红酒质量研究

Wine Quality Research

目录

Content

选题意义

数据表格

方案论证

模型评估

选题意义

确定葡萄酒质量时一般是通过聘请一批有资质的品酒员对酒进行品评，然后打分确定就的质量。而酒的质量在一定程度上受酒的理化指标的影响。然而这种传统方式需要聘请专业的品酒员，需要较高成本，从而使得酒的质量的划分方式往往受到大酒庄的控制。而各国规定的酒的分级制度，也仅仅是根据葡萄产地，酒的部分理化指标等数据将酒标上标签，并没有针对其质量。若能将葡萄酒的理化指标与其质量相联系起来，则可以实现葡萄酒质量评级的普及化，促进商家明码标价，为一般民众在选酒是提供重要参考。

特征提取

fixed.acidity：该变量指的是葡萄酒中的固定或者非挥发性酸度

volatile.acidity：挥发酸，葡萄酒中的醋酸含量过高，会导红酒味道变差。

citric.acid：柠檬酸，柠檬酸含量小，能给葡萄酒增添新鲜感和风味。

residual.sugar：剩余糖分，发酵结束后剩下的糖分，很少发现低于1克/升的葡萄酒，
超过45克/升的葡萄酒被认为是甜的。

chlorides：酒中的盐量。

free.sulfur.dioxide：酒中带硫元素的离子，它可以防止微生物的生长和葡萄酒的氧化。

total.sulfur.dioxide:二氧化硫，低浓度时检测不到，当浓度超过50 ppm时用鼻子可以闻到。

density：密度，大致接近于水，具体取决于酒精和糖的含量。

pH：用于描述酒的酸碱度。

sulphates：硫酸盐，葡萄酒的添加剂，用于控制二氧化硫比例。

alcohol：酒中的酒精浓度。

数据表格

winequality-red

fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5
7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	9.8	5
7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	9.8	5
11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58	9.8	6
7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5
7.4	0.66	0	1.8	0.075	13	40	0.9978	3.51	0.56	9.4	5
7.9	0.6	0.06	1.6	0.069	15	59	0.9964	3.3	0.46	9.4	5
7.3	0.65	0	1.2	0.065	15	21	0.9946	3.39	0.47	10	7
7.8	0.58	0.02	2	0.073	9	18	0.9968	3.36	0.57	9.5	7
7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	5
6.7	0.58	0.08	1.8	0.097	15	65	0.9959	3.28	0.54	9.2	5
7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	5
5.6	0.615	0	1.6	0.089	16	59	0.9943	3.58	0.52	9.9	5
7.8	0.61	0.29	1.6	0.114	9	29	0.9974	3.26	1.56	9.1	5
8.9	0.62	0.18	3.8	0.176	52	145	0.9986	3.16	0.88	9.2	5
8.9	0.62	0.19	3.9	0.17	51	148	0.9986	3.17	0.93	9.2	5
8.5	0.28	0.56	1.8	0.092	35	103	0.9969	3.3	0.75	10.5	7
8.1	0.56	0.28	1.7	0.368	16	56	0.9968	3.11	1.28	9.3	5
7.4	0.59	0.08	4.4	0.086	6	29	0.9974	3.38	0.5	9	4
7.9	0.32	0.51	1.8	0.341	17	56	0.9969	3.04	1.08	9.2	6
8.9	0.22	0.48	1.8	0.077	29	60	0.9968	3.39	0.53	9.4	6

数据预处理

```
#数据归一化
minVals = X.min(0)
maxVals = X.max(0)
X = (X-minVals)/(maxVals-minVals)
```

由于数据太过分散，我们将所有数据进行归一化处理，使它们均处在（0，1）内

数据预处理

```
for i in range(len(Y)):   #二值化标签
    if 0<Y[i]<6:
        Y[i] = 0
    else:
        Y[i] = 1

#划分训练集和测试集
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.3,random_state=32, stratify=Y)
```

我们将label划分成两个去区间：[6，10]（0，6）分别代表红酒质量的优劣。

方案论证

1、KNN算法实现

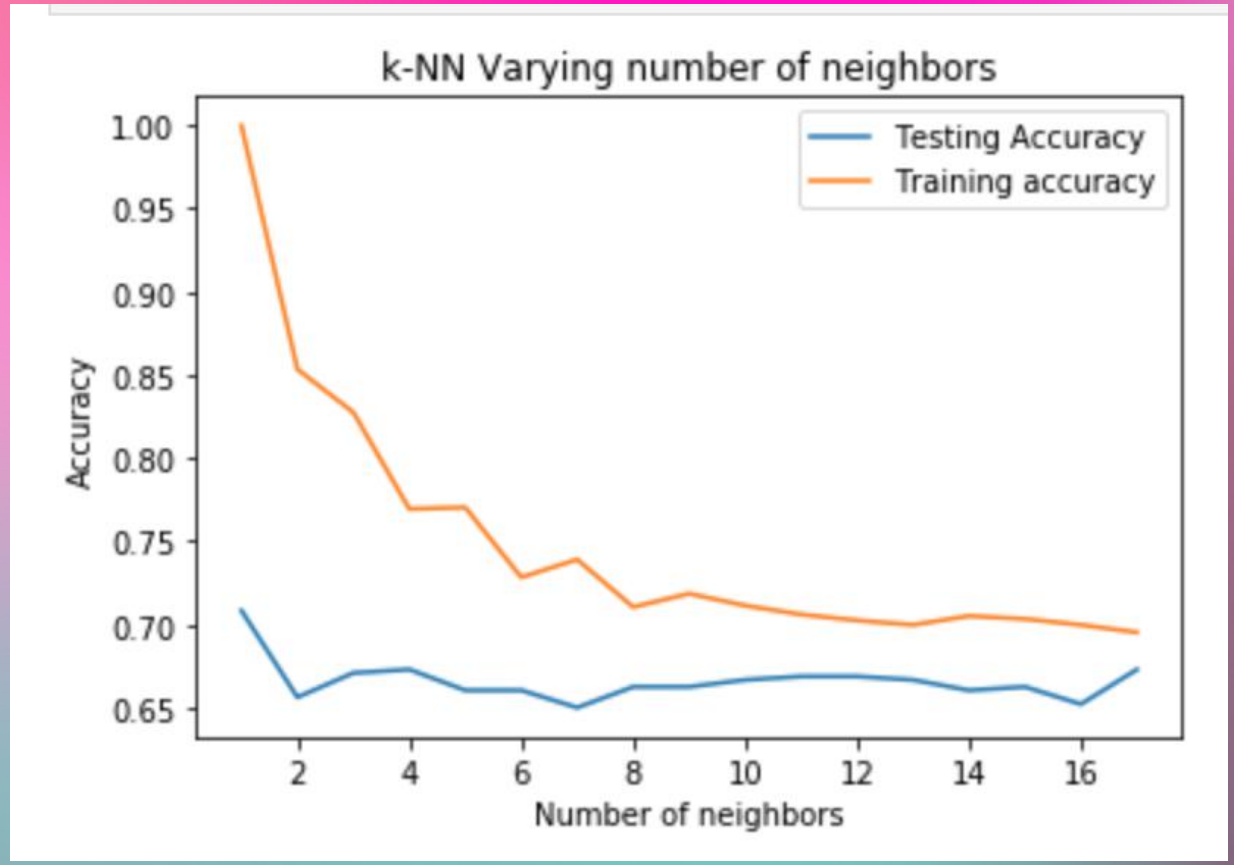
KNN算法实现

```
for i,k in enumerate(neighbors):
    #Setup a knn classifier with k neighbors
    knn = KNeighborsClassifier(n_neighbors=k)

    #Fit the model
    knn.fit(X_train, Y_train)

    #Compute accuracy on the training set
    train_accuracy[i] = knn.score(X_train, Y_train)

    #Compute accuracy on the test set
    test_accuracy[i] = knn.score(X_test, Y_test)
```



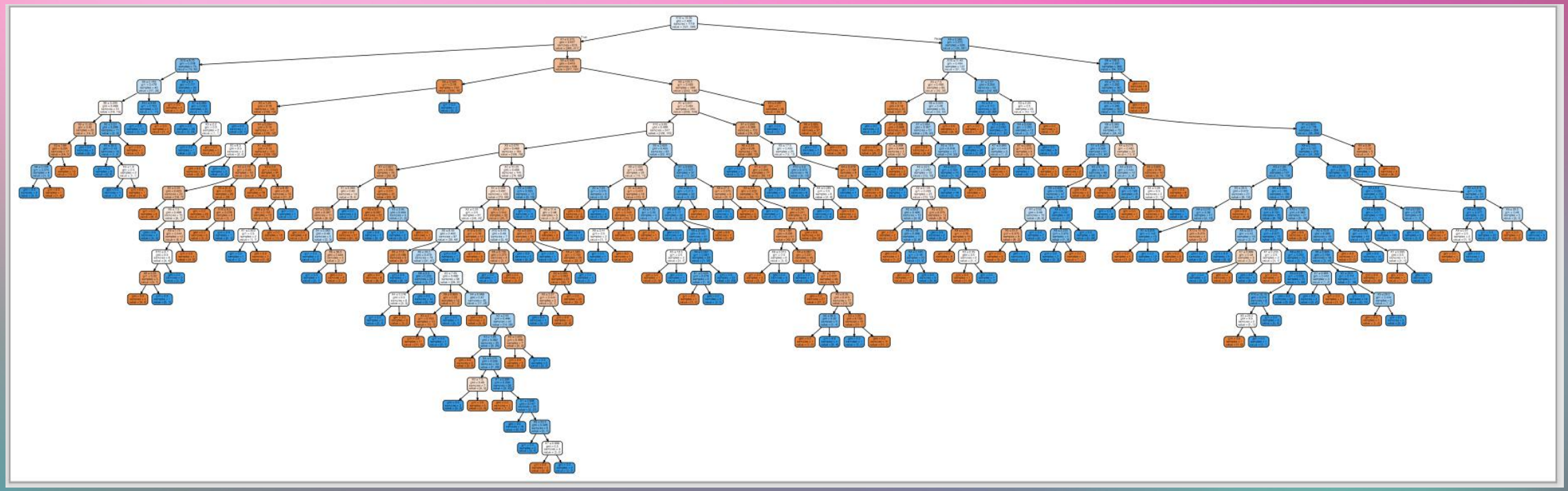
决策树算法实现

2. 决策树算法实现

决策树算法实现

```
from sklearn import tree
treeclf = tree.DecisionTreeClassifier(criterion='gini')
treeclf.fit(X_train,Y_train)
test_accuracy = treeclf.score(X_test,Y_test)
print(test_accuracy)
```

```
0.75
```



朴素贝叶斯算法实现

3、朴素贝叶斯算法实现

朴素贝叶斯算法实现

```
from sklearn.naive_bayes import GaussianNB

bayesclf = GaussianNB().fit(X_train, Y_train)
test_accuracy = bayesclf.score(X_test, Y_test)
```

```
test_accuracy
```

```
0.7234375
```

模型评估

```
#混淆函数
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
Y_pred = knn.predict(X_test)
#print(confusion_matrix(Y_test,Y_pred))
print(classification_report(Y_test,Y_pred))
```

```
             precision    recall  f1-score   support

        0.0       0.63      0.62      0.63       298
        1.0       0.67      0.68      0.68       342

avg / total       0.65      0.65      0.65       640
```

模型评估

```
#特征选择 树形选择法
from sklearn.ensemble import ExtraTreesClassifier
clf = ExtraTreesClassifier()
clf = clf.fit(X_train, Y_train)
clf.feature_importances_
```

```
array([0.06493354, 0.09689237, 0.06938514, 0.07137427, 0.06919244,
       0.06453383, 0.09306585, 0.07423613, 0.07125195, 0.09700279,
       0.22813169])
```

```
#方差选择法
from sklearn.feature_selection import VarianceThreshold
VarianceThreshold(threshold=3).fit_transform(X_train)
```

```
array([[ 8.8, 19. , 72. ],
       [ 7.6, 10. , 88. ],
       [ 9.9,  6. , 33. ],
       ...,
       [ 8.3,  6. , 12. ],
       [ 6.7, 15. , 36. ],
       [ 7.9, 23. , 49. ]])
```

```
#卡方检验
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
SelectKBest(chi2, k=5).fit_transform(X, Y)
```

```
array([[0.39726027, 0.        , 0.09893993, 0.13772455, 0.15384615],
       [0.52054795, 0.        , 0.2155477 , 0.20958084, 0.21538462],
       [0.43835616, 0.04      , 0.16961131, 0.19161677, 0.21538462],
       ...,
       [0.26712329, 0.13      , 0.12014134, 0.25149701, 0.4       ],
       [0.35958904, 0.12      , 0.13427562, 0.22754491, 0.27692308],
       [0.13013699, 0.47      , 0.12720848, 0.19760479, 0.4       ]])
```

非常感谢