

ЛАБОРАТОРНАЯ РАБОТА. ПРЕДСКАЗАНИЕ И РЕГРЕССИЯ. ОСНОВЫ

Цель. Изучить механизм регрессии и его использование (для предсказания).

Краткое теоретическое описание.

Регрессионная модель связывает наблюдаемые входы и выходы. Входы и выходы обычно задаются в форме массивов, например

```
X = [[6], [8], [10], [14], [18]]  
y = [[7], [9], [13], [17.5], [18]]
```

Затем нужно собственно создать модель

```
model = LinearRegression()  
model.fit(X, y)
```

Здесь создается линейная модель вида $y=AX+B$. Параметры модели A, B.

Теперь модель можно использовать для предсказания

```
print 'A 12" pizza should cost: $%.2f' % model.predict([12])[0]
```

Предсказание выполняем для входа $X=12$.

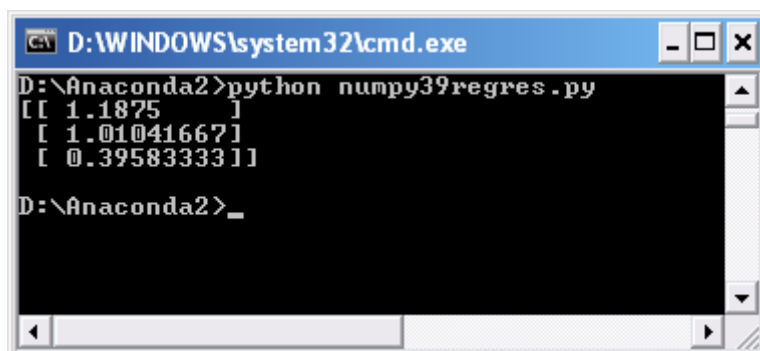
Полный текст скрипта

```
import numpy as np  
import matplotlib.pyplot as plt  
import pandas as pd  
import scipy.stats as stats  
import statsmodels.api as sm  
from scipy.optimize import fsolve  
from scipy.linalg import *  
from numpy import array  
from scipy.cluster.vq import vq, kmeans, whiten  
from sklearn import metrics  
from sklearn.metrics import pairwise_distances  
from sklearn import datasets
```

```
from sklearn import svm
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
# Training data
X = [[6], [8], [10], [14], [18]]
y = [[7], [9], [13], [17.5], [18]]
# Create and fit the model
model = LinearRegression()
model.fit(X, y)
print 'A 12" pizza should cost: $%.2f' % model.predict([12])[0]
```

Параметры модели можно определить с помощью функции `lstsq`. Пример скрипта

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.stats as stats
import statsmodels.api as sm
from scipy.optimize import fsolve
from scipy.linalg import *
from numpy import array
from scipy.cluster.vq import vq, kmeans, whiten
from sklearn import metrics
from sklearn.metrics import pairwise_distances
from sklearn import datasets
from sklearn import svm
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from numpy.linalg import lstsq
X = [[1, 6, 2], [1, 8, 1], [1, 10, 0], [1, 14, 2], [1, 18, 0]]
y = [[7], [9], [13], [17.5], [18]]
print lstsq(X, y)[0]
```



The screenshot shows a Windows command prompt window titled "D:\WINDOWS\system32\cmd.exe". The prompt is "D:\Anaconda2>". The user has entered the command "python numpy39regres.py". The output of the script is displayed on the next three lines: "[[1.1875", "[1.01041667]", and "[0.39583333]". The prompt is now "D:\Anaconda2>_".

+++++

Предсказание на основе модели

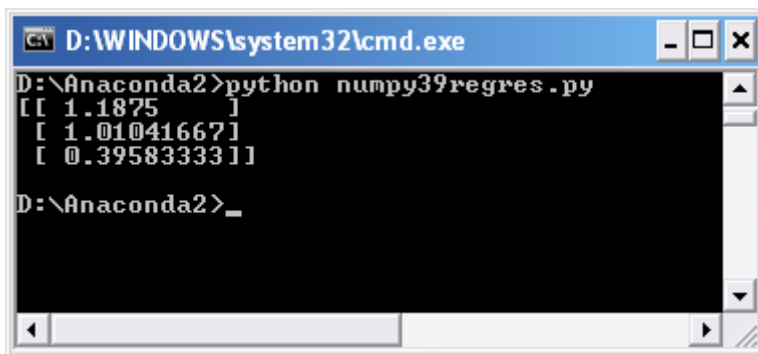
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.stats as stats
import statsmodels.api as sm
from scipy.optimize import fsolve
from scipy.linalg import *
from numpy import array
from scipy.cluster.vq import vq, kmeans, whiten
from sklearn import metrics
from sklearn.metrics import pairwise_distances
from sklearn import datasets
from sklearn import svm
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
# Training data
X = [[6], [8], [10], [14], [18]]
y = [[7], [9], [13], [17.5], [18]]
# Create and fit the model
model = LinearRegression()
model.fit(X, y)
print 'A 12" pizza should cost: $%.2f % model.predict([12])[0]
```

+++++

+

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.stats as stats
import statsmodels.api as sm
from scipy.optimize import fsolve
from scipy.linalg import *
from numpy import array
```

```
from scipy.cluster.vq import vq, kmeans, whiten
from sklearn import metrics
from sklearn.metrics import pairwise_distances
from sklearn import datasets
from sklearn import svm
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from numpy.linalg import lstsq
X = [[1, 6, 2], [1, 8, 1], [1, 10, 0], [1, 14, 2], [1, 18, 0]]
y = [[7], [9], [13], [17.5], [18]]
print lstsq(X, y)[0]
```



```
D:\WINDOWS\system32\cmd.exe
D:\Anaconda2>python numpy39regres.py
[[ 1.1875]
 [ 1.01041667]
 [ 0.39583333]]
D:\Anaconda2>_
```

ЗАДАНИЕ.

Отобразите два графика – первый для исходных данных (X, y) второй – для графика, соответствующего значениям Y, определяемым для тех же значений X на регрессионной модели. Для рисования графика можно использовать следующий скрипт

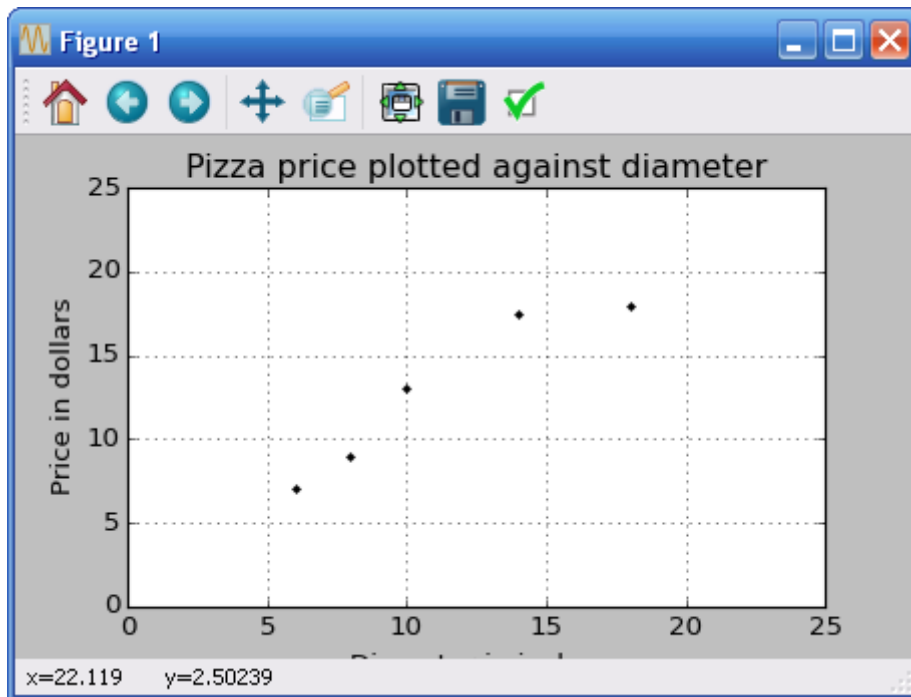
Вывод графика

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.stats as stats
import statsmodels.api as sm
from scipy.optimize import fsolve
from scipy.linalg import *
from numpy import array
from scipy.cluster.vq import vq, kmeans, whiten
```

```

from sklearn import metrics
from sklearn.metrics import pairwise_distances
from sklearn import datasets
from sklearn import svm
import matplotlib.pyplot as plt
X = [[6], [8], [10], [14], [18]]
y = [[7], [9], [13], [17.5], [18]]
plt.figure()
plt.title('Pizza price plotted against diameter')
plt.xlabel('Diameter in inches')
plt.ylabel('Price in dollars')
plt.plot(X, y, 'k.')
plt.axis([0, 25, 0, 25])
plt.grid(True)
plt.show()

```



+++++

Входные данные X могут быть многомерными, как показано в следующем скрипте

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.stats as stats
import statsmodels.api as sm
from scipy.optimize import fsolve
from scipy.linalg import *
from numpy import array
from scipy.cluster.vq import vq, kmeans, whiten
from sklearn import metrics
from sklearn.metrics import pairwise_distances
from sklearn import datasets
from sklearn import svm
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from numpy.linalg import lstsq
X = [[1, 6, 2], [1, 8, 1], [1, 10, 0], [1, 14, 2], [1, 18, 0]]
y = [[7], [9], [13], [17.5], [18]]
X_test = [[1,8, 2], [1,9, 0], [1,11, 2], [1,16, 2], [1,12, 0]]
y_test = [[11], [8.5], [15], [17], [11]]

print lstsq(X, y)[0]
model = LinearRegression()
model.fit(X, y)

predictions = model.predict(X_test)
for i, prediction in enumerate(predictions):
    print 'Predicted: %s, Target: %s' % (prediction, y[i])
print 'R-squared: %.2f' % model.score(X_test, y_test)
```

Этот скрипт, помимо прочего, выдает не только предсказанные, но и реальные значения:

```
D:\WINDOWS\system32\cmd.exe

D:\Anaconda2>python numpy40regres.py
[[ 1.1875 ]
 [ 1.01041667]
 [ 0.39583333]]
Predicted: [ 10.06250019], Target: [7]
Predicted: [ 10.28125019], Target: [9]
Predicted: [ 13.09375019], Target: [13]
Predicted: [ 18.14583353], Target: [17.5]
Predicted: [ 13.31250019], Target: [18]
R-squared: 0.69

D:\Anaconda2>_
```

ЗАДАНИЕ.

Программно определить максимальное по абсолютной величине отклонение предсказанного значения от реального. В хороших моделях оно должно быть не выше 12%.

+++++

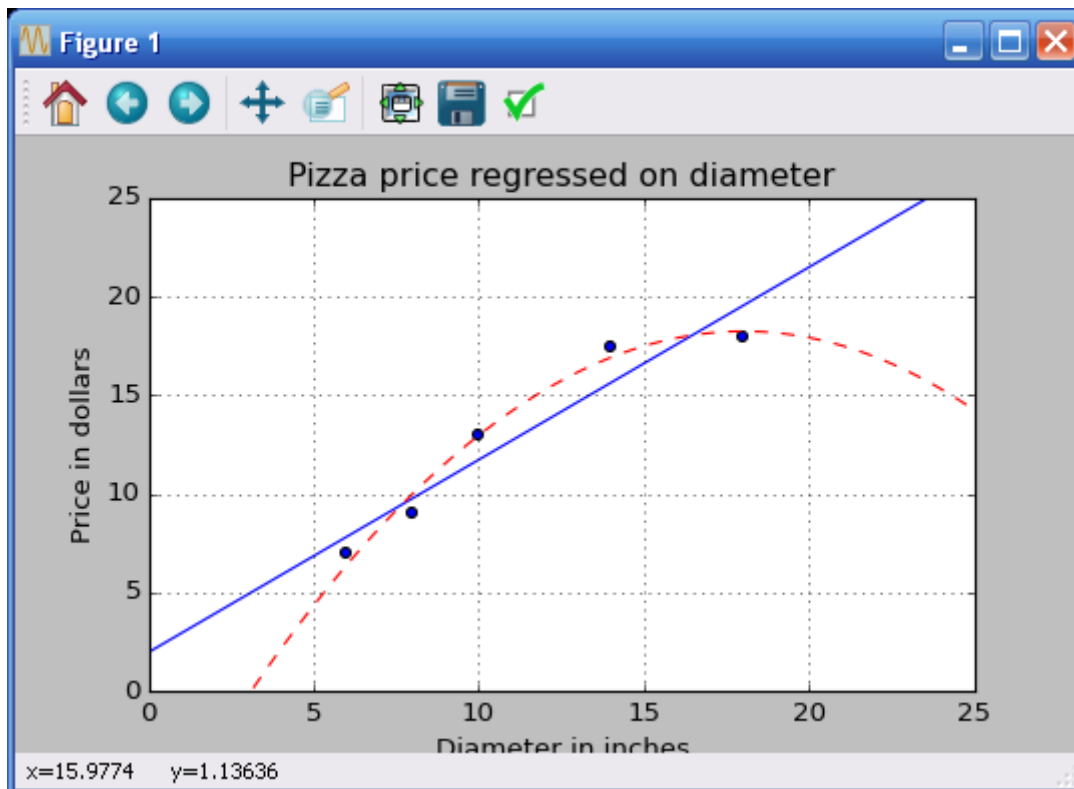
Полиномиальная квадратическая регрессия. Все то же, но вид регрессионной модели – полином второй степени. Пример

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.stats as stats
import statsmodels.api as sm
from scipy.optimize import fsolve
from scipy.linalg import *
from numpy import array
from scipy.cluster.vq import vq, kmeans, whiten
from sklearn import metrics
from sklearn.metrics import pairwise_distances
from sklearn import datasets
from sklearn import svm
```

```

import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from numpy.linalg import lstsq
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
X_train = [[6], [8], [10], [14], [18]]
y_train = [[7], [9], [13], [17.5], [18]]
X_test = [[6], [8], [11], [16]]
y_test = [[8], [12], [15], [18]]
regressor = LinearRegression()
regressor.fit(X_train, y_train)
xx = np.linspace(0, 26, 100)
yy = regressor.predict(xx.reshape(xx.shape[0], 1))
plt.plot(xx, yy)
quadratic_featurizer = PolynomialFeatures(degree=2)
X_train_quadratic = quadratic_featurizer.fit_transform(X_train)
X_test_quadratic = quadratic_featurizer.transform(X_test)
regressor_quadratic = LinearRegression()
regressor_quadratic.fit(X_train_quadratic, y_train)
xx_quadratic = quadratic_featurizer.transform(xx.reshape(xx.shape[0], 1))
plt.plot(xx, regressor_quadratic.predict(xx_quadratic), c='r', linestyle='--')
plt.title('Pizza price regressed on diameter')
plt.xlabel('Diameter in inches')
plt.ylabel('Price in dollars')
plt.axis([0, 25, 0, 25])
plt.grid(True)
plt.scatter(X_train, y_train)
plt.show()
print X_train
print X_train_quadratic
print X_test
print X_test_quadratic
print 'Simple linear regression r-squared', regressor.score(X_test, y_test)
print 'Quadratic regression r-squared', regressor_quadratic.score(X_test_quadratic,
y_test)

```

ЗАДАНИЕ.

Для заданных входных данных X, y построить линейную и полиномиальную регрессионную модели и найти, какая из них дает меньшее абсолютное отклонение и насколько.

+++++

ЗАДАНИЕ

Использовать регрессионную модель для приближенного отыскания корней уравнений. Пример, пусть дано уравнение

$$3x^2 - 8x + 14 = 0 \quad (*)$$

По этому уравнению сгенерировать множество точек (x_i, y_i) . Построить линейную регрессионную модель по этому множеству точек. Модель будет иметь вид $Y = AX + B$. Найти X . Вокруг этого значения X сгенерировать еще массив точек (x_i, y_i) . Найти новое значение линии регрессии $Y = AX + B$. Найти

новый корень X' . Повторять, пока очередное значение корня не будет выполнять (*) с точностью 98%.

Замените (*) на кубическое уравнение

$$x^3 - 8x + 22 = 0$$

Попытайтесь найти корень описанным методом (разумеется, речь идет хотя бы об одном корне).