In [0]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6
qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%
b&response_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2
www.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly
ttps%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/drive
```

In [0]:

```
import pandas as pd
df = pd.read_csv('/content/drive/My Drive/Train_data.csv')
print(df)
```

```
        duration protocol_type  ...  dst_host_srv_rerror_rate  xAttack
0              0          icmp  ...                      0.00   normal
1              0           udp  ...                      0.00   normal
2              0          icmp  ...                      0.00      dos
3              0          icmp  ...                      0.01   normal
4              0          icmp  ...                      0.00   normal
...          ...           ...  ...                       ...      ...
125968         0          icmp  ...                      0.00      dos
125969         8           udp  ...                      0.00   normal
125970         0          icmp  ...                      0.00   normal
125971         0          icmp  ...                      0.00      dos
125972         0          icmp  ...                      0.00   normal

[125973 rows x 42 columns]
```

In [0]:

```
X = df.iloc[:, :-1].values
print(X)
```

```
[[0 'icmp' 20 ... 0.0 0.05 0.0]
 [0 'udp' 45 ... 0.0 0.0 0.0]
 [0 'icmp' 50 ... 1.0 0.0 0.0]
 ...
 [0 'icmp' 55 ... 0.0 0.01 0.0]
 [0 'icmp' 31 ... 1.0 0.0 0.0]
 [0 'icmp' 20 ... 0.0 0.0 0.0]]
```

In [0]:

```
y = df.iloc[:, -1].values
print(y)
```

```
['normal' 'normal' 'dos' ... 'normal' 'dos' 'normal']
```

In [0]:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X = LabelEncoder()
X[:,0] = labelencoder_X.fit_transform(X[:,0])
X[:,1] = labelencoder_X.fit_transform(X[:,1])
onehotencoder = OneHotEncoder(categorical_features=[0,1])
X = onehotencoder.fit_transform(X).toarray()
print(X)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/preprocessing/_encoders.py:415: FutureWarning: The
handling of integer data will change in version 0.22. Currently, the categories are determined
based on the range [0, max(values)], while in the future they will be determined based on the
unique values.
If you want the future behaviour and silence this warning, you can specify "categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, t
hen you can now use the OneHotEncoder directly.
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/preprocessing/_encoders.py:451: DeprecationWarning:
The 'categorical_features' keyword is deprecated in version 0.20 and will be removed in 0.22. You
can use the ColumnTransformer instead.
  "use the ColumnTransformer instead.", DeprecationWarning)
```

```
[[1.   0.   0.   ... 0.   0.05 0.  ]
 [1.   0.   0.   ... 0.   0.   0.  ]
 [1.   0.   0.   ... 1.   0.   0.  ]
 ...
 [1.   0.   0.   ... 0.   0.01 0.  ]
 [1.   0.   0.   ... 1.   0.   0.  ]
 [1.   0.   0.   ... 0.   0.   0.  ]]
```

In [0]:

```
labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(y)
print(y)
```

```
[1 1 0 ... 1 0 1]
```

In [0]:

```
X.shape
```

Out[0]:

```
(125973, 3023)
```

In [0]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA, IncrementalPCA
X_st = StandardScaler().fit_transform(X)
```

In [0]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_st, y, test_size=0.2, random_state=0)
```

In [0]:

```
from sklearn.neighbors import KNeighborsClassifier
#create new a knn model
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

In [33]:

```
y_pred = knn.predict(X_test)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-33-08dc72747141> in <module>()
----> 1 y_pred = knn.predict(X_test)

/usr/local/lib/python3.6/dist-packages/sklearn/neighbors/classification.py in predict(self, X)
    147         X = check_array(X, accept_sparse='csr')
    148
--> 149         neigh_dist, neigh_ind = self.kneighbors(X)
```

```
    150         classes_ = self.classes_
    151         _y = self._y

/usr/local/lib/python3.6/dist-packages/sklearn/neighbors/base.py in kneighbors(self, X,
n_neighbors, return_distance)
    452                 delayed_query(
    453                     self._tree, X[s], n_neighbors, return_distance)
--> 454                 for s in gen_even_slices(X.shape[0], n_jobs)
    455             )
    456         else:

/usr/local/lib/python3.6/dist-packages/joblib/parallel.py in __call__(self, iterable)
   1001             # remaining jobs.
   1002             self._iterating = False
-> 1003             if self.dispatch_one_batch(iterator):
   1004                 self._iterating = self._original_iterator is not None
   1005

/usr/local/lib/python3.6/dist-packages/joblib/parallel.py in dispatch_one_batch(self, iterator)
    832                 return False
    833             else:
--> 834                 self._dispatch(tasks)
    835                 return True
    836

/usr/local/lib/python3.6/dist-packages/joblib/parallel.py in _dispatch(self, batch)
    751         with self._lock:
    752             job_idx = len(self._jobs)
--> 753             job = self._backend.apply_async(batch, callback=cb)
    754             # A job can complete so quickly than its callback is
    755             # called before we get here, causing self._jobs to

/usr/local/lib/python3.6/dist-packages/joblib/_parallel_backends.py in apply_async(self, func,
callback)
    199     def apply_async(self, func, callback=None):
    200         """Schedule a func to be run"""
--> 201         result = ImmediateResult(func)
    202         if callback:
    203             callback(result)

/usr/local/lib/python3.6/dist-packages/joblib/_parallel_backends.py in __init__(self, batch)
    580         # Don't delay the application, to avoid keeping the input
    581         # arguments in memory
--> 582         self.results = batch()
    583
    584     def get(self):

/usr/local/lib/python3.6/dist-packages/joblib/parallel.py in __call__(self)
    254         with parallel_backend(self._backend, n_jobs=self._n_jobs):
    255             return [func(*args, **kwargs)
--> 256                     for func, args, kwargs in self.items]
    257
    258     def __len__(self):

/usr/local/lib/python3.6/dist-packages/joblib/parallel.py in <listcomp>(.0)
    254         with parallel_backend(self._backend, n_jobs=self._n_jobs):
    255             return [func(*args, **kwargs)
--> 256                     for func, args, kwargs in self.items]
    257
    258     def __len__(self):

/usr/local/lib/python3.6/dist-packages/sklearn/neighbors/base.py in _tree_query_parallel_helper(tr
ee, data, n_neighbors, return_distance)
    289     under PyPy.
    290     """
--> 291     return tree.query(data, n_neighbors, return_distance)
    292
    293

KeyboardInterrupt:
```

In [34]:

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
print(classification_report(y_test, y_pred))
```

```
[[ 9146    19     1     0     0]
 [    6 13390    41    20     0]
 [    0    39  2303     0     0]
 [    2    31     0   182     0]
 [    0    15     0     0     0]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      9166
           1       0.99      1.00      0.99     13457
           2       0.98      0.98      0.98      2342
           3       0.90      0.85      0.87       215
           4       0.00      0.00      0.00        15

    accuracy                           0.99     25195
   macro avg       0.77      0.76      0.77     25195
weighted avg       0.99      0.99      0.99     25195
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/metrics/classification.py:1437:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples.
  'precision', 'predicted', average, warn_for)
```

In [35]:

```
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9930938678309188
```

In [0]:

```
from imblearn.metrics import sensitivity_score, specificity_score, geometric_mean_score
```

In [28]:

```
sensitivity_score(y_test, y_pred, average='macro')
```

Out[28]:

```
0.8637591298763134
```

In [37]:

```
specificity_score(y_test, y_pred, average='macro')
```

Out[37]:

```
0.9976004637070919
```

In [38]:

```
geometric_mean_score(y_test, y_pred, average='macro')
```

Out[38]:

```
0.8733299138641619
```

In [0]:

```

```