In [1]:

```
from __future__ import division
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn import svm, metrics
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np
import itertools
from sklearn.metrics import confusion_matrix
import seaborn as sns
import threading
from sklearn.datasets.samples_generator import make_classification
from sklearn.neighbors import KNeighborsClassifier
from bubbly.bubbly import bubbleplot
from plotly.offline import init_notebook_mode, iplot
from sanwei import k_means_run
```

```
/home/my/.local/lib/python3.6/site-packages/sklearn/ensemble/weight_bo
osting.py:29: DeprecationWarning: numpy.core.umath_tests is an interna
l NumPy module and should not be imported. It will be removed in a fut
ure NumPy release.
  from numpy.core.umath_tests import inner1d
```

In [2]:

```
def plot_confusion_matrix(cm, classes,normalize=False,title='Confusion matrix',cmap
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title, fontsize=30)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, fontsize=20)
    plt.yticks(tick_marks, classes, fontsize=20)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.

    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt), horizontalalignment="center",
                 color="white" if cm[i, j] < thresh else "black", fontsize=40)

    plt.tight_layout()
    plt.ylabel('True label', fontsize=30)
    plt.xlabel('Predicted label', fontsize=30)
    return plt
```

In [3]:

```
def JZ(juzheng):
    #confusion = np.array(([91, 0, 0,4], [0, 92, 1,4], [0, 0, 95,4],[1,1,1,1]))
    confusion = juzheng
    # 热度图，后面是指定的颜色块，可设置其他的不同颜色
    plt.imshow(confusion, cmap=plt.cm.Blues)
    # ticks 坐标轴的坐标点
    # label 坐标轴标签说明
    indices = range(len(confusion))
    plt.xticks(indices, [1, 3, 3,4,5,6,7,8,9,10])
    plt.yticks(indices, [1, 2, 3,4,5,6,7,8,9,10])
    plt.colorbar()
    plt.xlabel('预测值')
    plt.ylabel('真实值')
    plt.title('混淆矩阵')
    sns.set(font_scale=0.8)
    # plt.rcParams两行是用于解决标签不能显示汉字的问题
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False

    # 显示数据
    for first_index in range(len(confusion)):  # 第几行
        for second_index in range(len(confusion[first_index])):  # 第几列
            plt.text(first_index, second_index, confusion[first_index][second_index
    # 在matlab里面可以对矩阵直接imagesc(confusion)
    # 显示
    plt.show()
```

In [4]:

```
def xiao_ti_qing(a):
    #小提琴图
    a['categoryname'].value_counts()  # 对分类变量的类别进行计数
    # 后面将研究不同类型的'gearbox'对应'price'的差异
    x = a['categoryname']
    y = a['notified']  # 在原数据集中，'price'为目标变量
    # 绘制小提琴图
    sns.violinplot(x=x, y=y, data=a)
    plt.show()
    # 在sns.violinplot中，x是类别变量，y是数值型变量，data用于指定数据集
```

In [5]:

```
def xiang_guan(a):
    # 特征间相关系数热力图
    f = a.corr()
    sns.heatmap(f, annot=True)
    plt.show()
```

In [6]:

```
def categoryname(a):
    X = a.drop(["categoryname"], axis=1)  # 11829
    y = a["categoryname"]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random
    # 标准化
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)

    # Random Forest Classifier
    print("***************[categoryname]*****************")
    print("Random Forest Classifier")
    # n_estimators，表示选择多少棵树来构造随机森林；具体解释看《边学边练超系统掌握人工智机器学习
    rfc = RandomForestClassifier(n_estimators=200)
    rfc.fit(X_train, y_train)
    pred_rfc = rfc.predict(X_test)
    print("Random Forest classification_report==\n", classification_report(y_test,
    print("Random Forest confusion_matrix==\n", confusion_matrix(y_test, pred_rfc))
    # 混淆矩阵可视化
    # t1 = threading.Thread(target=JZ,args=[confusion_matrix(y_test, pred_rfc)])
    # t1.start()
    JZ(confusion_matrix(y_test, pred_rfc))
```

In [7]:

```
def notified(a):
    X = a.drop(["notified"], axis=1)  # 11829
    y = a["notified"]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random
    # 标准化
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)

    # Random Forest Classifier
    print("******************[notified]******************")
    print("随机森林测试")
    # n_estimators，表示选择多少棵树来构造随机森林；具体解释看《边学边练超系统掌握人工智机器学习
    rfc = RandomForestClassifier(n_estimators=200)
    rfc.fit(X_train, y_train)
    pred_rfc = rfc.predict(X_test)
    print("随机森林分类报告==\n", classification_report(y_test, pred_rfc))
    print("随机森林混淆矩阵==\n", confusion_matrix(y_test, pred_rfc))
    error = mean_squared_error(y_test, pred_rfc)
    print("随机森林预测的准确率为: ",rfc.score(X_test, y_test))
    print("随机森林-均方差为: \n", error)
    print('\n')

    estimator = KNeighborsClassifier(n_neighbors=3)
    estimator.fit(X_train, y_train)
    pred_est = estimator.predict(X_test)
    print("KNN 分类报告==\n", classification_report(y_test, pred_est))
    print('KNN混淆矩阵==\n',confusion_matrix(y_test, pred_est))
    error = mean_squared_error(y_test, estimator.predict(X_test))
    print("KNN预测的准确率为: ", estimator.score(X_test, y_test))
    print("KNN-均方差为: \n", error)
    print('\n')

    # SVM Classifier
    print("SVM 测试")
    svmc = svm.SVC()
    svmc.fit(X_train, y_train)
    pred_svmc = svmc.predict(X_test)
    print("SVM分类报告==\n", classification_report(y_test, pred_svmc))
    print("SVM混淆矩阵==\n", confusion_matrix(y_test, pred_svmc))
    error = mean_squared_error(y_test, pred_svmc)
    print("SVM预测的准确率为: ", svmc.score(X_test, y_test))
    print("SVM-均方误差为: \n", error)
    print('\n')

    # Neural Network
    print("神经网络")
    mlpc = MLPClassifier(hidden_layer_sizes=(11, 11, 11), max_iter=500)
    mlpc.fit(X_train, y_train)
    pred_mlpc = mlpc.predict(X_test)
    print("神经网络分类测试==\n", classification_report(y_test, pred_mlpc))
    print("神经网络混淆矩阵==\n", confusion_matrix(y_test, pred_mlpc))
    error = mean_squared_error(y_test, pred_mlpc)
    print("神经网络预测的准确率为: ", mlpc.score(X_test, y_test))
    print("神经网络-均方误差为: \n", error)
```

In [8]:

```
path = "cybersecurity_training.csv"
a = pd.read_csv(path, delimiter=",")
```

In [9]:

```
# drop () 方法是
a = a.drop(columns=['alert_ids', 'n1', 'n2', 'n3', 'n4', 'n5', 'n6', 'n7', 'n8', 'n
                    'grandparent_category', 'weekday', 'ip', 'ipcategory_name', 'ip
                    'parent_category',
                    'timestamp_dist', 'start_hour', 'start_minute', 'start_second',
                    'thrcnt_day'])

col_str = ['client_code', 'categoryname', 'dstipcategory_dominate', 'srcipcategory_

print(len(a))
```

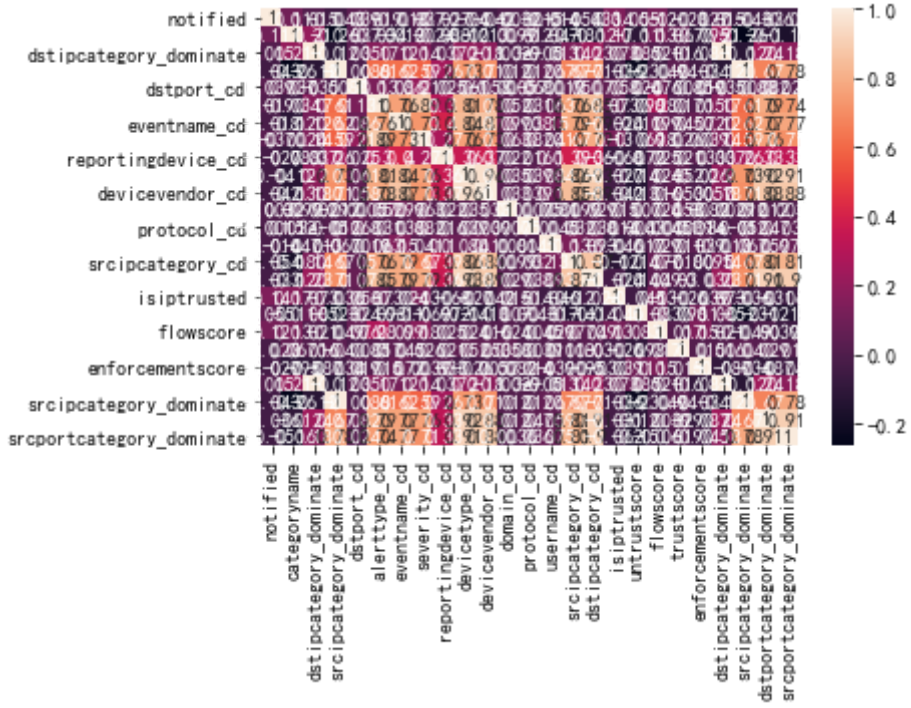
```
39427
```

In [10]:

```
# Converting non-numeric data into numeric data
## 离散型的数据转换成 0 0 0 到 n - 1 n-1 n-1 之间的数
for i in col_str:
    label_enc = LabelEncoder()
    label_enc.fit(a[i])
    a[i] = label_enc.transform(a[i])
```

In [11]:

```
# 特征间相关系数热力图
a_ = a[['notified','categoryname','dstipcategory_dominate','srcipcategory_dominate'
        'eventname_cd','severity_cd','reportingdevice_cd','devicetype_cd','deviceve
        'srcipcategory_cd','dstipcategory_cd','isiptrusted','untrustscore','flowsco
        'dstipcategory_dominate','srcipcategory_dominate','dstportcategory_dominate
xiang_guan(a_)
```



In [12]:

```
#受到攻击是否收到警告通知
notified(a)
```

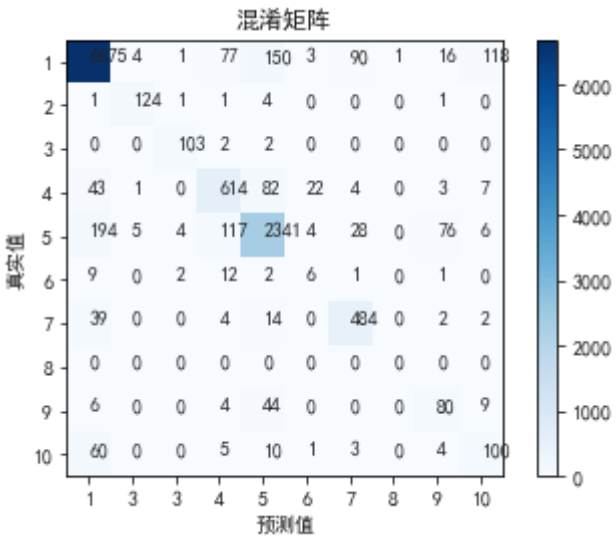
```
******************[notified]******************
随机森林测试
随机森林分类报告==
              precision    recall  f1-score   support

          0       0.95      0.99      0.97     11123
          1       0.52      0.21      0.30       706

avg / total       0.93      0.94      0.93     11829

随机森林混淆矩阵==
 [[10988   135]
 [  559   147]]
随机森林预测的准确率为:  0.9413306281173387
随机森林-均方差为:
 0.058669371882661255


KNN 分类报告==
              precision    recall  f1-score   support

          0       0.95      0.98      0.97     11123
          1       0.38      0.16      0.23       706

avg / total       0.91      0.93      0.92     11829

KNN混淆矩阵==
 [[10940   183]
 [  593   113]]
KNN预测的准确率为:  0.934398512131203
KNN-均方差为:
 0.06560148786879702


SVM 测试
SVM分类报告==
              precision    recall  f1-score   support

          0       0.94      1.00      0.97     11123
          1       0.71      0.01      0.01       706

avg / total       0.93      0.94      0.91     11829

SVM混淆矩阵==
 [[11121     2]
 [  701     5]]
SVM预测的准确率为:  0.9405697861188604
SVM-均方误差为:
 0.059430213881139574


神经网络
神经网络分类测试==
              precision    recall  f1-score   support

          0       0.94      1.00      0.97     11123
```

```
           1       0.52      0.05      0.10        706

avg / total       0.92      0.94      0.92     11829

神经网络混淆矩阵==
 [[11089    34]
 [  669    37]]
神经网络预测的准确率为:  0.9405697861188604
神经网络-均方误差为:
 0.059430213881139574
```

In [13]:

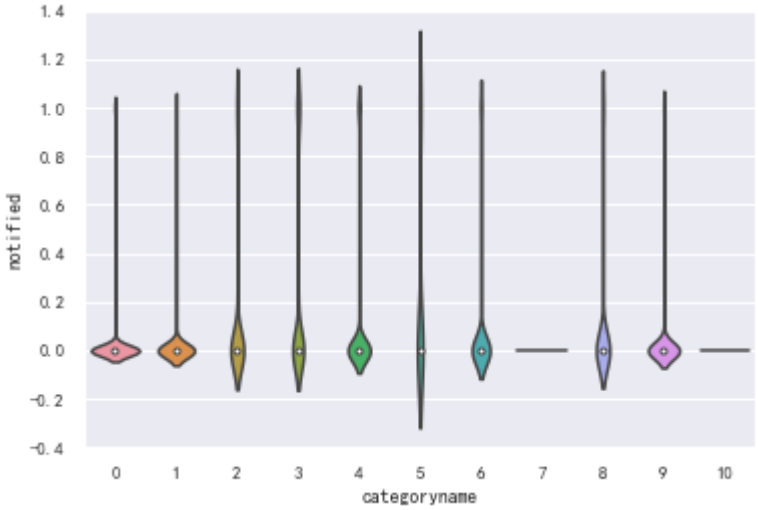
```
# categoryname混淆矩阵可视化
# t1 = threading.Thread(target=JZ,args=[confusion_matrix(y_test, pred_rfc)])
# t1.start()
categoryname(a)
```

```
***************[categoryname]*****************
Random Forest Classifier

/home/my/.local/lib/python3.6/site-packages/sklearn/metrics/classifica
tion.py:1135: UndefinedMetricWarning:

Precision and F-score are ill-defined and being set to 0.0 in labels w
ith no predicted samples.


Random Forest classification_report==
              precision    recall  f1-score   support

           0       0.94      0.95      0.94      7027
           1       0.94      0.93      0.93       134
           2       0.96      0.93      0.94       111
           3       0.79      0.73      0.76       836
           4       0.84      0.88      0.86      2649
           5       0.18      0.17      0.17        36
           6       0.89      0.79      0.84       610
           7       0.00      0.00      0.00         1
           8       0.56      0.44      0.49       183
           9       0.55      0.41      0.47       242

avg / total       0.89      0.89      0.89     11829

Random Forest confusion_matrix==
 [[6675    1    0   43  194    9   39    0    6   60]
 [   4  124    0    1    5    0    0    0    0    0]
 [   1    1  103    0    4    2    0    0    0    0]
 [  77    1    2  614  117   12    4    0    4    5]
 [ 150    4    2   82 2341    2   14    0   44   10]
 [   3    0    0   22    4    6    0    0    0    1]
 [  90    0    0    4   28    1  484    0    0    3]
 [   1    0    0    0    0    0    0    0    0    0]
 [  16    1    0    3   76    1    2    0   80    4]
 [ 118    0    0    7    6    0    2    0    9  100]]
```



In [14]:

```
# 哪几种类别攻击的严重程度更容易被检测到
xiao_ti_qing(a)
```



In [15]:

```
#k_means聚类三维可视化
k_means_run()
```

```
/usr/local/lib/python3.6/dist-packages/matplotlib/collections.py:902:
RuntimeWarning:

invalid value encountered in sqrt
```