@TOC

环境准备与基础知识

- Jupyter notebook

  > Jupyter Notebook是一块所见即所得的画布，通过在浏览器上编辑代码，让开发人员实现展示与快速迭代的利器。
  >
  > 更多内容参考：https://www.jianshu.com/p/91365f343585

- 决策树相关算法介绍
  - https://blog.csdn.net/qingqing7/article/details/78416708
- 完成任务回顾
  - 将 任务 分3块
    1. 需求，目标
    2. 确定方法，明确步骤，分步骤执行
    3. 输出结果与调优
  - 善用工具
    - 如Bob（翻译插件）、Dash（API文档）
  - 勤于总结
    - pd、np常用操作等
  - 积极沟通

第一次机器学习内容

Lab02 - Decision Tree - I

1、案例：泰坦尼克号幸存者预测

案例结构：

1. 引入依赖

```
# 数据分析和准备
import pandas as pd
import numpy as np
import random as rnd

# 可视化
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# 机器学习
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
```

2. 读取数据（训练数据和测试数据）

```
# 读取数据文件到dateframe
train_df = pd.read_csv('train.csv')

# 查看前5行
train_df.head()

# 查看数据类型和缺失值
train_df.info()
```

3. 人为分析数据（数据概况、缺失值等）

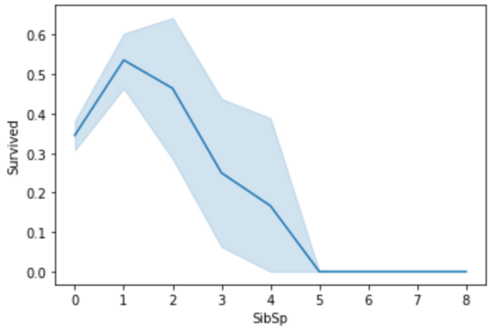
```
# 输出表头信息
print(train_df.columns.values.tolist())

# 根据百分比分布查看数据
train_df.describe(percentiles=[.1, .2,.3,.4, .6, .7, .8, .9, .95,
.98,.99])

# 查看连续特征的数据列分析
train_df.describe()

# 查看分类特征的数据列分析
train_df.describe(include=['O'])
```

4. 数据分析，清洗数据（相关性分析，缺失值填充等）

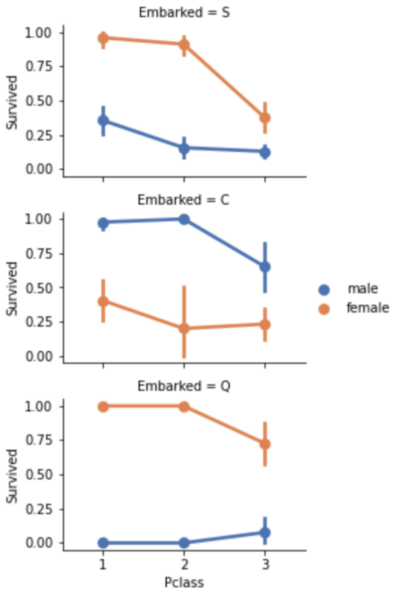
```
# 众数填充空值
df['Embarked'].fillna(df['Embarked'].mode(dropna=True)[0],
inplace=True)

# 数据表格 数据替换（数字编码分类特征）
num_encode = {
    'Sex': {'male': 0, "female": 1},
    'Embarked': {'S': 0, 'Q': 1, 'C': 2 }
}
df.replace(num_encode, inplace=True)
```

5. 可视化输出，方便分析

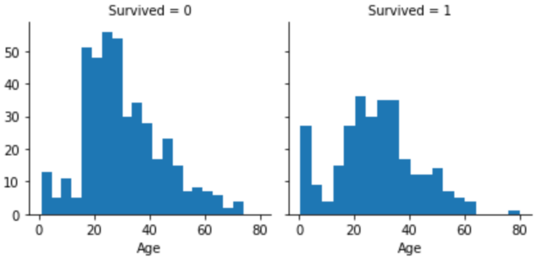
```
# 折线图
sns.lineplot(x='SibSp', y='Survived', data=train_df)
```



```
# 格点图
grid = sns.FacetGrid(train_df, row='Embarked', height=2.2,
aspect=1.6)
grid.map(sns.pointplot, 'Pclass', 'Survived', 'Sex',
palette='deep')
grid.add_legend()
```



```
# 柱状图
g = sns.FacetGrid(train_df, col='Survived')
g.map(plt.hist, 'Age', bins=20)
```



6. 建模、预测和评价

```
# 切分数据集，用于训练和验证
x_train, x_test, y_train, y_test = train_test_split(X, y,
test_size=0.3)

# 决策树建模和训练
tree = DecisionTreeClassifier(criterion='entropy')
tree.fit(x_train, y_train)

# 决策树预测
y_pred = tree.predict(x_test)

# 评价，输出准确度
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

2、任务：气缸带数据集band type预测

任务结构：

1. 引入依赖包

   依然使用决策树算法，引入包同上案例

2. 导入数据

```
# 指定列名和缺失值导入数据到dataframe
names = ['c1', 'c2', 'c3']
missing_values = ["?","nan"]
train_df = pd.read_csv('bands_init.data',names=names,na_values=missing_values)
```

3. 清洗数据

```
# 遍历替换
# 把df中的 ? 替换成 'nan'
# 把df中 str 类型的值全部转换成小写
temp = 'aa22'
feature_list = train_df.columns.tolist()
for index in train_df.index:
    for feature in feature_list:
        if type(train_df.loc[index, feature]) == type(temp):
            train_df.loc[index, feature] = train_df.loc[index, feature].lower()
        if train_df.loc[index, feature] == '?':
            # print(train_df.loc[index, feature])
            train_df.loc[index, feature] = np.nan
```

```
# 列类型转换
# 从20列到39列 将列属性转换成float类型
i=20
while(i<39):
    count = 0
    for row in train_df[feature_list[i]]:
        try:
            float(row)
        except ValueError:
            train_df.loc[count,feature_list[i]] = np.nan
            pass
        count=count+1
```

```
 train_df[feature_list[i]]=train_df[feature_list[i]].astype(np.float64)
    # 输出转换过的列名
    print(feature_list[i])
    i=i+1
```

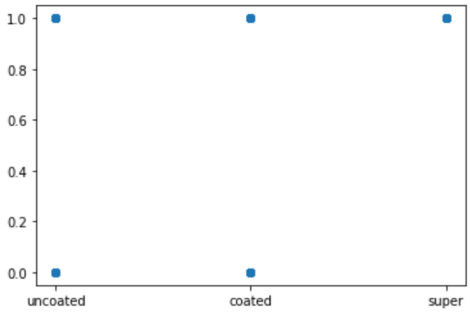
4. 数据分析

   同上案例

5. 数据可视化

> 画图时，当数据有空值会导致画图失败。
>
> 解决办法：先填充空值再画图分析。

```
# 散点图
x = train_df['paper type']
y = train_df['band type']
color = ['r','y','k','g','m']
plt.scatter(x, y)
plt.show()
```



6. 再次清洗数据

```
# 删除目标值为空的数据（错误数据（如出现错行现象））
for index in range(train_df.shape[0]):
    try:
        if np.isnan(train_df['band type'][index]):
            train_df=train_df.drop([index])
    except:
        pass
```

7. 准备数据

> 可以把encode的map定义大些，X可以从中选取几个训练，方便调整训练模型。

```
# 用大map数据替换
encode = {
    'grain screened':{'yes':0,'no':1},
    'proof on ctd ink':{'yes':0,'no':1},
    'blade mfg':{'uddeholm':0,'benton':1},
    'paper type':{'coated':0,'uncoated':1,'super':2},
    'ink type':{'coated':0,'uncoated':1,'cover':2},
    'direct steam':{'yes':0,'no':1},
    'type on cylinder':{'yes':0,'no':1},
    'plating tank':{'1911':0,'1910':1},
    'solvent type':{'naptha':0,'line':1,'xylol':2},
    'press type':{'motter70':0,'motter94':1,'albert70':2,'woodhoe70':3},
    'paper mill location':{'scandanavian':0,'canadian':1,'northus':2,'mideuropean':3,'southus':4}
}
train_df.replace(encode, inplace=True)

# 选取不同列作为训练数据集，之后进行训练
X = train_df[['grain screened','blade mfg','direct steam','paper type', 'ink type','press type','solvent type']]
# X = train_df[['grain screened','blade mfg','direct steam','type on cylinder','plating tank']]
y = train_df['band type']
```

8. 建模、预测和评价

   同上

第二次机器学习内容

Lab03 - Decision Tree - II

针对Lab02的调优

结构：

1. 导入数据

```
train_df = pd.read_csv('train.csv')
test_df = pd.read_csv('test_merged.csv')

combine = [train_df, test_df]
```

2. 处理连续特征值（年龄，船票，家庭人数）

> 1、相关性填充：案例中，我们注意到年龄，性别和Pclass之间的相关性。使用Pclass和Gender功能组合集的Age的中值来猜测Age值。因此，Pclass = 1和Gender = 0，Pclass = 1和Gender = 1的年龄中位数，依此类推...

```
# 准备一个空数组，以包含基于Pclass x Gender组合的年龄猜测值。使用船舱等级
和性别作为数据集，放到数组
guess_ages = np.zeros((2,3))

# 遍历Sex（0或1）和Pclass（1、2、3）以计算这六个组合的Age猜测值。
pd.set_option('mode.chained_assignment',None)

for dataset in combine:
    for i in range(0, 2):
        for j in range(0, 3):
            guess_df = dataset[(dataset['Sex'] == i) & \
                                  (dataset['Pclass'] == j+1)]
['Age'].dropna()
            age_guess = guess_df.median()

            guess_ages[i,j] = int(age_guess)

    for i in range(0, 2):
        for j in range(0, 3):
            mask = dataset[(dataset.Age.isnull()) & (dataset.Sex
== i) & (dataset.Pclass == j+1)]
            if mask.empty:
                continue

            dataset.loc[mask.index , 'Age'] = int(guess_ages[i,
j])

    # dataset.loc[:, 'Age'] = dataset['Age'].astype(int)
```

> 2、规律性强的连续数据转换成分类特征，使用cut。将年龄（有规律）根据常识切分成4个年龄段。

```
train_df.drop('AgeBand', inplace=True, axis=1)
for df in combine:
    category = pd.cut(df.Age,bins=[0,2,17,65,99], labels=[0, 1,
2, 3])
    df.insert(5,'AgeBand', category)
```

> 3、规律性不强的连续数据转换成分类特征，使用qcut。将票价根据百分比分部切分成5个分类值。

```
train_df['FareBand'] = pd.qcut(train_df['Fare'],[0, 0.25, 0.5,
0.75, 1])
train_df[['FareBand', 'Survived']].groupby(['FareBand'],
as_index=False).mean().sort_values(by='FareBand',
ascending=True)
train_df.drop(['FareBand'], axis=1, inplace=True)
for df in combine:
    category = pd.qcut(df.Fare, q=4, labels=[0, 1, 2, 3])
    df.insert(5,'FareBand', category)
```

> 4、从现有特征提取新特征：使用正则表达式截取人名，提取出身份作为新特征。

```
# 正则提取人名
train_df['Title'] = train_df.Name.str.extract(' ([A-Za-z]+)\.',
expand=False)
# 交叉列表取值，观察结果 该特征与年龄和生存相关
pd.crosstab(train_df['Title'], train_df['AgeBand'])
pd.crosstab(train_df['Title'], train_df['Survived'])

# 将名字分为几类
for dataset in combine:
    dataset['Title'] = dataset['Title'].replace(['Lady',
'Countess','Capt', 'Col',\
  'Don', 'Dr', 'Major', 'Rev', 'Sir', 'Jonkheer', 'Dona'],
'Rare')

    dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')
```

```
# 转换成数字类型
title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4,
"Rare": 5}
for dataset in combine:
    dataset['Title'] = dataset['Title'].map(title_mapping)
    dataset['Title'] = dataset['Title'].fillna(0)
```

3. 模型，预测可评估

> DecisionTreeClassifier参数详解：https://www.cnblogs.com/baby-lily/p/10646226.html

```
tree = DecisionTreeClassifier(criterion='entropy',
random_state=42)

# 通过混淆矩阵评估结果
print(metrics.confusion_matrix(y_valid, y_pred))
```

4. 超参数调整（优化模型），训练模型，取出各项指标最精确的模型

> 模型定义界面的几乎所有输入参数都是超参数。控制变量法调整参数以训练模型。

```
# ['entropy', '7']
entropy_tree_ent_seven =
DecisionTreeClassifier(criterion='entropy', max_depth=7,
random_state=42)
entropy_tree_ent_seven.fit(X_train, y_train)
y_pred = entropy_tree_ent_seven.predict(X_valid)

print(metrics.classification_report(y_valid, y_pred))

# ['entropy', '12']
entropy_tree_ent_twelve =
DecisionTreeClassifier(criterion='entropy', max_depth=12,
random_state=42)
entropy_tree_ent_twelve.fit(X_train, y_train)
y_pred = entropy_tree_ent_twelve.predict(X_valid)

print(metrics.classification_report(y_valid, y_pred))

# ['gini', '7']
```

```
entropy_tree_gini_seven =
DecisionTreeClassifier(criterion='gini', max_depth=7,
random_state=42)
entropy_tree_gini_seven.fit(X_train, y_train)
y_pred = entropy_tree_gini_seven.predict(X_valid)

print(metrics.classification_report(y_valid, y_pred))
```

Lab04 - K-Nearest Neighbours

1、案例：利用花瓣和萼片的数据预测虹膜的类型。

案例结构：

1. 导入所需的库

```
# 3维画图
from mpl_toolkits.mplot3d import Axes3D

# knn算法
from sklearn.neighbors import KNeighborsClassifier
# sklearn数据集
from sklearn import datasets
```

2. 获得数据

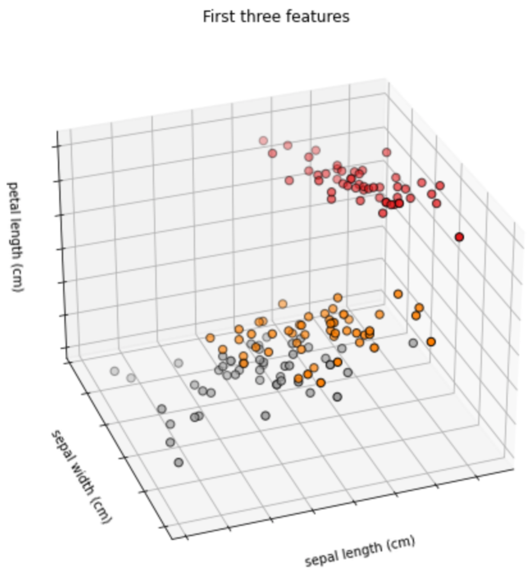
> 从sklearn加载数据集

```
# 加载数据集和目标
data, target = datasets.load_iris(return_X_y=True,
as_frame=True)
# 合并数据集和目标
combine = pd.concat([data, target], axis=1, sort=False)
```

3. 数据分析

> 3维图像绘制

```
# 3维图像绘制,使用前3个特征画图
fig = plt.figure(1, figsize=(8, 6))
ax = Axes3D(fig, elev=-150, azim=110)
ax.scatter(data["sepal length (cm)"], data["sepal width (cm)"],
data["petal length (cm)"], c=target,
           cmap=plt.cm.Set1, edgecolor='k', s=40)
ax.set_title("First three features")
ax.set_xlabel("sepal length (cm)")
ax.w_xaxis.set_ticklabels([])
ax.set_ylabel("sepal width (cm)")
ax.w_yaxis.set_ticklabels([])
ax.set_zlabel("petal length (cm)")
ax.w_zaxis.set_ticklabels([])
```



4. 数据预处理

> 数据归一化

```
# 对数据归一化，大小缩小到0到1内。
scaler = MinMaxScaler()
data_scaled = pd.DataFrame(scaler.fit_transform(data),
columns=data.columns, index=data.index)
```

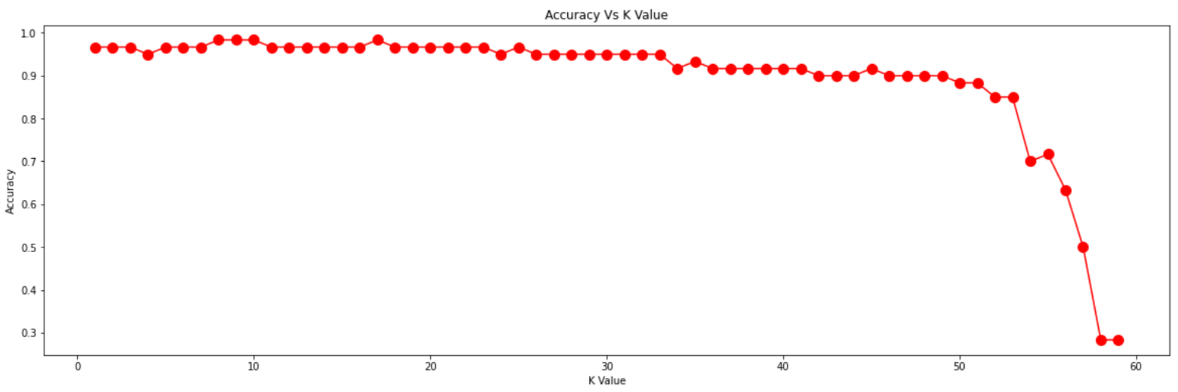
5. 建模、预测和评价

> knn建模和训练

```
# knn建模、训练、预测和评价
fivenn = KNeighborsClassifier(n_neighbors=5)
fivenn.fit(X_train, y_train)
pred = fivenn.predict(X_test)

default_acc = metrics.accuracy_score(y_test, pred)

print("Accuracy for model k=5:", default_acc)
```

> 评价与K值调优

```
# 将k值做变量，循环输出准确度
accuracy = []

# Calculating error for K values between 1 and 60
for i in range(1, 60):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    accuracy.append(metrics.accuracy_score(y_test, pred_i))


# 可视化
plt.figure(figsize=(20, 6))
plt.plot(range(1, 60), accuracy, color='red', marker='o',
         markersize=10)
plt.title('Accuracy Vs K Value')
plt.xlabel('K Value')
plt.ylabel('Accuracy')
```



2、任务：对气缸带数据集用knn训练并预测

任务结构：

1. 导入所需的库

   同上

2. 获得数据

   ```
   # 读取数据
   train_df =
   pd.read_csv("bands.data",names=names,na_values=missing_values,header=None)
   ```

3. 分析数据，可视化数据

   ```
   # 输出相关性 > 0.5的相关矩阵
   train_df.corr()[(train_df.corr() > 0.5) ]
   ```

   三维图同上

4. 数据预处理

   数据归一化同上

5. 建模、预测和评价

   > knn算法参数：p=1，设置为曼哈顿距离；p=2，设置为欧氏距离

   ```
   # 建模、预测和评价
   manhattanNN = KNeighborsClassifier(n_neighbors=5,
   p=1,weights="uniform")
   manhattanNN.fit(X_train, y_train)
   pred = manhattanNN.predict(X_test)

   manhattan_acc = metrics.accuracy_score(y_test, pred)
   print("Accuracy for model k=5, p=1:", manhattan_acc)
   ```

   控制变量法调参同Lab03

Lab05 - 特征值挑选

结构：

1. 导入开源数据集及算法

```
# 导入sklearn开源数据集，用到的算法
from sklearn import datasets
# knn
from sklearn.neighbors import KNeighborsClassifier
# 归一化
from sklearn.preprocessing import MinMaxScaler
# 挑选最好特征值
from sklearn.feature_selection import SelectKBest
# 卡方分布
from sklearn.feature_selection import chi2
# 随机森林
from sklearn.ensemble import ExtraTreesClassifier

# 加载开源数据集
dataset = datasets.load_breast_cancer()
```

2. 数据归一化

   同上

3. 测量相似度

   > 自定义拉塞尔差异实现方法，传参调用

```
# scipy中的距离计算模块，scipy.spatial.distance
from scipy.spatial import distance
def russel_rao(ins1, ins2):
    '''
    Russel-Rao similarity
    :param ins1: list - containing the features for instance 1
    :param ins2: list - containing the features for instance 2
    :return: float - russel_rao similarity index
    '''
    # TODO: implement it here.
    dis = 1-distance.russellrao(ins1, ins2)
    print(dis)
    return
```

```
# 调用算法，得到结果为0.4
ins_1 = [1, 1, 1, 0, 1]
ins_2 = [1, 0, 1, 0, 0]
rus_rao_similarity = russel_rao(ins_1, ins_2)
```

> pandas.DataFrame.sample 随机选取若干行

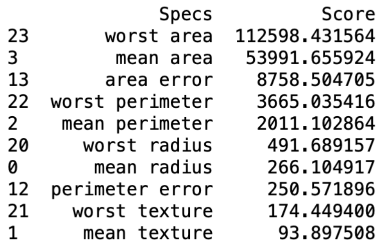
```
# 从df中随机取一行
ins_1 = data_scaled.sample(n=1)
```

4. 特征挑选

   1. 单变量选择

      > 使用卡方检验挑选topK的特征

```
# 使用卡方检验选择topK的特征
best_features = SelectKBest(score_func=chi2, k=10).fit(data, target)
print("best_features:")
print(best_features)
print('\n')

dfscores = pd.DataFrame(best_features.scores_)
dfcolumns = pd.DataFrame(data.columns)

# concat two dataframes for better visualization
# 合并两个数据集以获得更好的可视化效果
featureScores = pd.concat([dfcolumns,dfscores], axis=1)
featureScores.columns = ['Specs','Score']  # naming the dataframe columns 命名数据集列

print(featureScores.nlargest(10,'Score'))
```

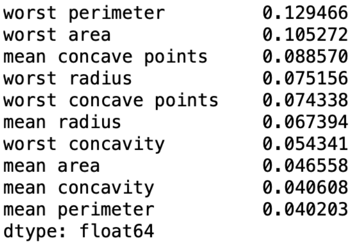


```
              Specs          Score
23       worst area  112598.431564
3         mean area   53991.655924
13       area error    8758.504705
22  worst perimeter    3665.035416
2    mean perimeter    2011.102864
20     worst radius     491.689157
0       mean radius     266.104917
12  perimeter error     250.571896
21    worst texture     174.449400
1      mean texture      93.897508
```

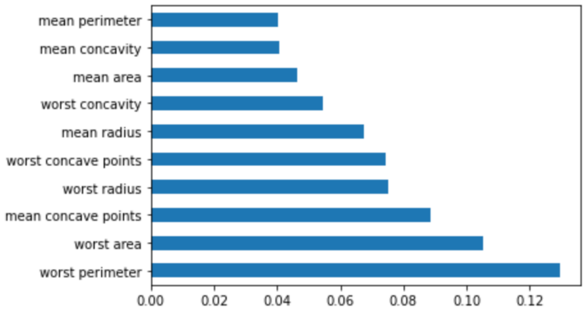
   2. 基于树的特征选择

      > 使用随机森林挑选最优特征值

```python
tree = ExtraTreesClassifier()
tree.fit(data, target)

df_feature_importances =
pd.Series(tree.feature_importances_, index=data.columns)
df_feature_importances.nlargest(10).plot(kind='barh')
df_feature_importances.nlargest(10)
```



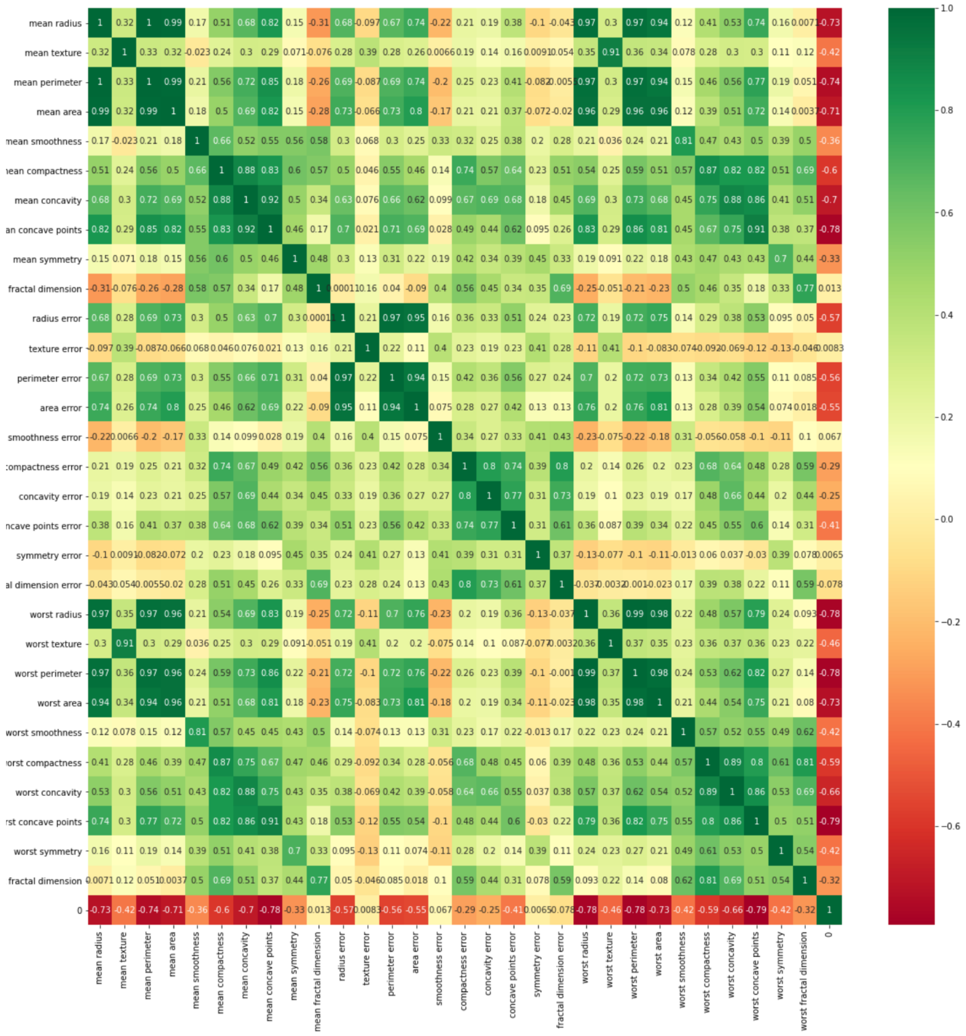
```
worst perimeter         0.129466
worst area              0.105272
mean concave points     0.088570
worst radius            0.075156
worst concave points    0.074338
mean radius             0.067394
worst concavity         0.054341
mean area               0.046558
mean concavity          0.040608
mean perimeter          0.040203
dtype: float64
```



3. 使用相关矩阵

> 使用相关矩阵热力图看特征间相关性

```python
combine = pd.concat([data, target], axis=1)
corr_mat = combine.corr()



plt.figure(figsize=(20,20))
g=sns.heatmap(corr_mat,annot=True,cmap="RdYlGn")
```



第三次机器学习内容

> 第三次开始取的是之前两次清洗后的数据，
>
> 所以没有数据清洗步骤，将 只对Lab重点和新出现内容总结。

Lab06 - 基于概率的学习

朴素贝叶斯算法：

- 训练数据要求：数值特征
- 预测目标：分类特征

```
# 引入朴素贝叶斯算法
from sklearn.naive_bayes import GaussianNB

# 生成分类器（建模）
gnb = GaussianNB()
# 训练分类器（训练）
gnb.fit(X_train, y_train)
# 预测和评估
pred = gnb.predict(X_test)

print(classification_report(y_pred=pred, y_true=y_test))
```

Lab07 - 线性回归和逻辑回归

1、线性回归

- 训练数据要求：数值特征
- 预测目标：数值特征

```
# 引入线性回归算法
from sklearn.linear_model import LinearRegression

# 生成线性回归模型（建模）
lin_reg = LinearRegression(normalize=True)

# 训练线性回归模型（训练）
lin_reg.fit(X=X_train, y=y_train)

# 预测和评估
pred = lin_reg.predict(X=X_test)

print("Mean squared error:", mean_squared_error(y_pred=pred,
y_true=y_test))
```

2、逻辑回归

- 训练数据要求：数值特征
- 预测目标：分类特征

```
# 引入逻辑回归算法
from sklearn.linear_model import LinearRegression,
LogisticRegression

# 生成逻辑回归模型（建模）
log_reg = LogisticRegression(max_iter=5)

# 训练逻辑回归模型（训练）
log_reg.fit(X_train, y_train)

# 使用逻辑回归模型进行预测（预测和评估）
pred = log_reg.predict(X_test)

# 输出预测结果的均方误差
print(metrics.accuracy_score(y_test, pred))
```

Lab08 - 机器学习模型评估与性能分析

> 评估维度：1、定时性能；2、预测性能；3、ROC曲线；4、交叉验证

举例评估逻辑回归和knn

```
# 引入依赖
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,
classification_report, plot_roc_curve, roc_auc_score
from sklearn.model_selection import train_test_split,
cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics

# 加载、划分数据
data = datasets.load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.DataFrame(data.target, columns=['cancer_type'])

X_train_val, X_test, y_train_val, y_test = train_test_split(X, y)
```

```
X_train, X_val, y_train, y_val = train_test_split(X_train_val,
y_train_val)

# 生成模型
log_reg = LogisticRegression(max_iter=5000)
knn = KNeighborsClassifier(n_neighbors=5)
```

1. 定时性能

```
%%time
log_reg.fit(X_train, y_train.values.ravel())
```

```
CPU times: user 2.28 s, sys: 603 ms, total: 2.88 s
Wall time: 493 ms

LogisticRegression(max_iter=5000)
```

```
%%time
knn.fit(X_train, y_train.values.ravel())
```

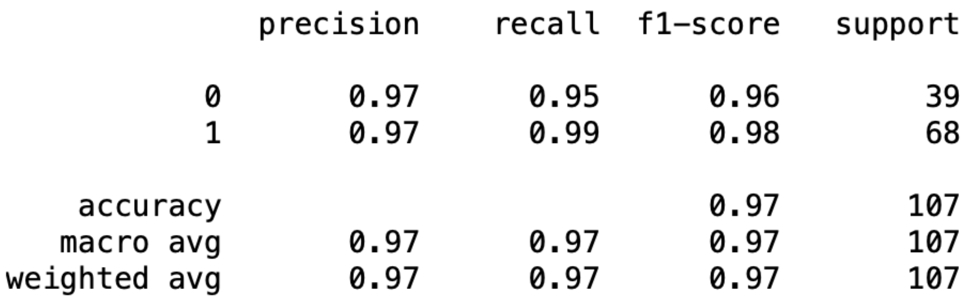
```
CPU times: user 10.3 ms, sys: 2.69 ms, total: 13 ms
Wall time: 2.2 ms
```

2. 预测性能

```
# 先为模型生成混淆矩阵和分类报告
confusion_matrix(y_true=y_val, y_pred=pred_log_reg)
print(classification_report(y_true=y_val, y_pred=pred_log_reg))
```

```
              precision    recall  f1-score   support

           0       0.97      0.95      0.96        39
           1       0.97      0.99      0.98        68

    accuracy                           0.97       107
   macro avg       0.97      0.97      0.97       107
weighted avg       0.97      0.97      0.97       107
```

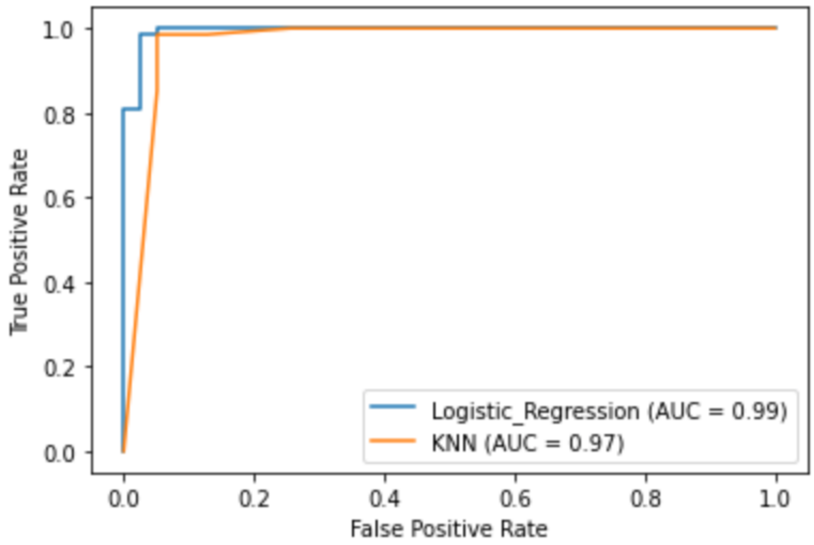
```
# 先为模型生成混淆矩阵和分类报告
confusion_matrix(y_true=y_val, y_pred=pred_knn)
print(classification_report(y_true=y_val, y_pred=pred_knn))
```



```
              precision    recall  f1-score   support

           0       0.97      0.95      0.96        39
           1       0.97      0.99      0.98        68

    accuracy                           0.97       107
   macro avg       0.97      0.97      0.97       107
weighted avg       0.97      0.97      0.97       107
```

3. ROC曲线

```
# 绘制两个模型的ROC曲线
ax = plt.gca()
plot_log_reg = plot_roc_curve(estimator=log_reg, X=X_val,
y=y_val, ax=ax, name='Logistic_Regression')
plot_knn = plot_roc_curve(estimator=knn, X=X_val, y=y_val,
ax=ax, name='KNN')
```



```
# 输出roc曲线下更精确的面积
# Area under the curve for logistic regression model:
0.9947209653092006
pred_scores = log_reg.predict_proba(X_val)
print('Area under the curve for logistic regression model:
{}'.format(roc_auc_score(y_true=y_val,

                               y_score=pred_scores[:, 1])))
```

```
# 输出roc曲线下更精确的面积
# Area under the curve for KNearestNeighbours model:
0.9685143288084465
pred_scores = knn.predict_proba(X_val)
print('Area under the curve for KNearestNeighbours model:
{}'.format(roc_auc_score(y_true=y_val,

                  y_score=pred_scores[:, 1])))
```

4. 交叉验证

> 分别利用两个模型的所有训练和验证数据进行10倍交叉验证。使用宏观平均f1分数作为评分方法。

```
# F1_score: 0.95 std: 0.05
scores = cross_val_score(estimator=log_reg, X=X_train_val,
y=y_train_val.values.ravel(), cv=10, scoring='f1_macro')
print('F1_score: %0.2f std: %0.2f'%(scores.mean(),
scores.std()))
```

```
# F1_score: 0.92 std: 0.05
scores = cross_val_score(estimator=knn, X=X_train_val,
y=y_train_val.values.ravel(), cv=10, scoring='f1_macro')
print('F1_score: %0.2f std: %0.2f'%(scores.mean(),
scores.std()))
```