#数据处理

import numpy as np

import pandas as pd

#数据可视化

import matplotlib

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

# 使用r2\_score 作为回归模型性能的评价

from sklearn.metrics import r2\_score

#显示中文

matplotlib.rcParams['font.sans-serif'] = ['SimHei']

matplotlib.rcParams['font.family'] = ['sans-serif']

#解决-变成方框问题

matplotlib.rcParams['axes.unicode\_minus'] = False

#读取数据

dpath = "./"

df = pd.read\_csv(dpath + "FE\_Advertising.csv")

#通过观察前5 行，了解数据每列（特征）的概况

df.head()

# 数据总体信息

df.info()

# 从原始数据中分离输入特征x 和输出y

y = df['sales']

X = df.drop('sales', axis = 1)

#特征名称，用于后续显示权重系数对应的特征

feat\_names = X.columns

# 从原始数据中分离输入特征x 和输出y

y = df['sales']

X = df.drop('sales', axis = 1)

#特征名称，用于后续显示权重系数对应的特征

feat\_names = X.columns

# 1、最小二乘线性回归

from sklearn.linear\_model import LinearRegression

# 1.使用默认配置初始化学习器实例

lr = LinearRegression()

# 2.用训练数据训练模型参数

lr.fit(X\_train, y\_train)

# 3. 用训练好的模型对测试集进行预测

y\_test\_pred\_lr = lr.predict(X\_test)

y\_train\_pred\_lr = lr.predict(X\_train)

#性能评估，R 方分数

print("The r2 score of LinearRegression on test is %f" %(r2\_score(y\_test, y\_test\_pred\_lr)))

print("The r2 score of LinearRegression on train is %f" %(r2\_score(y\_train, y\_train\_pred\_lr)))

# 看看各特征的权重系数，系数的绝对值大小可视为该特征的重要性

fs = pd.DataFrame({"columns":list(feat\_names), "coef":list((lr.coef\_.T))})

#lr.coef\_.T 权重 lr.intercept\_ 截距

#fs.sort\_values(by=['coef'],ascending=False)

fs = fs.append([{'columns':'intercept','coef':lr.intercept\_}], ignore\_index=True)

fs

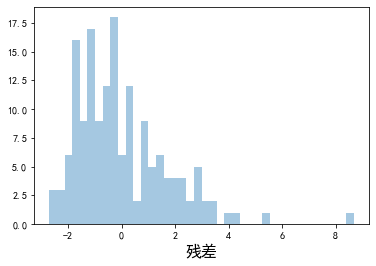
#在训练集上观察预测残差的分布，看是否符合模型假设：噪声为0 均值的高斯噪声

figsize = 11,9 # 设置输出的图片大小

res = y\_train\_pred\_lr - y\_train

sns.distplot(res, bins=40, kde = False)

plt.xlabel(u'残差', fontsize = 16)

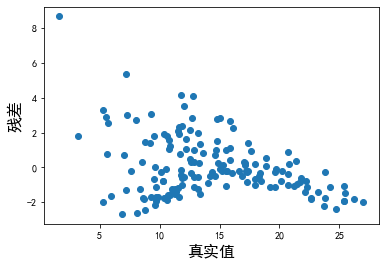


figsize = 11,9

plt.scatter(y\_train, res)

plt.xlabel(u'真实值', fontsize = 16)

plt.ylabel(u'残差', fontsize = 16)



#2、岭回归

from sklearn.linear\_model import RidgeCV

#1. 设置超参数（正则参数）范围

alphas = [ 0.01, 0.1, 1, 10, 100]

#2. 生成一个RidgeCV 实例

ridge = RidgeCV(alphas=alphas, store\_cv\_values=True)

#3. 模型训练

ridge.fit(X\_train, y\_train)

#4. 预测

y\_test\_pred\_ridge = ridge.predict(X\_test)

y\_train\_pred\_ridge = ridge.predict(X\_train)

#模型性能评估

print("The r2 score of Ridge on test is %f" %(r2\_score(y\_test, y\_test\_pred\_ridge)))

print("The r2 score of Ridge on train is %f" %(r2\_score(y\_train, y\_train\_pred\_ridge)))

# 看看各特征的权重系数，系数的绝对值大小可视为该特征的重要性

fs = pd.DataFrame({"columns":list(feat\_names), "coef":list((ridge.coef\_.T))})

#fs.sort\_values(by=['coef'],ascending=False)

fs = fs.append([{'columns':'intercept','coef':ridge.intercept\_}], ignore\_index=True)

fs

mse\_mean = np.mean(ridge.cv\_values\_, axis = 0)

plt.plot(np.log10(alphas), mse\_mean.reshape(len(alphas),1))

#最佳超参数

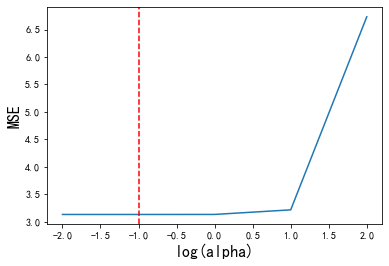
plt.axvline(np.log10(ridge.alpha\_), color='r', ls='--')

plt.xlabel('log(alpha)', fontsize = 16)

plt.ylabel('MSE', fontsize = 16)

plt.show()

print ('alpha is:', ridge.alpha\_)



#3、Lasso回归

from sklearn.linear\_model import LassoCV

#1. 设置超参数搜索范围

#Lasso 可以自动确定最大的alpha，所以另一种设置alpha 的方式是设置最小的alpha 值（eps） 和 超参数的数目（n\_alphas），

#然后LassoCV 对最小值和最大值之间在log 域上均匀取值n\_alphas 个

# np.logspace(np.log10(alpha\_max \* eps), np.log10(alpha\_max),num=n\_alphas)[::-1]

#2 生成LassoCV 实例（默认超参数搜索范围）

lasso = LassoCV()

#3. 训练（内含CV）

lasso.fit(X\_train, y\_train)

#4. 测试

y\_test\_pred\_lasso = lasso.predict(X\_test)

y\_train\_pred\_lasso = lasso.predict(X\_train)

# 评估，使用r2\_score 评价模型在测试集和训练集上的性能

print("The r2 score of lasso on test is %f" %(r2\_score(y\_test, y\_test\_pred\_lasso)))

print("The r2 score of lasso on train is %f" %(r2\_score(y\_train, y\_train\_pred\_lasso)))

# 看看各特征的权重系数，系数的绝对值大小可视为该特征的重要性

fs = pd.DataFrame({"columns":list(feat\_names), "coef\_lr":list((lr.coef\_.T)), "coef\_ridge":list((ridge.coef\_.T)), "coef\_lasso":list((lasso.coef\_.T))})

#fs.sort\_values(by=['coef\_lr'],ascending=False)

fs = fs.append([{'columns':'intercept','coef\_lr':lr.intercept\_, 'coef\_ridge':ridge.intercept\_, 'coef\_lasso':lasso.intercept\_}], ignore\_index=True)

fs

mses = np.mean(lasso.mse\_path\_, axis = 1)

plt.plot(np.log10(lasso.alphas\_), mses)

#最佳超参数

plt.axvline(np.log10(lasso.alpha\_), color='r', ls='--')

plt.xlabel('log(alpha)', fontsize = 16)

plt.ylabel('MSE', fontsize = 16)

plt.show()

print ('alpha is:', lasso.alpha\_)

形状

描述已自动生成

#4、弹性网络回归

from sklearn.linear\_model import ElasticNetCV

#1. 设置超参数搜索范围

#Lasso 可以自动确定最大的alpha，所以另一种设置alpha 的方式是设置最小的alpha 值（eps） 和 超参数的数目（n\_alphas），

#然后LassoCV 对最小值和最大值之间在log 域上均匀取值n\_alphas 个

# np.logspace(np.log10(alpha\_max \* eps), np.log10(alpha\_max),num=n\_alphas)[::-1]

l1\_ratio = [0.01, 0.1, .5, .7, .9, .95, .99, 1]

#2 ElasticNetCV（设置超参数搜索范围）

elastic\_net = ElasticNetCV(l1\_ratio = l1\_ratio )

#3. 训练（内含CV）

elastic\_net.fit(X\_train, y\_train)

#4. 测试

y\_test\_pred\_elastic\_net = elastic\_net.predict(X\_test)

y\_train\_pred\_elastic\_net = elastic\_net.predict(X\_train)

# 评估，使用r2\_score 评价模型在测试集和训练集上的性能

print("The r2 score of elastic\_net on test is %f" %(r2\_score(y\_test, y\_test\_pred\_elastic\_net)))

print("The r2 score of elastic\_net on train is %f" %(r2\_score(y\_train, y\_train\_pred\_elastic\_net)))

# 看看各特征的权重系数，系数的绝对值大小可视为该特征的重要性

fs = pd.DataFrame({"columns":list(feat\_names), "coef\_lr":list((lr.coef\_.T)), "coef\_ridge":list((ridge.coef\_.T)), "coef\_lasso":list((lasso.coef\_.T)), 'coef\_elastic\_net':list((elastic\_net.coef\_.T))})

#fs.sort\_values(by=['coef\_lr'],ascending=False)

fs = fs.append([{'columns':'intercept','coef\_lr':lr.intercept\_, 'coef\_ridge':ridge.intercept\_, 'coef\_lasso':lasso.intercept\_, 'coef\_elastic\_net':elastic\_net.intercept\_}], ignore\_index=True)

fs

mses = np.mean(elastic\_net.mse\_path\_, axis = 2)

# plot results

n\_l1\_ratio = len(l1\_ratio)

n\_alpha = elastic\_net.alphas\_.shape[1]

for i, l1 in enumerate(l1\_ratio):

plt.plot(np.log10(elastic\_net.alphas\_[i]), mses[i], label= u'l1 正则比例:'+str(l1))

#最佳超参数

plt.axvline(np.log10(elastic\_net.alpha\_), color='r', ls='--')

plt.xlabel('log(alpha)', fontsize = 16)

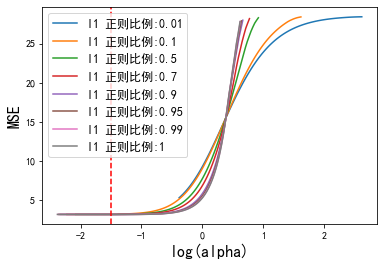
plt.ylabel('MSE', fontsize = 16)

plt.legend(fontsize = 12)

plt.show()

print ('alpha is:', elastic\_net.alpha\_)

print ('l1\_ratio is:', elastic\_net.l1\_ratio\_)



#最佳超参数

plt.axvline(np.log10(elastic\_net.alpha\_), color='r', ls='--')

plt.xlabel('log(alpha)', fontsize = 16)

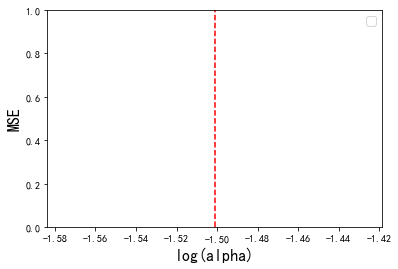
plt.ylabel('MSE', fontsize = 16)

plt.legend(fontsize = 12)

plt.show()

print ('alpha is:', elastic\_net.alpha\_)

print ('l1\_ratio is:', elastic\_net.l1\_ratio\_)



# 5、Huber 回归

from sklearn.linear\_model import HuberRegressor

# 1.使用默认配置初始化学习器实例

hr = HuberRegressor()

# 2.用训练数据训练模型参数

hr.fit(X\_train, y\_train)

# 3. 用训练好的模型对测试集进行预测

y\_test\_pred\_hr = hr.predict(X\_test)

y\_train\_pred\_hr = hr.predict(X\_train)

# 看看各特征的权重系数，系数的绝对值大小可视为该特征的重要性

fs = pd.DataFrame({"columns":list(feat\_names), "coef":list((hr.coef\_.T))})

#fs.sort\_values(by=['coef'],ascending=False)

fs = fs.append([{'columns':'intercept','coef':hr.intercept\_}], ignore\_index=True)

fs

#在训练集上观察预测残差的分布，看是否符合模型假设：噪声为0 均值的高斯噪声

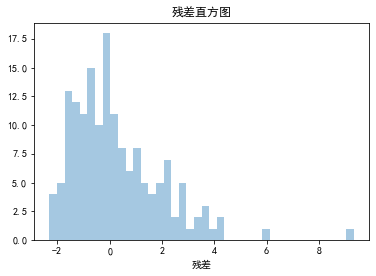
figsize = 11,9

res = y\_train\_pred\_hr - y\_train

sns.distplot(res, bins=40, kde = False)

plt.xlabel(u'残差')

plt.title(u'残差直方图')



figsize = 11,9

res = y\_train - y\_train\_pred\_hr

plt.scatter(y\_train\_pred\_hr, res)

plt.xlabel(u'预测值')

plt.ylabel(u'残差')

