```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

```
dataset = pd.read_csv('/content/drive/My Drive/kddcup99_csv.csv')
```

```
dataset.head(5)
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragmen
0	0	tcp	http	SF	181	5450	0	
1	0	tcp	http	SF	239	486	0	
2	0	tcp	http	SF	235	1337	0	
3	0	tcp	http	SF	219	1337	0	
4	0	tcp	http	SF	217	2032	0	

```
print(dataset.var()['src_bytes'])
print(dataset.var()['dst_bytes'])
print(dataset.var()['land'])
```

```
976576992036.6654
1091643891.0952706
4.453071700180602e-05
```

```
print(dataset.var()['wrong_fragment'])
print(dataset.var()['urgent'])
print(dataset.var()['hot'])
```

```
0.018172491590816044
3.03630038804738e-05
0.6116856844184144
```

```
print(dataset.var()['num_failed_logins'])
print(dataset.var()['logged_in'])
print(dataset.var()['lnum_compromised'])
print(dataset.var()['lroot_shell'])
print(dataset.var()['lsu_attempted'])
```

```
0.00024085837553570224
0.12626868283082365
3.2339838746190672
0.0001113193556638224
6.072508173886537e-05
```

```
print(dataset.var()['lnum_root'])
print(dataset.var()['lnum_file_creations'])
print(dataset.var()['lnum_shells'])
print(dataset.var()['lnum_access_files'])
#print(dataset.var()['lnum_outbound_cmds'])
#print(dataset.var()['is_host_login'])
print(dataset.var()['is_guest_login'])
```

```
4.051043258019034
0.009296040477405787
0.000121440870502456
0.001330916397839991
0.001384663728080717
```

```
print(dataset.var()['count'])
print(dataset.var()['srv_count'])
print(dataset.var()['serror_rate'])
print(dataset.var()['srv_serror_rate'])
print(dataset.var()['rerror_rate'])
print(dataset.var()['srv_rerror_rate'])
print(dataset.var()['same_srv_rate'])
print(dataset.var()['diff_srv_rate'])
print(dataset.var()['srv_diff_host_rate'])
print(dataset.var()['dst_host_count'])
print(dataset.var()['dst_host_srv_count'])
print(dataset.var()['dst_host_same_srv_rate'])
print(dataset.var()['dst_host_diff_srv_rate'])
```

```
45431.6891034859
60674.89003505529
0.1449456310521699
0.14517386836860224
0.0536495355281265
0.05389232330783698
0.15069129958002264
0.006757755850526054
0.020276896258574477
4191.863370949552
11244.525010698932
0.1687402028545784
0.011937575833318366
```

```
print(dataset.var()['dst_host_same_src_port_rate'])
print(dataset.var()['dst_host_srv_diff_host_rate'])
print(dataset.var()['dst_host_serror_rate'])
print(dataset.var()['dst_host_srv_serror_rate'])
```

```
print(dataset.var()['dst_host_rerror_rate'])
print(dataset.var()['dst_host_srv_rerror_rate'])
```

```
0.2316583307814032
0.0017751826094751953
0.14485133584224127
0.1450998566016115
0.053171621563146344
0.052964669234579015
```

```
dataset['lsu_attempted'].value_counts()
dataset.drop('lsu_attempted', axis=1, inplace=True)
dataset['urgent'].value_counts()
dataset.drop('urgent', axis=1, inplace=True)
dataset['lnum_outbound_cmds'].value_counts()
dataset.drop('lnum_outbound_cmds', axis=1, inplace=True)
dataset['is_host_login'].value_counts()
dataset.drop('is_host_login', axis=1, inplace=True)
dataset['wrong_fragment'].value_counts()
dataset.drop('wrong_fragment', axis=1, inplace=True)
dataset['hot'].value_counts()
dataset.drop('hot', axis=1, inplace=True)
dataset['num_failed_logins'].value_counts()
dataset.drop('num_failed_logins', axis=1, inplace=True)
dataset['logged_in'].value_counts()
dataset.drop('logged_in', axis=1, inplace=True)
dataset['lroot_shell'].value_counts()
dataset.drop('lroot_shell', axis=1, inplace=True)
dataset['lnum_file_creations'].value_counts()
dataset.drop('lnum_file_creations', axis=1, inplace=True)
dataset['lnum_shells'].value_counts()
dataset.drop('lnum_shells', axis=1, inplace=True)
dataset['lnum_access_files'].value_counts()
dataset.drop('lnum_access_files', axis=1, inplace=True)
dataset['is_guest_login'].value_counts()
dataset.drop('is_guest_login', axis=1, inplace=True)
dataset['serror_rate'].value_counts()
dataset.drop('serror_rate', axis=1, inplace=True)
dataset['srv_serror_rate'].value_counts()
dataset.drop('srv_serror_rate', axis=1, inplace=True)
```

```
dataset['rerror_rate'].value_counts()
dataset.drop('rerror_rate', axis=1, inplace=True)
dataset['srv_rerror_rate'].value_counts()
dataset.drop('srv_rerror_rate', axis=1, inplace=True)
dataset['same_srv_rate'].value_counts()
dataset.drop('same_srv_rate', axis=1, inplace=True)
dataset['diff_srv_rate'].value_counts()
dataset.drop('diff_srv_rate', axis=1, inplace=True)
dataset['srv_diff_host_rate'].value_counts()
dataset.drop('srv_diff_host_rate', axis=1, inplace=True)
dataset['dst_host_same_srv_rate'].value_counts()
dataset.drop('dst_host_same_srv_rate', axis=1, inplace=True)
```

```
dataset.drop('dst_host_same_srv_rate', axis=1, inplace=True)
dataset['dst_host_diff_srv_rate'].value_counts()
dataset.drop('dst_host_diff_srv_rate', axis=1, inplace=True)
dataset['dst_host_same_src_port_rate'].value_counts()
dataset.drop('dst_host_same_src_port_rate', axis=1, inplace=True)
dataset['dst_host_srv_diff_host_rate'].value_counts()
dataset.drop('dst_host_srv_diff_host_rate', axis=1, inplace=True)
dataset['dst_host_serror_rate'].value_counts()
dataset.drop('dst_host_serror_rate', axis=1, inplace=True)
dataset['dst_host_srv_serror_rate'].value_counts()
dataset.drop('dst_host_srv_serror_rate', axis=1, inplace=True)
dataset['dst_host_rerror_rate'].value_counts()
dataset.drop('dst_host_rerror_rate', axis=1, inplace=True)
dataset['dst_host_srv_rerror_rate'].value_counts()
dataset.drop('dst_host_srv_rerror_rate', axis=1, inplace=True)
```

```
dataset.head()
#dataset.describe()
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	lnum_compromi
0	0	tcp	http	SF	181	5450	0	
1	0	tcp	http	SF	239	486	0	
2	0	tcp	http	SF	235	1337	0	
3	0	tcp	http	SF	219	1337	0	
4	0	tcp	http	SF	217	2032	0	

```
dataset['label'] = dataset['label'].replace(['back', 'buffer_overflow', 'ftp_write', 'gues
```

```
dataset.describe()
```

	duration	src_bytes	dst_bytes	land	lnum_compromised	
count	494020.000000	4.940200e+05	4.940200e+05	494020.000000	494020.000000	49
mean	47.979400	3.025616e+03	8.685308e+02	0.000045	0.010212	
std	707.747185	9.882191e+05	3.304003e+04	0.006673	1.798328	
min	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	
25%	0.000000	4.500000e+01	0.000000e+00	0.000000	0.000000	
50%	0.000000	5.200000e+02	0.000000e+00	0.000000	0.000000	
75%	0.000000	1.032000e+03	0.000000e+00	0.000000	0.000000	
max	58329.000000	6.933756e+08	5.155468e+06	1.000000	884.000000	

```
x = dataset.iloc[:, :-1].values
#x
y = dataset.iloc[:, 13].values
#y
```

```
x
```

```
array([[0, 'tcp', 'http', ..., 8, 9, 9],
       [0, 'tcp', 'http', ..., 8, 19, 19],
       [0, 'tcp', 'http', ..., 8, 29, 29],
       ...,
       [0, 'tcp', 'http', ..., 18, 16, 255],
       [0, 'tcp', 'http', ..., 12, 26, 255],
       [0, 'tcp', 'http', ..., 35, 6, 255]], dtype=object)
```

```
x.shape
```

```
(494020, 13)
```

```
y.shape
```

```
(494020,)
```

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer
labelencoder_x_1 = LabelEncoder()
labelencoder_x_2 = LabelEncoder()
labelencoder_x_3 = LabelEncoder()
x[:, 1] = labelencoder_x_1.fit_transform(x[:, 1])
x[:, 2] = labelencoder_x_2.fit_transform(x[:, 2])
x[:, 3] = labelencoder_x_3.fit_transform(x[:, 3])
```

```
x
```

```
array([[0, 1, 22, ..., 8, 9, 9],
       [0, 1, 22, ..., 8, 19, 19],
       [0, 1, 22, ..., 8, 29, 29],
       ...,
       [0, 1, 22, ..., 18, 16, 255],
       [0, 1, 22, ..., 12, 26, 255],
       [0, 1, 22, ..., 35, 6, 255]], dtype=object)
```

```
 from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state =
```

```
pipeline_lr=Pipeline([('scalar1',StandardScaler()),
                      ('pca1',PCA(n_components=2)),
                      ('lr_classifier',LogisticRegression(random_state=0))])
```

```
pipeline_dt=Pipeline([('scalar2',StandardScaler()),
                      ('pca2',PCA(n_components=2)),
```

```
                      ('dt_classifier',DecisionTreeClassifier())])


pipeline_randomforest=Pipeline([('scalar3',StandardScaler()),
                      ('pca3',PCA(n_components=2)),
                      ('rf_classifier',RandomForestClassifier())])


## LEts make the list of pipelines
pipelines = [pipeline_lr, pipeline_dt, pipeline_randomforest]


best_accuracy=0.0
best_classifier=0
best_pipeline=""


# Dictionary of pipelines and classifier types for ease of reference
pipe_dict = {0: 'Logistic Regression', 1: 'Decision Tree', 2: 'RandomForest'}

# Fit the pipelines
for pipe in pipelines:
  pipe.fit(x_train, y_train)


for i,model in enumerate(pipelines):
    print("{} Test Accuracy: {}".format(pipe_dict[i],model.score(x_test,y_test)))
```

```
Logistic Regression Test Accuracy: 0.9417499966263174
Decision Tree Test Accuracy: 0.9958166336045774
RandomForest Test Accuracy: 0.9965183595805838
```

```
for i,model in enumerate(pipelines):
    if model.score(x_test,y_test)>best_accuracy:
        best_accuracy=model.score(x_test,y_test)
        best_pipeline=model
        best_classifier=i
print('Classifier with best accuracy:{}'.format(pipe_dict[best_classifier]))
```

```
Classifier with best accuracy:RandomForest
```

Pipelines Perform Hyperparameter Tuning Using Grid SearchCV

```
from sklearn.model_selection import GridSearchCV


# Create a pipeline
pipe = Pipeline([("classifier", RandomForestClassifier())])
# Create dictionary with candidate learning algorithms and their hyperparameters
grid_param = [
                {"classifier": [LogisticRegression()],
                 "classifier__penalty": ['l2','l1'],
                 "classifier__C": np.logspace(0, 4, 6)
                 },
                {"classifier": [LogisticRegression()],
```

```
                  "classifier__penalty": ['l2'],
                  "classifier__C": np.logspace(0, 4, 6),
                  "classifier__solver":['newton-cg','saga','sag','liblinear'] ##This solver
                  },
                 {"classifier": [RandomForestClassifier()],
                  "classifier__n_estimators": [10, 15],
                  "classifier__max_depth":[5,8,15,25,30,None],
                  "classifier__min_samples_leaf":[1,2,5,10,15,30],
                  "classifier__max_leaf_nodes": [2, 5,10]}]
# create a gridsearch of the pipeline, the fit the best model
gridsearch = GridSearchCV(pipe, grid_param, cv=3, verbose=0,n_jobs=-1) # Fit grid search
best_model = gridsearch.fit(x_train,y_train)


print(best_model.best_estimator_)
print("The mean accuracy of the model is:",best_model.score(x_test,y_test)*100)
```

```
Pipeline(memory=None,
         steps=[('classifier',
                 RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                        class_weight=None, criterion='gini',
                                        max_depth=25, max_features='auto',
                                        max_leaf_nodes=10, max_samples=None,
                                        min_impurity_decrease=0.0,
                                        min_impurity_split=None,
                                        min_samples_leaf=30,
                                        min_samples_split=2,
                                        min_weight_fraction_leaf=0.0,
                                        n_estimators=10, n_jobs=None,
                                        oob_score=False, random_state=None,
                                        verbose=0, warm_start=False))],
         verbose=False)
The mean accuracy of the model is: 99.54455285211125
```