```
import pandas as pd


# ADMIN 1: Load Input dataset


dataset=pd.read_csv('cybermg.csv')

print("==============================\nimbalanced Data with 10000
records\n==============================\n",dataset)


# ADMIN 2: Convert categorical to numerical data using Label Encoding technique


from sklearn import preprocessing

label_encoder = preprocessing.LabelEncoder()

cols = ['protocol_type','service','flag','class']

dataset[cols] = dataset[cols].apply(label_encoder.fit_transform)

print("\n\nback-->0\nbuffer_overflow-->1\nftp_write-->2\nguess_passwd-->3\nimap-->4\nipsweep--
>5\nland-->6\nloadmodule-->7\nmultihop-->8\nneptune-->9\nnmap-->10\nnormal-->11\nperl--
>12\nphf-->13\npod-->14\nportsweep-->15\nrootkit-->16\nsatan-->17\nsmurf-->18\nspy--
>19\nteardrop-->20\nwarezclient-->21\nwarezmaster-->22\n")

print("\n================================================================================\nDatas
et after categorical to numerical
conversion\n================================================================================\n",
dataset)


X = dataset.iloc[:, :-1]   # data without class label

Y = dataset.iloc[:, 41].values     # class label values only


# ADMIN 3: Transform the dataset using Synthetic Minority Oversampling Technique (SMOTE)


from imblearn.over_sampling import SMOTE

oversample = SMOTE(k_neighbors=1)
```

```
X1, Y1 = oversample.fit_resample(X, Y)


print("\n=============================================================================
==\nDataset without class
label\n=============================================================================
=\n",X)

print("\n=============================================================================
======\nDataset class label values
only\n=============================================================================
====\n",Y)



# ADMIN 4: Split dataset into training and testing


from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.1, shuffle=True)

# What does Test_size 0.1 mean? We have passed test_size as 0.1 which means 10% of data will

# be in the test part and rest will be in train part.


print("\n=============================================================================
============\nTraining Dataset without class
label\n=============================================================================
===========\n",X_train)

print("\n=============================================================================
==============\nTraining Dataset class label values
only\n=============================================================================
=============\n",Y_train)

print("\n=============================================================================
============\nTesting Dataset without class
label\n=============================================================================
===========\n",X_test)

print("\n=============================================================================
==============\nActual class label values for Testing
Dataset\n=============================================================================
================\n",Y_test)
```

```
# ADMIN 5: Principal Component Analysis (PCA) for feature selection and dimensionlaity reduction

from sklearn.decomposition import PCA

pca = PCA(n_components=1)

X_train = pca.fit_transform(X_train)

X_test = pca.transform(X_test)


# ADMIN 6: Predict Intrusion with Attack Type using Principal Component Analysis with Random Forest

from sklearn.ensemble import RandomForestClassifier

PCARF = RandomForestClassifier()

PCARF.fit(X_train, Y_train)

Y_pred = PCARF.predict(X_test)

print("\n============================================================================
========================\nPCA-RF Predicted class label values for Testing
Dataset\n============================================================================
========================\n",Y_pred)

from scipy import spatial

PCARFAccuracy = (1 - spatial.distance.cosine(Y_test, Y_pred))*100;

print("PCA-RF Accuracy: ",PCARFAccuracy," %\n")


# ADMIN 7: Predict Intrusion with Attack Type using Support Vector Machine

from sklearn.svm import SVC

svm=SVC();

svm.fit(X_train, Y_train)
```

```
Y_pred = svm.predict(X_test);

#
print("\n============================================================================
=======================\nSVM Predicted class label values for Testing
Dataset\n============================================================================
=========================\n",Y_pred)

from scipy import spatial

svmAccuracy = (1 - spatial.distance.cosine(Y_test, Y_pred))*100;

print("SVM Accuracy: ",svmAccuracy," %\n")


# ADMIN 8: Accuracy Comparison


import matplotlib.pyplot as plt

x1 = ['SVM','PCA-RF']

y1 = [svmAccuracy,PCARFAccuracy]

plt.plot(x1, y1, label = "Accuracy")

plt.xlabel('Algorithm')

plt.ylabel('Accuracy (in %)')

plt.title('Accuracy Comparison')

plt.legend()

plt.show()
```