

## ЛАБОРАТОРНАЯ РАБОТА. ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ

Цель. Изучить возможности использования логистической регрессии в языке python.

Краткая теория.

### Логистическая регрессия

Линейная регрессионная модель не всегда способна качественно предсказывать значения зависимой переменной. Выбирая для построения модели линейное уравнение, мы не накладываем никаких ограничений на значения зависимой переменной. А такие ограничения могут быть существенными.

Линейная регрессионная модель может дать результаты, несовместимые с реальностью. С целью решения данных проблем полезно изменить вид уравнения регрессии и подстроить его для решения конкретной задачи.

Вообще, логит регрессионная модель предназначена для решения задач предсказания значения непрерывной зависимой переменной, при условии, что эта зависимая переменная может принимать значения на интервале от 0 до 1.

Приведем конкретный пример. Пусть требуется предсказать эффективность операции по пересадке сердца. Такие операции очень сложны и результата от их проведения может быть только два: пациент жив или умер (точнее, пережил ли он месяц после трансплантации - этот срок является определяющим).

В качестве предикторов используются данные предоперационного обследования и клинические параметры, например, возраст, уровень холестерина в крови, давление, группа крови и т.д. Задача свелась к классификации пациентов на две группы.

### Математическая основа логистической регрессии

Итак, как уже было сказано, в логит регрессионной модели предсказанные значения зависимой переменной или переменной отклика не могут быть меньше (или равными) 0, или больше (или равными) 1, не зависимо от значений независимых переменных; поэтому, эта модель часто используется для анализа бинарных зависимых переменных или переменных отклика.

При этом используется следующее уравнение регрессии (термин логит был впервые использован Berkson, 1944):

$$y = \exp(b_0 + b_1 \cdot x_1 + \dots + b_n \cdot x_n) / [1 + \exp(b_0 + b_1 \cdot x_1 + \dots + b_n \cdot x_n)]$$

Легко увидеть, что независимо от регрессионных коэффициентов или величин  $x$ , предсказанные значения ( $y$ ) в этой модели всегда будут лежать в диапазоне от 0 до 1.

Термин логит произошел от того, что эту модель легко линеаризовать с помощью логит преобразования. Предположим, что бинарная зависимая переменная  $y$  является непрерывной вероятностью  $p$ , лежащей в диапазоне от 0 до 1. Тогда можно преобразовать эту вероятность  $p$  следующим образом:

$$p' = \log_e \{p/(1-p)\}$$

Это преобразование называется логит или логистическим преобразованием.

Заметим, что  $p'$  теоретически может принимать любые значения от минус до плюс бесконечности. Поскольку логит преобразование решает проблему 0/1 границ для исходной зависимой переменной (вероятности), то можно использовать эти (логит преобразованные) значения в обычном линейном уравнении регрессии.

Фактически, при проведении логит преобразования обеих частей логит регрессионного уравнения, приведенного выше, мы получим стандартную линейную модель множественной регрессии:

$$p' = b_0 + b_1 * x_1 + b_2 * x_2 + \dots + b_n * x_n$$

Подобное уравнение нам уже знакомо. Решив его, мы получим значения регрессионных коэффициентов, по которым затем можно восстановить вероятность  $p$ .

ПРИМЕР ПРОГРАММЫ.

```
import pandas as pd

import numpy as np

from sklearn import preprocessing

import matplotlib.pyplot as plt


from sklearn.datasets import load_iris

iris = load_iris()

X, y = iris.data[:-1,:], iris.target[:-1]

print X

print y

from sklearn.linear_model import LogisticRegression

logistic = LogisticRegression()

logistic.fit(X,y)

print 'Predicted class %s, real class %s' % ( logistic.predict(iris.data[:-1,:]),iris.target[:-1])
```



## ЗАДАНИЕ.

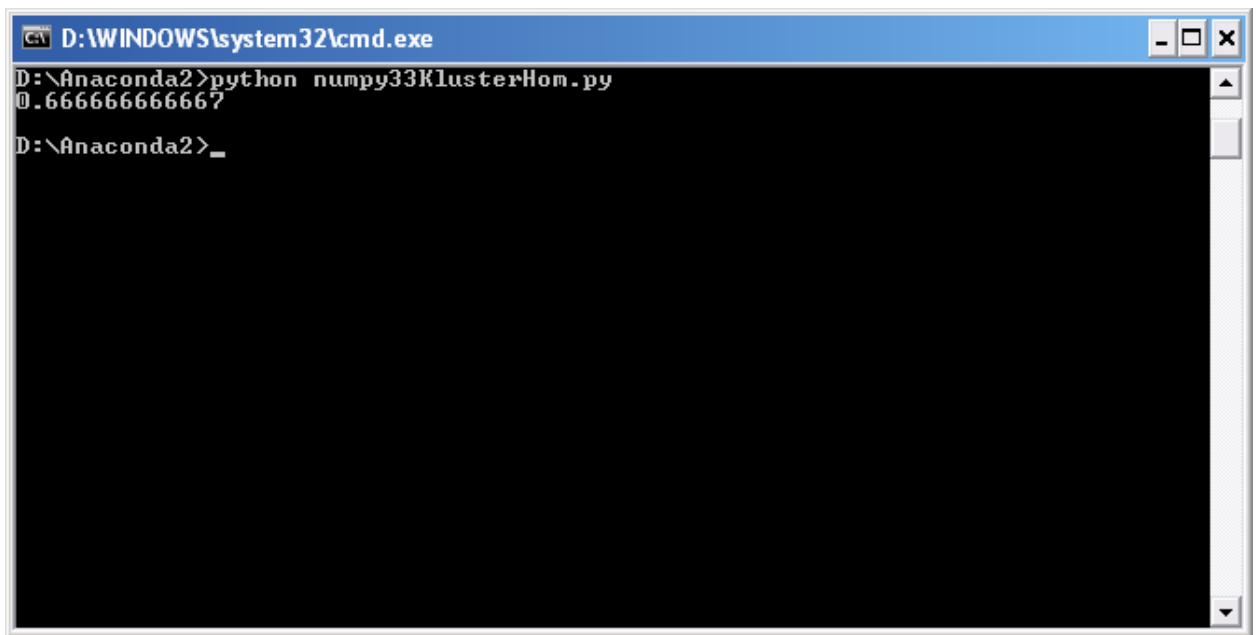
1. Сформировать массив  $X$  и  $y$  вручную в программе. Создать логистическую модель. Обучить ее. Сделать предсказание. Использовать приведенный выше пример программы как образец. Исходные данные должны описывать данные по росту и весу человека и его полу (м или ж). Данные подобрать самостоятельно.
2. Проверить качество логистической регрессионной модели на известных данных (данные сформировать самостоятельно).

Далее рассмотрим технику работы с кластерами. Ниже приводятся примеры и результирующие скриншоты. Выполните эти примеры.

Следующий код проверяет однородность кластера (нужен эталон для сравнения)

```
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
labels_true=[0,0,0,1,1,1]
labels_pred=[0,0,1,1,2,2]
z=metrics.homogeneity_score(labels_true,labels_pred)
print z
```

Чем ближе к 1, тем лучше. У нас



```
D:\WINDOWS\system32\cmd.exe
D:\Anaconda2>python numpy33KlusterHom.py
0.6666666666666667
D:\Anaconda2>_
```

Оценка степени схожести двух кластеров

```
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
labels_true=[0,0,0,1,1,1]
labels_pred=[0,0,1,1,2,2]
z=metrics.adjusted_rand_score(labels_true,labels_pred)
print z
Совершенная схожесть есть 1.0
```

Оценка близости объектов внутри кластеров и их удаленности от других кластеров выполняется с помощью метрики силуэта:

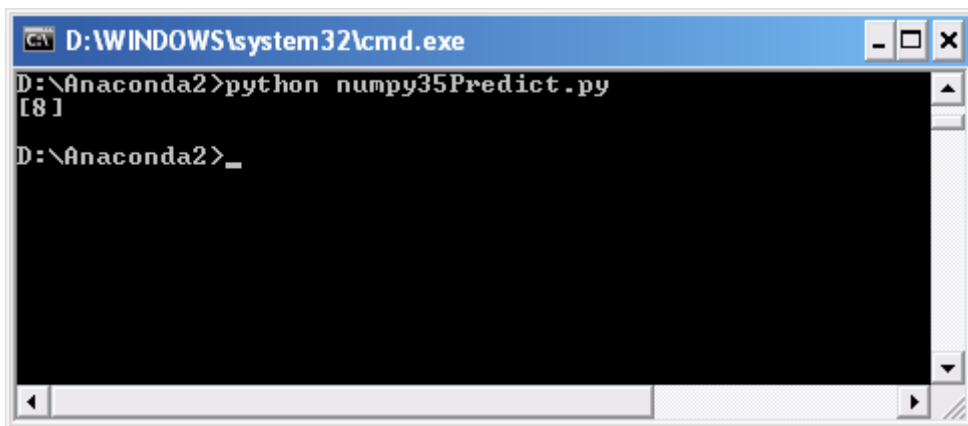
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.stats as stats
import statsmodels.api as sm
from scipy.optimize import fsolve
from scipy.linalg import *
from numpy import array
from scipy.cluster.vq import vq, kmeans, whiten
from sklearn import metrics
from sklearn.metrics import pairwise_distances
from sklearn import datasets
dataset=datasets.load_iris()
x=dataset.data
from sklearn.cluster import KMeans
kmodel= KMeans(n_clusters=3,random_state=1).fit(x)
labels=kmodel.labels_
print labels
z=metrics.silhouette_score(x,labels,metric='euclidean')
```

```
print z
```

[illegible]

## ПРЕДСКАЗАНИЕ

```
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
iris = datasets.load_iris()
digits = datasets.load_digits()
from sklearn import svm
clf = svm.SVC(gamma=0.001, C=100.)
clf.fit(digits.data[:-1], digits.target[:-1])
z=clf.predict(digits.data[-1:])
print z
```

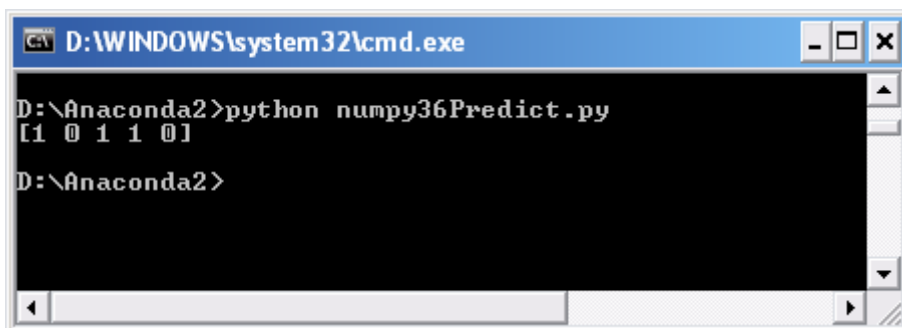


```
D:\WINDOWS\system32\cmd.exe
D:\Anaconda2>python numpy35Predict.py
[8]
D:\Anaconda2>_
```

+++++  
ПРЕДСКАЗАНИЕ СПИСКА ЗНАЧЕНИЙ

```
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
rng = np.random.RandomState(0)
X = rng.rand(100, 10)
y = rng.binomial(1, 0.5, 100)
X_test = rng.rand(5, 10)

clf = svm.SVC()
clf.set_params(kernel='linear').fit(X, y)
z = clf.predict(X_test)
print z
```



```
D:\WINDOWS\system32\cmd.exe
D:\Anaconda2>python numpy36Predict.py
[1 0 1 1 0]
D:\Anaconda2>
```

Предсказание на основе логистической регрессии

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

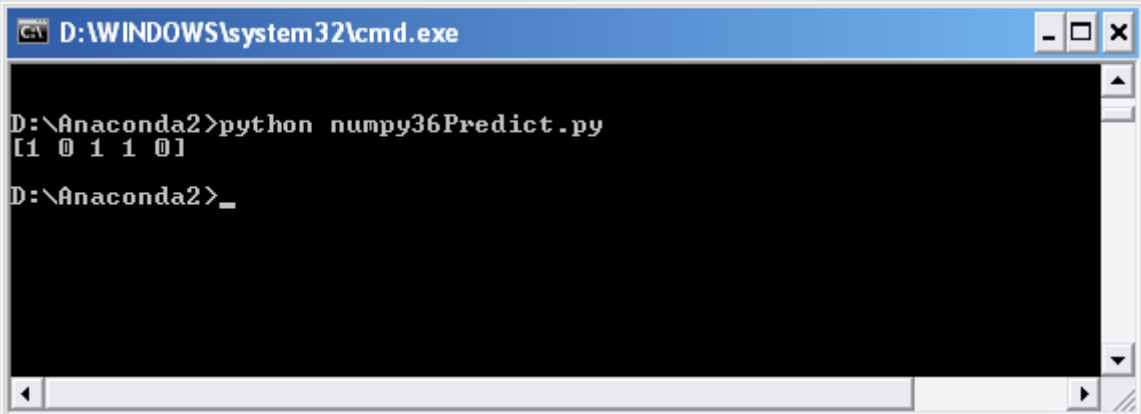
```

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
rng = np.random.RandomState(0)
X = rng.rand(100, 10)
y = rng.binomial(1, 0.5, 100)
X_test = rng.rand(5, 10)
from sklearn import linear_model
logreg = linear_model.LogisticRegression(C=1e8)
# This class implements regularized logistic regression. C is the Inverse of

# Large value => no regularization.
logreg.fit(X, y)
z = logreg.predict(X_test)
print z

```

Результат тот же:



The screenshot shows a Windows command prompt window titled "D:\WINDOWS\system32\cmd.exe". The prompt is at "D:\Anaconda2>". The user has entered the command "python numpy36Predict.py", and the output is "[1 0 1 1 0]". The prompt is now "D:\Anaconda2>\_".

Прогнозирование на ОСНОВЕ ДЕРЕВА

```

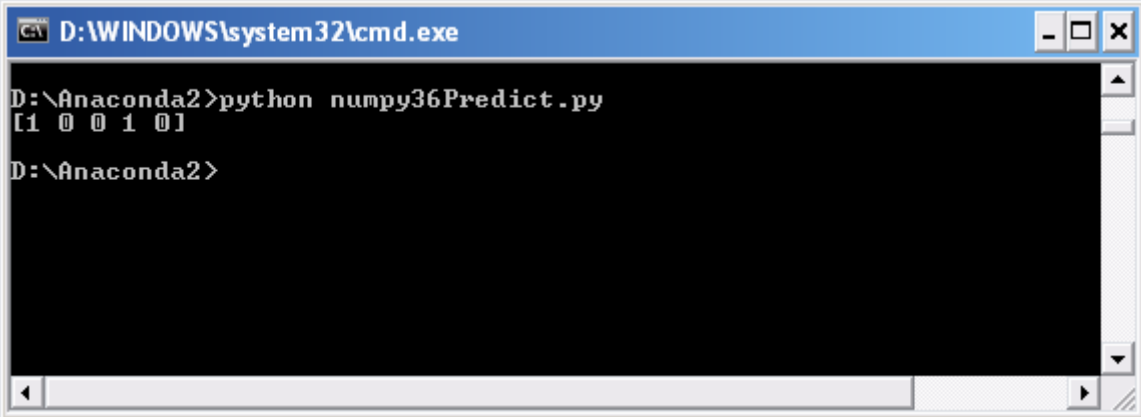
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.stats as stats
import statsmodels.api as sm
from scipy.optimize import fsolve
from scipy.linalg import *
from numpy import array
from scipy.cluster.vq import vq, kmeans, whiten
from sklearn import metrics
from sklearn.metrics import pairwise_distances
from sklearn import datasets
from sklearn import svm
rng = np.random.RandomState(0)
X = rng.rand(100, 10)
y = rng.binomial(1, 0.5, 100)
X_test = rng.rand(5, 10)
from sklearn.ensemble import RandomForestClassifier

```



```
forest = RandomForestClassifier(n_estimators = 100)
forest.fit(X, y)
z= forest.predict(X)
print z[0:5]
```

Результат уже несколько отличается



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 numpy36Predict.py". The output of the script is displayed on the next line: "[1 0 0 1 0]". The prompt is now "D:\Anaconda2>".

На основе приведенных примеров решите следующие задачи.

1. Составьте два кластера: мужчины и женщины. Возьмите их рост. Проверьте, насколько согласованы значения в кластерах и между кластерами. Возьмите какое-нибудь новое значение для роста, включите его поочередно в каждый кластер и посмотрите, как на основании изменения оценки согласованности можно сделать заключение о принадлежности объекта к кластеру.
2. Возьмите множество значений, например, значений роста попеременно, мужчин, женщин, детей. Разбейте сначала множество на два кластера и оцените согласованность. Затем разбейте на три кластера и снова оцените согласованность. Наконец, разбейте на 4 кластера. Выполните те же действия. Сделайте заключение, какое разбиение наилучшее.