Statistics and Machine Learning on HackerRank

Outline

Statistics computation with formula

- **Pearson correlation coefficient PCC, r_{xy}**
- **Confidence interval, standard error**
- **std and variance**
- **generate polynomial feature matrix (for nonlinear problem)**
- **Slope of simple linear regression**
- **Intercept of simple linear regression**
- **Bivariate distribution**

ML

- **text multi-class classification**
- **Feature selection: most correlated / important feature**
- **multi linear regression; randomforestregressor to predict future values**
- **interpolate with polynomial, order=2 for missing value in the middle**
- **interpolate value with cubic for missing value the middle**
- Time series forecasting
- **sklearn tf-idf + cosine similarity of a pair**

Coding

- **scikit pipeline**
- Train/test data build with **numpy nd-array**
- Read from sys input, file input, json file input
- python: **map, zip**, **enumerate**, strip, split, etc.

Easy - 4 questions

Polynomial Regression: Office Prices
EasyMax Score: 10Success Rate: 73.72%
In this problem, we will help Charlie estimate the per-square-foot prices of Office-space.

Laptop Battery Life
EasyMax Score: 10Success Rate: 88.33%

Stock Predictions
EasyMax Score: 100Success Rate: 35.42%

Basic Statistics Warmup

Medium - 11 questions

Correlation and Regression Lines - A Quick Recap #1
MediumMax Score: 5Success Rate: 73.52%
Computing Karl Pearson’s coefficient of correlation.

Correlation and Regression Lines - A Quick Recap #2
MediumMax Score: 5Success Rate: 85.79%

Correlation and Regression Lines - A quick recap #3
MediumMax Score: 5Success Rate: 89.40%

Correlation and Regression Lines - A Quick Recap #4
MediumMax Score: 5Success Rate: 84.49%

Correlation and Regression Lines - A Quick Recap #5
MediumMax Score: 5Success Rate: 85.24%

Markov's Snakes And Ladders
MediumMax Score: 40Success Rate: 69.05%

Document Classification
MediumMax Score: 100Success Rate: 9.81%

The Best Aptitude Test
MediumMax Score: 15Success Rate: 70.47%

Stack Exchange Question Classifier
MediumMax Score: 100Success Rate: 82.31%

Time Series: Predict the Web Traffic
MediumMax Score: 100Success Rate: 26.11%

Forecasting passenger traffic

Hard - 10 questions

Day 6: Multiple Linear Regression: Predicting House Prices
ExpertMax Score: 10Success Rate: 84.06%
Help Charlie predict housing prices.

Day 5: Computing the Correlation
ExpertMax Score: 20Success Rate: 94.34%

Quora Answer Classifier
HardMax Score: 100Success Rate: 43.27%

Craigslist Post Classifier: Identify the Category
HardMax Score: 100Success Rate: 41.60%

Matching Questions with their Answers
HardMax Score: 50Success Rate: 78.59%

Shakespeare Cuts It Short
ExpertMax Score: 100Success Rate: 17.46%

Missing Stock Prices
HardMax Score: 100Success Rate: 28.43%

Dota 2 Game Prediction
ExpertMax Score: 100Success Rate: 28.54%

Predict the Missing Grade
ExpertMax Score: 100Success Rate: 77.83%

Day 7: Temperature Predictions

HardMax Score: 100Success Rate: 63.40%

Statistics symbols

Symbol	Symbol Name	Meaning / definition
σ^2	variance	variance of population values
std(X)	standard deviation	standard deviation of random variable X
σ_X	standard deviation	standard deviation value of random variable X

μ	population mean	mean of population values	$\mu = 10$
σ^2	variance	variance of population values	$\sigma^2 = 4$
$\tilde{x}$	median	middle value of random variable x	$\tilde{x} = 5$
$\bar{x}$	sample mean	average / arithmetic mean	$\bar{x} = (2+5+9) / 3 = 5.333$
σ_X	standard deviation	standard deviation value of random variable X	$\sigma_X = 2$

Basic Statistics Warmup

confidence interval

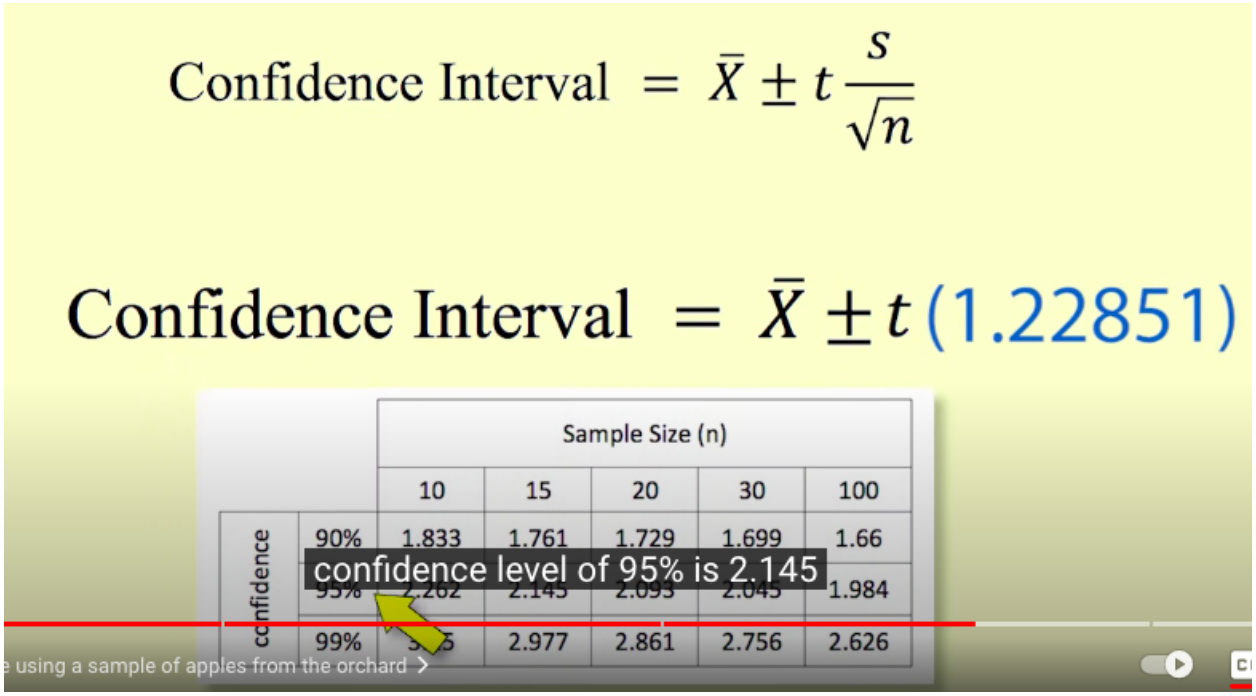
instead use mean, a single number, confidence interval, usually 95%, is a better way to describe the range of mean.

sample error

standard error: $\frac{s}{\sqrt{n}}$

the width are related to

- variation: more variated the sample, larger the width
- sample size: small sample has higher variance



standard deviation

is the average distance from the mean value of all values in a set of data.

$$s = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \overline{x})^2}{N - 1}}$$

s = sample standard deviation

N = the number of observations

x_i = the observed values of a sample item

$\overline{x}$ = the mean value of the observations

variance

The variance measures the average degree to which each point differs from the mean. While standard deviation is the square root of the variance, variance is the average of all data points within a group.

$$S^2 = \frac{\sum (x_i - \bar{x})^2}{n - 1}$$

S^2 = sample variance

x_i = the value of the one observation

$\bar{x}$ = the mean value of all observations

n = the number of observations

Sample Input

```
10
64630 11735 14216 99233 14470 4978 734
```

Sample Output

```
43900.6
44627.5
4978
30466.9
25017.0 62784.2
```

```
N=input()
arr=[int(i) for i in input().split()]

##mean
mean=sum(arr)/float(N)
print mean


##median
arrS=sorted(arr)
if N%2==1: #if N is odd
    median=arrS[N/2]
else:#n is even
    median=(arrS[(N-1)/2]+arrS[N/2])/2.

print median

##mode
best=0
```

```
mode=-1
cur=0
curNum=-1
for i in arrS:
    if i!=curNum:
        cur=1
        curNum=i
    else:
        cur+=1
    if cur>best:
        best=cur
        mode=curNum
print mode

##standard deviation

SD= (sum(((i-mean)**2 for i in arr)) /float(N))**.5
print "%.1f"%SD


## 95% confidence interval

print "%.1f"%(-1.96*SD/N**.5 +mean), "%.1f"%(1.96*SD/N**.5
+mean)
```

Polynomial Regression: Office Prices

Polynomial Regression is a form of Linear regression known as **a special case of Multiple linear regression** which estimates the relationship as an **nth degree polynomial**. Polynomial Regression is **sensitive to outliers** so the presence of one or two outliers can also badly affect the performance.

Sample Input

```
2 100
0.44 0.68 511.14
0.99 0.23 717.1
0.84 0.29 607.91
0.28 0.45 270.4
0.07 0.83 289.88
0.66 0.8 830.85
0.73 0.92 1038.09
0.57 0.43 455.19
```

Sample Output

```
180.38
1312.07
440.13
343.72
```

construct numpy 2d array

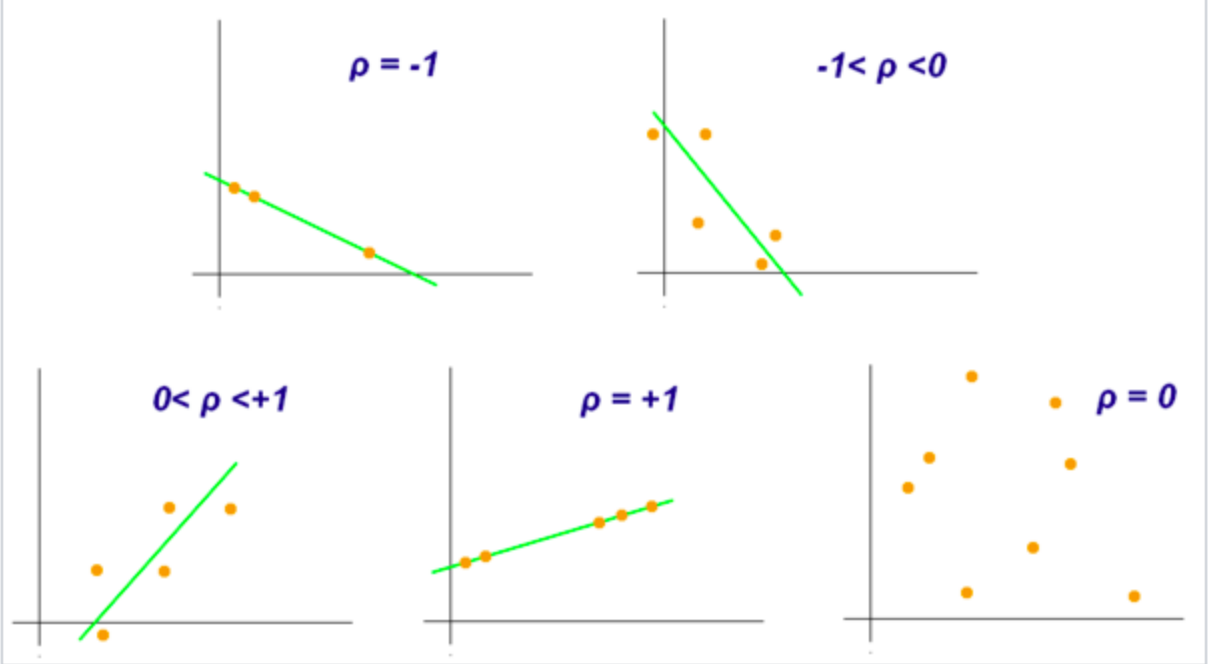
```
import numpy as np
from sklearn import linear_model
from sklearn.preprocessing import PolynomialFeatures

params = [int(s) for s in input().strip().split()]
num_features = params[0]
num_samples = params[1]

# construct numpy 2d array X
X = np.zeros((num_samples, num_features))
y = np.zeros(num_samples)
```

```
for i in range(num_samples):
    row = [float(s) for s in input().strip().split()]
    X[i, :] = row[:-1]
    y[i] = row[-1]

poly = PolynomialFeatures(3)
X = poly.fit_transform(X)

clf = linear_model.RidgeCV()
clf.fit(X, y)
num_test_samples = int(input())
X_test = np.zeros((num_test_samples, num_features))
for i in range(num_test_samples):
    X_test[i, :] = [float(s) for s in
input().strip().split()]
X_test = poly.fit_transform(X_test)
print("\n".join(str(y) for y in clf.predict(X_test)))
```

Correlation and Regression Lines - A Quick Recap #1

Pearson's correlation coefficient (PCC)

is a measure of linear correlation between two sets of data
the result always has a value between −1 and 1.



Examples of scatter diagrams with different values of correlation coefficient (ρ)

above. Given paired data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ consisting of n pairs, r_{xy} is defined as:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where:

- n is sample size
- x_i, y_i are the individual sample points indexed with i
- $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ (the sample mean); and analogously for $\bar{y}$

An equivalent expression gives the formula for {\displaystyle r_{xy}}r_{xy} as the mean of the products of the standard scores as follows:

$$r_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{s_x}\right)\left(\frac{y_i - \bar{y}}{s_y}\right)$$

where:

- $n, x_i, y_i, \bar{x}, \bar{y}$ are defined as above, and s_x, s_y are defined below
- $\left(\frac{x_i - \bar{x}}{s_x}\right)$ is the standard score (and analogously for the standard score of y)

Alternative formulae for r_{xy} are also available. For example. one can use the following formula for r_{xy}:

$$r_{xy} = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{(n-1)s_x s_y}$$

where:

- $n, x_i, y_i, \bar{x}, \bar{y}$ are defined as above and:
- $s_x = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$ (the sample standard deviation); and analogously for s_y

```
import math

def P(A,B):
    n = float(len(A))
    muA = sum(A)/n
    muB = sum(B)/n
    diffA = map(lambda x: x - muA, A)
    diffB = map(lambda x: x - muB, B)
    stdA = math.sqrt((1/(n-1))* sum([d*d for d in diffA]))
    stdB = math.sqrt((1/(n-1))* sum([d*d for d in diffB]))
    return (sum([A[i]*B[i] for i in range(int(n))]) - n *
muA * muB) / ((n-1) * stdA * stdB)

A = [15,12,8,8,7,7,7,6,5,3]
B = [10,25,17,11,13,17,20,13,9,15]
print('%.3f' % (P(A,B)))
```

python map() function

The `map()` function executes a specified function for **each item in an iterable**. **The item is sent to the function as a parameter.**

```
def myfunc(a, b):
  return a + b

x = map(myfunc, ('apple', 'banana', 'cherry'), ('orange', 'lemon',
'pineapple'))

print(x)

#convert the map into a list, for readability:
print(list(x))
```

```
<map object at 0x151c0df9c2b0>
['appleorange', 'bananalemon', 'cherrypineapple']
```

Correlation and Regression Lines - A Quick Recap #2 and Correlation and Regression Lines - A quick recap #3

slope of simple linear regression

$$y = \alpha + \beta x,$$

which describes a line with slope β and y-intercept α.

$$\widehat{\alpha} = \bar{y} - (\widehat{\beta}\,\bar{x}),$$

$$\widehat{\beta} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

```
def mean(L):
    return sum(L) / len(L)

X = [15, 12, 8, 8, 7, 7, 7, 6, 5, 3]
Y = [10, 25, 17, 11, 13, 17, 20, 13, 9, 15]
mean_X = mean(X)
mean_Y = mean(Y)

slope = sum([(x - mean_X) * (y - mean_Y) for x, y in zip(X, Y)]) /
sum([(x - mean_X) ** 2 for x in X])
print("%.3f" % slope)
```

intercept of simple linear regression

```
def mean(L):
    return sum(L) / len(L)

X = [15, 12, 8, 8, 7, 7, 7, 6, 5, 3]
Y = [10, 25, 17, 11, 13, 17, 20, 13, 9, 15]
mean_X = mean(X)
mean_Y = mean(Y)

beta = sum([(x - mean_X) * (y - mean_Y) for x, y in zip(X, Y)]) /
sum([(x - mean_X) ** 2 for x in X])
alpha = mean_Y - beta * mean_X

x=10
y = alpha + beta * x
```

```
print("%.1f" % y)
```

Correlation and Regression Lines - A Quick Recap #4

bivariate distribution

双变量分布

The two regression lines of a bivariate distribution are:
4x– 5y + 33 = 0 (line of y on x; y is the dependent variable for 4x – 5y + 33 = 0)
20x – 9y – 107 = 0 (line of x on y; x is the dependent variable).
Estimate the value of when y = 7. Compute the correct answer to one decimal place.
when y = 7, use the line of x on y

In regression analysis, there are usually two regression lines to show the average relationship between X and Y variables. It means that if there are two variables X and Y, then one line represents regression of Y upon x and the other shows the regression of x upon Y (Fig. 35.2).

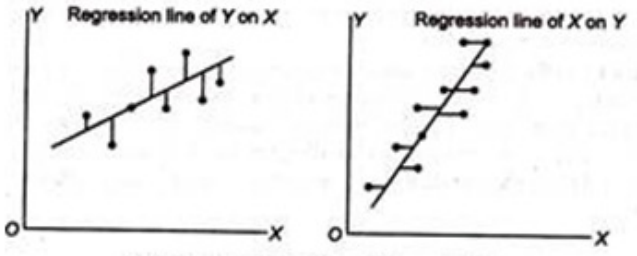


Fig. 35.2 Regression lines of two variables.

Correlation and Regression Lines - A Quick Recap #5

replace with another question:

variance of y

The two lines of regressions are $x+2y-5=0$ and $2x+3y-8=0$ and the variance of x is 12. Find the variance of y and the coefficient of correlation.

The given equation of the lines of regression are

$x+2y-5=0$.....(i)

and $2x+3y-8=0$.....(ii)

slope 1= -½=-0.5
slope 2=-3/2=-1.5
variance of x =12
variance of y = slope1/slope2 * variance of x = 4

variance of x = (standard deviation of x) ** 2

Document Classification

You have been given a stack of documents that have already been processed and some that have not. Your task is to classify these documents into one of eight categories: [1,2,3,...8]. You look at the specifications on what a document must contain and are baffled by the jargon. However, you notice that you already have a large amount of documents which have already been correctly categorize (training data). You decide to use Machine Learning on this data in order to categorize the uncategorized documents.

Sample Input

3

This is a document

this is another document

documents are seperated by newlines

Sample Output

1

4

8

```
import sys
from sklearn.feature_extraction import text
```

```
from sklearn import pipeline
from sklearn import linear_model
import numpy


def make_model():
    clf = pipeline.Pipeline([
        ('vectorization',
         text.TfidfVectorizer(stop_words='english',ngram_range=(1, 3),

min_df=4,max_df=0.95,strip_accents='ascii',
lowercase=True)), # Ignore terms with higher frequency than 0.95, low
frequency than twice. ngram: 1-3. convert to lowercase before
tokenizing.
        ('clf',
         linear_model.SGDClassifier(class_weight='balanced',tol=0.001))
# balanced means if the class shows less frequent in the training data,
increase the weight of this class in prediction. stopping criteria when
(loss > best_loss - tol).
    ])
    return clf


def run():
    known = [('Business means risk!', 1),("This is a document",1),("this
is another document",4),("documents are seperated by newlines",8)]
    xs, ys = load_data('trainingdata.txt')
    mdl = make_model()
    mdl.fit(xs, ys)
    txs = list(line for line in sys.stdin)[1:]
    for y, x in zip(mdl.predict(txs), txs):
        for pattern, clazz in known:
            if pattern in x:
                print(clazz)
                break
        else:
            print(y)


def load_data(filename):
    with open(filename, 'r') as data_file:
```

```
        sz = int(data_file.readline())
        xs = numpy.zeros(sz, dtype=object)
        ys = numpy.zeros(sz, dtype=int)
        for i, line in enumerate(data_file):
            idx = line.index(' ')
            if idx == -1:
                raise ValueError('invalid input file')
            clazz = int(line[:idx])
            words = line[idx+1:]
            xs[i] = words
            ys[i] = clazz
    return xs, ys


if __name__ == '__main__':
    run()
```

Stack Exchange Question Classifier

Input Format

The first line will be an integer N. N lines follow each line being a valid JSON object. The following fields of raw data are given in json

- question (string) : The text in the title of the question.
- excerpt (string) : Excerpt of the question body.
- topic (string) : The topic under which the question was posted.

The input for the program has all the fields but topic which you have to predict as the answer.

Constraints

1 <= N <= 22000

topic is of ascii format

question is of UTF-8 format

excerpt is of UTF-8 format

Output Format

For each question that is given as a JSON object, output the topic of the question as predicted by your model separated by newlines.

training data


```
20219
{"topic":"electronics","question":"What is the effective differencial effective of this circuit","excerpt":"I'm trying to
capacitance of this circuit (see diagram: http://i.stack.imgur.com/BS85b.png).  \n\nWhat is the effective capacitance of t
{"topic":"electronics","question":"Heat sensor with fan cooling","excerpt":"Can I know which component senses heat or acts
the given diagram, it is said that the 4148 diode acts as the sensor. But basically it is a zener diode and ...\r\n
{"topic":"electronics","question":"Outlet Installation--more wires than my new outlet can use [on hold]","excerpt":"I am r
USB outlet (TR7745).  The new outlet has 3 wires coming out of it--a black, a white, and a green.  Each one needs to be at
{"topic":"electronics","question":"Buck Converter Operation Question","excerpt":"i have been reading about the buck conver
online resources like here.\n\n\n\nIn the above circuit, as I understand, when switch closes, current starts to increase .
{"topic":"electronics","question":"Urgent help in area of ASIC design, verification, SoC [on hold]","excerpt":"I need help
need some ideas related to the field of ASIC Design/ verification or something related to SoC's, FPGA and or combination.
{"topic":"electronics","question":"Slowly supplying power to a very high load","excerpt":"Is it possible to supply power t
\n\nThis question particularly regards MOSFETs.\nI know that a MOSFET must switch fast so that a little power would develo
{"topic":"electronics","question":"I have a 110 VAC solenoid and want to know what kind of circuit i should build to contr
```

Sample Input

```
12345
json_object
json_object
json_object
.
.
.
json_object
```

Sample Output

```
electronics
security
photo
.
.
.
mathematica
```

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
import json
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import SGDClassifier

def cleanText(string):
    string = string.strip() # Remove space at the beginning and ending
    string = " ".join(string.split()) # Replace continuous multiple
white spaces with single space
    string = string.lower()
    string = string.replace('\n', ' ')
    string = string.replace('\t', ' ')
```

```
    string = string.replace('\r', ' ')
    string = string.replace('-', ' ')
    string = string.replace('/', ' ')
    string = string.replace('*', ' ')
    string = string.replace('_', ' ')
    string = string.replace('(', ' ')
    string = string.replace(')', ' ')
    string = string.replace('!', ' ')
    string = string.replace('|', ' ')
    string = string.replace(',', ' ')
    string = string.replace(';', ' ')
    string = string.replace(':', ' ')
    string = string.replace('+', ' ')
    string = string.replace('@', ' ')
    return string


f=open('training.json', 'r')
N=int(f.readline())
i=0
train_ans=[]
train=[]
topic_list=[]
for line in f:
    j = json.loads(line)
    topic = (j['topic'])
    if topic in topic_list:
        topicN = topic_list.index(topic)
    else:
        topic_list.append(topic)
        topicN = topic_list.index(topic)
    train_ans.append(topicN)
    train_ans.append(topicN)
    train.append(cleanText(j['question']))
    train.append(cleanText(j['excerpt']))
test=[]
N=int(input())
for i in range(N):
    j = json.loads(input())
    test.append(cleanText(j['question']+ " " + j['excerpt']))
```

```
v =
TfidfVectorizer(ngram_range=(1,3),stop_words="english",min_df=2,max_df=0
.95)
X = v.fit_transform(train)
Y = train_ans
X_test = v.transform(test)

clf = SGDClassifier(alpha=.00002, penalty="l2", class_weight='balanced')
clf.fit(X, Y)

predictions = clf.predict(X_test)
for i in predictions:
    print (topic_list[i])
```

The Best Aptitude Test

A college admits students on the basis of their scores in a series of 5 aptitude tests. The academic performance of the admitted students is then tracked, and when they complete the first year of their academic program in college, their Grade Point Average (GPA) is recorded. A student's GPA is an indicator of his/her academic performance. Higher the GPA, the better. This exercise is conducted on one batch of students. Absolute GPA is used for this problem.

The next year, the college realizes that it does not have the necessary resources to conduct all 5 aptitude tests, it decides to conduct only 1 of the 5 tests. Which of the aptitude tests would you recommend them to conduct for the next round of admissions?

You are given the recorded data. Identify the aptitude test which has clearly the best predictive value in determining the relative academic standing of the students after they enter the college.

Input Format

The 1st line in the file is an integer T, T testcases follow.

Each testcase has 7 lines.

The 1st line of every testcase contains the number of admitted students, N.

The 2nd line of every testcase has N decimal numbers (upto 2 decimal places) separated by a single space which are the Grade Point Averages (GPAs) of the admitted students at the end of the 1st academic year. The ith score, is the GPA of the ith student at the end of the 1st year.

This next 5 lines of every testcase has N decimal numbers (upto 2 decimal places) separated by a single space in each line, which is the performance of the students in the 5 aptitude tests conducted for the entrance exam. The ith integer in each line, is what the ith student scored in that aptitude test.

Constraints

1 <= T <= 10

4 <= N <= 100

0.0 <= k <= 10.0, where k is the GPA of every student.

0.0 <= s <= 100.0, where s is the score of every student in any of the 5 aptitude tests.

Output Format

T integers, each on a new line.

For each test case, output the Aptitude Test (1-5) which appears to be the best predictor of the relative academic standing of the students after they enter the college.

Sample Input

```
1
5
7.5 7.7 7.9 8.1 8.3
10 30 20 40 50
11 9 5 19 29
21 9 15 19 39
91 9 75 19 89
81 99 55 59 89
```

Example Output

```
1
```

Feature selection

REF: Recursive feature elimination
RFECV: performs RFE in a cross-validation loop to find the optimal number of features.

```
import numpy as np
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression

num_cases = int(input())

for _ in range(num_cases):
    estimator = LinearRegression()
    selector = RFE(estimator, n_features_to_select=1, step=1, verbose=0)

    N = int(input())
    y = np.array(input().split(), float)
    X = np.array([input().split() for _ in range(5)], float)

    # transform the numpy ndarray
```

```
    # [[1.,2.],[3.,4.]] -> [[ 1.,  3.],[ 2.,  4.]]
    # [1.,2.,3.,4.] -> [1.,2.,3.,4.]
    selector.fit(X.T, y.T)

    print(np.argmin(selector.ranking_) + 1)
```

Temperature Predictions

Sample Input

```
20
yyyy   month   tmax    tmin
1908   January 5.0 -1.4
1908   February    7.3 1.9
1908   March   6.2 0.3
1908   April   Missing_1   2.1
1908   May Missing_2   7.7
1908   June    17.7    8.7
1908   July    Missing_3   11.0
1908   August  17.5    9.7
1908   September   16.3    8.4
```

Sample Output

The four missing values ($Missing\_1$, $Missing\_2$, $Missing\_3$, and $Missing\_4$) are:

```
8.6
15.8
18.9
0.0
```

interpolate missing values in the middle

```
import pandas as pd

N = int(input())
```

```
columns = input().split()

df = pd.DataFrame(columns=columns)

for i in range(N):
    df.loc[i, :] = input().split()

# Trimmed Maximum
df.tmax = pd.to_numeric(df.tmax, errors='coerce')
df.tmin = pd.to_numeric(df.tmin, errors='coerce')

tmax_missing = df.tmax.isna()
tmin_missing = df.tmin.isna()

tmax_interp = df.tmax.interpolate(method='cubic')
tmin_interp = df.tmin.interpolate(method='cubic')

for i in range(df.shape[0]):
    if tmax_missing[i]:
        print(round(tmax_interp[i],1))
    if tmin_missing[i]:
        print(round(tmin_interp[i],1))
```

Multiple Linear Regression: Predicting House Prices

Sample Input

```
2 7
0.18 0.89 109.85
1.0 0.26 155.72
0.92 0.11 137.66
0.07 0.37 76.17
0.85 0.16 139.75
0.99 0.41 162.6
0.87 0.47 151.77
4
0.49 0.18
0.57 0.83
0.56 0.64
0.76 0.18
```

Sample Output

```
105.22
142.68
132.94
129.71
```

linear_model.LinearRegression()

RandomForestRegressor()

```
from sklearn import linear_model
from sklearn.ensemble import RandomForestRegressor
import numpy as np
```

```
fn = input().split()
fn = [int(i) for i in fn]
f = fn[0]
n = fn[1]
X = []
y = []
test = []
for i in range(n):
    xy = [float(t) for t in input().split()]
    X.append(xy[:-1])
    y.append(xy[f])
X = np.array(X)
ntest = int(input())
for i in range(ntest):
    test.append([float(t) for t in input().split()])

lm = linear_model.LinearRegression()
#lm = RandomForestRegressor(random_state=42)
lm.fit(X,y)
predictions = lm.predict(test)
for i in range(ntest):
    print(round(predictions[i],2))
```

Missing Stock Prices

Sample Input

```
250
1/3/2012 16:00:00  Missing_1
1/4/2012 16:00:00  27.47
1/5/2012 16:00:00  27.728
1/6/2012 16:00:00  28.19
1/9/2012 16:00:00  28.1
1/10/2012 16:00:00  28.15
....
....
....
12/13/2012 16:00:00 27.52
12/14/2012 16:00:00 Missing_19
12/17/2012 16:00:00 27.215
12/18/2012 16:00:00 27.63
12/19/2012 16:00:00 27.73
12/20/2012 16:00:00 Missing_20
12/21/2012 16:00:00 27.49
12/24/2012 13:00:00 27.25
12/26/2012 16:00:00 27.2
12/27/2012 16:00:00 27.09
12/28/2012 16:00:00 26.9
12/31/2012 16:00:00 26.77
```

Sample Output

```
26.96
31.98
32.69
32.41
32.32
30.5
29.18
30.8
30.46
```

interpolate with polynomial, order=2 for missing value in the middle of a list

```
import pandas as pd
import numpy as np
n = int(input())
data = []
for _ in range(n):
    data.append(input().split())

data = pd.DataFrame(data)
data.columns = ['date', 'time', 'price']
data.price = pd.to_numeric(data.price, errors='coerce')
del data['date']
del data['time']

missingidxlist = data.loc[pd.isna(data["price"]), :].index
data.interpolate(method='polynomial',order=2,inplace=True)

for i in missingidxlist:
    print(round(data.at[i, 'price'],2))
```

Matching Questions with their Answers

Input Format
The first line contains a short paragraph of text from Wikipedia.
Line Number 2 to 6 contain a group of 5 simple questions, expecting factual answers, which can be directly answered by reading the provided paragraph.
Line Number 7 contains the five answers which have been jumbled up. The answers are grouped into one string and separated by semi-colons.

Sample Input

```
Zebras are several species of African equids (horse family) united by the
Which Zebras are endangered?
What is the aim of the Quagga Project?
Which animals are some of their closest relatives?
Which are the three species of zebras?
Which subgenus do the plains zebra and the mountain zebra belong to?
subgenus Hippotigris;the plains zebra, the Grévy's zebra and the mountai
```

Sample Output

```
Grévy's zebra and the mountain zebra
aims to breed zebras that are phenotypically similar to the quagga
horses and donkeys
the plains zebra, the Grévy's zebra and the mountain zebra
subgenus Hippotigris
```

sklearn cosine similarity

sklearn tfidf

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import pandas as pd
import numpy as np

corpus = []
corpus.append(input())
q = []
for i in range(5):
    q.append(input())

a = input().split(';')
# print(q)
# print(a)

q_a =[]
for i in range(5):
    for j in range(5):
        q_a.append(q[i] + ' ' + a[j])

# print(q_a)
for i in q_a:
    corpus.append(i)
print(len(corpus))
#print(corpus)

vectorizer = TfidfVectorizer()
trsfm = vectorizer.fit_transform(corpus)
#
print(pd.DataFrame(trsfm.toarray(),columns=vectorizer.get_feature_names(
)))
print(cosine_similarity(trsfm[0:1], trsfm).flatten())
```

----------END----------