一、 实验过程

1. 数据集及预处理

1.1 数据描述

该数据集包含从 6 年（2006-2011）的联合循环发电厂收集的 9568 个数据点，一共包含五个特征属性：AP、V、T、RH、PE，这五个数据均为连续形数值变量。

属性名	属性说明	类型
AP	环境压力	continuous.
V	排气真空度	continuous.
T	每小时平均环境变量温度	continuous.
RH	相对湿度	continuous.
PE	设备的净小时电能输出	continuous.

预测任务是根据这四个特征 AP、T、V、RH 以预测设备的净小时电能输出（PE）。

1.2 数据处理

- 去重处理

查找出数据重复行，进行删除

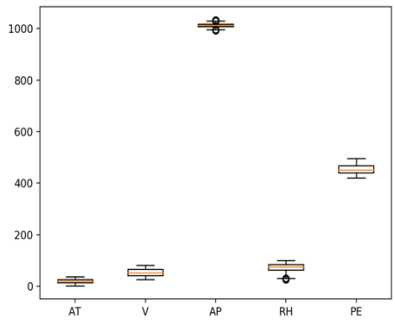
```
#查找出数据重复行
dIndex = data.duplicated()
print(dIndex)
#去除重复
data = data.drop_duplicates()
```

- 异常值处理

异常值处理主要是对这四个连续型属性数据进行处理，主要依靠箱型图的方式来观察异常值。

```
# 全部特征箱形图
 plt.boxplot(x=data.values,labels=data.columns,whis=1.5)
 plt.show()
 data.head()
 print(data.head())
```

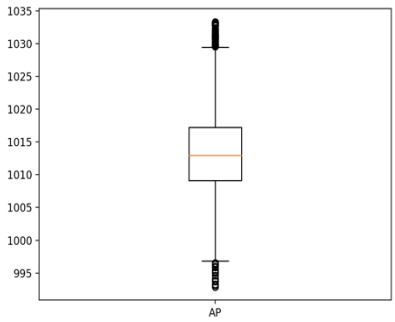
通过绘制图形特征的箱型图，可以判断 AP，RH 存在异常值。

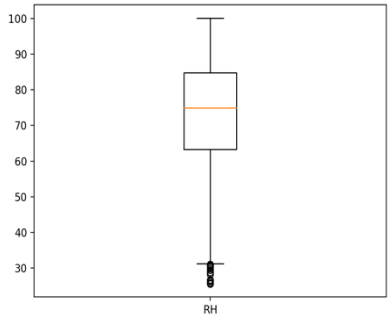


分别画出 AP,RH 的箱型图进一步观察。

```
1.   plt.boxplot(x=data['AP'].values,labels=['AP'],whis=1.5)
2.   plt.show()
3.  data.head()
4.   print(data.head())
```

观察 AP 的箱型图，AP 的异常范围判定为：大于 1029，小于 996
观察 RH 的箱型图，RH 的异常范围判定为：小于 31





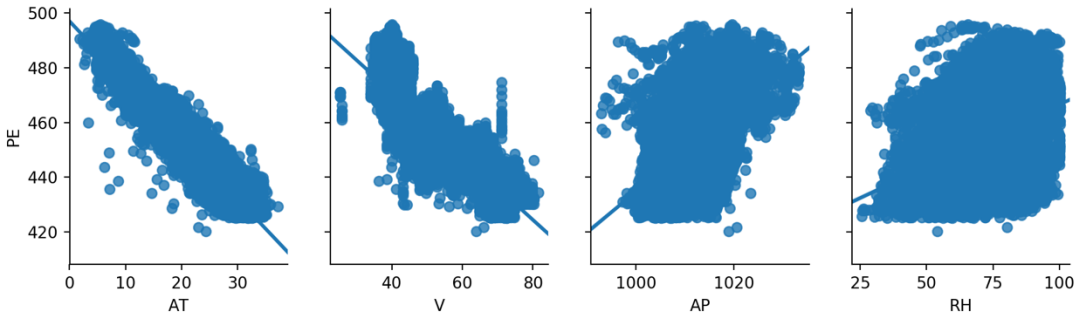

```
1.  # 处理异常值
2.  print(len(data[data.AP>1029]))
3.  print(len(data[data.AP<996]))
4.  print(len(data[data.RH<31]))
```

首先查找出存在异常值的行的数目，分别得出 AP 大于 1029 的行数有 83 个，小于 996 的行数有 11 个，RH 小于 31 的行数有 12 个，总远小于数据总数，因此选择直接去掉存在异常值的行

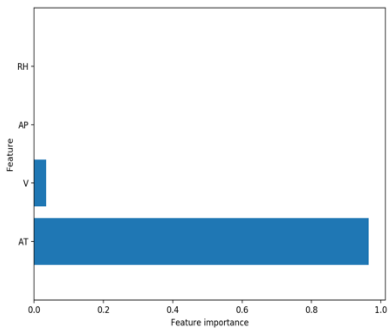
```
1.  data[data.AP>1029]=None
2.  data[data.AP<996]=None
3.  data[data.RH<31]=None
4.  data=data.dropna()
```

- 线性特征分析

通过 seaborn 绘制 X 特征（AP、V、T、RH、PE）的每一维度和对于 Y（PE）的散点图，通过参数 kind 可以添加各个特征与 Y 之间一条最佳拟合曲线，分析特征向量 AT、V、AP、RH 与 PE 之间的关系，发现 AT、V 与 PE 之间线性关系较强，而 AP、RH 关系较弱。



同时通过随机森林回归来判断各个特征的重要程度，从图中可以看出，AT、V 的重要程度较高，而 AP、RH 的重要程度几乎忽略不计。



因此我们先初步选择四个特征 AT、V，AP，RH 作为 X 数据，然后选择 AT、V，而 AP，RH 暂不考虑作为 X 数据，比较特征的不同组合带来的不同效果。

- Spearman 相关系数分析

```
1.  # spearman 相关系数分析（显示特征之间的关系）
2.  print(data.corr(method='spearman'))
```

根据 Spearman 相关系数分析输出了如下的相关系数矩阵

```
/anaconda3/bin/python3.6 /Users/miya/PycharmProjects/miya/ccpp.py
          AT         V        AP        RH        PE
AT  1.000000  0.849301 -0.513912 -0.544303 -0.943276
V   0.849301  1.000000 -0.423202 -0.303292 -0.883264
AP -0.513912 -0.423202  1.000000  0.087790  0.539287
RH -0.544303 -0.303292  0.087790  1.000000  0.388854
PE -0.943276 -0.883264  0.539287  0.388854  1.000000
```

由相关系数矩阵可以看出：AT 与 PE 存在较强的负相关性，AT 与 V 存在较强的相关性，V 与 PE 存在较强的负相关性，AP 与 AT、V 存在一般的负相关性，RH 与 AT 存在一般的负相关性，PE 与 AT、V 存在较强的负相关性,与 AP 存在一般的负相关性

- 划分测试集和训练集

按照 train_test_split 随机划分测试集，设置样本比例为 40%
（此处数据集 X 选取全部特征）

```
1. from sklearn.cross_validation import train_test_split
2. X = data[['AT', 'V','RH','AP']]
3. Y = data[['PE']]
4. X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.4,
5.  random_state=1)
```

2. 模型构建

2.1 随机森林（回归）

第一次我们采用 100 颗决策树

```
1. from sklearn.ensemble import RandomForestRegressor
2. rf= RandomForestRegressor(max_depth=None,n_estimators=100,random_state=0)
3. rf.fit(X_train,Y_train)
4. from sklearn.model_selection import cross_val_predict
5. predicted = cross_val_predict(rf, X, Y, cv=10)
```

调整参数
考虑 maxdepth，因为该数据集只有四个特征，因此 max_depth 不设置特殊值，简单地选取所有特征，每颗树都可以利用他们，每颗树都没有任何的限制。
考虑增大 n_estimators，运行速度变慢，n_estimators=1000

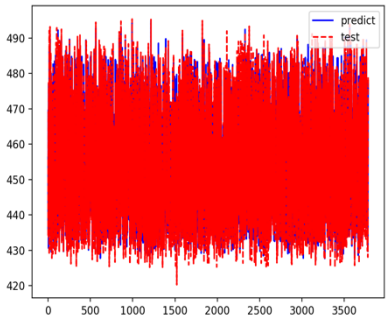
```
6. rf= RandomForestRegressor(max_depth=None,n_estimators=1000,random_state=0)
```

不考虑全部特征的情况有以下几种，因为 AT、V 具有明显的线性关系，因此只考虑特征里面是否包含 RH 和 AP，一共有以下几种情况

```
1. X = data[['AT', 'V','RH']]
2. X = data[['AT', 'V']]
3. X = data[['AT', 'V','AP']]
```

这三种情况下，求出的 MSE 均偏大，大于考略全部特征的情况，效果不好，因此，我们考虑全部特征，进行分析。

绘制 ROC 曲线：ROC 曲线被广泛用于二分类输出模型的性能评估，这里红线代表测试集的值，蓝线代表预测值，从两者的分布可以看出我们分类器效果。

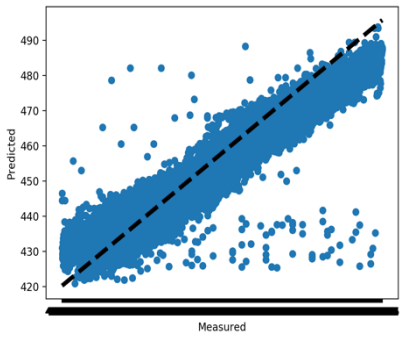


2.2 线性回归

```
#线性回归
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(X_train, Y_train)
print(linreg.intercept_)
print(linreg.coef_)
 #模拟测试集
 y_pred = linreg.predict(X_test)
 from sklearn import metrics
print ("MSE:",metrics.mean_squared_error(Y_test, y_pred))
print ("RMSE:",np.sqrt(metrics.mean_squared_error(Y_test, y_pred))
from sklearn.model_selection import cross_val_predict
predicted = cross_val_predict(linreg, X, Y, cv=10)
print( "MSE:",metrics.mean_squared_error(Y, predicted))
```

这里画出：真实值和预测值的变化关系，离中间的直线 y=x 直接越近的点代表预测损失越低

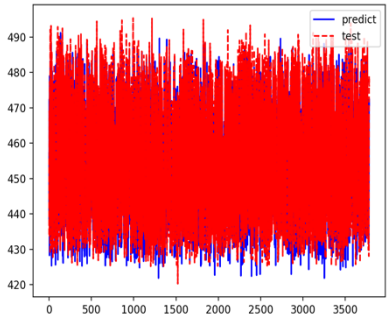
```
plt.scatter(Y, predicted)
plt.plot([Y.min(), Y.max()], [Y.min(), Y.max()], 'k--', lw=4)
plt.xlabel('Measured')
plt.ylabel('Predicted') plt.show()
```



输出得到的线性回归的权重和偏重

```
15. [445.35907985]
16. [[-1.99266307 -0.22627567  0.07124649 -0.16018131]]
```

绘制 ROC 曲线：ROC 曲线被广泛用于二分类输出模型的性能评估，这里红线代表测试集的值，蓝线代表预测值，从两者的分布可以看出我们分类器效果。



2.3 梯度提升

```
1.  # 梯度提升
2.  from sklearn.ensemble import GradientBoostingRegressor
3.  gb=GradientBoostingRegressor(random_state=0)
4.  gb.fit(X_train,Y_train)
5.  predicted = cross_val_predict(gb, X, Y, cv=10)
6.  #绘制 ROC 曲线
7.   y_pred=gb.predict(X_test)
8.  plt.figure()
9.  plt.plot(range(len(y_pred)),y_pred,'b',label="predict")
10. plt.plot(range(len(y_pred)),Y_test,'r',label="test")
11. plt.legend(loc="upper right") #显示图中的标签
12. plt.xlabel(" ")
13. plt.ylabel(' ')
14. plt.show()
```

梯度提升中两个参数，为了降低过拟合，我们可以通过限制最大深度或降低学习速率来进行更强的修剪。

特征数量较少，不考虑额最大深度，默认不输入

```
1.  gb=GradientBoostingRegressor(random_state=0, max_depth=None)
```

对于同样的训练集拟合效果，较小的 learning-rate 意味着我们需要更多的弱学习器的迭代次数。通常我们用步长和迭代最大次数一起来决定算法的拟合效果。
从以下几个数值开始试验 learning-rate 的值，0.001 ,0.003, 0.01, 0.03, 0.1, 0.3, 1, …

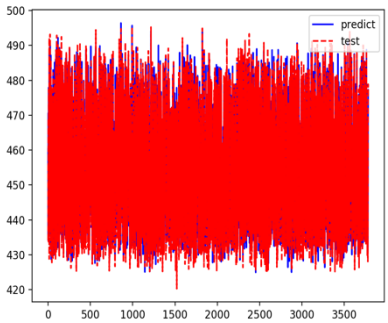
```
15. gb=GradientBoostingRegressor(random_state=0,learning_rate=0.7)
```

以 MSE 作为损失函数来衡量 learning-rate 的值
根据 MSE 的走向，取 learning-rate=0.7 时，效果最好

Learning-rate	0.001	0.1	0.3	0.6	0.7	0.8	0.9	1
MSE	242	15.09	12.61	12.27	12.11	13.02	12.51	13.39

当 learning-rate 等于 0.7 时，绘制出 ROC 曲线

```
1.  # 红色的线是真实的值曲线，蓝色的是预测值曲线
2.  y_pred=gb.predict(X_test)
3.  plt.figure()
4.  plt.plot(range(len(y_pred)),y_pred,'b',label="predict")
5.  plt.plot(range(len(y_pred)),Y_test,'r--',label="test")
6.  plt.legend(loc="upper right") #显示图中的标签
7.  plt.show()
```



2.4 MLPRegressor

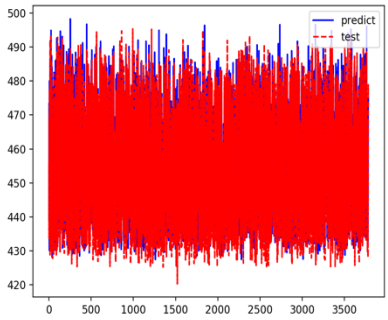
```
# # 深度学习
from sklearn.neural_network import MLPRegressor

from sklearn.preprocessing import StandardScaler
scaler =StandardScaler()
X=scaler.fit_transform(X)
X_train_scaled =scaler.fit_transform(X_train)
X_test_scaled =scaler.fit_transform(X_test)
mlp =MLPRegressor()
mlp.fit(X_train_scaled,Y_train)
predicted = cross_val_predict(mlp, X, Y, cv=10)
print( "MSE:",metrics.mean_squared_error(Y, predicted))
print ("RMSE:",np.sqrt(metrics.mean_squared_error(Y, predicted)))
```

在深度学习回归中，不设置任何特殊参数时间的 MSE 值为 25，与其他几种算法相比过高，不太准确，应调整参数。
根据多次试验，调整最大迭代次数 max_iter=10000，可以提升准确度

```
mlp =MLPRegressor(max_iter=10000,random_state=20)
```

max_iter=10000 时，MSE: 17.91422403445372

绘制出 ROC 曲线



3.模型评估与分析

在回归算法中我们采用两个指标来衡量算法的优劣：MSE,RMSE。
MSE(均方差):该统计参数是预测数据和原始数据对应点误差的平方和的均值
RMSE(均方根):该统计参数，也叫回归系统的拟合标准差，是 MSE 的平方根

MSE 的值越小，说明预测模型描述实验数据具有更好的精确度。

```
from sklearn.model_selection import cross_val_predict
```

```
predicted = cross_val_predict(model, X, Y, cv=10)
# 用 scikit-learn 计算 MSE
print( "MSE:",metrics.mean_squared_error(Y, predicted))
# 用 scikit-learn 计算 RMSE
print ("RMSE:",np.sqrt(metrics.mean_squared_error(Y, predicted)))
```

算法	MSE	RMSE
随机森林（回归）	10.792559044338399	3.2852030446135894
线性回归	20.881163300615956	4.569591152457291
梯度提升	12.118266958697106	3.481130126653858
MLP	17.91422403445372	4.232519820916816

根据计算 MSE 得到的结果，可以看出各个算法的效果：随机森林>梯度提升>MLP>线性回归。

二、 实验小结

本实验通过一组联合发电厂的发电数据，预测任务是根据这四个特征 AP、T、V、RH 以预测设备的净小时电能输出（PE），转化为回归问题，即是找到这四个特征：AP、T、V、RH 与 PE 之间的关系，采用四种回归算法：随机森林（回归）、线性回归、梯度提升（回归）、深度学习 MLP（回归）来进行分析，算法效果通过计算 MSE(均方差)和 RMSE(均方根)来衡量，判断条件为 MSE 的值越小，说明预测模型描述实验数据具有更好的精确度，以此来看出随机森林的效果好于梯度提升，线性回归最差。