# SVC Assignment

# Part 1 Dos vs Non Dos

In [*]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import time
from sklearn.linear_model import LogisticRegression, Lasso, Ridge
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import scale
from sklearn.model_selection import train_test_split
from sklearn import svm, datasets
from sklearn import metrics
from sklearn.svm import SVC
from sklearn.preprocessing import MinMaxScaler
```

In [74]:
```
df = pd.read_csv('kddcup99_csv.csv')
```

In [75]:
```
df
```

| | duration | protocol_type | service | flag | src_bytes | dst_bytes | land | wrong_fragment | urgent |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | tcp | http | SF | 181 | 5450 | 0 | 0 | 0 |
| **1** | 0 | tcp | http | SF | 239 | 486 | 0 | 0 | 0 |
| **2** | 0 | tcp | http | SF | 235 | 1337 | 0 | 0 | 0 |
| **3** | 0 | tcp | http | SF | 219 | 1337 | 0 | 0 | 0 |
| **4** | 0 | tcp | http | SF | 217 | 2032 | 0 | 0 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **494015** | 0 | tcp | http | SF | 310 | 1881 | 0 | 0 | 0 |
| **494016** | 0 | tcp | http | SF | 282 | 2286 | 0 | 0 | 0 |
| **494017** | 0 | tcp | http | SF | 203 | 1200 | 0 | 0 | 0 |
| **494018** | 0 | tcp | http | SF | 291 | 1200 | 0 | 0 | 0 |
| **494019** | 0 | tcp | http | SF | 219 | 1234 | 0 | 0 | 0 |

In [4]:
```
y = df.iloc[0:,41].values
#X = df.iloc[0:,4:40].values
#probe = ['ipsweep.','satan.','nmap.','portsweep.']
y = np.where(((df['label']== 'back') | (df['label'] == 'land') | (df['label
r = np.unique(y)
r
```

Out[4]:
```
array([0, 1])
```

In [5]:
```
dos = ['back.', 'land.', 'teardrop.', 'pod.', 'smurf.']
non_dos = ['back.', 'buffer_overflow.',',ftp_write.','guess_passwd.','imap.
```

In [6]:
```
X=df
X.drop('label', axis=1, inplace=True)
column_trans = make_column_transformer(
    (OneHotEncoder(), ['protocol_type', 'service', 'flag']),
    remainder='passthrough')
X1=column_trans.fit_transform(X)
```

In [7]:
```
X.shape
```

Out[7]:
```
(494020, 41)
```

In [8]:
```
y.shape
```

Out[8]:
```
(494020,)
```

# Part 2: Running SVM model four times with different kernels

In [9]:
```


Xs = MinMaxScaler().fit_transform(X1)
from imblearn.under_sampling import RandomUnderSampler
ros = RandomUnderSampler(random_state=0)
X_underresampled, y_underresampled = ros.fit_resample (Xs,y)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_underresampled,y_unde
X_train
```

Out[9]:
```
array([[0., 1., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 1., 0., 0.],
       ...,
       [0., 0., 1., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.]])
```

In [ ]:
```
svc = svm.SVC(probability=False,  kernel="linear", C=2.8, gamma=.0073,verbo
svc.fit(X_train, y_train)
svc.score(X_test,y_test)
```

```
[LibSVM]
```

In [12]:
```
svc.score(X_train,y_train)
```

Out[12]:
```
0.9994515505883643
```

In [13]:
```
svc = svm.SVC(probability=False,  kernel="rbf", C=2.8, gamma=.0073,verbose=
svc.fit(X_train, y_train)
svc.score(X_test,y_test)
```

```
[LibSVM]
```

Out[13]: 0.9958074344911639

In [14]:
```
svc.score(X_train,y_train)
```

Out[14]: 0.9965691442361013

In [15]:
```
svc = svm.SVC(probability=False,  kernel="sigmoid", C=2.8, gamma=.0073,verb
svc.fit(X_train, y_train)
svc.score(X_test,y_test)
```

```
[LibSVM]
```

Out[15]: 0.9957586837294333

In [16]:
```
svc.score(X_train,y_train)
```

Out[16]: 0.9964533604714227

In [17]:
```
svc = svm.SVC(probability=False,  kernel="poly", C=2.8, gamma=.0073,verbose
svc.fit(X_train, y_train)
svc.score(X_test,y_test)
```

```
[LibSVM]
```

Out[17]: 0.9926630103595369

In [18]:
```
svc.score(X_train,y_train)
```

Out[18]: 0.9931565701192573

# Compare the results for each of the kernels. Discuss the pros and cons of using each of the kernels that you've chosen.

Linear: Score test: 0.9992931139549055 Linear: Score train: 0.9994515505883643

RBF: Score test: 0.9994515505883643 Score train: 0.9965691442361013

Sigmoid: Score test: 0.9957586837294333 Score train: 0.9964533604714227

Poly: Score test: 0.9926630103595369 train test: 0.9931565701192573

The cons of using all these kernels is that they all took a very long time to load and show the outputs. Poly took the longest out of all but had the lowest numbers. Linear had the highest numbers compared to all of them. Sigmoids score test and train test were close together. RBF had numbers very apart from each other with the score test and score train.

# Part 3: Pick two features

In [77]:
```
X=df.iloc[0:, [5, 23]].values
X
```

Out[77]:
```
array([[5450,    8],
       [ 486,    8],
       [1337,    8],
       ...,
       [1200,   18],
       [1200,   12],
       [1234,   35]])
```

In [78]:
```
Xs = MinMaxScaler().fit_transform(X)
from imblearn.under_sampling import RandomUnderSampler
ros = RandomUnderSampler(random_state=0)
X_underresampled, y_underresampled = ros.fit_resample (Xs,y)
```

In [79]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_underresampled,y_unde
X_train
```

Out[79]:
```
array([[0.00000000e+00, 1.00000000e+00],
       [1.72826211e-04, 1.76125245e-02],
       [4.74641681e-04, 1.95694716e-03],
       ...,
       [0.00000000e+00, 1.00000000e+00],
       [2.71168398e-04, 5.87084149e-03],
       [5.12077662e-05, 5.28375734e-02]])
```

In [*]:
```
svc = svm.SVC(probability=False,  kernel="rbf", C=2.8, gamma=.0073,verbose=
svc.fit(X_train, y_train)
svc.score(X_test,y_test)
```

```
[LibSVM]
```

In [*]:
```
from matplotlib.colors import ListedColormap


def plot_decision_regions(X, y, classifier, resolution=0.02):

    # setup marker generator and color map
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])

    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.3, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())

    # plot class examples
    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0],
                    y=X[y == cl, 1],
                    alpha=0.8,
                    c=colors[idx],
                    marker=markers[idx],
                    label=cl,
                    edgecolor='black')
```

In [*]:
```
plot_decision_regions(X_train, y_train, classifier=svc)
plt.xlabel('sepal length [cm]')
plt.ylabel('petal length [cm]')
plt.legend(loc='upper left')


# plt.savefig('images/02_08.png', dpi=300)
plt.show()
```

In [*]:
```
svc = svm.SVC(probability=False,  kernel="linear", C=2.8, gamma=.0073,verbo
svc.fit(X_train, y_train)
svc.score(X_test,y_test)
```

In [*]:
```python
from matplotlib.colors import ListedColormap


def plot_decision_regions(X, y, classifier, resolution=0.02):

    # setup marker generator and color map
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])

    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.3, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())

    # plot class examples
    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0],
                    y=X[y == cl, 1],
                    alpha=0.8,
                    c=colors[idx],
                    marker=markers[idx],
                    label=cl,
                    edgecolor='black')
```

In [*]:
```python
plot_decision_regions(X_test,y_test, classifier=svc)
plt.xlabel('sepal length [cm]')
plt.ylabel('petal length [cm]')
plt.legend(loc='upper left')


# plt.savefig('images/02_08.png', dpi=300)
plt.show()
```

# Discuss your observations.

I saw that the linear feature has a higher percentage than the other feature that is RBF. The decision boundaries however looked very similar between these two.