1. 数据集及代码的分析，预测目标及预测所使用的特征

(1) 数据集及代码分析

jupyter 20221024 (1) 最新检查点: 21 分钟前 (已自动保存)

Logout

File Edit View Insert Cell Kernel Widgets Help 可信 Python 3 (ipykernel)

运行 代码

In [6]:
```
import numpy as np
import pandas as pd
import hvplot.pandas
import matplotlib.pyplot as plt
%matplotlib inline
# 使输出的图像以更高清的方式显示
%config InlineBackend.figure_format = 'retina'
import seaborn as sns
#用 matplotlib能够完成一些基本的图表操作，Seaborn 库可以让这些图的表现更加丰富
# plt.style.use('ggplot')
plt.style.use("fivethirtyeight")
# Pandas中只显示3位小数
pd.set_option('display.float_format', lambda x: '{:.3f}'.format(x))

from sklearn import datasets  # 导入数据集
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

import warnings
```

In [7]:
```
boston = datasets.load_boston()
print(boston.keys())
#  data 代表特征矩阵，target 代表目标结果，feature_names 代表 data 对应的特征名称，
# DESCR 是对数据集的描述，filename 对应的是 boston 这个数据在本地的存放文件路径。
```

```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename', 'data_module'])
```

In [8]:
```
X = boston.data   # 数据
y = boston.target  # 目标
# X为输入，y为输出
#一个Datarame表示一个表格，类似电子表格的数据结构，包含一个经过排序的列表集，它的每一列都可以有不同的类型值（数字，字符串，布尔等等）。
#Datarame有行和列的索引；它可以被看作是一个Series的字典（Series们共享一个索引）。
df = pd.DataFrame(
    X,
    columns = boston.feature_names#特征名称
)
df.head() #df.head()会将excel表格中的第一行看作列名，并默认输出之后的五行
#CRIM: per capita crime rate by town 城镇人均犯罪率
#ZN:  proportion of residential land zoned for lots over 25,000 sq.ft. 占地面积超过2.5万平方英尺的住宅用地比例
#INDUS: proportion of non-retail business acres per town 城镇上非零售业务地区的 比例
#CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise) 虚拟变量；如果土地在查尔斯河，取值1；否则为0
#NOX:  nitric oxides concentration (parts per 10 million) 一氧化氮浓度
#RM: average number of rooms per dwelling 平均每个居民房数
#AGE:  proportion of owner-occupied units built prior to 1940 在1940年之前建成的所有者占用单位的比例
#DIS:  weighted distances to five Boston employment centres 与波士顿的5个就业中心之间的加权距离
#RAD:   index of accessibility to radial highways 辐距离住房最近的公路入口编号
#TAX: full -value property-tax rate per $10,000 每10,000美元的全额物业税
#PTRATIO: pupil-teacher ratio by town 城镇师生比例大小
#B: 1000(Bk - 0.63)^2 where Bk is the proportion of black people by town 1000(Bk-0.63)^2,其中 Bk 指代城镇中黑人的比例
#LSTAT: % lower status of the population 全部人口中地位较低人群的百分数大小
#MEDV: M edian value of owner-occupied homes in $1000's 目标变量，以1000美元来进行计算的自由住房的中位数大小
```

Out[8]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.006	18.000	2.310	0.000	0.538	6.575	65.200	4.090	1.000	296.000	15.300	396.900	4.980
1	0.027	0.000	7.070	0.000	0.469	6.421	78.900	4.967	2.000	242.000	17.800	396.900	9.140
2	0.027	0.000	7.070	0.000	0.469	7.185	61.100	4.967	2.000	242.000	17.800	392.830	4.030
3	0.032	0.000	2.180	0.000	0.458	6.998	45.800	6.062	3.000	222.000	18.700	394.630	2.940
4	0.069	0.000	2.180	0.000	0.458	7.147	54.200	6.062	3.000	222.000	18.700	396.900	5.330

In [9]:
```
df["MEDV"] = y #MEDV: M edian value of owner-occupied homes in $1000's 目标变量，以1000美元来进行计算的自由住房的中位数大小
df.head() #df.head()会将excel表格中的第一行看作列名，并默认输出之后的五行
```

Out[9]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.006	18.000	2.310	0.000	0.538	6.575	65.200	4.090	1.000	296.000	15.300	396.900	4.980	24.000
1	0.027	0.000	7.070	0.000	0.469	6.421	78.900	4.967	2.000	242.000	17.800	396.900	9.140	21.600
2	0.027	0.000	7.070	0.000	0.469	7.185	61.100	4.967	2.000	242.000	17.800	392.830	4.030	34.700
3	0.032	0.000	2.180	0.000	0.458	6.998	45.800	6.062	3.000	222.000	18.700	394.630	2.940	33.400
4	0.069	0.000	2.180	0.000	0.458	7.147	54.200	6.062	3.000	222.000	18.700	396.900	5.330	36.200

In [10]:
```
#查看数据字段，类型
df.columns
```

Out[10]:
```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MEDV'],
      dtype='object')
```

In [11]:
```
df.dtypes
```

Out[11]:
```
CRIM       float64
ZN         float64
INDUS      float64
CHAS       float64
NOX        float64
RM         float64
AGE        float64
DIS        float64
RAD        float64
TAX        float64
PTRATIO    float64
B          float64
LSTAT      float64
MEDV       float64
dtype: object
```

In [12]:
```
df.info()
#打印DataFrame的简要摘要，显示有关DataFrame的信息，
#包括索引的数据类型dtype和列的数据类型dtype，非空值的数量和内存使用情况。
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

In [13]:
```
#数据形状
df.shape #df.shape会返回一个元组，该元组的第一个元素代表行数，第二个元素代表列数
```

Out[13]:
```
(506, 14)
```

In [14]:
```
#数据缺失情况
df.isnull().sum()
```

Out[14]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

mean：平均数
std:样本标准偏差
min:最小值
25%：四分之一分位数

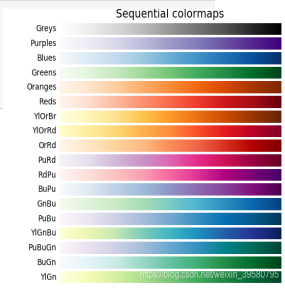
In [15]:
```
df.describe()
#describe()函数用于生成描述性统计信息。
#描述性统计数据：数值类型的包括均值，标准差，最大值，最小值，分位数等；
#类别的包括个数，类别的数目，最高数量的类别及出现次数等；输出将根据提供的内容而有所不同。
```

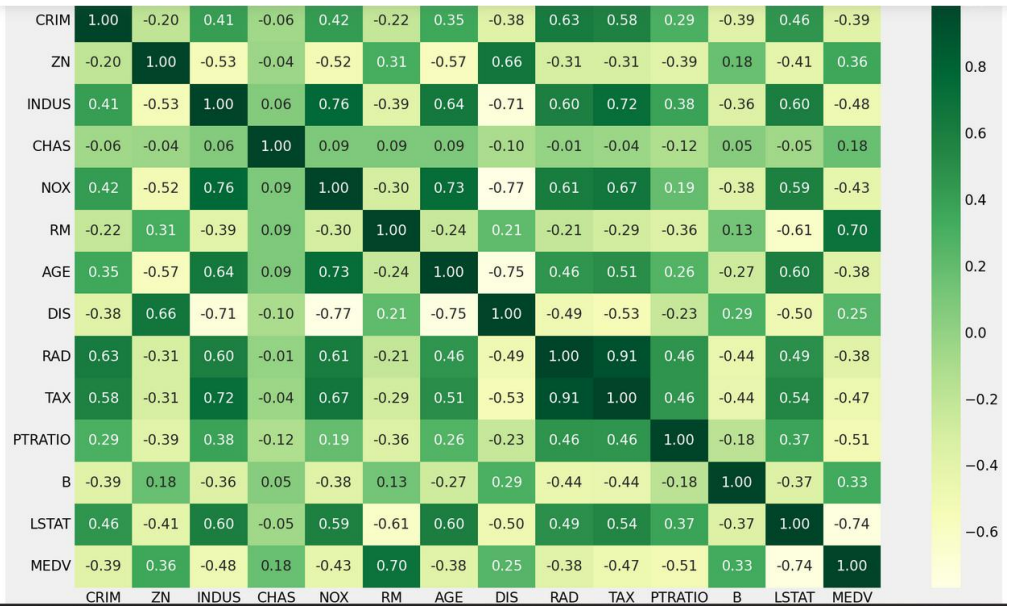
Out[15]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
count	506.000	506.000	506.000	506.000	506.000	506.000	506.000	506.000	506.000	506.000	506.000	506.000	506.000	506.000
mean	3.614	11.364	11.137	0.069	0.555	6.285	68.575	3.795	9.549	408.237	18.456	356.674	12.653	22.533
std	8.602	23.322	6.860	0.254	0.116	0.703	28.149	2.106	8.707	168.537	2.165	91.295	7.141	9.197
min	0.006	0.000	0.460	0.000	0.385	3.561	2.900	1.130	1.000	187.000	12.600	0.320	1.730	5.000
25%	0.082	0.000	5.190	0.000	0.449	5.885	45.025	2.100	4.000	279.000	17.400	375.377	6.950	17.025
50%	0.257	0.000	9.690	0.000	0.538	6.208	77.500	3.207	5.000	330.000	19.050	391.440	11.360	21.200
75%	3.677	12.500	18.100	0.000	0.624	6.623	94.075	5.188	24.000	666.000	20.200	396.225	16.955	25.000
max	88.976	100.000	27.740	1.000	0.871	8.780	100.000	12.127	24.000	711.000	22.000	396.900	37.970	50.000

In [16]:
```
corr=df.corr()
#计算相关系数
```

In [17]:
```
#绘制相关系数的热力分布图
plt.figure(figsize=(16,10))
sns.heatmap(corr,annot=True,fmt=".2f",cmap="YlGn")
# annot=True：显示相关系数的数据，fmt=".2f"：只显示两位小数，cmap="YlGn"：设置热力图颜色
plt.show()
```

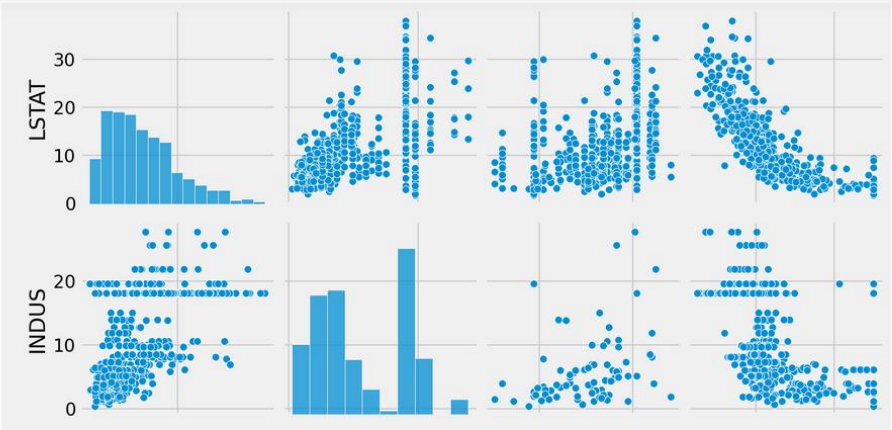


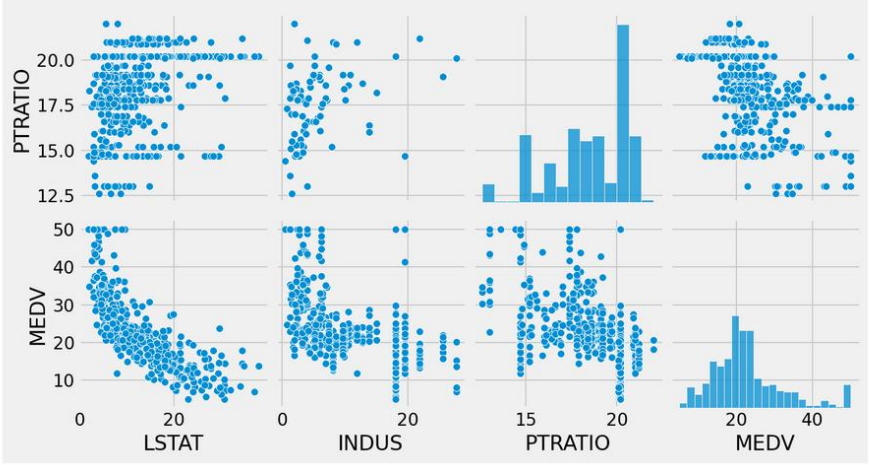


In [18]:
```
#查看每个特征和目标变量MEDV之间的相关系数
corr["MEDV"].sort_values()#默认axis=0，按照指定列中数据大小排序
```

Out[18]:
```
LSTAT     -0.738
PTRATIO   -0.508
INDUS     -0.484
TAX       -0.469
NOX       -0.427
CRIM      -0.388
RAD       -0.382
AGE       -0.377
CHAS       0.175
DIS        0.250
B          0.333
ZN         0.360
RM         0.695
MEDV       1.000
Name: MEDV, dtype: float64
```

In [19]:
```
sns.pairplot(df[["LSTAT","INDUS","PTRATIO","MEDV"]]) # 绝对值靠前3的特征
#sns.pairplot() 用来展示两两特征之间的关系
#对角线上是各个属性的直方图（分布图），而非对角线上是两个不同属性之间的相关图
plt.show()
```





（2） 预测目标及预测使用的特征

```
In [28]: #划分给定的数据集，比例是8：2
X = df.drop("MEDV",axis=1)#代表将 "MEDV" 对应的列标签沿着水平的方向依次删掉
y = df[["MEDV"]]

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2, #表示切分的训练集和测试集的比例
    random_state=8)
#random_stste:随机数种子——其实就是该组随机数的编号，在需要重复试验的时候，保证得到一组一样的随机数。
```

```
In [29]: print("X_train shape: ",X_train.shape)
print("y_train shape: ",y_train.shape)
print("X_test shape: ",X_test.shape)
print("y_test shape: ",y_test.shape)
#df.shape会返回一个元组，该元组的第一个元素代表行数，第二个元素代表列数
```

```
X_train shape:  (404, 13)
y_train shape:  (404, 1)
X_test shape:  (102, 13)
y_test shape:  (102, 1)
```

```
In [17]: #导入支持向量机回归模型
from sklearn.svm import SVR
from sklearn.metrics import r2_score
from sklearn.metrics import explained_variance_score
#分别测试linear核函数和rbf核函数
for kernel in ['linear','rbf']:
    svr = SVR(kernel = kernel,gamma = 'auto')
    #gamma:该参数为核函数系数，只对 'rbf' ，'poly' ，'sigmod' 有效。
    #如果gamma设置为auto，代表其值为样本特征数的倒数，即1/n_features，也有其他值可设定。gamma默认auto

    #kernel:该参数用于选择模型所使用的核函数，
    #算法中常用的核函数有：— linear：线性核函数— poly：多项式核函数 - rbf：
    #径像核函数/高斯核 - sigmod：sigmod核函数 - precomputed：核矩阵，该矩阵表示自己事先计算好的，输入后算法内部将使用你提供的矩阵进行计算
    svr.fit(X_train,y_train) # 训练
    y_pre = svr.predict(X_test)
    print(kernel,'核函数的模型训练集得分：{:.4f}'.format(svr.score(X_train,y_train)))
    print(kernel,'核函数的模型测试集得分：{:.4f}'.format(svr.score(X_test,y_test)))
    print('均方根误差RMSE:{:.4f}'.format(np.sqrt(mean_squared_error(y_test,y_pre))))
    print('均方误差MSE:{:.4f}'.format(np.round(mean_squared_error(y_test,y_pre),2)))
    print('解释方差分:{:.4f}'.format(np.round(explained_variance_score(y_test,y_pre),2)))
    print('R^2得分:{:.4f}'.format(np.round(r2_score(y_test,y_pre),2)))
```

```
linear 核函数的模型训练集得分:0.7095
linear 核函数的模型测试集得分:0.6918
均方根误差RMSE:4.7790
均方误差MSE:22.8400
解释方差分:0.6900
R^2得分:0.6900
rbf 核函数的模型训练集得分:0.1441
rbf 核函数的模型测试集得分:0.0207
均方根误差RMSE:8.5182
均方误差MSE:72.5600
解释方差分:0.0300
R^2得分:0.0200
```

R^2又名决定系数，这个公式的作为指标的意图就相对明显了。决定系数计算出来越是接近1，预测值越接近真实样本值。

$$R^2 = 1 - \frac{MSE}{Var(Y))} = 1 - \frac{MSE}{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y}))}$$

均方根误差，亦称标准误差，其定义为i=1，2，3，…n。在有限测量次数中，均方根误差常用下式表示：√[∑di^2/n]=Re，式中：n为测量次数；di为一组测量值与真值的偏差。标准误差能够很好地反映出测量的精密度。

均方误差（mean-square error，MSE）是反映估计量与被估计量之间差异程度的一种度量。设t是根据子样确定的总体参数θ的一个估计量，(θ-t)2的数学期望，称为估计量t的均方误差。

$$MSE(y, y') = \frac{\sum_{i=1}^{n}(y_i - y_i')^2}{n}$$

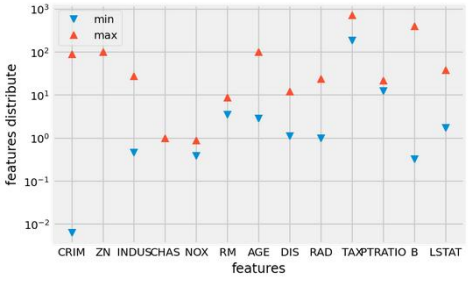
可解释方差是(1- 样本值与预测值之差的方差/样本方差)

$$Evar = 1 - \frac{\sum_{i=1}^{n}((y_i - \hat{y}) - E(\vec{y} - \hat{y}))^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

可解释方差指标衡量的是所有预测值和样本之间的差的分散程度与样本本身的分散程度的相近程度。本身是分散程度的对比。最后用1-这个值，最终值越大表示预测和样本值的分散分布程度越相近。

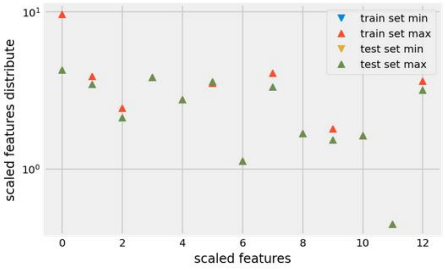
In [32]:
```
#SVM算法对于数据预处理的要求比较高，比如：数据特征量级相差较大，则就要对数据进行预处理。
# 查看数据集中各个特征的数量级分布情况，使用散点图进行可视化
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(X.min(axis = 0), 'v', label='min')
plt.plot(X.max(axis = 0), '^', label='max')
# 设定纵坐标为对数形式
plt.yscale('log')
# 设置图注位置为最佳
plt.legend(loc='best')
# 设置最标轴标题
plt.xlabel('features')
plt.ylabel('features distribute')
# 显示图像
plt.show()
```

%matplotlib incline 作用就是这样图象就会出现在Notebook里面，而不是一个新窗口里



In [34]:
```
# 导入数据处理工具
from sklearn.preprocessing import StandardScaler
#StandardScaler类是一个用来讲数据进行归一化和标准化的类。
#-计算训练集的平均值和标准差，以便测试数据集使用相同的变换。
#标准差标准化（standardScale）使得经过处理的数据符合标准正态分布，对于每个属性/每列来说所有数据都聚集在0附近，标准差为1，
#使得新的X数据集方差为1，均值为0

# 对训练集和测试集进行数据预处理
scaler = StandardScaler()
scaler.fit(X_train)
#应用到训练集、测试集
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 处理之后的数据进行可视化
plt.plot(X_train_scaled.min(axis=0),'v',label = 'train set min')
plt.plot(X_train_scaled.max(axis=0),'^',label = 'train set max')
plt.plot(X_test_scaled.min(axis=0),'v',label = 'test set min')
plt.plot(X_test_scaled.max(axis=0),'^',label = 'test set max')
plt.yscale('log')
# 设置图标注位置，为最佳
plt.legend(loc = 'best')

# 设置坐标轴标题
plt.xlabel('scaled features')
plt.ylabel('scaled features distribute')
plt.show()
```

简而言之，归一化的目的就是使得预处理的数据被限定在一定的范围内（比如[0,1]或者[-1,1]），从而消除奇异样本数据导致的不良影响。
奇异样本数据是指相对于其他输入样本特别大或特别小的样本矢量（即特征向量）
奇异样本数据的存在会引起训练时间增大，同时也可能导致无法收敛，因此，当存在奇异样本数据时，在进行训练之前需要对预处理数据进行归一化；反之，不存在奇异样本数据时，则可以不进行归一化。



In [35]:
```
# 使用线性核和rbf核
for kernel in ['linear', 'rbf']:
    svr = SVR(kernel = kernel)
    svr.fit(X_train_scaled, y_train)
    print(kernel,"训练集得分:{}".format(svr.score(X_train_scaled, y_train)))
    print(kernel,"测试集得分:{}".format(svr.score(X_test_scaled, y_test)))
```

```
linear 训练集得分:0.7101563377300559
linear 测试集得分:0.6987138959194498
rbf 训练集得分:0.6902628301980511
rbf 测试集得分:0.6749681608781942
```

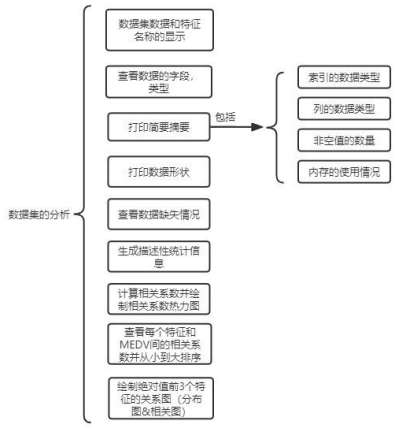
C表示模型对误差的惩罚系数，gamma反映了数据映射到高维特征空间后的分布；C越大，模型越容易过拟合；C越小，模型越容易欠拟合。
gamma越大，支持向量越多，gamma值越小，支持向量越少。gamma越小，模型的泛化性变好，但过小，模型实际上会退化为线性模型；gamma越大，理论上SVM可以拟合任何非线性数据。

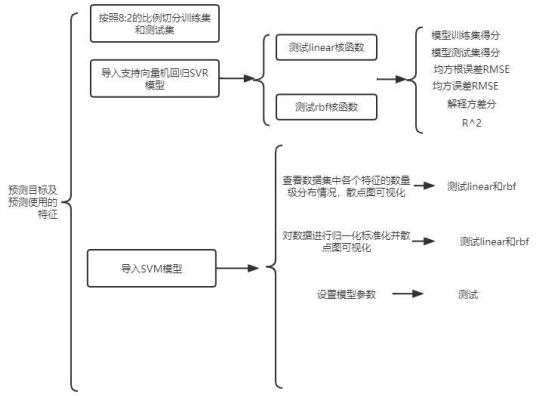
In [51]:
```
# 设置模型的参数
svr = SVR(C=100, gamma=0.2)

#C:表示错误项的惩罚系数C越大，即对分错样本的惩罚程度越大，因此在训练样本中准确率越高，但是泛化能力降低；
#相反，减小C的话，容许训练样本中有一些误分类错误样本，泛化能力强。
#对于训练样本带有噪声的情况，一般采用后者，把训练样本集中错误分类的样本作为噪声。

svr.fit(X_train_scaled, y_train)
print("训练集得分:{}".format(svr.score(X_train_scaled, y_train)))
print("测试集得分:{}".format(svr.score(X_test_scaled, y_test)))
```

```
训练集得分:0.9837888802764432
测试集得分:0.9045050499726706
```

```
D:\Anaconda 3 destination folder_\lib\site-packages\sklearn\utils\validation.py:1111: DataConversionWarning: A column-vector y was passed wh
en a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

二. 实现流程分析（绘制流程图）





三. 代码分析，介绍关键类和关键方法的使用方法，体现如何从对工具包一无所知到掌握工具包基本用法这个过程

1.

```
import numpy as np
import pandas as pd
import hvplot.pandas
import matplotlib.pyplot as plt
%matplotlib inline
# 使输出的图像以更高清的方式显示
%config InlineBackend.figure_format = 'retina'
import seaborn as sns
#用 matplotlib 能够完成一些基本的图表操作，Seaborn 库可以让这些图的表现更加丰富
```

```
# plt.style.use('ggplot')
plt.style.use("fivethirtyeight")
# Pandas 中只显示 3 位小数
pd.set_option('display.float_format', lambda x: '{:.3f}'.format(x))
from sklearn import datasets  # 导入数据集
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import warnings

boston = datasets.load_boston()
print(boston.keys())
#  data 代表特征矩阵，target 代表目标结果，feature_names 代表 data 对应的特征名称，
#DESCR 是对数据集的描述，filename 对应的是 boston 这个数据在本地的存放文件路径。

X = boston.data   # 数据
y = boston.target  # 目标
# X 为输入，y 为输出

#一个 Datarame 表示一个表格，类似电子表格的数据结构，包含一个经过排序的列表集，它的每一列都可以有不同的类型值（数字，字符串，布尔等等）。
#Datarame 有行和列的索引;它可以被看作是一个 Series 的字典(Series 们共享一个索引)。
df = pd.DataFrame(
    X,
    columns = boston.feature_names#特征名称
)
df.head() #df.head()会将 excel 表格中的第一行看作列名，并默认输出之后的五行

#CRIM：per capita crime rate by town 城镇人均犯罪率
#ZN： proportion of residential land zoned for lots over 25,000 sq.ft. 占地面积超过 2.5 万平方英尺的住宅用地比例
#INDUS：proportion of non-retail business acres per town 城镇上非零售业务地区的 比例
#CHAS：Charles River dummy variable (= 1 if tract bounds river; 0 otherwise) 虚拟变量；如果土地在查尔斯河，取值 1；否则为 0
#NOX： nitric oxides concentration (parts per 10 million) 一氧化氮浓度
#RM：average number of rooms per dwelling 平均每个居民房数
#AGE： proportion of owner-occupied units built prior to 1940 在 1940 年之前建成的所有者占用单位的比例
#DIS： weighted distances to five Boston employment centres 与波士顿的 5 个就业中心之间的加权距离
#RAD：  index of accessibility to radial highways 辐距离住房最近的公路入口编号
#TAX：full -value property-tax rate per $10,000 每 10,000 美元的全额物业税
#PTRATIO：pupil-teacher ratio by town 城镇师生比例大小
```

#B：1000(Bk - 0.63)^2 where Bk is the proportion of black people by town
1000(Bk-0.63)^2,其中 Bk 指代城镇中黑人的比例
#LSTAT：% lower status of the population 全部人口中地位较低人群的百分数大小
#MEDV：M edian value of owner-occupied homes in $1000's 目标变量，以 1000 美元来进行计算的自由住房的中位数大小

df["MEDV"] = y #MEDV：M edian value of owner-occupied homes in $1000's 目标变量，以 1000 美元来进行计算的自由住房的中位数大小
df.head() #df.head()会将 excel 表格中的第一行看作列名，并默认输出之后的五行

#查看数据字段，类型
df.columns

df.dtypes

df.info()
#打印 DataFrame 的简要摘要，显示有关 DataFrame 的信息，
#包括索引的数据类型 dtype 和列的数据类型 dtype，非空值的数量和内存使用情况。

#数据形状
df.shape #df.shape 会返回一个元组，该元组的第一个元素代表行数，第二个元素代表列数

#数据缺失情况
df.isnull().sum()

df.describe()
#describe()函数用于生成描述性统计信息。
#描述性统计数据：数值类型的包括均值，标准差，最大值，最小值，分位数等；
#类别的包括个数，类别的数目，最高数量的类别及出现次数等；输出将根据提供的内容而有所不同。

corr=df.corr()
#计算相关系数

#绘制相关系数的热力分布图
plt.figure(figsize=(16,10))
sns.heatmap(corr,annot=True,fmt=".2f",cmap="YlGn")
# annot=True：显示相关系数的数据，fmt=".2f"：只显示两位小数，cmap="YlGn"：设置热力图颜色
plt.show()

#查看每个特征和目标变量 MEDV 之间的相关系数
corr["MEDV"].sort_values()#默认 axis=0，按照指定列中数据大小排序

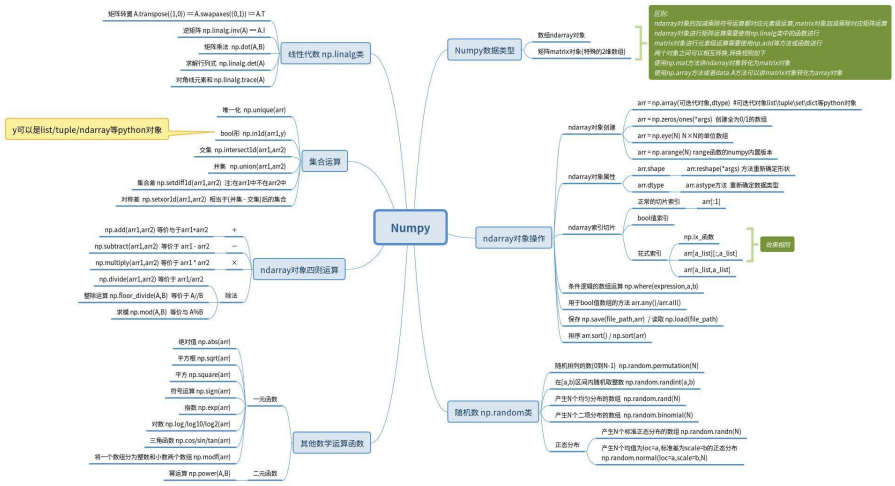
```
sns.pairplot(df[["LSTAT","INDUS","PTRATIO","MEDV"]]) # 绝对值靠前 3 的特征
#sns.pairplot() 用来展示两两特征之间的关系
#对角线上是各个属性的直方图（分布图），而非对角线上是两个不同属性之间的相关图
plt.show()

#划分给定的数据集，比例是 8：2
X = df.drop("MEDV",axis=1)#代表将“MEDV”对应的列标签沿着水平的方向依次删掉，axis = 0，按行计算，得到列的性质；axis = 1，按列计算，得到行的性质
y = df[["MEDV"]]

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2, #表示切分的训练集和测试集的比例
    random_state=8)
#random_stste:随机数种子——其实就是该组随机数的编号，在需要重复试验的时候，保证得到一组一样的随机数。

print("X_train shape: ",X_train.shape)
print("y_train shape: ",y_train.shape)
print("X_test shape: ",X_test.shape)
print("y_test shape: ",y_test.shape)
#df.shape 会返回一个元组，该元组的第一个元素代表行数，第二个元素代表列数

#导入支持向量机回归模型
from sklearn.svm import SVR
from sklearn.metrics import r2_score
from sklearn.metrics import explained_variance_score
#分别测试 linear 核函数和 rbf 核函数
for kernel in ['linear','rbf']:
    svr=SVR(kernel=kernel,gamma='auto')
    #gamma:该参数为核函数参数，只对 ‘rbf’,'poly'， ‘sigmod’ 有效
    #如果 gamma 设置为 auto,代表其值为样本特征值数的倒数，即 1/n_features，也有其他值可以设定，gamma 默认 auto
    #kernel :该参数用于选择模型所使用的核函数
    #算法中常用的核函数：linear ：线性核函数，poly:多项式核函数
    #rbf：径像核函数/高斯核，sigmod: sigmod 核函数
    #precomputed:核矩阵，该矩阵表示自己事先计算好的，输入后算法内部将使用你提供的矩阵进行计算
    svr.fit(X_train,y_train)#训练
    y_pre=svr.predict(X_test)
    print(kernel,"核函数的模型训练集得分:{:.4f}".format(svr.score(X_train, y_train)))
    print(kernel,"核函数的模型测试集得分:{:.4f}".format(svr.score(X_test,
```

```
y_test)))
    print('均方根误差
RMSE:{:.4f}'.format(np.sqrt(mean_squared_error(y_test,y_pre))))
    print('均方误差
MSE:{:.4f}'.format(np.round(mean_squared_error(y_test,y_pre),2)))
    print('解释方差
分:{:.4f}'.format(np.round(explained_variance_score(y_test,y_pre),2)))
    print('R^2 得分:{:.4f}'.format(np.round(r2_score(y_test,y_pre),2)))

    #SVM 算法对于数据预处理的要求比较高，比如：数据特征量级相差较大，则就要对数
据进行预处理。
    # 查看数据集中各个特征的数量级分布情况,使用散点图进行可视化
    import matplotlib.pyplot as plt
    %matplotlib inline
    plt.plot(X.min(axis = 0), 'v', label='min')
    plt.plot(X.max(axis = 0), '^', label='max')
    # 设定纵坐标为对数形式
    plt.yscale('log')
    # 设置图注位置为最佳
    plt.legend(loc='best')
    # 设置最标轴标题
    plt.xlabel('features')
    plt.ylabel('features distribute')
    # 显示图像
    plt.show()

    # 导入数据处理工具
    from sklearn.preprocessing import StandardScaler
    #StandardScaler 类是一个用来讲数据进行归一化和标准化的类。
    #-计算训练集的平均值和标准差，以便测试数据集使用相同的变换。
    #标准差标准化（standardScale）使得经过处理的数据符合标准正态分布，对于每个属
性/每列来说所有数据都聚集在 0 附近，标准差为 1，
    #使得新的 X 数据集方差为 1，均值为 0

    # 对训练集和测试集进行数据预处理
    scaler = StandardScaler()
    scaler.fit(X_train)
    #应用到训练集、测试集
    X_train_scaled = scaler.transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    # 处理之后的数据进行可视化
    plt.plot(X_train_scaled.min(axis=0),'v',label = 'train set min')
    plt.plot(X_train_scaled.max(axis=0),'^',label = 'train set max')
```

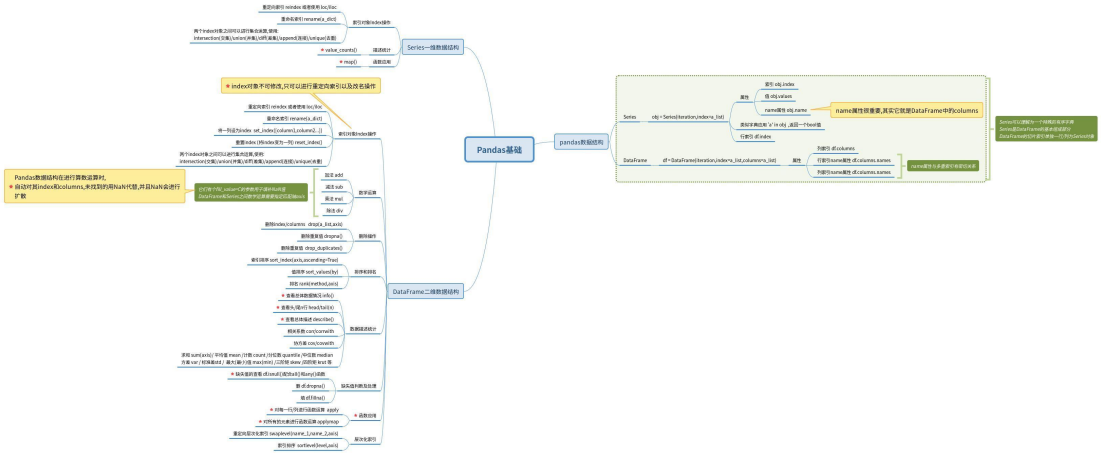
```
    plt.plot(X_test_scaled.min(axis=0),'v',label = 'test set min')
    plt.plot(X_test_scaled.max(axis=0),'^',label = 'test set max')
    plt.yscale('log')
    # 设置图标注位置，为最佳
    plt.legend(loc = 'best')

    # 设置坐标轴标题
    plt.xlabel('scaled features')
    plt.ylabel('scaled features distribute')
    plt.show()

    # 使用线性核和 rbf 核
    for kernel in ['linear', 'rbf']:
        svr = SVR(kernel = kernel)
        svr.fit(X_train_scaled, y_train)
        print(kernel,"训练集得分:{}".format(svr.score(X_train_scaled,
y_train)))
        print(kernel,"测试集得分:{}".format(svr.score(X_test_scaled, y_test)))

    # 设置模型的参数
    svr = SVR(C=100, gamma=0.2)

    #C:表示错误项的惩罚系数 C 越大，即对分错样本的惩罚程度越大，因此在训练样本中准确率越高，但是泛化能力降低；
    #相反，减小 C 的话，容许训练样本中有一些误分类错误样本，泛化能力强。
    #对于训练样本带有噪声的情况，一般采用后者，把训练样本集中错误分类的样本作为噪声。

    svr.fit(X_train_scaled, y_train)
    print("训练集得分:{}".format(svr.score(X_train_scaled, y_train)))
    print("测试集得分:{}".format(svr.score(X_test_scaled, y_test)))
```

2. 关键类和关键方法的使用方法，及工具包的使用

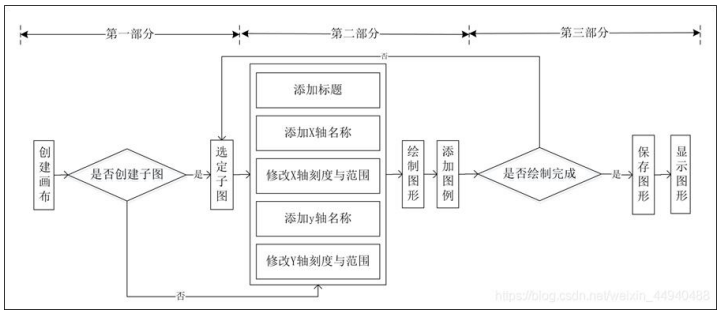
（1） Numpy



(2) Pandas



(3) Matplotlib.pyplot



函数名称	函数作用
plt. **figure**(figsize, facecolor)	**创建一个空白画布**，figsize参数可以指定画布大小，像素，单位为英寸。
figure.**add_subplot**()	**创建并选中子图**，可以指定子图的行数，列数，与选中图片编号。

函数名称	函数作用
plt.**plot**(x,y,ls,lw,lable,color)	根据x，y数据**绘制直线、曲线**、标记点，ls为线型linestyle，lw为线宽linewidth，lable为标签文本内容，color为颜色。
plt. **scatter**(x, y, c, marker, label, color)	**绘制散点图**：x、y为相同长度的序列，c为单个颜色字符或颜色序列，marker为标记的样式，默认的是'o'，label为标签文本内容，color为颜色
plt. **bar**(x, height, width, bottom)	**绘制条形图**
plt. **pie**(x, explode, labels, autopct, shadow = False, startangle)	**绘制饼图**
Plt. **stem**(x, y, linefmt, markerfmt, use_line_collection)	**绘制stem图**
plt.**title**(string)	在当前图形中**添加标题**，可以指定标题的名称、位置、颜色、字体大小等参数。
plt.**xlabel**(string)	在当前图形中**添加x轴名称**，可以指定位置、颜色、字体大小等参数。
plt.**ylabel**(string)	在当前图形中**添加y轴名称**，可以指定位置、颜色、字体大小等参数。
plt.**xlim**(xmin,xmax)	**指定当前图形x轴的范围**，只能确定一个数值区间，而无法使用字符串标识。
plt.**ylim**(ymin,ymax)	**指定当前图形y轴的范围**，只能确定一个数值区间，而无法使用字符串标识。
plt.**xticks**()	**指定x轴刻度的数目与取值**。
plt.**yticks**()	**指定y轴刻度的数目与取值**。
plt.**legend**()	**指定当前图形的图例**，可以指定图例的大小、位置、标签。

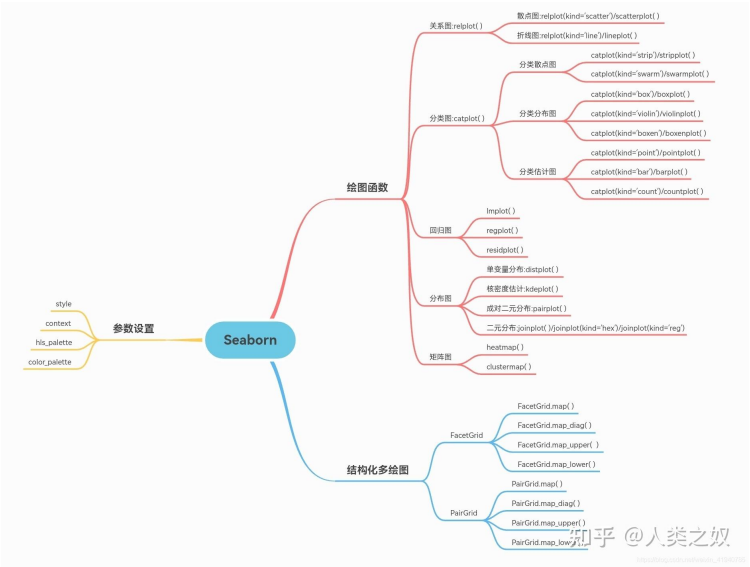
plt.**xlim**(xmin,xmax)	**指定当前图形x轴的范围**，只能确定一个数值区间，而无法使用字符串标识。
plt.**ylim**(ymin,ymax)	**指定当前图形y轴的范围**，只能确定一个数值区间，而无法使用字符串标识。
plt.**xticks**()	**指定x轴刻度的数目与取值**。
plt.**yticks**()	**指定y轴刻度的数目与取值**。
plt.**legend**()	**指定当前图形的图例**，可以指定图例的大小、位置、标签。

函数名称	函数作用
plt.**savafig**()	**保存绘制的图片**，可以指定图片的分辨率、边缘的颜色等参数。
plt.**show**()	**在本机显示图形**。

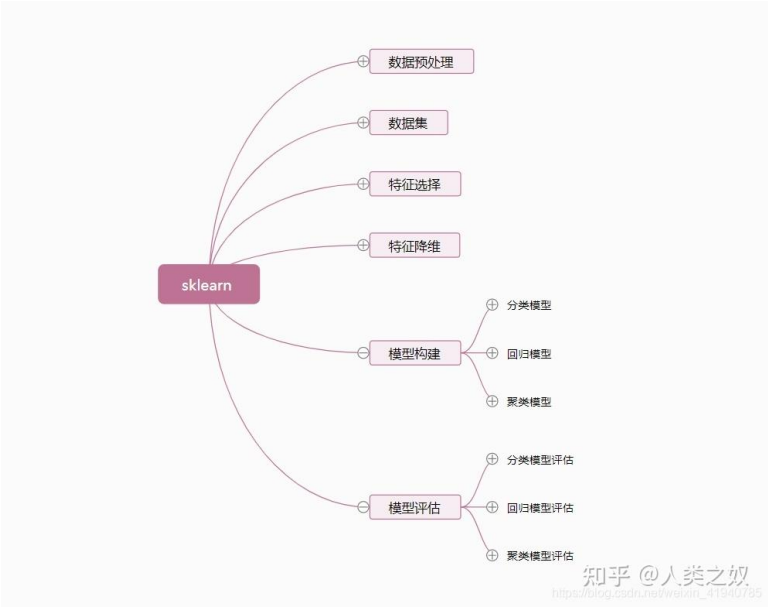
函数	说明
plt.plot()	绘制直线、曲线图
plt.boxplot()	绘制箱形图
plt.bar()	绘制条形图
plt.barh()	绘制横向条形图
plt.polar()	绘制极坐标图
plt.pie()	绘制饼图
plt.psd()	绘制功率谱密度图
plt.specgram()	绘制谱图
plt.cohere()	绘制相关性函数
plt.scatter()	绘制散点图
plt.step()	绘制步阶图
plt.hist()	绘制直方图

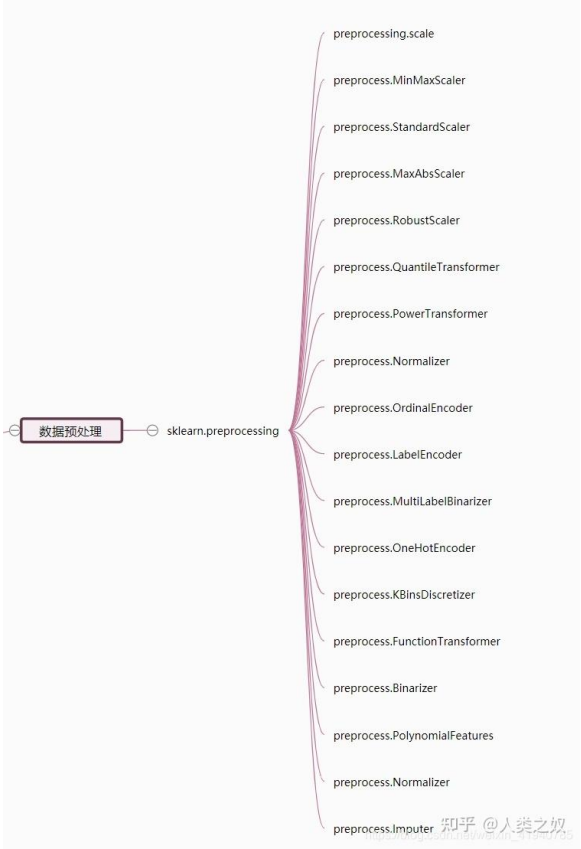
plt.contour()	绘制等值图
plt.vlines()	绘制垂直图
plt.stem()	绘制柴火图
plt.plot_date()	绘制数据日期
plt.clabel()	绘制轮廓图
plt.hist2d()	绘制2D直方图
plt.quiverkey()	绘制颤动图
plt.stackplot()	绘制堆积面积图
plt.Violinplot()	绘制小提琴图

(4) Seaborn

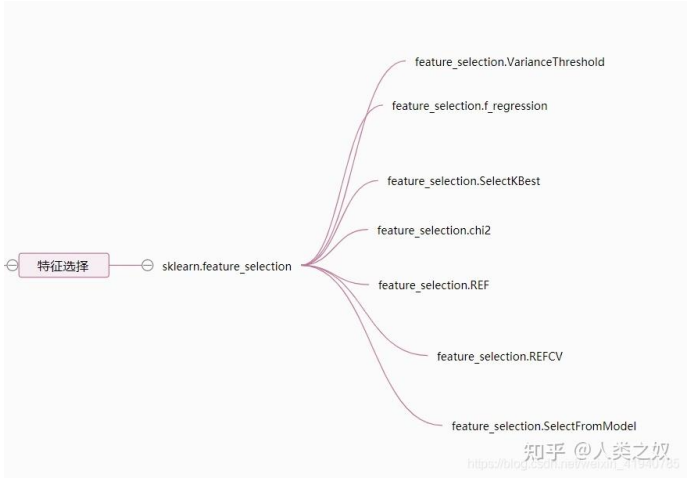

Seaborn
参数设置
style
context
his_palette
color_palette
绘图函数
关系图:relplot( )
散点图:relplot(kind='scatter')/scatterplot( )
折线图:relplot(kind='line')/lineplot( )
分类图:catplot( )
分类散点图
catplot(kind='strip')/stripplot( )
catplot(kind='swarm')/swarmplot( )
分类分布图
catplot(kind='box')/boxplot( )
catplot(kind='violin')/violinplot( )
catplot(kind='boxen')/boxenplot( )
分类估计图
catplot(kind='point')/pointplot( )
catplot(kind='bar')/barplot( )
catplot(kind='count')/countplot( )
回归图
lmplot( )
regplot( )
residplot( )
分布图
单变量分布:distplot( )
核密度估计:kdeplot( )
成对二元分布:pairplot( )
二元分布:jointplot( )/joinplot(kind='hex')/joinplot(kind='reg')
矩阵图
heatmap( )
clustermap( )
结构化多绘图
FacetGrid
FacetGrid.map( )
FacetGrid.map_diag( )
FacetGrid.map_upper( )
FacetGrid.map_lower( )
PairGrid
PairGrid.map( )
PairGrid.map_diag( )
PairGrid.map_upper( )
PairGrid.map_lower( )
知乎 @人类之奴

(5) Sklearn

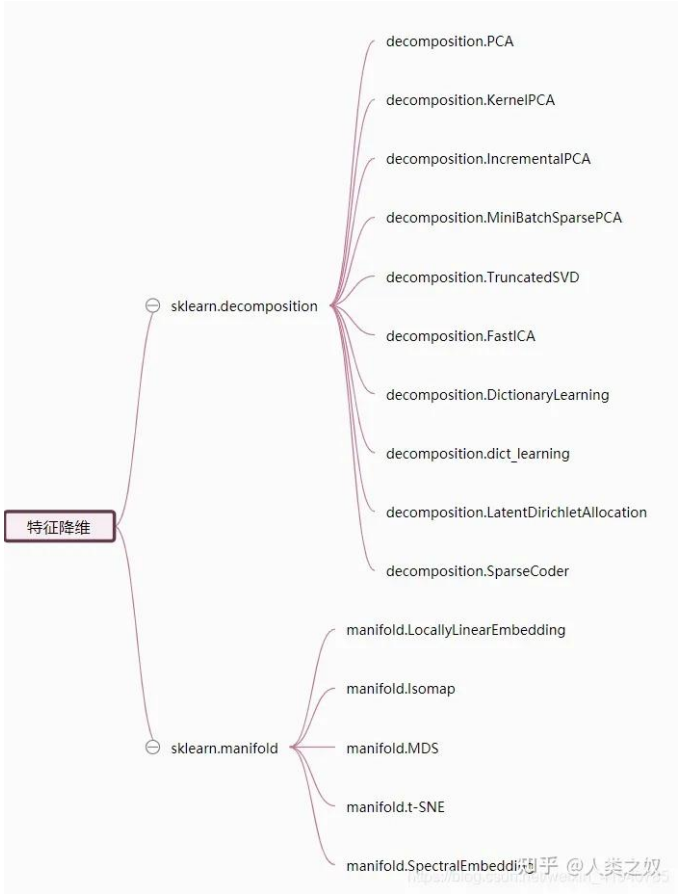

sklearn
数据预处理
数据集
特征选择
特征降维
模型构建
分类模型
回归模型
聚类模型
模型评估
分类模型评估
回归模型评估
聚类模型评估
知乎 @人类之奴
https://blog.csdn.net/weixin_41940755



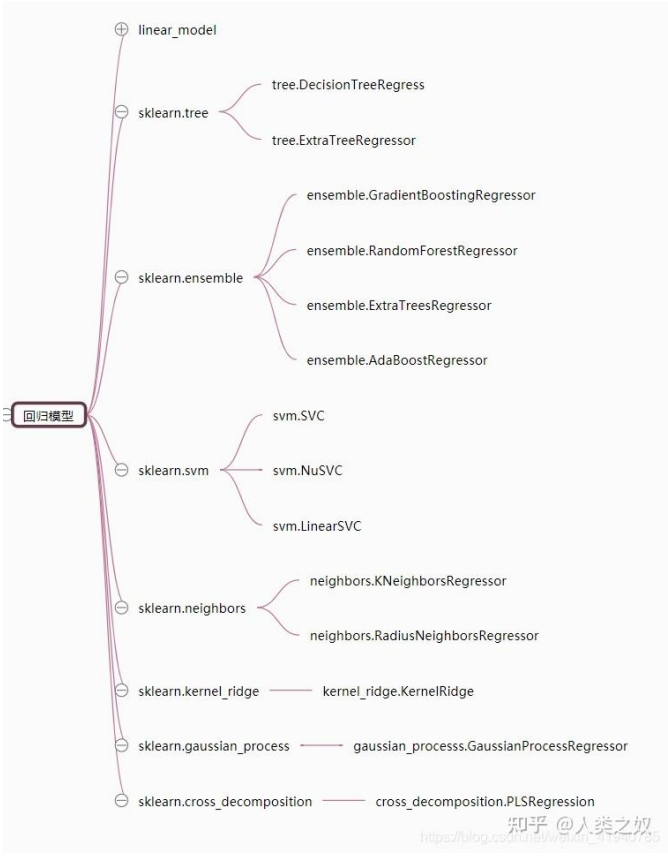
函数	功能
preprocessing.scale( )	标准化
preprocessing.MinMaxScaler( )	最大最小值标准化
preprocessing.StandardScaler( )	数据标准化
preprocessing.MaxAbsScaler( )	绝对值最大标准化
preprocessing.RobustScaler( )	带离群值数据集标准化
preprocessing.QuantileTransformer( )	使用分位数信息变换特征
preprocessing.PowerTransformer( )	使用幂变换执行到正态分布的映射
preprocessing.Normalizer( )	正则化
preprocessing.OrdinalEncoder( )	将分类特征转换为分类数值
preprocessing.LabelEncoder( )	将分类特征转换为分类数值
preprocessing.MultiLabelBinarizer( )	多标签二值化
preprocessing.OneHotEncoder( )	独热编码
preprocessing.KBinsDiscretizer( )	将连续数据离散化
preprocessing.FunctionTransformer( )	自定义特征处理函数
preprocessing.Binarizer( )	特征二值化
preprocessing.PolynomialFeatures( )	创建多项式特征
preprocesssing.Normalizer( )	正则化
preprocessing.Imputer( )	弥补缺失值

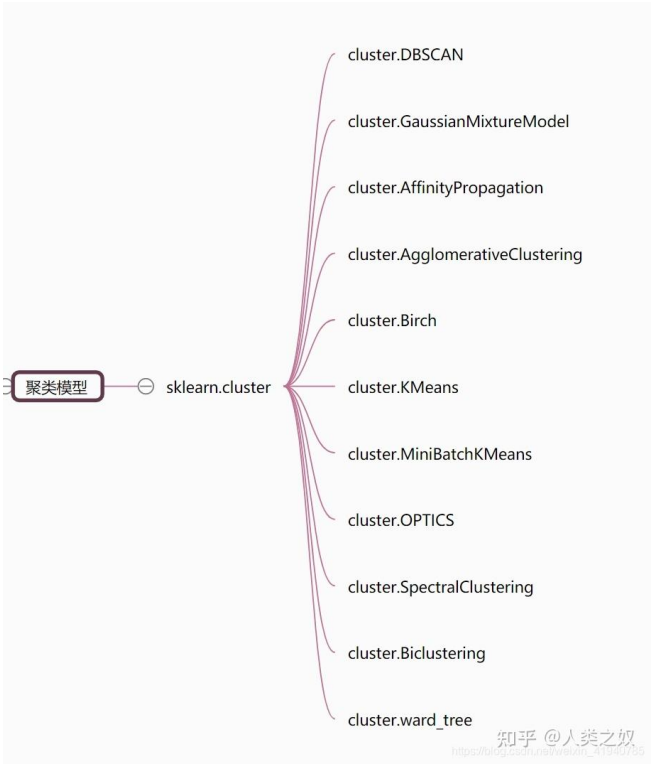


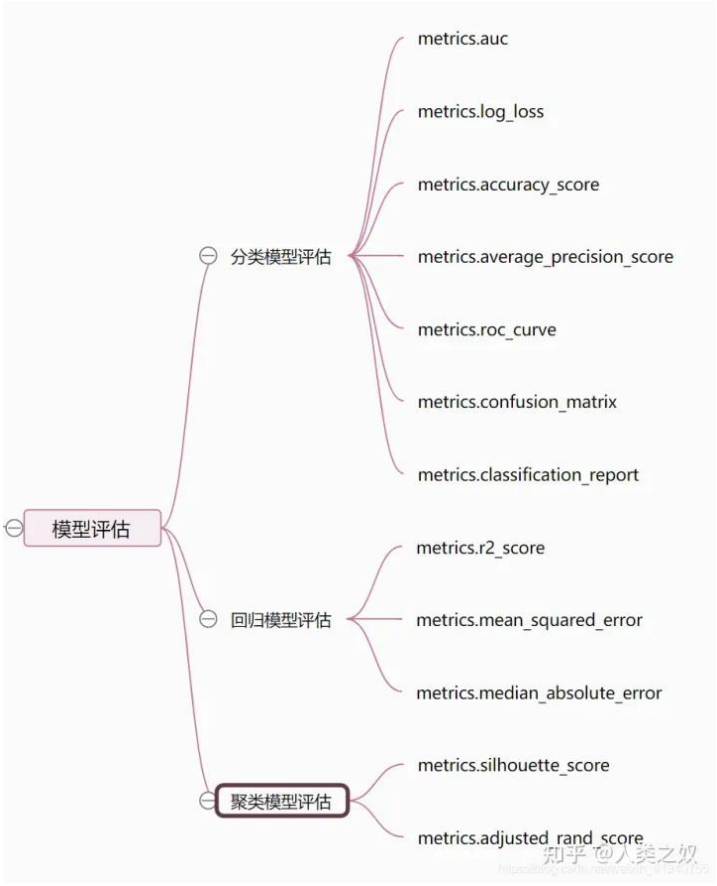
函数	功能
feature_selection.SelectKBest( ) feature_selection.chi2 feature_selection.f_regression feature_selection.mutual_info_regression	选择K个得分最高的特征
feature_selection.VarianceThreshold( )	无监督特征选择
feature_selection.REF( )	递归式特征消除
feature_selection.REFCV( )	递归式特征消除交叉验证法
feature_selection.SelectFromModel( )	特征选择



函数	功能
decomposition.PCA( )	主成分分析
decomposition.KernelPCA( )	核主成分分析
decomposition.IncrementalPCA( )	增量主成分分析
decomposition.MiniBatchSparsePCA( )	小批量稀疏主成分分析
decomposition.SparsePCA( )	稀疏主成分分析
decomposition.FactorAnalysis( )	因子分析
decomposition.TruncatedSVD( )	截断的奇异值分解
decomposition.FastICA( )	独立成分分析的快速算法
decomposition.DictionaryLearning( )	字典学习
decomposition.MiniBatchDictonaryLearning ( )	小批量字典学习
decomposition.dict_learning( )	字典学习用于矩阵分解
decomposition.dict_learning_online( )	在线字典学习用于矩阵分解
decomposition.LatentDirichletAllocation( )	在线变分贝叶斯算法的隐含迪利克雷分布
decomposition.NMF( )	非负矩阵分解
decomposition.SparseCoder( )	稀疏编码


回归模型
linear_model
sklearn.tree
tree.DecisionTreeRegress
tree.ExtraTreeRegressor
sklearn.ensemble
ensemble.GradientBoostingRegressor
ensemble.RandomForestRegressor
ensemble.ExtraTreesRegressor
ensemble.AdaBoostRegressor
sklearn.svm
svm.SVC
svm.NuSVC
svm.LinearSVC
sklearn.neighbors
neighbors.KNeighborsRegressor
neighbors.RadiusNeighborsRegressor
sklearn.kernel_ridge
kernel_ridge.KernelRidge
sklearn.gaussian_process
gaussian_processs.GaussianProcessRegressor
sklearn.cross_decomposition
cross_decomposition.PLSRegression
知乎 @人类之奴


聚类模型
sklearn.cluster
cluster.DBSCAN
cluster.GaussianMixtureModel
cluster.AffinityPropagation
cluster.AgglomerativeClustering
cluster.Birch
cluster.KMeans
cluster.MiniBatchKMeans
cluster.OPTICS
cluster.SpectralClustering
cluster.Biclustering
cluster.ward_tree
知乎 @人类之奴



四． 结束语，对本次实验遇到的问题及解决方法进行总结

1. SVM 优点

非线性映射是 SVM 方法的理论基础,SVM 利用内积核函数代替向高维空间的非线性映射；

对特征空间划分的最优超平面是 SVM 的目标,最大化分类边际的思想是 SVM 方法的核心；

支持向量是 SVM 的训练结果,在 SVM 分类决策中起决定作用的是支持向量；

SVM 的最终决策函数只由少数的支持向量所确定,计算的复杂性取决于支持向量的数目,而不是样本空间的维数,这在某种意义上避免了“维数灾难”。

小样本集上分类效果通常比较好。

少数支持向量决定了最终结果,这不但可以帮助我们抓住关键样本、“剔除”大量冗余样本,而且注定了该方法不但算法简单,而且具有较好的“鲁棒”性。这种“鲁棒”性主要体现在:

①增、删非支持向量样本对模型没有影响;

②支持向量样本集具有一定的鲁棒性;

③有些成功的应用中,SVM 方法对核的选取不敏感

SVM 是一种有坚实理论基础的新颖的小样本学习方法。它基本上不涉及概率测度及大数定律等,因此不同于现有的统计方法。从本质上看,它避开了从归纳到演绎的传统过程,实现了高效的从训练样本到预报样本的“转导推理”,大大简化了通常的分类和回归等问题。

2. SVM 是一种二分类模型。它的基本模型是在特征空间中寻找间隔最大化的分离超平面的线性分类器。（间隔最大化是它的独特之处），通过该超平面实现对未知样本集的分类。

当训练样本线性可分时，通过硬间隔最大化，学习一个线性分类器，即线性可分支持向量机。

当训练数据近似线性可分时，引入松弛变量，通过软间隔最大化，学习一个线性分类器，即线性支持向量机。

当训练数据线性不可分时，通过使用核技巧及软间隔最大化，学习非线性支持向量机。

3. 本实验通过对波士顿房价，了解了线性回归模型的原理，学会了数据集的划分和数据标准化处理，学会使用 sklearn 中的线性回归模型进行训练和预测。

在完成实验内容的过程中，遇到了一些困难，如在计算 svr.score，均方误差，均方根误差，解释方差时由于对于函数不熟悉，在添加参数时出了问题，最终通查阅官方文档得到解决。在进行房价预测结果的可视化时，通过查阅关于 matplotlib 相关资料，修改了图像的格式，并添加图例使视图更加清晰