.pyplot as pltobj
from sklearn.linear_model import LinearRegression
npobj.random.seed(2)
x = 2 - 3 * npobj.random.normal(0, 1, 20)
y = x - 2 * (x ** 2) + 0.5 * (x ** 3) + npobj.random.normal(-3, 3, 20)
# transform data to include another axis 	
x = x[:, npobj.newaxis]
y = y[:, npobj.newaxis]
model = LinearRegression()
model.fit(x , y) y_pred = model.predict(x)
pltobj.scatter(x , y, s=10)
pltobj.plot(x , y_pred, color='r')
pltobj.show()
	

Machine learning / Decision Tree Using Numpy

It is used to take decisions on a given data set.

Consider the below data set("products.csv").

Product	price	quantity	profit	category	productconditiongood
A1	200	190	49	x	YES
A2	400	560	45	x	YES
A3	200	329	45	x	YES
A4	100	265	40	x	YES
A5	700	540	55	x	YES
A6	200	329	55	x	YES
A7	600	509	40	x	YES
A8	700	765	42	x	YES
A9	700	512	48	x	YES
A10	800	550	49	x	YES
A11	300	380	49	x	YES
A12	500	390	51	x	YES
A13	200	512	49	x	YES
A14	800	652	44	x	YES
A15	800	726	47	x	YES
A16	800	730	47	y	YES
A17	800	765	49	y	YES
A18	1400	680	54	y	YES
A19	800	519	54	y	YES

A20	1200	728	55	y	YES
A21	800	984	44	y	NO
A22	1200	828	49	y	NO
A23	1300	765	49	y	NO
A24	800	815	49	y	NO
A25	1200	815	49	y	NO
A26	700	865	52	y	NO
A27	1200	890	54	y	NO
A28	1200	1125	64	y	NO
A29	800	923	59	z	NO
A30	1200	1105	64	z	NO
A31	1300	1005	65	z	NO
A32	1200	1146	67	z	NO
A33	800	635	54	z	NO
A34	800	790	58	z	NO
A35	800	805	59	z	NO
A36	1700	795	70	z	NO

Read and print the data set: using pandas

Statistical Calculation	Python Program

pandas read and print	import pandas
	from sklearn import tree
	import pydotplus
	from sklearn.tree import DecisionTreeClassifier
	import matplotlib.pyplot as plt
	import matplotlib.image as pltimg
	df = pandas.read_csv("products.csv")
	print(df)

Result

```
0    A1    200    190    49    x    YES
1    A2    400    560    45    x    YES
2    A3    200    329    45    x    YES
3    A4    100    265    40    x    YES
4    A5    700    540    55    x    YES
5    A6    200    329    55    x    YES
6    A7    600    509    40    x    YES
7    A8    700    765    42    x    YES
8    A9    700    512    48    x    YES
9    A10    800    550    49    x    YES
10    A11    300    380    49    x    YES
11    A12    500    390    51    x    YES
```

```
12    A13    200    512    49    x    YES
13    A14    800    652    44    x    YES
14    A15    800    726    47    x    YES
15    A16    800    730    47    y    YES
16    A17    800    765    49    y    YES
17    A18    1400    680    54    y    YES
18    A19    800    519    54    y    YES
19    A20    1200    728    55    y    YES
20    A21    800    984    44    y    NO
21    A22    1200    828    49    y    NO
22    A23    1300    765    49    y    NO
23    A24    800    815    49    y    NO
24    A25    1200    815    49    y    NO
25    A26    700    865    52    y    NO
26    A27    1200    890    54    y    NO
27    A28    1200    1125    64    y    NO
28    A29    800    923    59    z    NO
29    A30    1200    1105    64    z    NO
30    A31    1300    1005    65    z    NO
31    A32    1200    1146    67    z    NO
32    A33    800    635    54    z    NO
33    A34    800    790    58    z    NO
34    A35    800    805    59    z    NO
35    A36    1700    795    70    z    NO
```

Statistical Calculation	**Python Program**

pandas numerical to strings	import pandas
	from sklearn import tree
	import pydotplus
	from sklearn.tree import DecisionTreeClassifier
	import matplotlib.pyplot as plt
	import matplotlib.image as pltimg
	fp = pandas.read_csv("products.csv")
	points = {'x': 2, 'y': 4, 'z': 6}
	fp['category'] = fp['category'].map(points)
	points = {'YES': 1, 'NO': 0}
	fp['productconditiongood'] = fp['productconditiongood'].map(points)
	print(fp)

Result

```
0    A1    200    190    49    x    1
1    A2    400    560    45    x    1
2    A3    200    329    45    x    1
3    A4    100    265    40    x    1
4    A5    700    540    55    x    1
5    A6    200    329    55    x    1
```

6	A7	600	509	40	x	1
7	A8	700	765	42	x	1
8	A9	700	512	48	x	1
9	A10	800	550	49	x	1
10	A11	300	380	49	x	1
11	A12	500	390	51	x	1
12	A13	200	512	49	x	1
13	A14	800	652	44	x	1
14	A15	800	726	47	x	1
15	A16	800	730	47	y	1
16	A17	800	765	49	y	1
17	A18	1400	680	54	y	1
18	A19	800	519	54	y	1
19	A20	1200	728	55	y	1
20	A21	800	984	44	y	0
21	A22	1200	828	49	y	0
22	A23	1300	765	49	y	0
23	A24	800	815	49	y	0
24	A25	1200	815	49	y	0
25	A26	700	865	52	y	0
26	A27	1200	890	54	y	0
27	A28	1200	1125	64	y	0
28	A29	800	923	59	z	0
29	A30	1200	1105	64	z	0
30	A31	1300	1005	65	z	0
31	A32	1200	1146	67	z	0
32	A33	800	635	54	z	0
33	A34	800	790	58	z	0
34	A35	800	805	59	z	0
35	A36	1700	795	70	z	0