加载数据

In [2]:

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import numpy as np

# 加载数据
df = pd.read_csv('D:\KDD_CUP_99\kdd_cup\AllData.csv',header=None)
df
```

Out[2]:

	0	1	2	3	4	5	6	7	8	9	...	32	33	34	35	36	37	38
0	0	tcp	http	SF	181	5450	0	0	[8]	0	...	9	1.0	0.0	0.11	0.00	0.00	0.00
1	0	tcp	http	SF	239	486	0	0	[8]	0	...	19	1.0	0.0	0.05	0.00	0.00	0.00
2	0	tcp	http	SF	235	1337	0	0	[8]	0	...	29	1.0	0.0	0.03	0.00	0.00	0.00
3	0	tcp	http	SF	219	1337	0	0	[8]	0	...	39	1.0	0.0	0.03	0.00	0.00	0.00
4	0	tcp	http	SF	217	2032	0	0	[8]	0	...	49	1.0	0.0	0.02	0.00	0.00	0.00
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
494016	0	tcp	http	SF	310	1881	0	0	[8]	0	...	255	1.0	0.0	0.01	0.05	0.00	0.01
494017	0	tcp	http	SF	282	2286	0	0	[8]	0	...	255	1.0	0.0	0.17	0.05	0.00	0.01
494018	0	tcp	http	SF	203	1200	0	0	[8]	0	...	255	1.0	0.0	0.06	0.05	0.06	0.01
494019	0	tcp	http	SF	291	1200	0	0	[8]	0	...	255	1.0	0.0	0.04	0.05	0.04	0.01
494020	0	tcp	http	SF	219	1234	0	0	[8]	0	...	255	1.0	0.0	0.17	0.05	0.00	0.01

494021 rows × 42 columns

一、对数据集中正常连接和非正常连接数量进行统计

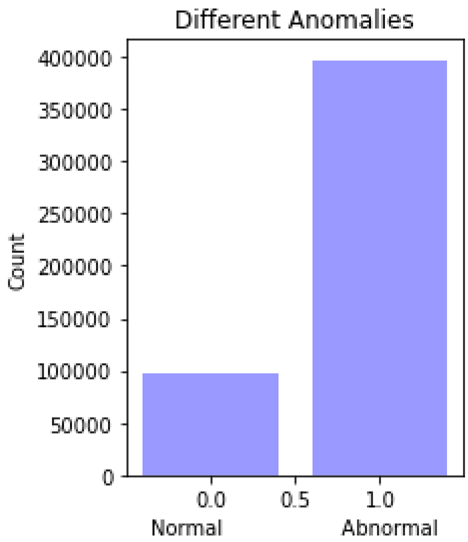
In [2]:

```
abnormal_num_list = []

normal_count = 0
abnormal_count = 0

for i in range(0,len(df)):
    if df[41][i] == 'normal.':
        normal_count += 1
    else:
        abnormal_count += 1
        abnormal_num_list.append(i)

print('正常连接数目：',normal_count)
print('非正常连接数目：',abnormal_count)
```

正常连接数目： 97278
非正常连接数目： 396743

In [3]:

```
%matplotlib inline
import matplotlib.pyplot as plt

# 绘制数目图，正常连接 vs 非正常连接
x = [0,1]
y = [normal_count,abnormal_count]
plt.figure(figsize=(3,4))
plt.bar(x,y,color = '#9999ff')
plt.title('Different Anomalies')
plt.xlabel('Normal                   Abnormal')
plt.ylabel('Count')
plt.show()
```



二、查看非正常连接的分布

In [4]:

```
buffer_overflow = []
loadmodule = []
perl = []
neptune = []
smurf = []
guess_passwd = []
pod = []
teardrop = []
portsweep = []
satan = []
phf = []
back = []
warezclient = []
ipsweep = []
nmap = []
rootkit = []
land = []
ftp_write = []
spy = []
imap = []
warezmaster = []
multihop = []
land = []

for i in abnormal_num_list:
    if df[41][i] == 'buffer_overflow.':
        buffer_overflow.append(df.loc[i])
    elif df[41][i] == 'loadmodule.':
        loadmodule.append(df.loc[i])
    elif df[41][i] == 'perl.':
        perl.append(df.loc[i])
    elif df[41][i] == 'neptune.':
        neptune.append(df.loc[i])
    elif df[41][i] == 'smurf.':
        smurf.append(df.loc[i])
    elif df[41][i] == 'guess_passwd.':
        guess_passwd.append(df.loc[i])
    elif df[41][i] == 'pod.':
        pod.append(df.loc[i])
    elif df[41][i] == 'teardrop.':
        teardrop.append(df.loc[i])
    elif df[41][i] == 'portsweep.':
        portsweep.append(df.loc[i])
    elif df[41][i] == 'satan.':
        satan.append(df.loc[i])
    elif df[41][i] == 'back.':
        back.append(df.loc[i])
    elif df[41][i] == 'ipsweep.':
        ipsweep.append(df.loc[i])
    elif df[41][i] == 'phf.':
        phf.append(df.loc[i])
    elif df[41][i] == 'nmap.':
        nmap.append(df.loc[i])
    elif df[41][i] == 'warezclient.':
        warezclient.append(df.loc[i])
    elif df[41][i] == 'rootkit.':
        rootkit.append(df.loc[i])
    elif df[41][i] == 'land.':
        land.append(df.loc[i])
```

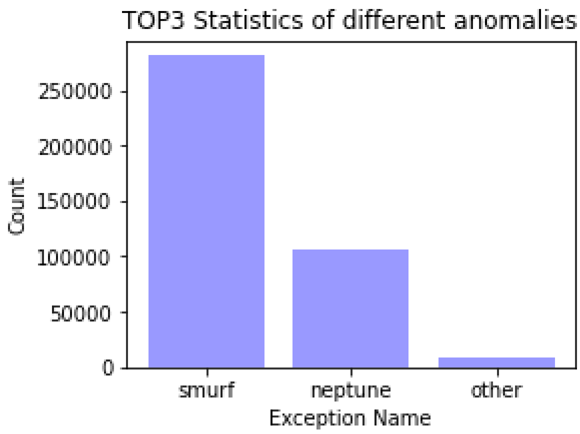
```
    elif df[41][i] == 'ftp_write.':
        ftp_write.append(df.loc[i])
    elif df[41][i] == 'imap.':
        imap.append(df.loc[i])
    elif df[41][i] == 'multihop.':
        multihop.append(df.loc[i])
    elif df[41][i] == 'warezmaster.':
        warezmaster.append(df.loc[i])
    elif df[41][i] == 'spy.':
        spy.append(df.loc[i])
    else :
        print(df[41][i])
```

In [14]:

```
anomalies_count = []
anomalies_count.append(len(back))
anomalies_count.append(len(satan))
# anomalies_count.append(len(neptune))
# anomalies_count.append(len(smurf))
anomalies_count.append(len(teardrop))
anomalies_count.append(len(portsweep))
anomalies_count.append(len(warezclient))
anomalies_count.append(len(ipsweep))
anomalies_count.append(len(pod))
anomalies_count.append(len(nmap))
anomalies_count.append(len(multihop))
anomalies_count.append(len(land))
anomalies_count.append(len(phf))
anomalies_count.append(len(ftp_write))
anomalies_count.append(len(perl))
anomalies_count.append(len(guess_passwd))
anomalies_count.append(len(warezmaster))
anomalies_count.append(len(rootkit))
anomalies_count.append(len(land))
anomalies_count.append(len(buffer_overflow))
anomalies_count.append(len(loadmodule))
anomalies_count.append(len(spy))
anomalies_count.append(len(imap))
```

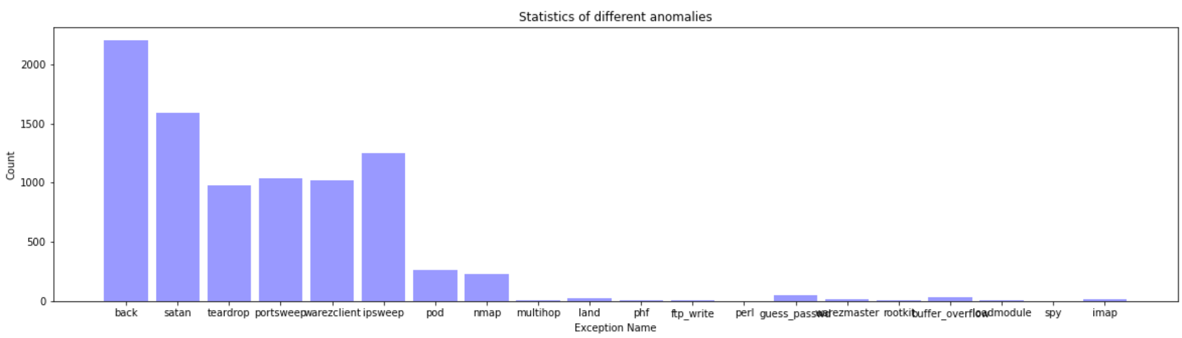
In [5]:

```
top3 = []
top3.append(len(smurf))
top3.append(len(neptune))
top3.append(abnormal_count - len(smurf) - len(neptune))

x = ['smurf','neptune','other']
y = top3
plt.figure(figsize=(4,3))
plt.bar(x,y,color = '#9999ff')
plt.title('TOP3 Statistics of different anomalies')
plt.xlabel('Exception Name')
plt.ylabel('Count')
plt.show()
```



In [16]:

```
x = ['back','satan','teardrop','portsweep','warezclient','ipsweep','pod','nmap','multihop','land','p
y = anomalies_count
plt.figure(figsize=(20,5))
plt.bar(x,y,color = '#9999ff',width = 0.85)
plt.title('Statistics of different anomalies')
plt.xlabel('Exception Name')
plt.ylabel('Count')
plt.show()
```



不同协议与连接持续时间，发送字节数，接受字节数关系

In [6]:

```
df.pivot_table(index=1,values=[0,4,5],aggfunc='mean')
```

Out[6]:

	0	4	5
1			
icmp	0.000000	928.318351	0.000000
tcp	18.299576	6468.998132	2248.436461
udp	993.646163	93.935000	84.709689

In [22]:

```
df.pivot_table(index=1,values=[0,4,5],aggfunc='min')
```

Out[22]:

	0	4	5
1			
icmp	0	8	0
tcp	0	0	0
udp	0	1	0

In [8]:

```
df.pivot_table(index=1,values=[0,4,5],aggfunc='max')
```

Out[8]:

	0	4	5
1			
icmp	0	1480	0
tcp	42448	693375640	5155468
udp	58329	516	516

异常状况时用户登录情况

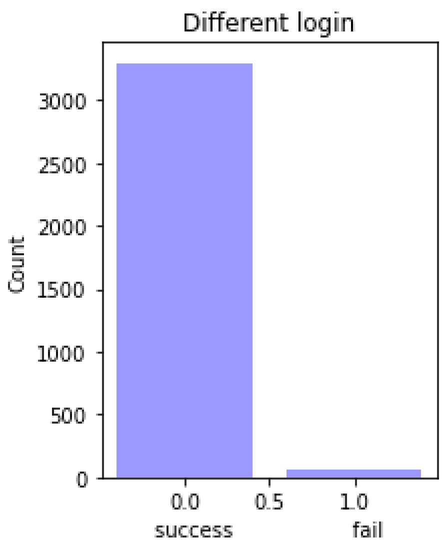
In [71]:

```
success_login = 0
fail_login = 0

df.pivot_table(index=1,values=[10,11,41],aggfunc='mean')
for i in range(0,len(df)):
    if df[41][i] != 'normal.':
        if df[11][i] == 1:
            success_login += 1
        else:
            fail_login += df[10][i]
print('网络发生异常时，登录成功次数：',success_login)
print('网络发生异常时，登录失败次数：',fail_login)
```

```
网络发生异常时，登录成功次数： 3298
网络发生异常时，登录失败次数： 57
```

In [73]:

```
# 绘制数目图，登录成功 vs 登录失败
x = [0,1]
y = [success_login,fail_login]
plt.figure(figsize=(3,4))
plt.bar(x,y,color = '#9999ff')
plt.title('Different login')
plt.xlabel('success                    fail')
plt.ylabel('Count')
plt.show()
```



异常状况时root用户访问占比

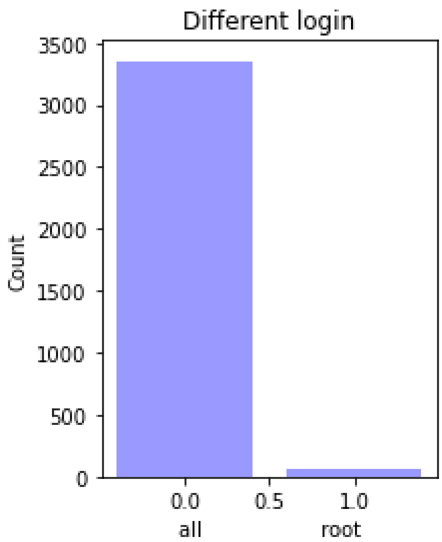
In [74]:

```
root_post = 0
df.pivot_table(index=1,values=[15,41],aggfunc='mean')

for i in range(0,len(df)):
    if df[41][i] != 'normal.':
        root_post += df[10][i]
print('异常状况时root用户访问次数为：',fail_login)
```

异常状况时root用户访问次数为： 57

In [75]:

```
x = [0,1]
y = [success_login + fail_login ,root_post]
plt.figure(figsize=(3,4))
plt.bar(x,y,color = '#9999ff')
plt.title('Different login')
plt.xlabel('all                        root')
plt.ylabel('Count')
plt.show()
```



十二、预测发送字节数是否大于200

In [33]:

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import numpy as np

# 加载数据
df = pd.read_csv('D:\KDD_CUP_99\kdd_cup\DataConvy.csv')
df
```

Out[33]:

	持续时间	协议类型	连接状态	发送字节数	接收字节数	加急包个数	访问敏感文件和目录次数	登录失败次数	登录成功次数
0	0	tcp	SF	181	5450	0	0	0	1
1	0	tcp	SF	239	486	0	0	0	1
2	0	tcp	SF	235	1337	0	0	0	1
3	0	tcp	SF	219	1337	0	0	0	1
4	0	tcp	SF	217	2032	0	0	0	1
...	...	...	...	...	...	...	...	...	...
494016	0	tcp	SF	310	1881	0	0	0	1
494017	0	tcp	SF	282	2286	0	0	0	1
494018	0	tcp	SF	203	1200	0	0	0	1
494019	0	tcp	SF	291	1200	0	0	0	1
494020	0	tcp	SF	219	1234	0	0	0	1

494021 rows × 9 columns

In [17]:

```
# 标准化
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
data = [anomalies_count]
ss.fit_transform(data)
```

Out[17]:

```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.]])
```

In [34]:

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

# 提取样本数据
target = df['发送字节数']

# 有许多特征与发送字节数无关，所以需要手动抽取关联特征
# 提取出特征：1. 持续时间 2. 协议类型 3. 接收字节数 4. 加急包个数
feature = df[['持续时间','协议类型','接收字节数','加急包个数']]
# feature.shape  # (494021, 4)  494021行 4列
# target.shape   # (494021,)

# 数据集拆分：拆分完观察样本数据中的特征是否需要特征工程。10%比例
x_train, x_test, y_train, y_test = train_test_split(feature,target,test_size=0.1,random_state=2020)

# 观察特征数据是否需要特征工程。协议类型为非数值型数据，需要特征值化，转换为数值型数据
# x_train

# 对训练集特征进行手动onehot编码
occ_one_hot = pd.get_dummies(x_train['协议类型'])
# occ_one_hot

# 将 occ_one_hot 与 x_train 进行级联。x_train 为 DataFrame，axis=0表示行，axis=1表示列
# pd.concat((x_train, occ_one_hot),axis=1)
x_train = pd.concat((x_train, occ_one_hot),axis=1).drop(labels='协议类型',axis=1)
# x_train

# 对测试集特征进行手动onehot编码
occ_one_hot_test = pd.get_dummies(x_test['协议类型'])
# occ_one_hot_test

# 对测试集级联
x_test = pd.concat((x_test, occ_one_hot_test),axis=1).drop(labels='协议类型',axis=1)
```

进行训练

In [31]:

```
import time
start_time=time.perf_counter()

# 实例化KNN，并传入训练集数据
knn = KNeighborsClassifier(n_neighbors=10, n_jobs = -1).fit(x_train, y_train)

# 查看训练结果
reslt_score = knn.score(x_test, y_test)
print('模型得分为：',reslt_score)

end_time=time.perf_counter()
print("Running time:",(end_time-start_time))  #输出程序运行时间
```

```
模型得分为： 0.7302997793656256
Running time: 99.3656662000003
```

探索模型训练最适线程数

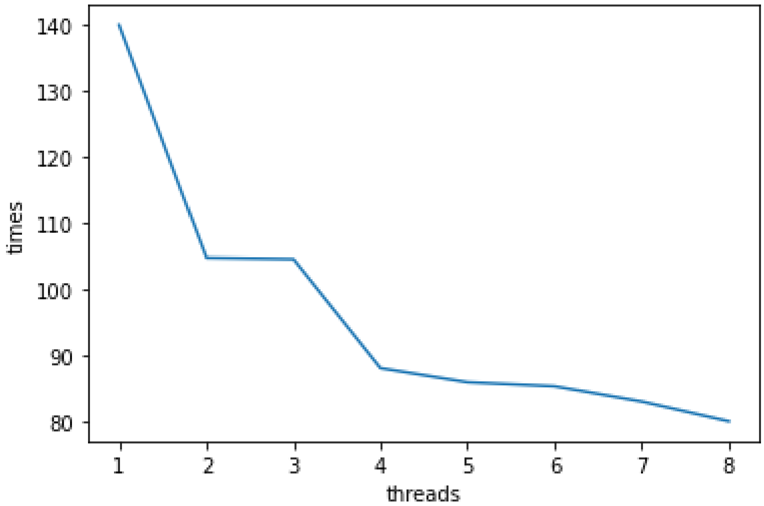
In [12]:

```
import time
scores = []
threads = []
times = []

# 用学习曲线，寻找最适线程数
for i in range(1,9):
    start_time = time.perf_counter()
    # 实例化
    knn = KNeighborsClassifier(n_neighbors=10, n_jobs = i)

    # 训练模型
    knn.fit(x_train, y_train)

    # 训练好模型后进行评分
    score = knn.score(x_test, y_test)

    end_time = time.perf_counter()

    # 拿到不同threads的时间
    scores.append(score)
    threads.append(i)
    times.append(end_time-start_time)

# 转换为np数组
scores_arr = np.array(scores)
threads_arr = np.array(threads)
times_arr = np.array(times)
```

In [13]:

```
# 绘图  参数：(自变量，因变量)
plt.plot(threads, times)
plt.xlabel('threads')
plt.ylabel('times')

# 找出最大值。scores_arr.argmax() 最大值下标
min_time = threads_arr[times_arr.argmin()]
print('最短时间的线程数为：', min_time)
```

最短时间的线程数为： 8

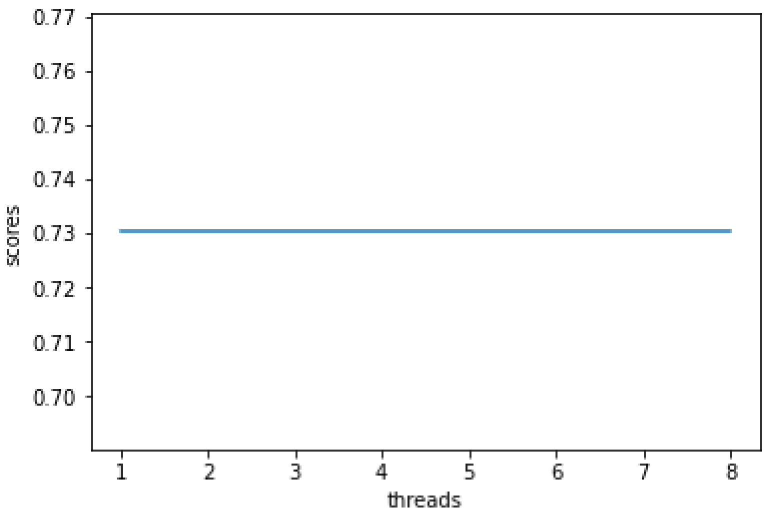


In [15]:

```
# 绘图  参数：(自变量，因变量)
plt.plot(threads, scores)
plt.xlabel('threads')
plt.ylabel('scores')
```

Out[15]:

Text(0, 0.5, 'scores')



优化KNN模型，探索最适K值

In [16]:

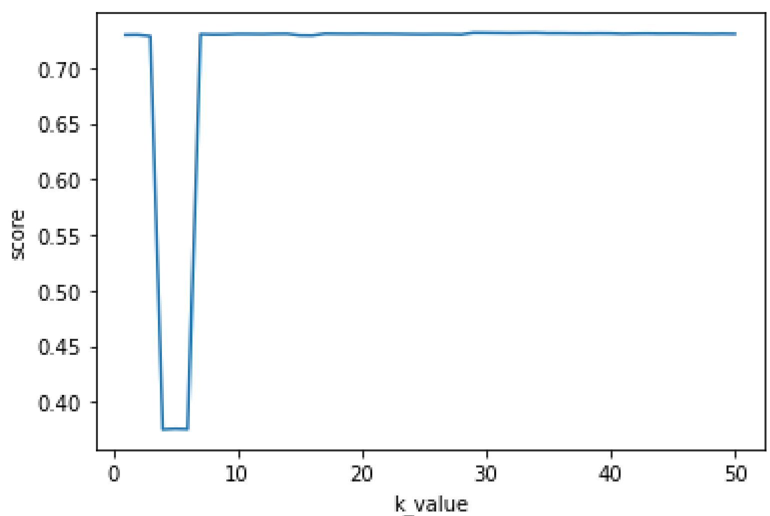
```
start_time=time.perf_counter()
scores = []
ks = []

# 用学习曲线，寻找最优K值
for i in range(1,51):
    # 实例化
    knn = KNeighborsClassifier(n_neighbors=i, n_jobs = -1)

    # 训练模型
    knn.fit(x_train, y_train)

    # 训练好模型后进行评分
    score = knn.score(x_test, y_test)

    # 拿到不同K的得分
    scores.append(score)
    ks.append(i)

# 转换为np数组
scores_arr = np.array(scores)
ks_arr = np.array(ks)

end_time=time.perf_counter()
print("Running time:",(end_time-start_time))  #输出程序运行时间
```

```
Running time: 5100.5635838
```

In [17]:

```
# 绘图  参数：(自变量，因变量)
plt.plot(ks_arr, scores_arr)
plt.xlabel('k_value')
plt.ylabel('score')

# 找出最大值。scores_arr.argmax() 最大值下标
max_k = ks_arr[scores_arr.argmax()]
print('最优的K为', max_k)
```

```
最优的K为 29
```



In [28]:

```
# 实例化
knn1 = KNeighborsClassifier(n_neighbors=29, n_jobs = -1)
# 训练模型
knn1.fit(x_train, y_train)
# 训练好模型后进行评分
score = knn1.score(x_test, y_test)
```

用训练好的模型对发送字节数进行预测

In [61]:

```
knn1.predict([[0,3222,0,0,1,0]])
```

```
D:\DATA\ProgramData\Anaconda\lib\site-packages\sklearn\base.py:450: UserWarning: X d
oes not have valid feature names, but KNeighborsClassifier was fitted with feature n
ames
  warnings.warn(
```

Out[61]:

```
array([232], dtype=int64)
```

In [60]:

```
x_train.loc[74274]
```

Out[60]:

```
持续时间          0
接收字节数     3222
加急包个数        0
icmp            0
tcp             1
udp             0
Name: 74274, dtype: int64
```

rootkit 模型重建

In [9]:

```
# 加载数据
df = pd.read_csv('D:\KDD_CUP_99\kdd_cup\AllData.csv',header=None)
df
```

Out[9]:

	0	1	2	3	4	5	6	7	8	9	...	32	33	34	35	36	37	38
0	0	tcp	http	SF	181	5450	0	0	[8]	0	...	9	1.0	0.0	0.11	0.00	0.00	0.00
1	0	tcp	http	SF	239	486	0	0	[8]	0	...	19	1.0	0.0	0.05	0.00	0.00	0.00
2	0	tcp	http	SF	235	1337	0	0	[8]	0	...	29	1.0	0.0	0.03	0.00	0.00	0.00
3	0	tcp	http	SF	219	1337	0	0	[8]	0	...	39	1.0	0.0	0.03	0.00	0.00	0.00
4	0	tcp	http	SF	217	2032	0	0	[8]	0	...	49	1.0	0.0	0.02	0.00	0.00	0.00
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
494016	0	tcp	http	SF	310	1881	0	0	[8]	0	...	255	1.0	0.0	0.01	0.05	0.00	0.01
494017	0	tcp	http	SF	282	2286	0	0	[8]	0	...	255	1.0	0.0	0.17	0.05	0.00	0.01
494018	0	tcp	http	SF	203	1200	0	0	[8]	0	...	255	1.0	0.0	0.06	0.05	0.06	0.01
494019	0	tcp	http	SF	291	1200	0	0	[8]	0	...	255	1.0	0.0	0.04	0.05	0.04	0.01
494020	0	tcp	http	SF	219	1234	0	0	[8]	0	...	255	1.0	0.0	0.17	0.05	0.00	0.01

494021 rows × 42 columns

In [112]:

```
v=[]
w=[]
y=[]
# 筛选标记为KDD99和normal且是telent的数据
for x1 in range(0,len(df)):
    if ( df.loc[x1][41] in ['rootkit.','normal.'] ) and ( df.loc[x1][2] == 'telnet' ):
        if df.loc[x1][41] == 'rootkit.':
            y.append(1)
        else:
            y.append(0)

        x1 = df.loc[x1][9:21]
        v.append(x1)

#挑选与Rookit相关的特征作为样本特征
for x1 in v :
    v1=[]
    for x2 in x1:
        v1.append(float(x2))
    w.append(v1)
```

In [113]:

```
clf = KNeighborsClassifier(n_neighbors=3)
print(cv.cross_val_score(clf, w, y, n_jobs=-1, cv=10).mean())
```

```
0.982411067193676

D:\DATA\ProgramData\Anaconda\lib\site-packages\sklearn\model_selection\_split.py:67
6: UserWarning: The least populated class in y has only 5 members, which is less tha
n n_splits=10.
  warnings.warn(
```

neptune 模型

In [11]:

```
from sklearn import model_selection as cv
v = [] #  每条数据的具体特征
w = []
y = []
# 筛选标记为KDD99和normal且是telent的数据
for x1 in range(0,len(df)):
    if ( df.loc[x1][41] in ['neptune.','normal.'] ) and ( df.loc[x1][2] == 'telnet' ):
        if df.loc[x1][41] == 'neptune.':
            y.append(1)
        else:
            y.append(0)

        x1 = df.loc[x1][9:21]
        v.append(x1)

#挑选与 neptune 相关的特征作为样本特征
for x1 in v :
    v1=[]
    for x2 in x1:
        v1.append(float(x2))
    w.append(v1)
```

交叉验证

In [12]:

```
clf = KNeighborsClassifier(n_neighbors=3)
print(cv.cross_val_score(clf, w, y, n_jobs=-1, cv=10).mean())
```

```
0.9259001161440186
```

获取交叉验证超参数

In [13]:

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import model_selection as cv

# 提取样本数据
target = df[41]

# 有许多特征与发送字节数无关，所以需要手动抽取关联特征
feature = df[[9,10,11,12,13,14,15,16,17,18,19,20,21]]

# 数据集拆分：拆分完观察样本数据中的特征是否需要特征工程。10%比例
x_train, x_test, y_train, y_test = train_test_split(feature,target,test_size=0.1,random_state=2020)

# 观察特征数据是否需要特征工程。协议类型为非数值型数据，需要特征值化，转换为数值型数据
# x_train
```

In [24]:

```
scores = []
ks = []
for k in range(7,8):
    knn = KNeighborsClassifier(n_neighbors=k)
    score = cv.cross_val_score(knn, x_train, y_train, n_jobs=-1).mean()
    scores.append(score)
    ks.append(k)
```

```
D:\DATA\ProgramData\Anaconda\lib\site-packages\sklearn\model_selection\_split.py:67
6: UserWarning: The least populated class in y has only 2 members, which is less tha
n n_splits=5.
  warnings.warn(
```

In [22]:

```
import matplotlib.pyplot as plt
plt.plot(ks,scores)
```

Out[22]:

```
[<matplotlib.lines.Line2D at 0x22c20fb1e80>]
```

In [30]:

```
# 训练模型
knn.fit(x_train, y_train)
```

Out[30]:

```
KNeighborsClassifier(n_neighbors=7)
```

In [31]:

```
knn.predict([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
```

Out[31]:

```
array(['smurf.'], dtype=object)
```

In [ ]: