linear regression_raman

June 6, 2019

```
In [1]: import os

In [2]: cd

C:\Users\raman


In [3]: import pandas as pd

In [5]: import matplotlib.pyplot as plt

In [6]: import seaborn as sns

In [7]: %matplotlib inline

In [9]: train = pd.read_csv("train.csv")

In [11]: train.head()
```

Out[11]:

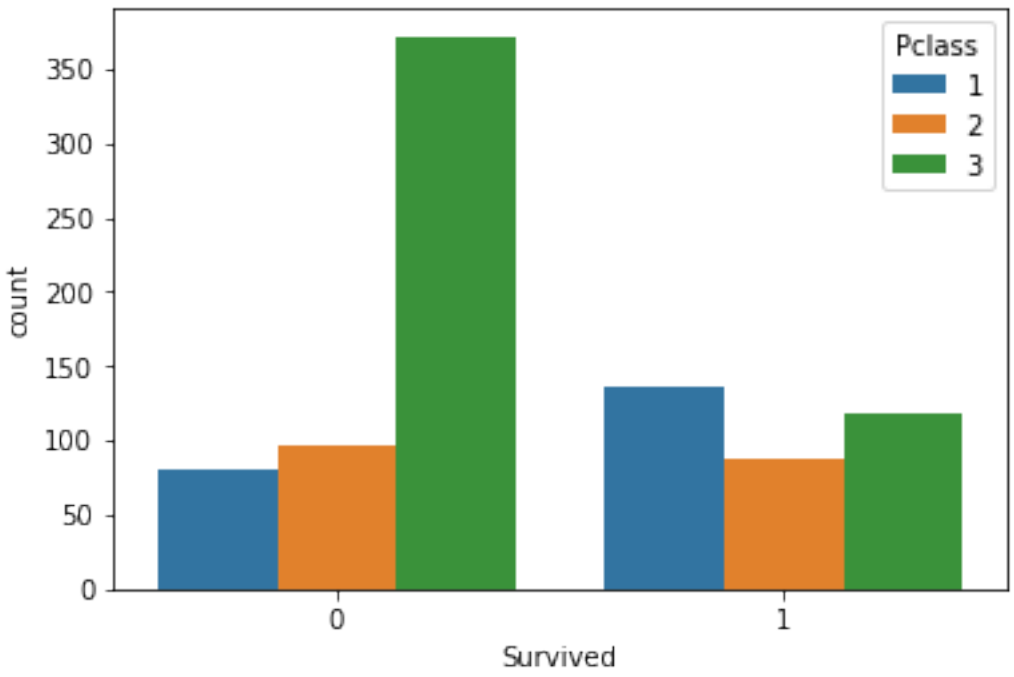
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
In [12]: train.count()

Out[12]: PassengerId    891
         Survived       891
         Pclass         891
         Name           891
         Sex            891
         Age            714
         SibSp          891
         Parch          891
         Ticket         891
         Fare           891
         Cabin          204
         Embarked       889
         dtype: int64

In [13]: train[train['Sex'].str.match("female")].count()

Out[13]: PassengerId    314
         Survived       314
         Pclass         314
         Name           314
         Sex            314
         Age            261
         SibSp          314
         Parch          314
         Ticket         314
         Fare           314
         Cabin           97
         Embarked       312
         dtype: int64

In [14]: train[train['Sex'].str.match("male")].count()

Out[14]: PassengerId    577
         Survived       577
         Pclass         577
         Name           577
         Sex            577
         Age            453
         SibSp          577
         Parch          577
         Ticket         577
         Fare           577
         Cabin          107
         Embarked       577
         dtype: int64

In [15]: train[train["Name"].str.contains("Dawson")]
```

```
Out[15]: Empty DataFrame
         Columns: [PassengerId, Survived, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, C
         Index: []
```

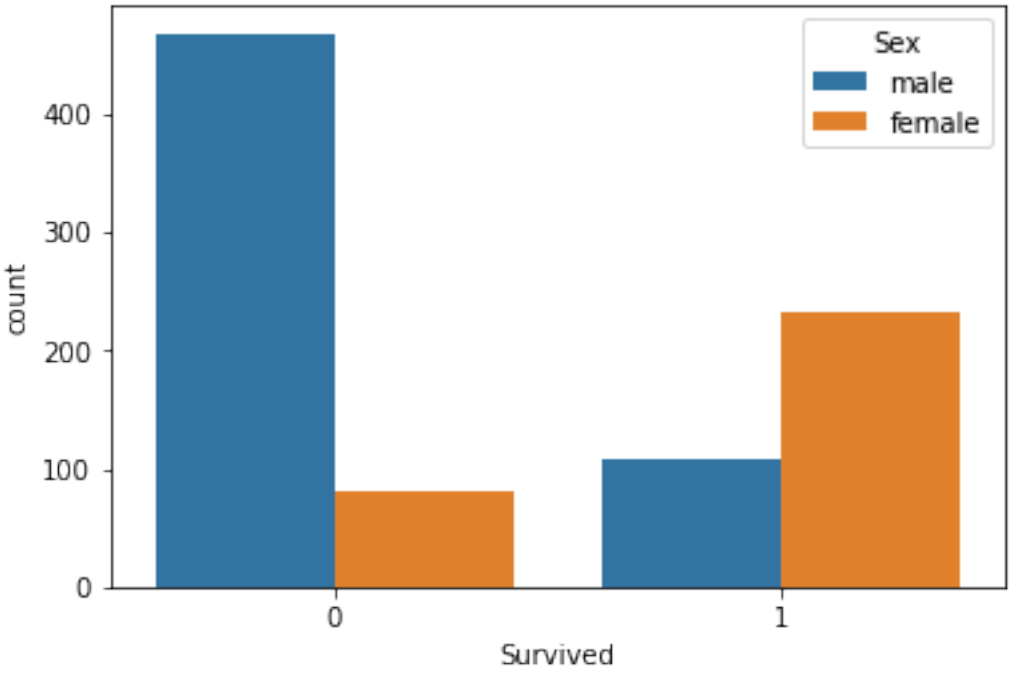
```
In [16]: sns.countplot(x='Survived', hue='Pclass', data=train)
```

```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x215f6525ac8>
```

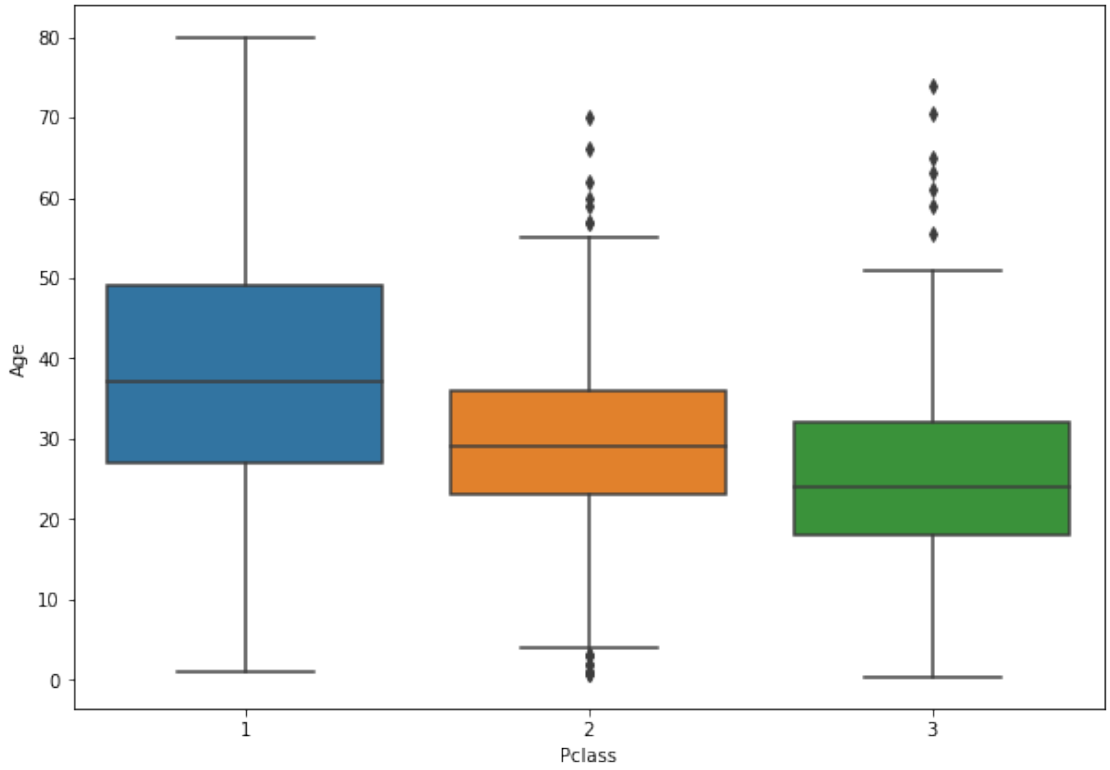



```
In [17]: sns.countplot(x='Survived', hue='Sex', data=train)
```

```
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x215f68557f0>
```




```
In [18]: plt.figure(figsize=(10,7))
         sns.boxplot(x='Pclass',y='Age',data=train)

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x215f68a29e8>
```




```
In [19]: def add_age(cols):
             Age = cols[0]
             Pclass = cols[1]
             if pd.isnull(Age):
                 return int(train[train["Pclass"] == Pclass]["Age"].mean())
             else:
                 return Age

In [20]: train["Age"] = train[["Age", "Pclass"]].apply(add_age,axis=1)

In [21]: train.drop("Cabin",inplace=True,axis=1)

In [22]: train.dropna(inplace=True)

In [23]: pd.get_dummies(train["Sex"])

Out[23]:      female  male
          0         0     1
          1         1     0
          2         1     0
          3         1     0
          4         0     1
          5         0     1
```

6	0	1
7	0	1
8	1	0
9	1	0
10	1	0
11	1	0
12	0	1
13	0	1
14	1	0
15	1	0
16	0	1
17	0	1
18	1	0
19	1	0
20	0	1
21	0	1
22	1	0
23	0	1
24	1	0
25	1	0
26	0	1
27	0	1
28	1	0
29	0	1
..	...	...
861	0	1
862	1	0
863	1	0
864	0	1
865	1	0
866	1	0
867	0	1
868	0	1
869	0	1
870	0	1
871	1	0
872	0	1
873	0	1
874	1	0
875	1	0
876	0	1
877	0	1
878	0	1
879	1	0
880	1	0
881	0	1
882	1	0
883	0	1

```
884       0     1
885       1     0
886       0     1
887       1     0
888       1     0
889       0     1
890       0     1

[889 rows x 2 columns]
```

```
In [24]: sex = pd.get_dummies(train["Sex"],drop_first=True)
```

```
In [25]: embarked = pd.get_dummies(train["Embarked"],drop_first=True)
         embarked = pd.get_dummies(train["Pclass"],drop_first=True)
```

```
In [27]: train["Age"] = train[["Age", "Pclass"]].apply(add_age,axis=1)
```

```
In [30]: train.drop(["PassengerId","Pclass","Name","Sex","Ticket","Embarked"],axis=1,inplace=T
```

```
In [33]: X = train.drop("Survived",axis=1)
         y = train["Survived"]
```

```
In [38]: X.head()
```

Out[38]:

	Age	SibSp	Parch	Fare
0	22.0	1	0	7.2500
1	38.0	1	0	71.2833
2	26.0	0	0	7.9250
3	35.0	1	0	53.1000
4	35.0	0	0	8.0500

```
In [39]: y.head()
```

```
Out[39]: 0    0
         1    1
         2    1
         3    1
         4    0
         Name: Survived, dtype: int64
```

```
In [40]: import numpy as np
```

```
In [41]: x1 = np.linspace(0.0, 6.0, 4)
```

```
In [42]: print('x1:',x1)
```

```
x1: [0. 2. 4. 6.]
```

```
In [43]: np.linspace(10.0, 12.0, 3)
```

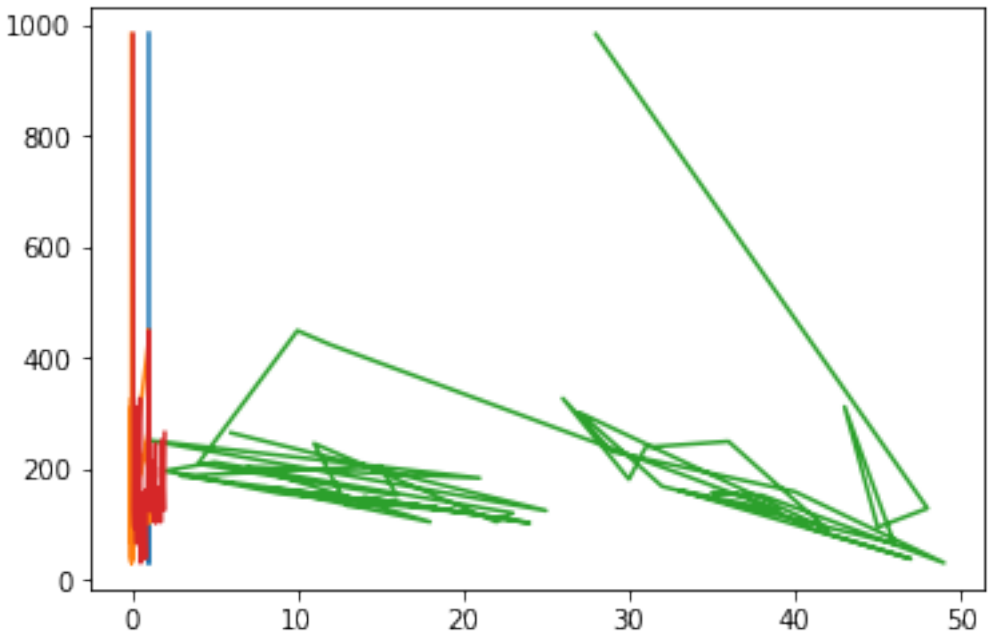
```
Out[43]: array([10., 11., 12.])

In [44]: np.linspace(10.0, 12.0, 11)

Out[44]: array([10. , 10.2, 10.4, 10.6, 10.8, 11. , 11.2, 11.4, 11.6, 11.8, 12. ])

In [45]: np.linspace(0.0, 10.0, 11)

Out[45]: array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.])

In [46]: x = np.array([0.0, 1.0, 2.0, 3.0,  4.0,  5.0])

In [47]: y = np.array([0.0, 0.8, 0.9, 0.1, -0.8, -1.0])

In [69]: z = np.polyfit(x, y,3)

In [49]: print("z:",z)

z: [ 0.08703704 -0.81349206  1.69312169 -0.03968254]


In [50]: x = [-0.018,-0.008,0.011,0.017,-0.008,-0.002]

In [51]: y = [-0.006,-0.001,0.015,0.017,-0.0019,-0.005]

In [52]: print("x:",x)

x: [-0.018, -0.008, 0.011, 0.017, -0.008, -0.002]


In [53]: print("y:",y)

y: [-0.006, -0.001, 0.015, 0.017, -0.0019, -0.005]


In [54]: from scipy import stats

In [60]: print("Gradient and intercept:",x,y)

Gradient and intercept: [-0.018, -0.008, 0.011, 0.017, -0.008, -0.002] [-0.006, -0.001, 0.015,


In [74]: import numpy as np

In [75]: x1 = np.linspace(0.0, 6.0, 4)

In [76]: print('x1:',x1)

x1: [0. 2. 4. 6.]


In [84]: x2 = np.linspace(-10.0, 10, 21)
```

```
In [78]: print('x2:',x2)

x2: [-10.  -9.  -8.  -7.  -6.  -5.  -4.  -3.  -2.  -1.   0.   1.   2.   3.
   4.   5.   6.   7.   8.   9.  10.]


In [85]: from scipy import stats

In [86]: import numpy as np

In [87]: import statsmodels.api as sm

In [88]: import statsmodels.formula.api as smf

In [89]: y = [984.5410628019, 128.8194444444, 94.495412844, 312.0331950207,
         65.1127819549,  168.3289588801, 301.7441860465, 90.1408450704,
         249.4573643411, 239.0361445783, 181.1775200714, 327.2440944882,
         230.9523809524, 158.9442815249, 30.3759398496, 152.5783619818,
         157.5938566553, 150.9933774834, 92.2413793103, 37.5706214689,
         161.2958226769, 125.1546391753, 181.0394610202, 423.9678899083,
         449.3975903614, 208.9949518128, 151.5695067265, 205.4315027158,
         187.1388998813, 121.733615222, 155.1660516605, 196.2901896125,
         211.9626168224, 120.3170028818, 104.2812254517, 245.7815565729,
         133.5061088486, 135.6054191363, 102.4313372355, 148.6814566764,
         125.9274357929, 166.783352781, 104.2042042042, 203.8586703859,
         183.3114323259, 251.4944939696, 124.5541022592, 264.9880095923]

In [90]: x1 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
               0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1]

In [91]: x2 = [28,48,45,43,46,32,27,42,36,31,30,26,29,40,49,38,35,
         37,41,47,33,39,34,12,10,4,16,15,3,20,9,2,5,23,22,11,
         13,17,24,14,19,8,18,7,21,1,25,6]

In [92]: x3 = [0.09,0.12,0.17,0.23,0.31,0.32,0.33,0.34,0.38,0.42,0.43,
         0.5,0.51,0.52,0.55,0.74,0.78,0.79,0.8,0.83,0.84,0.93,0.95,1.05,1.05,1.09,1.11,1.13,1.1

In [93]: x = np.column_stack((x1,x2,x3))

In [94]: x = sm.add_constant(x, prepend=True)

In [95]: results = smf.OLS(y,x).fit()

In [96]: print(results.summary())

                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.363
Model:                            OLS   Adj. R-squared:                  0.319
Method:                 Least Squares   F-statistic:                     8.342
```

```
Date:                Thu, 06 Jun 2019   Prob (F-statistic):           0.000168
Time:                        22:39:47   Log-Likelihood:                -295.45
No. Observations:                  48   AIC:                             598.9
Df Residuals:                      44   BIC:                             606.4
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        680.7824    102.383      6.649      0.000     474.443     887.122
x1          -116.6710     76.995     -1.515      0.137    -271.844      38.502
x2           -10.5126      2.459     -4.275      0.000     -15.469      -5.556
x3          -168.7207     52.589     -3.208      0.002    -274.706     -62.735
==============================================================================
Omnibus:                       62.282   Durbin-Watson:                   1.256
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              530.769
Skew:                           3.245   Prob(JB):                    5.56e-116
Kurtosis:                      17.942   Cond. No.                         189.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

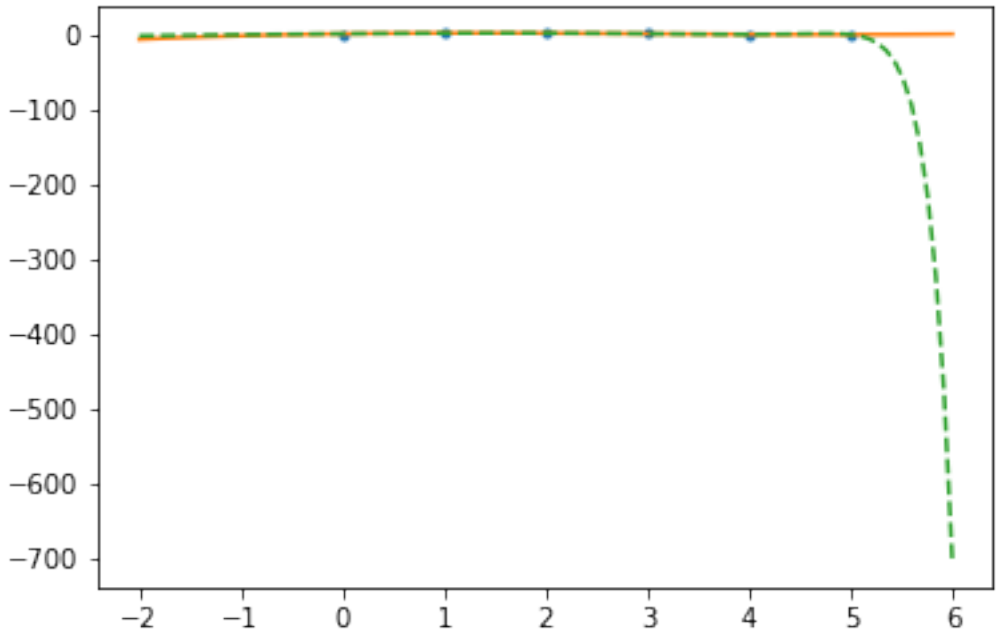
```
In [102]: plt.plot(x,y)
```

```
Out[102]: [<matplotlib.lines.Line2D at 0x215f7cdc5c0>,
           <matplotlib.lines.Line2D at 0x215f7cdc710>,
           <matplotlib.lines.Line2D at 0x215f7cdc860>,
           <matplotlib.lines.Line2D at 0x215f7cdc9b0>]
```

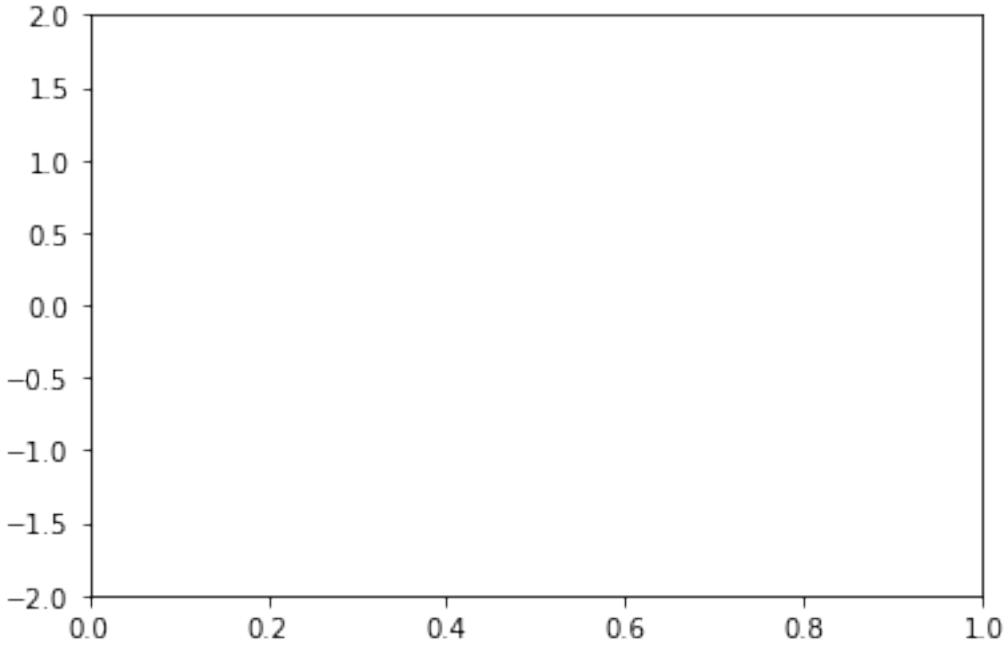



```
In [103]: x = np.array([0.0, 1.0, 2.0, 3.0,  4.0,  5.0])

In [104]: y = np.array([0.0, 0.8, 0.9, 0.1, -0.8, -1.0])

In [105]: z = np.polyfit(x, y, 3)

In [106]: print("z:",z)

z: [ 0.08703704 -0.81349206  1.69312169 -0.03968254]

In [107]: p = np.poly1d(z)

In [108]: print("p(0.5):",p(0.5))

p(0.5): 0.6143849206349201

In [109]: print("p(3.5):",p(3.5))

p(3.5): -0.347321428571432

In [110]: print("p(10):",p(10))

p(10): 22.579365079365022
```

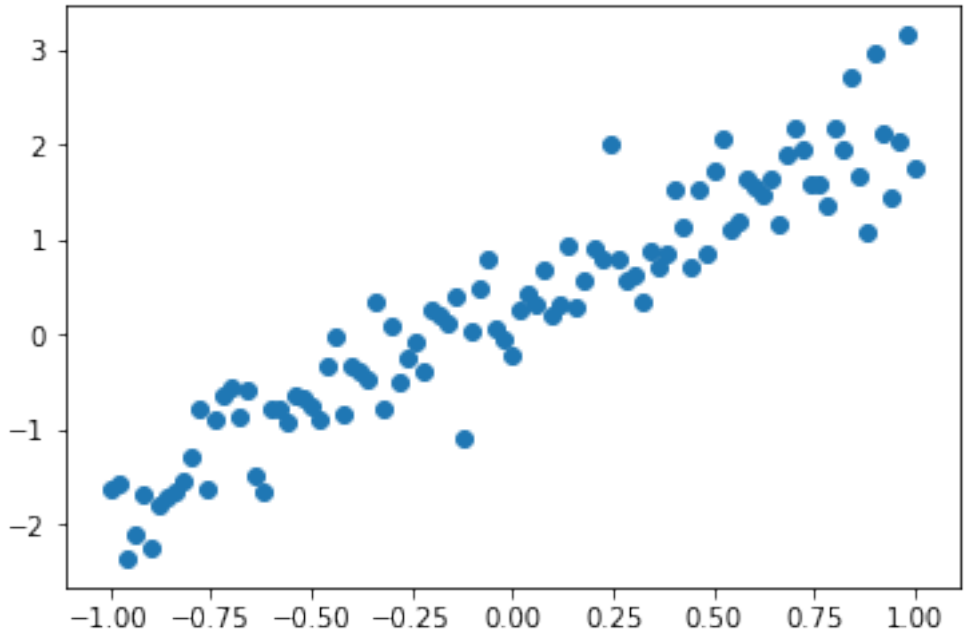
```
In [113]: import matplotlib.pyplot as plt
In [114]: xp = np.linspace(-2, 6, 100)
In [115]: _ = plt.plot(x, y, '.', xp, p(xp), '-', xp, p30(xp), '--')
```




```
In [116]: plt.ylim(-2,2)
Out[116]: (-2, 2)
```

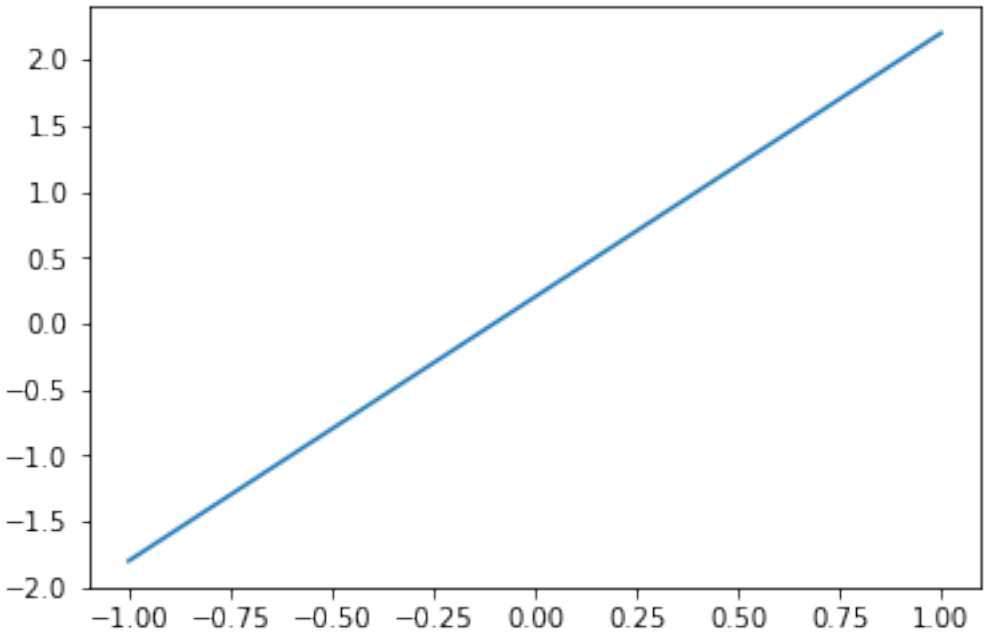


```
In [117]: plt.show()

In [119]: import numpy as np

In [120]: import matplotlib.pyplot as plt

In [121]: trX = np.linspace(-1, 1, 101)

In [122]: import numpy as np

In [123]: print(trX)

[-1.   -0.98 -0.96 -0.94 -0.92 -0.9  -0.88 -0.86 -0.84 -0.82 -0.8  -0.78
 -0.76 -0.74 -0.72 -0.7  -0.68 -0.66 -0.64 -0.62 -0.6  -0.58 -0.56 -0.54
 -0.52 -0.5  -0.48 -0.46 -0.44 -0.42 -0.4  -0.38 -0.36 -0.34 -0.32 -0.3
 -0.28 -0.26 -0.24 -0.22 -0.2  -0.18 -0.16 -0.14 -0.12 -0.1  -0.08 -0.06
 -0.04 -0.02  0.    0.02  0.04  0.06  0.08  0.1   0.12  0.14  0.16  0.18
  0.2   0.22  0.24  0.26  0.28  0.3   0.32  0.34  0.36  0.38  0.4   0.42
  0.44  0.46  0.48  0.5   0.52  0.54  0.56  0.58  0.6   0.62  0.64  0.66
  0.68  0.7   0.72  0.74  0.76  0.78  0.8   0.82  0.84  0.86  0.88  0.9
  0.92  0.94  0.96  0.98  1.  ]


In [124]: trY = 2*trX + np.random.randn(*trX.shape)*0.4+0.2

In [126]: plt.figure()

Out[126]: <Figure size 432x288 with 0 Axes>

<Figure size 432x288 with 0 Axes>


In [127]: plt.scatter(trX,trY)

Out[127]: <matplotlib.collections.PathCollection at 0x215f7d61748>
```

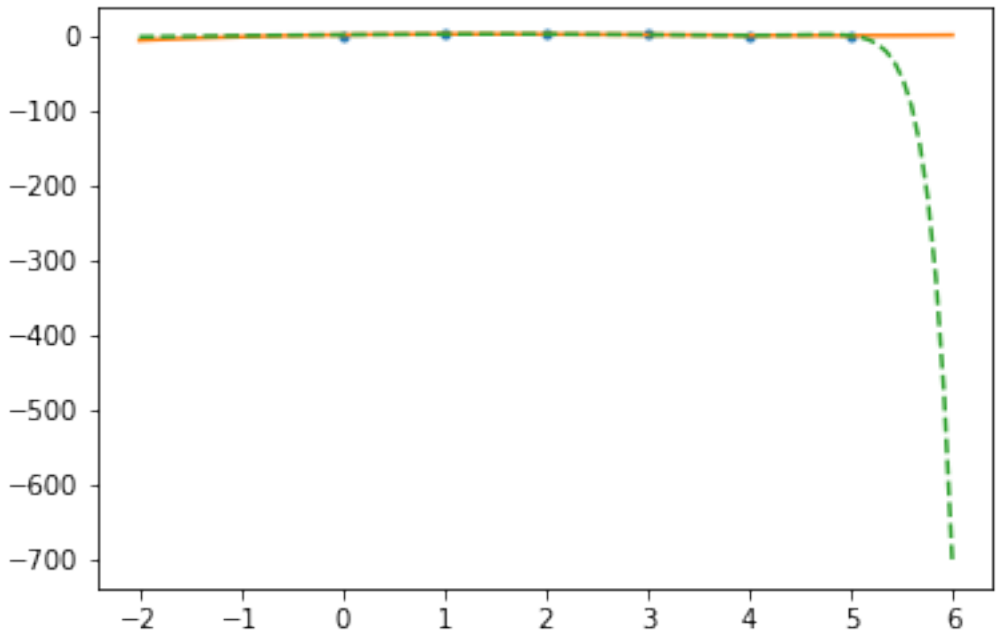



```
In [128]: plt.plot (trX, .2 + 2 * trX)
```

```
Out[128]: [<matplotlib.lines.Line2D at 0x215f7e73898>]
```



```
In [129]: plt.show()

In [130]: import numpy as np

In [131]: x = np.array([0.0, 1.0, 2.0, 3.0,  4.0,  5.0])

In [132]: y = np.array([0.0, 0.8, 0.9, 0.1, -0.8, -1.0])

In [133]: z = np.polyfit(x, y, 3)

In [134]: print("z:",z)

z: [ 0.08703704 -0.81349206  1.69312169 -0.03968254]

In [135]: p = np.poly1d(z)

In [136]: print("p(0.5):",p(0.5))

p(0.5): 0.6143849206349201

In [138]: xp = np.linspace(-2, 6, 100)

In [139]: _ = plt.plot(x, y, '.', xp, p(xp), '-', xp, p30(xp), '--')
```




```
In [140]: x = [-0.018,-0.008,0.011,0.017,-0.008,-0.002]
```

```
In [141]: y = [-0.006,-0.001,0.015,0.017,-0.0019,-0.005]

In [142]: gradient,intercept,r_value,p_value,std_err=stats.linregress(x,y)

In [143]: print("x:",x)

x: [-0.018, -0.008, 0.011, 0.017, -0.008, -0.002]


In [144]: print("y:",y)

y: [-0.006, -0.001, 0.015, 0.017, -0.0019, -0.005]


In [145]: print("Gradient and intercept:")

Gradient and intercept:


In [146]: print(gradient,intercept)

0.7240841777084958 0.003982112236944661


In [147]: print("R-squared",r_value**2)

R-squared 0.8541759033554293


In [148]: print("p-value",p_value)

p-value 0.008397118596107471


In [149]: from math import sqrt

In [150]: def rmse_metric(actual, predicted):
              sum_error = 0.0
              for i in range(len(actual)):
                prediction_error = predicted[i] - actual[i]
                sum_error += (prediction_error ** 2)
              mean_error = sum_error / float(len(actual))
              return sqrt(mean_error)


In [151]: def evaluate_algorithm(dataset, algorithm):
              test_set = list()
              for row in dataset:
                row_copy = list(row)
                row_copy[-1] = None
                test_set.append(row_copy)
```

```
In [156]:  from scipy import stats

In [157]: import numpy as np

In [158]: x = [-0.018,-0.008,0.011,0.017,-0.008,-0.002]

In [159]: y = [-0.006,-0.001,0.015,0.017,-0.0019,-0.005]

In [160]: gradient,intercept,r_value,p_value,std_err=stats.linregress(x,y)

In [161]: print("Gradient and intercept",gradient,intercept)

Gradient and intercept 0.7240841777084958 0.003982112236944661


In [162]: print("R-squared",r_value**2)

R-squared 0.8541759033554293


In [163]: print("p-value",p_value)

p-value 0.008397118596107471


In [ ]:
```