1. Lambda作为一个表达式，定义了一个匿名函数，比如说

Func = lambda x:x+1

在该代码中x为入口参数，x+1为函数体，lambda简化了函数定义的书写方式

2． Python数据可视化的工具包，Wordcloud是python的慈云生成类库，plotly据说很强大，还是交互式的，seaborn是生成主题？squarify和构建树结构有关？

3． Python中忽略warnings可以使用下面两行代码：

Import warning warning.filterwarnings(‘ignore’)

4． dataFrame.ix[row,col || indr,indc]，通过行列索引定位dataFrame中的数据元素

5． 上围篱值（top whiskers），下围篱值（bottom whiskers），上围篱值在1.5倍四分位距处。

6． Python2.6开始，新增了一种格式化字符串函数str.format()，例如：

>>> "{1} {0} {1}".format("hello", "world") # 设置指定位置

'world hello world'

7． Python创建DataFrame对象：

Df = pd.DataFrame([1,2,3,4,5],columns = [‘cols’],index = [‘a’,’b’,’c’,’d’,’e’])

Print(df)  
 cols

a 1

b 2

c 3

d 4

e 5

8． Numpy中的nan为np.nan，去除dataframe类型中的nan可以用pd.dropna（subset），subset是为了define in which columns to look for missing values

9． DataFrame去除重复项可以用drop\_duplicates([columns]),当columns组合列中存在重复项时保留第一个出现的值的组合，删除其他重复项

10． Python对字典进行排序可以使用sorted（x.items(),key=operator.itemgetter(0)）,导入operator包，itemgetter（0）表示按照item的第一个字符也就是key进行排序，而itemgetter（1）表示按照item的value进行排序。

11． Numpy.percentile（data，percentage），可以计算输入数据的累计百分位，

12． 3δ法则，异常值定义为一组测定值中与平均值的偏差超过3倍标准差的值，四分位距（interquartile range， IQR），又称四分差，相当于是一种稳健统计（robust statistic），IQR=Q3 – Q1，在Q3+1.5IQR和Q1－1.5IQR处画两条与中位线一样的线段，这两条线段为异常值截断点，称其为内限；在Q3+3IQR和Q1－3IQR处画两条线段，称其为外限。处于内限以外位置的点表示的数据都是异常值，其中在内限与外限之间的异常值为温和的异常值（mild outliers），在外限以外的为极端的异常值(extreme outliers)。四分位距IQR=Q3-Q1。[Turkey method]

13． Numpy，series，DataFrame类都具有Concat，默认axis=0为沿y轴方向，axis=1为沿x轴方向。

14． DataFrame检测nan并且进行处理的方法：dataset.gillna(np.nan)，dataset.isnull().sum()，DataFrame类summarize and statistic 数据可以使DataFrame.describe()

15． 相关矩阵（correlation matrix），可以使用DataFrame.corr（）函数

16． 可以使用DataFrame类型中的skew函数，DataFrame.skew(axis=None, skipna=None, level=None, numeric\_only=None, \*\*kwargs)

17． 可以采用seaborn包中的sns.heatmap(dataset[[‘’…columns]].corr(),cmap = “BrBG”,annot = True)来画出相关矩阵寻找不同变量之间的相关关系

18． Pandas中DataFrame.loc根据index来对数据进行索引，而DataFrame.iloc根据数据的行号来对数据进行索引

19． Seaborn中的carplot（kind = strip）：

20． Pandas.cut(x,y)以及Pandas.qcut(x,y)都是用来对原始数据x进行离散化的函数，y是离散化后得到的分数，不同的地方在于cut函数是遵照在x区间上等分的原则进行离散化，qcut函数是遵照在x区间上按百分位等分的原则进行离散化，也就是qcut离散化后每个分段上的数据点数是一样的，而cut是不一样的，但是区间长度是一样的。返回的是一个categories类型或者series类型

21． Python中的lambda用来创建匿名函数，如下面代码所示：

# 可写函数说明

sum = lambda arg1, arg2: arg1 + arg2;

# 调用sum函数

print "相加后的值为 : ", sum( 10, 20 )

print "相加后的值为 : ", sum( 20, 20 )

22． 在对原始数据进行数值转化的过程中可以使用sklearn的labelencode函数，作用如下：

from sklearn import preprocessing

...: le =preprocessing.LabelEncoder()

...: le.fit(["paris", "paris", "tokyo", "amsterdam"])

...: print('标签个数:%s'% le.classes\_)

...: print('标签值标准化:%s' % le.transform(["tokyo", "tokyo", "paris"]))

23． Get\_dummies，Convert categorical variable into dummy/indicator variables，将标签数据转换成为指标变量，如下面的例子所示：

s = pd.Series(list('abca')

pd.get\_dummies(s)

a b c

0 1 0 0

1 0 1 0

2 0 0 1

3 1 0 0

df = pd.DataFrame({'A': ['a', 'b', 'a'], 'B': ['b', 'a', 'c'],'C': [1, 2, 3]})

pd.get\_dummies(df, prefix=['col1', 'col2'])

C col1\_a col1\_b col2\_a col2\_b col2\_c

0 1 1 0 0 1 0

1 2 0 1 1 0 0

2 3 1 0 0 0 1

24． 可以使用sklearn.model\_selection.train\_test\_split函数对数据进行训练数据和测试数据的切割

X\_train,X\_test,y\_train,y\_test=train\_test\_split(train\_data,train\_target,test\_size=0.3,random\_state=0)

25． 在数据可视化方面可以选择的集中图像的类型，包括

Barplot

Boxplot

Violinplot

Hist

Pointplot

Kdeplot：核密度图像

26．Sklearn.model\_selection.ShuffleSplit函数可以将原始数据随机切割成训练数据以及测试数据用于交叉验证，同时与其他切割办法不同的是在该方法中不同的训练数据集是有可能相同的，他返回的是相应数据集的索引。

from sklearn.model\_selection import ShuffleSplit

X = np.array([[1, 2], [3, 4], [5, 6], [7, 8]])

y = np.array([1, 2, 1, 2])

rs = ShuffleSplit(n\_splits=3, test\_size=.25, random\_state=0)

rs.get\_n\_splits(X)

print(rs)

for train\_index, test\_index in rs.split(X):

print("TRAIN:", train\_index, "TEST:", test\_index)

rs = ShuffleSplit(n\_splits=3, train\_size=0.5, test\_size=.25,

random\_state=0)

for train\_index, test\_index in rs.split(X):

print("TRAIN:", train\_index, "TEST:", test\_index)

TRAIN: [3 1] TEST: [2]

TRAIN: [2 1] TEST: [0]

TRAIN: [0 2] TEST: [3]

27． 可以使用sklearn.metrics.confusion\_matrix(y\_true, y\_pred)函数生成混淆矩阵对分类器的性能进行评估，然后使用plot\_confusion\_matrix()函数生成混响矩阵图像

28． 可以使用data\_copy[item].isnull().sum()函数计算原始数据中缺失数据的个数

29． Python中的下划线与变量：

1. 前带\_的变量: 标明是一个私有变量, 只用于标明, 外部类还是可以访问到这个变量

2. 前带两个\_ ,后带两个\_ 的变量: 标明是内置变量,

3. 大写加下划线的变量: 标明是 不会发生改变的全局变量

函数:

1. 前带\_的变量: 标明是一个私有函数, 只用于标明,

2. 前带两个\_ ,后带两个\_ 的函数: 标明是特殊函数

常量 : 大写加下划线

USER\_CONSTANT

对于不会发生改变的全局变量，使用大写加下划线。

私有变量 : 小写和一个前导下划线

\_private\_value

Python 中不存在私有变量一说，若是遇到需要保护的变量，使用小写和一个前导下划线。但这只是程序员之间的一个约定，用于警告说明这是一个私有变量，外部类不要去访问它。但实际上，外部类还是可以访问到这个变量。

内置变量 : 小写，两个前导下划线和两个后置下划线

\_\_class\_\_

两个前导下划线会导致变量在解释期间被更名。这是为了避免内置变量和其他变量产生冲突。用户定义的变量要严格避免这种风格。以免导致混乱。

**另外一个版本：**

**1、 \_xx 以单下划线开头的表示的是protected类型的变量。即保护类型只能允许其本身与子类进行访问。若内部变量标示，如： 当使用“from M import”时，不会将以一个下划线开头的对象引入 。**

**2、 \_\_xx 双下划线的表示的是私有类型的变量。只能允许这个类本身进行访问了，连子类也不可以用于命名一个类属性（类变量），调用时名字被改变（在类FooBar内部，\_\_boo变成\_FooBar\_\_boo,如self.\_FooBar\_\_boo）**

**3、 \_\_xx\_\_定义的是特列方法。用户控制的命名空间内的变量或是属性，如init , \_\_import\_\_或是file 。只有当文档有说明时使用，不要自己定义这类变量。 （就是说这些是python内部定义的变量名）**

**例子：**

**class A(object):**

**def \_\_init\_\_(self):**

**self.\_\_private()**

**self.public()**

**def \_\_private(self):**

**print 'A.\_\_private()'**

**def public(self):**

**print 'A.public()'**

**class B(A):**

**def \_\_private(self):**

**print 'B.\_\_private()'**

**def public(self):**

**print 'B.public()'**

**a = A() # 实例化对象a**

**b = B() # 实例化对象b**

**print a.\_\_private() # 这里会报错,说 AttributeError: 'a' object has no attribute '\_\_private'**

**print b.\_\_private() # 这里会报错,说 AttributeError: 'B' object has no attribute '\_\_private'**

**print '======'**

**print b.\_A\_\_private() # 因为私有变量轧压机制,\_\_private方法变成\_A\_\_private()**

**print dir(b) # 通过自省方法dir(),查看b所具有的属性**

**print '======'**

**# --->**

**A.\_\_private()**

**B.public()**

**======**

**A.\_\_private()**

**======**

**['\_A\_\_private', '\_B\_\_private', '\_\_class\_\_', '\_\_delattr\_\_', '\_\_dict\_\_', '\_\_doc\_\_', '\_\_format\_\_', '\_\_getattribute\_\_', '\_\_hash\_\_', '\_\_init\_\_', '\_\_module\_\_', '\_\_new\_\_', '\_\_reduce\_\_', '\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_setattr\_\_', '\_\_sizeof\_\_', '\_\_str\_\_', '\_\_subclasshook\_\_', '\_\_weakref\_\_', '\_test', 'get\_pri', 'public']**

**类A和B都有一个私有方法,命名为\_ \_private(),因为B是继承于A,在用B实例化对象b时,程序需要先执行A的\_ \_ init\_ \_ (),执行到\_ \_ private时, 是A的\_ \_private();到public()方法时候,B里面重写了,因此调用的是B的public()方法;**

**当 执行到print a.\_\_private( )时候,会出现找不到属性错误,自己本身调用私有方法也会出错,因为私有变量轧压机制,\_ \_private方法变成\_A \_ \_ private(),所以就找不到这个方法.**

30． 可以使用sklearn中的feature\_selection.RFECV函数来对原始数据特征进行特征选择工作。class sklearn.feature\_selection.RFE(estimator, n\_features\_to\_select=None, step=1, verbose=0)[source]¶

特征选择大致可以分为三种方式：

Filter：过滤法，按照发散性或者相关性对各个特征进行评分，设定阈值或者待选择阈值的个数，选择特征。Filter方法中又包括集中方式：移除低方差的特征（Removing features with low variance），单变量特征选择（Univariate feature selection），

Wrapper：包装法，根据目标函数（通常是预测效果评分），每次选择若干特征，或者排除若干特征。Warpper又包含几种方式：递归特征消除（Recursive Feature Elimination，RFE）：对于一个数量为d的feature的集合，他的所有的自己的个数是2的d次方减1，递归特征消除会计算所有子集的validation error，选择error最小的那个子集作为所挑选的特征。

Embedded：嵌入法，先使用某些机器学习的算法和模型进行训练，得到各个特征的权值系数，根据系数从大到小选择特征。类似于Filter方法，但是是通过训练来确定特征的优劣。该方法中又包括了使用SelectFromModel选择特征，根据pearson相关系数等指标对特征进行打分从而实现对特征的选择，使用L1货柜产生稀疏模型进行特征选择

31． 对于一个小的数据集，自制的分类算法往往会取得最好的结果，来自于kaggle博主，url=https://www.kaggle.com/ldfreeman3/a-data-science-framework-to-achieve-99-accuracy

32． 可以使用sklearn.model\_selection.learning\_curve函数来构建模型的性能曲线，例：

>>> from sklearn.model\_selection import learning\_curve

>>> from sklearn.svm import SVC

>>> train\_sizes, train\_scores, valid\_scores = learning\_curve(

... SVC(kernel='linear'), X, y, train\_sizes=[50, 80, 110], cv=5)

>>> train\_sizes

array([ 50, 80, 110])

>>> train\_scores

array([[ 0.98..., 0.98 , 0.98..., 0.98..., 0.98...],

[ 0.98..., 1. , 0.98..., 0.98..., 0.98...],

[ 0.98..., 1. , 0.98..., 0.98..., 0.99...]])

>>> valid\_scores

array([[ 1. , 0.93..., 1. , 1. , 0.96...],

[ 1. , 0.96..., 1. , 1. , 0.96...],

[ 1. , 0.96..., 1. , 1. , 0.96...]])

33． \*args可以接收序列的输入参数，当函数的参数不确定时，可以使用\*args，实际上是所有参数的数组，而\*kwargs是存储的字典

34． Python3将文本（text）和二进制数据做了更清晰的区分，文本纵使用unicode进行编码，以str类型进行表示，二进制数据以bytes类型进行表示，他们之间的关系可以表示为：

Strings可以本编码（encode）成字节bytes，bytes也可以解码（decode）成strings

>>> '€20'.encode('utf-8')

b'\xe2\x82\xac20'

>>> b'\xe2\x82\xac20'.decode('utf-8')

'€20'

35. python中的pass语句是空语句，是为了保持程序结构的完整性而存在的，pass语句不会做任何事情，一般只用作占位语句。

36．Python中可以使用time.sleep(secs

)函数推迟调用线程的运行，可以通过参数secs指秒数，表示进程挂起的时间。

37．PanedWindow是一个窗口布局管理的插件，可以用鼠标移动上面的分割线来改变每个子控件的大小。