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

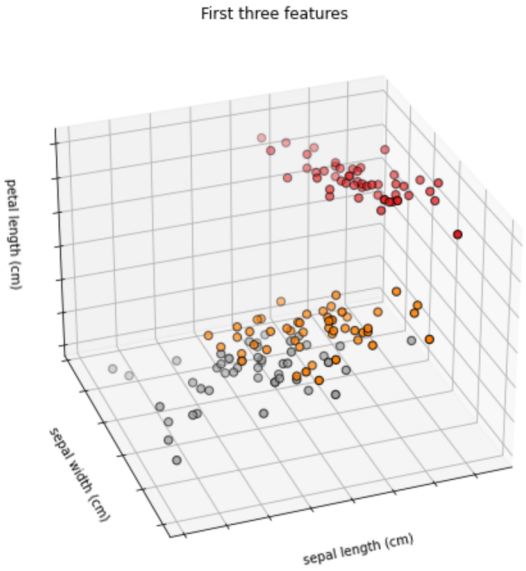
> 从sklearn加载数据集

```
# 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)
```

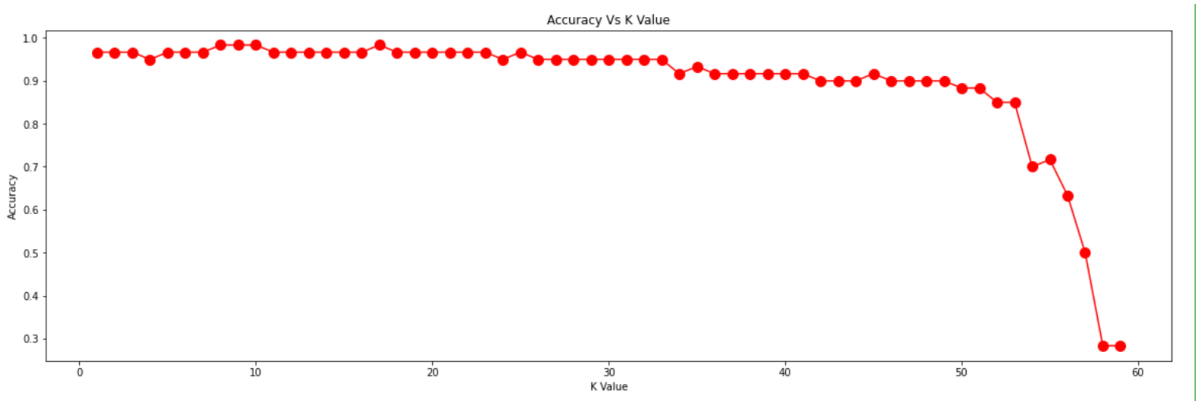
5. 建模、预测和评价

> 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,he
   ader=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'))
```

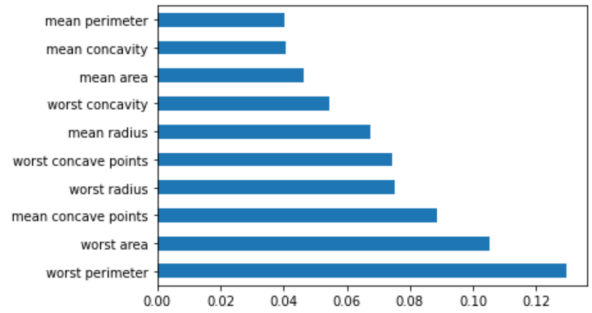
```
              Specs          Score
23       worst area  112598.431564
3         mean area   53991.655924
13       area error    8758.504705
22  worst perimeter    3665.035416
2    mean perimeter    2011.102864
20     worst radius     491.689157
0       mean radius     266.104917
12  perimeter error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

```
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

```
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")
```