華東理工大學

模式识别大作业

题　　目	毒蘑菇分类识别
学　　院	信息科学与工程
专　　业	控制科学与工程
组　　员	罗　粤
指导教师	赵海涛

完成日期：　2019 年 12 月 10 日

1 毒蘑菇分类识别问题简介

每年由于误食毒蘑菇而亡的居民不在少数，因此对于毒蘑菇进行分类识别显得十分重要。根据Kaggle给的毒蘑菇数据集，是一个包含8123个样本的数据集，有23个特征，为菌盖颜色、菌盖形状、菌盖表面形状、气味、菌褶等。每个种类都被确定为绝对可食用，绝对有毒或食用不明（不建议食用），后一类与有毒类结合在一起。同时也明确指出，没有简单的规则来确定蘑菇的食用性。因此，所要解决的问题就是，利用一个包含蘑菇各种特征参数的数据集，选取一定的分类方法，对蘑菇进行进行分类识别，将有毒蘑菇和可食用的蘑菇区分开来。

2 数据分析处理

2.1 数据结构分析

从Kaggle上下载数据后，得到一个毒蘑菇数据集，数据大致情况如下：

	class edible=e, poisonous=p	cap-shape bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s	cap-surface fibrous=f,grooves=g,scaly=y,smooth=s	cap-color brown=n,buff=b,cinnamon=c,gray=g,green=r,pink=p,purple=u,red=e,white=w,yellow=y	bruises bruises=t,no=f	odor almond=e=c,fishy m,none=y=s
	e 52% p 48%	x 45% f 39% Other (4) 16%	y 40% s 31% Other (2) 29%	n 28% g 23% Other (8) 49%	true 0 0% false 0 0%	n f Other (7
17	e	f	f	w	f	n
18	p	x	s	n	t	p
19	p	x	y	w	t	p
20	p	x	s	n	t	p
21	e	b	s	y	t	a
22	p	x	y	n	t	p
23	e	b	y	y	t	l
24	e	b	y	w	t	a
25	e	b	s	w	t	l
26	p	f	s	w	t	p

图2-1 毒蘑菇特征分布情况

class	cap-shape	cap-surfac	cap-color	bruises	odor	gill-attachi	gill-spacin	gill-size	gill-color	stalk-shap	stalk-root	stalk-surfa	s
p	x	s	n	t	p	f	c	n	k	e	e	s	s
e	x	s	y	t	a	f	c	b	k	e	c	s	s
e	b	s	w	t	l	f	c	b	n	e	c	s	s
p	x	y	w	t	p	f	c	n	n	e	e	s	s
e	x	s	g	f	n	f	w	b	k	t	e	s	s
e	x	y	y	t	a	f	c	b	n	e	c	s	s
e	b	s	w	t	a	f	c	b	g	e	c	s	s
e	b	y	w	t	l	f	c	b	n	e	c	s	s
p	x	y	w	t	p	f	c	n	p	e	e	s	s
e	b	s	y	t	a	f	c	b	g	e	c	s	s
e	x	y	y	t	l	f	c	b	g	e	c	s	s
e	x	y	y	t	a	f	c	b	n	e	c	s	s
e	b	s	y	t	a	f	c	b	w	e	c	s	s
p	x	y	w	t	p	f	c	n	k	e	e	s	s
e	x	f	n	f	n	f	w	b	n	t	e	s	f
e	s	f	g	f	n	f	c	n	k	e	e	s	s
e	f	f	w	f	n	f	w	b	k	t	e	s	s
p	x	s	n	t	p	f	c	n	n	e	e	s	s
p	x	y	w	t	p	f	c	n	n	e	e	s	s
p	x	s	n	t	p	f	c	n	k	e	e	s	s

图2-2 毒蘑菇特征数据情况

其中，共有8123个样本的数据集，有23个特征，各特征表示如下：

（1）class（类型）：可食用= e，有毒= p；

（2）cap-shape（帽型）：凸形= x，平面= f，钟形= b，圆锥形= c，球形= k，下沉= s；

（3）cap-surface（表面）：纤维= f，槽= g，鳞片= y，光滑= s；

（4）cap-color（颜色）：棕色= n，浅黄色= b，肉桂色= c，灰色= g，绿色= r，粉红色= p，紫色= u，红色= e，白色= w，黄色= y；

（5）bruises（是否有瘀伤）：有瘀伤= t，没有瘀伤 = f；

（6）odor（气味）：杏仁= a，茴香= l，杂油= c，腥= y，污垢= f，麝香= m，无= n，辛辣= p，辣= s；

（7）gill-attachment（附着物）：有附着物= a，无附着物= f，有缺口= n；

（8）gill-spacing（间距）：接近=c,拥挤=w,遥远=d；

（9）gill-size（大小）：宽= b，窄= n；

（10）gill-color（颜色）：黑色= k，棕色= n，浅黄色= b，巧克力= h，灰色= g，绿色= r，橙色= o，粉红色= p，紫色= u，红色= e，白色= w，黄色= y；

（11）stalk-shape（秸秆形状）：扩大= e，渐细= t；

（12）stalk-root（茎根）：球根= b，棍棒= c，杯= u，等于= e，根茎= z，根= r，缺失=？；

（13）stalk-surface-above-ring（茎表面上方环）：纤维= f，鳞片= y，丝质= k，光滑= s；

（14）stalk-surface-below-ring（茎表面以下环）：纤维= f，鳞片= y，丝质= k，光滑= s；

（15）stalk-color-above-ring（柄上颜色）：棕色= n，浅黄色= b，肉桂色= c，灰色= g，橙色= o，粉红色= p，红色= e，白色= w，黄色= y；

（16）stalk-color-below-ring（茎杆颜色-下方环）：棕色= n，浅黄色= b，肉桂色= c，灰色= g，橙色= o，粉红色= p，红色= e，白色= w，黄色= y；

（17）veil-type（面纱类型）：局部= p，通用= u；

（18）veil-color（面纱颜色）：棕色= n，橙色= o，白色= w，黄色= y；

（19）ring-number(环数)：none = n，one = o，two = t；
（20）ring-type（环形）：蜘蛛网= c，消逝= e，喇叭形= f，大= l，无= n，
吊坠= p，护套= s，区域= z；
（21）spore-print-color（孢子颜色）：黑色= k，棕色= n，浅黄色= b，巧克力= h，
绿色= r，橙色= o，紫色= u，白色= w，黄色= y；
（22）population（密度）：丰富= a，成群= c，大量= n，分散= s，几个= v，
单独= y；
（23）habitat（生长环境）：草= g，叶= l，草甸= m，路径= p，城市= u，废物= w，
森林= d；

2.2 数据预处理

在对数据进行分析处理前，首先要将数据进行清理。通过观察数据可以发现，数据集没有缺失值。

(1) 利用label_encoder.fit_transform将数据集数字化编码；

cap-shape	cap-surfac	cap-color	bruises	odor	gill-attach	gill-spacin	gill-size	gill-color	stalk-shap	stalk-root	stalk-surfa	stalk-surfa
5	0	9	0	2	1	0	0	3	0	1	1	1
2	3	2	0	7	1	0	1	0	1	0	1	1
2	0	3	1	5	1	0	0	5	1	1	2	2
3	2	4	0	2	1	0	1	0	1	0	2	2
2	3	3	0	2	1	0	0	2	0	1	1	1
5	0	3	0	2	1	0	0	2	0	1	1	1
2	3	2	0	2	1	0	1	0	1	0	2	1
5	3	9	1	3	1	0	0	5	0	4	2	3
5	3	8	1	3	1	0	0	2	0	2	2	2
2	2	4	0	2	1	0	1	0	1	0	2	1
5	0	8	0	5	1	1	0	5	1	3	2	0
2	3	3	1	5	1	0	0	10	1	1	2	2
5	0	2	1	5	1	0	0	9	1	1	2	2
3	3	4	0	2	1	0	1	0	1	0	1	1
0	3	9	1	0	1	0	0	4	0	2	2	2
2	0	2	1	5	1	0	0	9	1	1	2	2
5	3	4	1	5	1	0	0	10	1	1	2	2
2	0	8	0	5	1	1	0	5	1	3	2	2

图2-3 数据集数字化

(2) 利用data.drop将数据集中的class一列删除，便于之后进行特征分析；

(3) 利用train_test_split将数据集分为测试集和训练集，其中测试集占20%，训练集占80%。

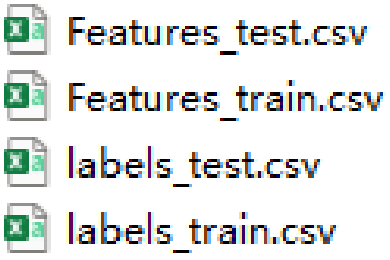


图2-4 测试集与训练集

3 问题的求解与分析

3.1 模型构建

3.1.1 决策树

决策树（Decision Tree）是一种基本的分类与回归方法，本文主要讨论分类决策树。决策树模型呈树形结构，在分类问题中，表示基于特征对数据进行分类的过程。它可以认为是if-then规则的集合。每个内部节点表示在属性上的一个测试，每个分支代表一个测试输出，每个叶节点代表一种类别。

决策树的优点：

1）可以自学习。在学习过程中不需要使用者了解过多的背景知识，只需要对训练数据进行较好的标注，就能进行学习。

2）决策树模型可读性好，具有描述性，有助于人工分析；

3）效率高，决策树只需要一次构建，就可以反复使用，每一次预测的最大计算次数不超过决策树的深度。

决策树构建的基本步骤如下：

1. 开始，所有记录看作一个节点
2. 遍历每个特征的每一种分裂方式，找到最好的分裂特征（分裂点）
3. 分裂成两个或多个节点
4. 对分裂后的节点分别继续执行2-3步，直到每个节点足够“纯”为止

如何评估分裂点的好坏？如果一个分裂点可以将当前的所有节点分为两类，使得每一类都很“纯”，也就是同一类的记录较多，那么就是一个好分裂点。

具体实践中，到底选择哪个特征作为当前分裂特征，常用的有下面三种算法：

ID3：使用信息增益g(D,A)进行特征选择

C4.5：信息增益率 =g(D,A)/H(A)

CART：基尼系数

一个特征的信息增益(或信息增益率，或基尼系数)越大，表明特征对样本的熵的减少能力更强，这个特征使得数据由不确定性到确定性的能力越强。

3.1.2 朴素贝叶斯

朴素贝叶斯（Naive Bayesian）是基于贝叶斯定理和特征条件独立假设原则的分类方法。通过给出的特征计算分类的概率，选取概率大的情况进行分类。也是基于概率论的一种机器学习分类方法。分类目标确定，属于监督学习。

分类原理：基于概率论，二分类问题如下：

如果 $P_1 > P_2$，分入类别1否则分入类别2。

其次，贝叶斯定理，有 $p(c_i \mid x,y) = p(x,y \mid c_i) * p(c_i) / p(x,y)$

其中，x,y表示特征变量，c_i 表示类别。$p(c_i \mid x,y)$ 即表示在特征x,y出现的情况下，分入类别 c_i 的概率。结合如上：

如果 $p(c_i \mid x, y) > p(c_j \mid x, y)$，分入类别i，否则分入类别j。

贝叶斯定理最大的好处是可以用已知的三个概率去计算未知的概率，而如果仅仅是为了比较 $p(c_i \mid x, y)$ 和 $p(c_j \mid x, y)$ 的大小，只需要已知两个概率即可，分母相同，比较 $p(x,y \mid c_i)p(c_i)$ 和 $p(x,y \mid c_j)p(c_j)$ 即可。

3.1.3 支持向量机

SVM定义：支持向量机（英语：support vector machine，常简称为SVM，又名支持向量网络）是在分类与回归分析中分析数据的监督式学习模型与相关的学习算法。给定一组训练实例，每个训练实例被标记为属于两个类别中的一个或另一个，SVM训练算法创建一个将新的实例分配给两个类别之一的模型，使其成为非概率二元线性分类器。SVM模型是将实例表示为空间中的点，这样映射就使得单独类别的实例被尽可能宽的明显的间隔分开。然后，将新的实例映射到同一空间，并基于它们落在间隔的哪一侧来预测所属类别。简言之，SVM就是一种二类分类模型，它的基本模型是定义在特征空间的间隔最大的线性分类器，SVM的学习策略就是间隔最大化。

支持向量机思想：为了把两组数据分开，在空心点的类别找到一个或多个点离实心点最近，在实心点中找到一个或多个点与空心点最近，分类实心和空心点取决于这些边界上的点而与离边界较远的点，即在分割两类别点的时候，只需要考虑支持向量，通过支持向量确定分割直线。

3.1.4 K近邻法

k近邻法（k-nearest neighbor, k-NN）是一种基本分类与回归方法，由Cover和Hart于1968年提出。分类时，对于新的实例，根据与它最接近的k个训练实例的类别，通过多数表决等方式，进行预测。对于给定的训练集，当k值，距离度量和分类决策规则（统称三要素）确定后，基于k近邻法的模型就已经确定了。所以，它实际上利用训练集对特征向量空间进行划分，并没有显示的学习过程。k近邻法，符合我们基本的认知，即“物以类聚，人以群分”，一件事物的类别通常与它附近的事物具有相似性。

KNN方法虽然从原理上也依赖于极限定理，但在类别决策时，只与极少量的相邻

样本有关。由于KNN方法主要靠周围有限的邻近的样本，而不是靠判别类域的方法来确定所属类别的，因此对于类域的交叉或重叠较多的待分样本集来说，KNN方法较其他方法更为适合。

3.2 模型评估

引入混淆矩阵用来评价模型分类效果，混淆矩阵是ROC曲线绘制的基础，同时它也是衡量分类型模型准确度中最基本，最直观，计算最简单的方法。

混淆矩阵就是分别统计分类模型归错类，归对类的观测值个数，然后把结果放在一个表里展示出来，这个表就是混淆矩阵。

混淆矩阵是评判模型结果的指标，属于模型评估的一部分。此外，混淆矩阵多用于判断分类器（Classifier）的优劣，适用于分类型的数据模型，如分类树（Classification Tree）、逻辑回归（logistic regression）、线性判别分析（Linear Discriminant Analysis）等方法。

在分类型模型评判的指标中，常见的方法有如下三种：混淆矩阵（也称误差矩阵，Confusion Matrix）、ROC曲线、AUC面积。本次主要利用第一种方法，即混淆矩阵，也称误差矩阵。

混淆矩阵（Confusion Matrix），它的本质远没有它的名字听上去那么拉风。矩阵，可以理解为就是一张表格，混淆矩阵其实就是一张表格而已。

以分类模型中最简单的二分类为例，对于这种问题，我们的模型最终需要判断样本的结果是0还是1，或者说是positive还是negative。

我们通过样本的采集，能够直接知道真实情况下，哪些数据结果是positive，哪些结果是negative。同时，我们通过用样本数据跑出分类型模型的结果，也可以知道模型认为这些数据哪些是positive，哪些是negative。

因此，我们就能得到这样四个基础指标，我称它们是一级指标（最底层的）：

真实值是positive，模型认为是positive的数量（True Positive=TP）

真实值是positive，模型认为是negative的数量（False Negative=FN）：这就是统计学上的第一类错误（Type I ERROR）

真实值是negative，模型认为是positive的数量（False Positive=FP）：这就是统计学上的第二类错误（Type II Error）

真实值是negative，模型认为是negative的数量（True Negative=TN）

将这四个指标一起呈现在表格中，就能得到如下这样一个矩阵，我们称它为混淆矩阵（Confusion Matrix）：

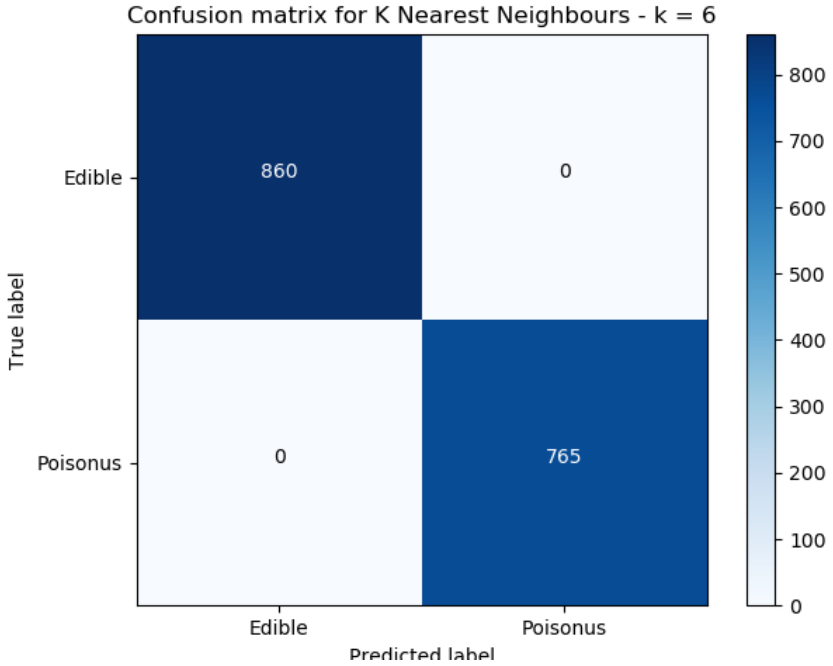


图3-1 混淆矩阵

混淆矩阵的指标

预测性分类模型，肯定是希望越准越好。那么，对应到混淆矩阵中，那肯定是希望TP与TN的数量大，而FP与FN的数量小。所以当我们得到了模型的混淆矩阵后，就需要去看有多少观测值在第二、四象限对应的位置，这里的数值越多越好；反之，在第一、三四象限对应位置出现的观测值肯定是越少越好。

二级指标

但是，混淆矩阵里面统计的是个数，有时候面对大量的数据，光凭算个数，很难衡量模型的优劣。因此混淆矩阵在基本的统计结果上又延伸了如下4个指标，我称它们是二级指标（通过最底层指标加减乘除得到的）：

准确率（Accuracy）：针对整个模型

精确率（Precision）

灵敏度（sensitivity）：就是召回率（Recall）

特异度（Specificity）

```
Metrics for K Nearest Neighbours - k = 20
Precision:  0.997364953886693
Recall:  0.9895424836601308
Accuracy: 0.9938461538461538
Average Precision:  0.9918580705076869
```

图3-2 混淆矩阵二级评价指标

4 实验结果分析

4.1 决策树

```
def classify_by_decision_tree(Features_train, labels_train, Features_test,
labels_test):
    # #Decision Tree
    dt = tree.DecisionTreeClassifier()
    dt = dt.fit(Features_train, labels_train)
    predictions_dt = dt.predict(Features_test)
    predictions_dt = predictions_dt.reshape((-1, 1))
    print_metrics(labels_test, predictions_dt, 'Decision Tree')
    create_confusion_matrix(labels_test, predictions_dt, 'Decision Tree')
```

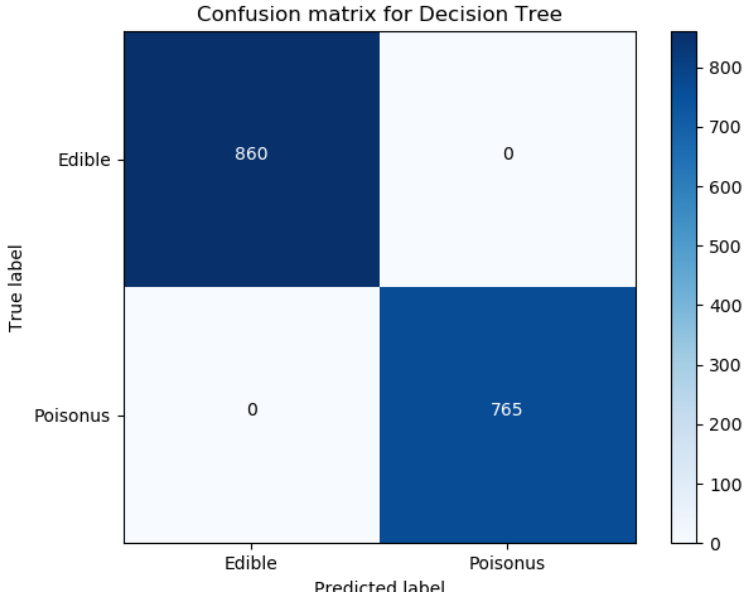


图4-1 决策树法分类结果

```
Metrics for Decision Tree
Precision:  1.0
Recall:  1.0
Accuracy: 1.0
Average Precision:  1.0
```

图4-2 决策树法二级指标情况

可以看出决策树法有很好的分类的效果，无论是混淆矩阵一级指标还是二级指标，都完美的将有毒和无毒的蘑菇区分开来。

4.2 朴素贝叶斯

```
def classify_by_bayesian(Features_train, labels_train, Features_test, labels_test):
    #Gaussian Naive Bayes
    gnb = GaussianNB()
    gnb.fit(Features_train, labels_train.values.ravel())
    predictions_gnb = gnb.predict(Features_test)

    print_metrics(labels_test, predictions_gnb,f'Gaussian Naive Bayes')
    create_confusion_matrix(labels_test, predictions_gnb, f'Gaussian Naive Bayes')
```

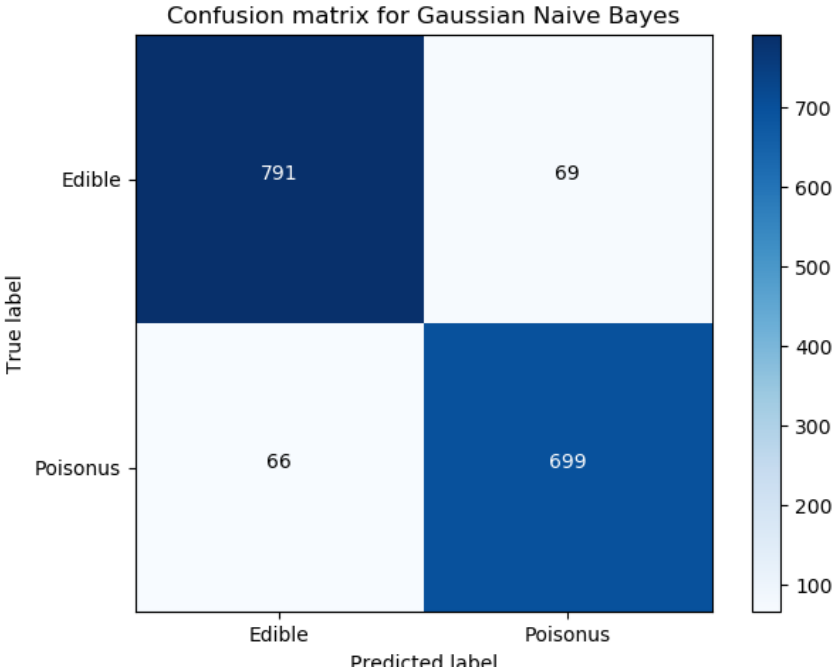


图4-3 贝叶斯法分类结果

```
Metrics for Gaussian Naive Bayes
Precision:  0.91015625
Recall:  0.9137254901960784
Accuracy: 0.916923076923077
Average Precision:  0.8722483503016591
```

图4-4 贝叶斯法二级指标情况

可以看出朴素贝叶斯法分类效果一般，从混淆矩阵上就可以看出错误率较大，量化后，可以发现准确度只有91.015%，不可以用作毒蘑菇分类的方法。

4.3 支持向量机

```
def classify_by_svm(Features_train, labels_train, Features_test, labels_test):
    #Support Vector Machines
    svc = svm.SVC(gamma='scale')
    svc.fit(Features_train, labels_train.values.ravel())
    predictions_svc = svc.predict(Features_test)
    print_metrics(labels_test, predictions_svc, f'Support Vector Machines')
    create_confusion_matrix(labels_test, predictions_svc, f'Support Vector Machines')
```

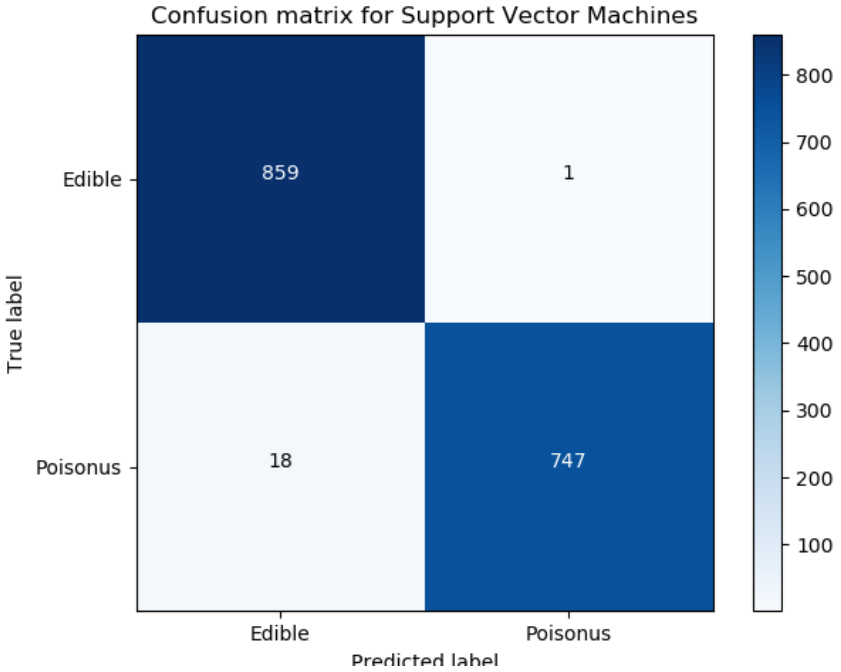


图4-5 支持向量机分类结果

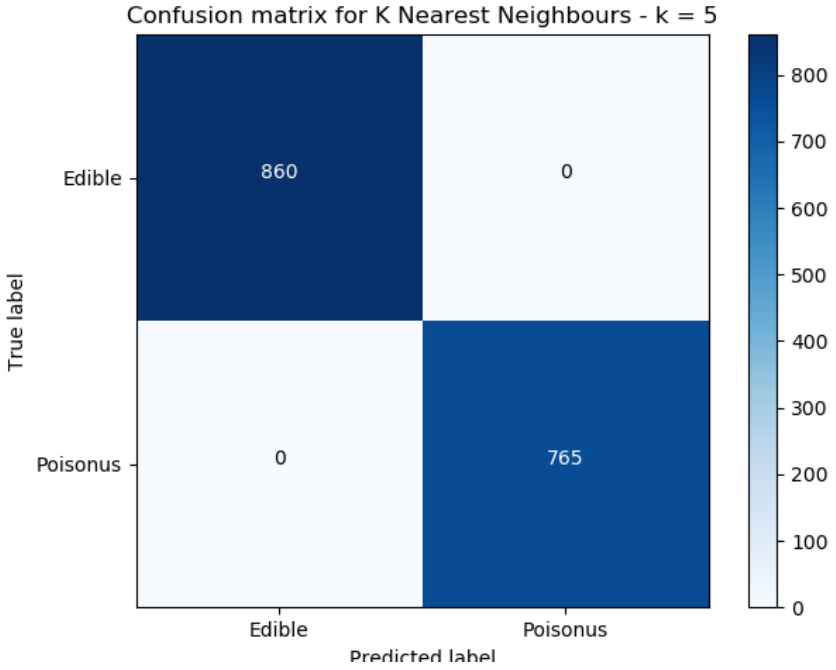
```
Metrics for Support Vector Machines
Precision:  0.9986631016042781
Recall:  0.9764705882352941
Accuracy: 0.9883076923076923
Average Precision:  0.9862420693493358
```

图4-6 贝叶斯法二级指标情况

从上表可以看出，贝叶斯法分类效果较好，从混淆矩阵上看出分类错误较少，从二级指标上看，准确率有99.866%，可以用作毒蘑菇分类。

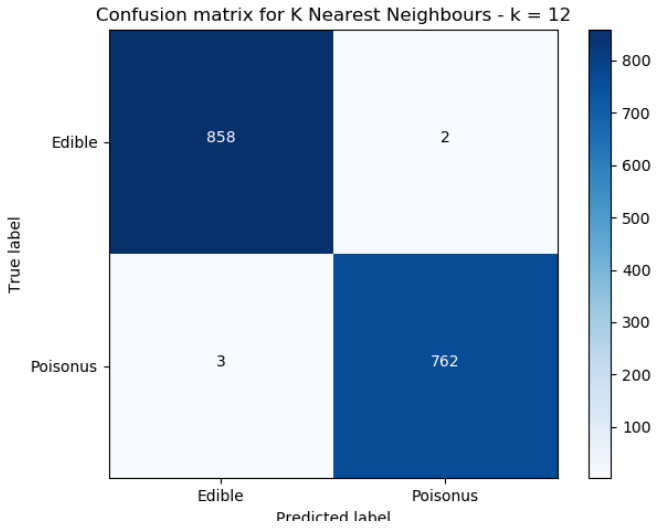
4.4 K近邻法

```
def classify_by_knn(Features_train, labels_train, Features_test, labels_test):
    #K Nearest Neighbours
    for i in range(5,21):
        knn = KNeighborsClassifier(n_neighbors=i)
        knn.fit(Features_train, labels_train.values.ravel())
        predictions_knn = knn.predict(Features_test)
        print_metrics(labels_test, predictions_knn,f'K Nearest Neighbours - k = {i}')
  create_confusion_matrix(labels_test, predictions_knn, f'K Nearest Neighbours - k = {i}')
```



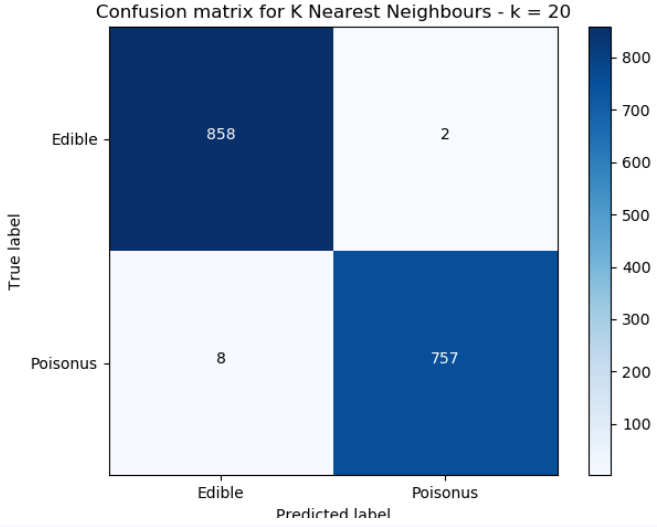

```
Metrics for K Nearest Neighbours - k = 5
Precision:  1.0
Recall:  1.0
Accuracy: 1.0
Average Precision:  1.0
```

图4-7 K近邻法分类结果(K=5)




```
Metrics for K Nearest Neighbours - k = 12
Precision:  0.9973821989528796
Recall:  0.996078431372549
Accuracy: 0.9969230769230769
Average Precision:  0.9953170500580417
```

图4-8 K近邻法分类结果(K=12)




```
Metrics for K Nearest Neighbours - k = 20
Precision:  0.997364953886693
Recall:  0.9895424836601308
Accuracy: 0.9938461538461538
Average Precision:  0.9918580705076869
```

图4-9 K近邻法分类结果(K=20)

从上表可以看出，K近邻法分类效果和分的类别数关系较大，在分5类时，效果很好，能完美区分毒蘑菇和可食用蘑菇；分为12类、20类时，效果开始变差，准确率下降到99.736%。

5 总结

KNN算法，对于数值型的数据适配度比较高，且预处理较少，一般单一类数据归一化即可，选择求距离的公式也可以依托实际进行。注意，KNN有算法的一个优点为对异常值不敏感，但是注意，我们在预处理的时候，如果能将极端数据去掉后再进行归一化，分类效果会更好。决策树算法对于数据预处理要求高一点，需要预分类，分类过程如果面向问题本身会更加好，用于制作实际算法投入运用的话，由用户来进行一个标度效果对于特定用户本身会更好。但是数据过多的时候，决策树剪枝方面要多进行考虑，但是势必带来准确度降低。支持向量机适用于二类分类问题，对于多分类问题解决效果并不好，同时小集群分类效果好，但观测样本太多时，效率较低。朴素贝叶斯方法对于数据的要求较高，预处理的过程较多，但是效果不错，对于数据较少的情况依旧适用。