Machine Learning (CodeWithHarry)

Playlist Link: https://youtube.com/playlist?list=PLu0W_9lII9ai6fAMHp-acBmJONT7Y4BSG (https://youtube.com/playlist?list=PLu0W_9lII9ai6fAMHp-acBmJONT7Y4BSG)

Lectures

1. (Link: https://youtu.be/_u-PaJCpwiU (https://youtu.be/_u-PaJCpwiU))
2. (Link: https://youtu.be/suCgvxW27og (https://youtu.be/suCgvxW27og))
3. (Link: https://youtu.be/6pUkiLnpYC4 (https://youtu.be/6pUkiLnpYC4))
4. (Link: https://youtu.be/DA0dIVeXpzQ (https://youtu.be/DA0dIVeXpzQ))
5. (Link: https://youtu.be/SJRojOq5P_4 (https://youtu.be/SJRojOq5P_4))
6. (Link: https://youtu.be/U3YAUQUdwiI (https://youtu.be/U3YAUQUdwiI))
7. (Link: https://youtu.be/sAabARyfYZg (https://youtu.be/sAabARyfYZg))
8. (Link: https://youtu.be/Ni5QwU-xaUs (https://youtu.be/Ni5QwU-xaUs))
9. (Link: https://youtu.be/0Kha6KIto28 (https://youtu.be/0Kha6KIto28))
10. (Link: https://youtu.be/pfmJtWEMEpo (https://youtu.be/pfmJtWEMEpo))
11. (Link: https://youtu.be/vKdgg8O2me8 (https://youtu.be/vKdgg8O2me8))
12. (Link: https://youtu.be/BZ10SGWtU-s (https://youtu.be/BZ10SGWtU-s))
13. (Link: https://youtu.be/cJVkqh7-EZs (https://youtu.be/cJVkqh7-EZs))
14. (Link: https://youtu.be/U3SfXcPqeTo (https://youtu.be/U3SfXcPqeTo))
15. (Link: https://youtu.be/lCUMtthAr38 (https://youtu.be/lCUMtthAr38))
16. (Link: https://youtu.be/P4KD9GhA15Y (https://youtu.be/P4KD9GhA15Y))
17. (Link: https://youtu.be/nkakQ4_ouKE (https://youtu.be/nkakQ4_ouKE))
18. (Link: https://youtu.be/D-ZZa6DS8aw (https://youtu.be/D-ZZa6DS8aw))
19. (Link: https://youtu.be/eL6dukE4f-4 (https://youtu.be/eL6dukE4f-4))
20. (Link:)
21. (Link:)
22. (Link:)

#10 Linear Regression Code

In [1]:
```
import sklearn
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error
```

In [3]:
```
diabetes = datasets.load_diabetes()
display(diabetes)
```

{'data': array([[0.03807591, 0.05068012, 0.06169621, ..., -0.00259226,
         0.01990842, -0.01764613],
        [-0.00188202, -0.04464164, -0.05147406, ..., -0.03949338,
         -0.06832974, -0.09220405],
        [0.08529891, 0.05068012, 0.04445121, ..., -0.00259226,
         0.00286377, -0.02593034],
        ...,
        [0.04170844, 0.05068012, -0.01590626, ..., -0.01107952,
         -0.04687948, 0.01549073],
        [-0.04547248, -0.04464164, 0.03906215, ..., 0.02655962,
         0.04452837, -0.02593034],
        [-0.04547248, -0.04464164, -0.0730303 , ..., -0.03949338,
         -0.00421986, 0.00306441]]),
 'target': array([151., 75., 141., 206., 135., 97., 138., 63., 110., 310., 101.,
         69., 179., 185., 118., 171., 166., 144., 97., 168., 68., 49.,
         68., 245., 184., 202., 137., 85., 131., 283., 129., 59., 341.,
         87., 65., 102., 265., 276., 252., 90., 100., 55., 61., 92.,
        259., 53., 190., 142., 75., 142., 155., 225., 59., 104., 182.,
        128., 52., 37., 170., 170., 61., 144., 52., 128., 71., 163.,
        150., 97., 160., 178., 48., 270., 202., 111., 85., 42., 170.,
        200., 252., 113., 143., 51., 52., 210., 65., 141., 55., 134.,
         42., 111., 98., 164., 48., 96., 90., 162., 150., 279., 92.,
         83., 128., 102., 302., 198., 95., 53., 134., 144., 232., 81.,
        104., 59., 246., 297., 258., 229., 275., 281., 179., 200., 200.,
        173., 180., 84., 121., 161., 99., 109., 115., 268., 274., 158.,
        107., 83., 103., 272., 85., 280., 336., 281., 118., 317., 235.,
         60., 174., 259., 178., 128., 96., 126., 288., 88., 292., 71.,
        197., 186., 25., 84., 96., 195., 53., 217., 172., 131., 214.,
         59., 70., 220., 268., 152., 47., 74., 295., 101., 151., 127.,
        237., 225., 81., 151., 107., 64., 138., 185., 265., 101., 137.,
        143., 141., 79., 292., 178., 91., 116., 86., 122., 72., 129.,
        142., 90., 158., 39., 196., 222., 277., 99., 196., 202., 155.,
         77., 191., 70., 73., 49., 65., 263., 248., 296., 214., 185.,
         78., 93., 252., 150., 77., 208., 77., 108., 160., 53., 220.,
        154., 259., 90., 246., 124., 67., 72., 257., 262., 275., 177.,
         71., 47., 187., 125., 78., 51., 258., 215., 303., 243., 91.,
        150., 310., 153., 346., 63., 89., 50., 39., 103., 308., 116.,
        145., 74., 45., 115., 264., 87., 202., 127., 182., 241., 66.,
         94., 283., 64., 102., 200., 265., 94., 230., 181., 156., 233.,
         60., 219., 80., 68., 332., 248., 84., 200., 55., 85., 89.,
         31., 129., 83., 275., 65., 198., 236., 253., 124., 44., 172.,
        114., 142., 109., 180., 144., 163., 147., 97., 220., 190., 109.,
        191., 122., 230., 242., 248., 249., 192., 131., 237., 78., 135.,
        244., 199., 270., 164., 72., 96., 306., 91., 214., 95., 216.,
        263., 178., 113., 200., 139., 139., 88., 148., 88., 243., 71.,
         77., 109., 272., 60., 54., 221., 90., 311., 281., 182., 321.,
         58., 262., 206., 233., 242., 123., 167., 63., 197., 71., 168.,
        140., 217., 121., 235., 245., 40., 52., 104., 132., 88., 69.,
        219., 72., 201., 110., 51., 277., 63., 118., 69., 273., 258.,
         43., 198., 242., 232., 175., 93., 168., 275., 293., 281., 72.,
        140., 189., 181., 209., 136., 261., 113., 131., 174., 257., 55.,
         84., 42., 146., 212., 233., 91., 111., 152., 120., 67., 310.,
         94., 183., 66., 173., 72., 49., 64., 48., 178., 104., 132.,
        220., 57.]),
 'frame': None,
 'DESCR': '.. _diabetes_dataset:\n\nDiabetes dataset\n----------------\n\nTen basel
ine variables, age, sex, body mass index, average blood\npressure, and six blood se
rum measurements were obtained for each of n =\n442 diabetes patients, as well as t
he response of interest, a\nquantitative measure of disease progression one year af
ter baseline.\n\n**Data Set Characteristics:**\n\n :Number of Instances: 442\n\n
:Number of Attributes: First 10 columns are numeric predictive values\n\n :Target:
Column 11 is a quantitative measure of disease progression one year after baseline
\n\n :Attribute Information:\n   - age   age in years\n   - sex\n   - b
mi   body mass index\n   - bp   average blood pressure\n   - s1   tc,
total serum cholesterol\n   - s2   ldl, low-density lipoproteins\n   - s3
hdl, high-density lipoproteins\n   - s4   tch, total cholesterol / HDL\n
- s5   ltg, possibly log of serum triglycerides level\n   - s6   glu, bloo
d sugar level\n\nNote: Each of these 10 feature variables have been mean centered a

```
nd scaled by the standard deviation times `n_samples` (i.e. the sum of squares of e
ach column totals 1).\n\nSource URL:\nhttps://www4.stat.ncsu.edu/~boos/var.select/d
iabetes.html\n\nFor more information see:\nBradley Efron, Trevor Hastie, Iain Johns
tone and Robert Tibshirani (2004) "Least Angle Regression," Annals of Statistics (w
ith discussion), 407-499.\n(https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle
_2002.pdf)',
 'feature_names': ['age',
  'sex',
  'bmi',
  'bp',
  's1',
  's2',
  's3',
  's4',
  's5',
  's6'],
 'data_filename': 'diabetes_data.csv.gz',
 'target_filename': 'diabetes_target.csv.gz',
 'data_module': 'sklearn.datasets.data'}
```

In [4]:
```
print(diabetes.keys())
#print(diabetes.DESCR)
```

```
dict_keys(['data', 'target', 'frame', 'DESCR', 'feature_names', 'data_filename', 't
arget_filename', 'data_module'])
```

independent and dependent variables

In [5]:
```
diabetes1 = diabetes.data[:, np.newaxis,2] # data is in columns for np.newaxis
#display(diabetes1) # taking as x or independent variable
```

In [6]:
```
diabetes1train = diabetes1[:-30] # 1st 30
diabetes1test = diabetes1[-20:] # last 20
#display(diabetes1train, diabetes1test)
```

In [7]:
```
diabetes2 = diabetes.target
#display(diabetes2)  # taking as y or dependent variable
```

In [8]:
```
diabetes2train = diabetes1[:-30] # 1st 30
diabetes2test = diabetes1[-20:] # last 20
#display(diabetes1train, diabetes1test)
```

training and testing and finding mean squared error

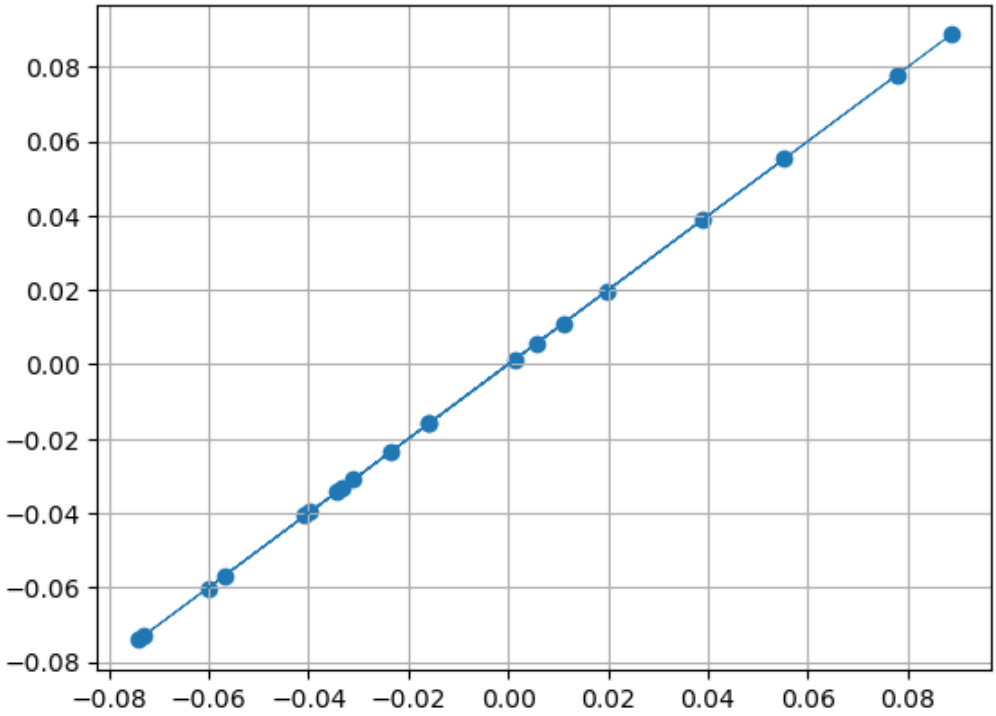
In [9]:
```
model12 = linear_model.LinearRegression()
model12.fit(diabetes1train, diabetes2train) # (x,y) for training
diabetes2predict = model12.predict(diabetes1test) # (x,y) for testing

err = mean_squared_error(diabetes2test, diabetes2predict)
print('mean squared error is', err)
print('intercept =', model12.intercept_, ', wights =', model12.coef_)
```

```
mean squared error is 2.830308278135871e-34
intercept = [4.06575815e-20] , wights = [[1.]]
```

press `ctrl` and click on a function to see the codes on which the function is written.

In [10]:
```
plt.scatter(diabetes1test, diabetes2test)
plt.plot(diabetes1test, diabetes2predict, lw=0.5)
plt.grid()
plt.show()
```



#11 How does linear regression work ?

variables

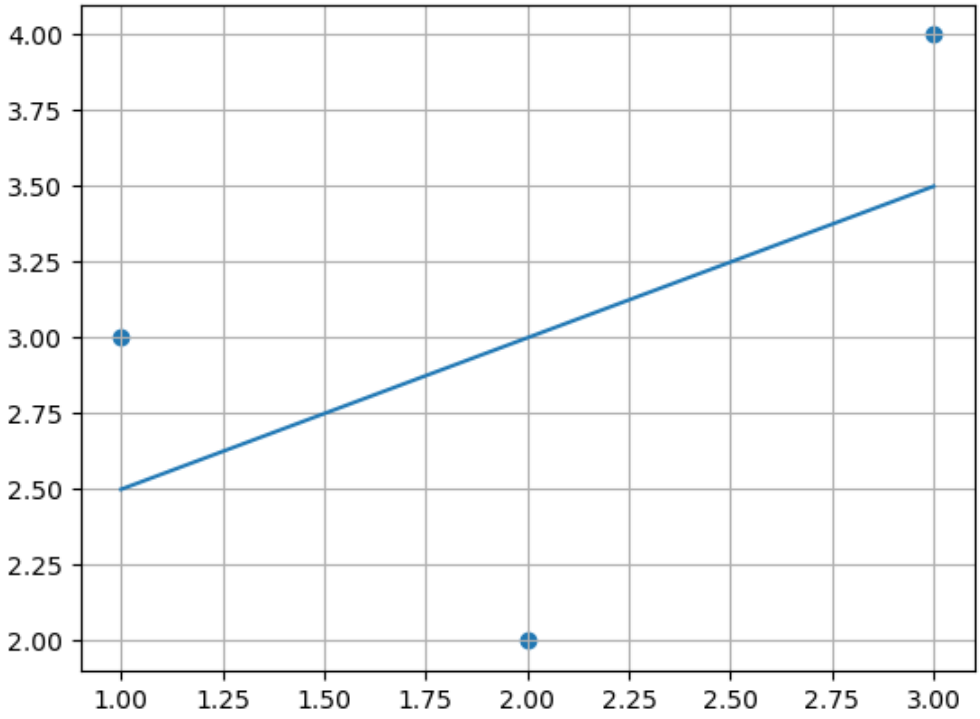
In [11]:
```
x1 = np.array([[1],[2],[3]]) # column
y1 = np.array([3,2,4]) # row
```

modelling

In [12]:
```
model1 = linear_model.LinearRegression()
model1.fit(x1, y1)
y1line = model1.predict(x1)
err1 = mean_squared_error(y1, y1line)
print('sum of squared error =', err1)
print('intercept =', model1.intercept_, ', weight =', model1.coef_)

plt.scatter(x1, y1)
plt.plot(x1, y1line)
plt.grid()
plt.show()
```

```
sum of squared error = 0.5000000000000001
intercept = 2.0 , weight = [0.5]
```



#15 K Nearest Neighbor Classification in python

In [13]:
```
from sklearn import datasets
from sklearn.neighbors import KNeighborsClassifier

iris = datasets.load_iris()
print(iris.DESCR)
```

```
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

.. topic:: References

   - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
     Structure and Classification Rule for Recognition in Partially Exposed
     Environments".  IEEE Transactions on Pattern Analysis and Machine
     Intelligence, Vol. PAMI-2, No. 1, 67-71.
   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
     on Information Theory, May 1972, 431-433.
   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
     conceptual clustering system finds 3 classes in the data.
   - Many, many more ...
```

features and labels

In [14]:
```
features = iris.data
labels = iris.target
# display(features, labels)
print(features.shape, labels.shape)
```

```
(150, 4) (150,)
```

Training the classifier and predictions

In [15]:
```
clf = KNeighborsClassifier()
clf.fit(features, labels)

preds = clf.predict([[10.3, 1.2, 14.7, 17.1]])
print(preds)
```

```
[2]

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:22
8: FutureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the d
efault behavior of `mode` typically preserves the axis it acts along. In SciPy 1.1
1.0, this behavior will change: the default value of `keepdims` will become False,
the `axis` over which the statistic is taken will be eliminated, and the value None
will no longer be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

In [ ]:
```
```