```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from pandas.api.types import is_numeric_dtype
import warnings
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.tree  import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, VotingClassifier, GradientBoostingClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.naive_bayes import BernoulliNB
from lightgbm import LGBMClassifier
from sklearn.feature_selection import RFE
import itertools
from xgboost import XGBClassifier
from tabulate import tabulate
```

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
train=pd.read_csv('/content/drive/MyDrive/CS/Train_data.csv')
```

```
test=pd.read_csv('/content/drive/MyDrive/CS/Test_data.csv')
```

```
train.head()
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_sam
0	0	tcp	ftp_data	SF	491	0	0	0	0	0	...	25	
1	0	udp	other	SF	146	0	0	0	0	0	...	1	
2	0	tcp	private	S0	0	0	0	0	0	0	...	26	
3	0	tcp	http	SF	232	8153	0	0	0	0	...	255	
4	0	tcp	http	SF	199	420	0	0	0	0	...	255	

5 rows × 42 columns

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25192 entries, 0 to 25191
Data columns (total 42 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   duration                     25192 non-null  int64
 1   protocol_type                25192 non-null  object
 2   service                      25192 non-null  object
 3   flag                         25192 non-null  object
 4   src_bytes                    25192 non-null  int64
 5   dst_bytes                    25192 non-null  int64
 6   land                         25192 non-null  int64
 7   wrong_fragment               25192 non-null  int64
 8   urgent                       25192 non-null  int64
 9   hot                          25192 non-null  int64
 10  num_failed_logins            25192 non-null  int64
 11  logged_in                    25192 non-null  int64
 12  num_compromised              25192 non-null  int64
 13  root_shell                   25192 non-null  int64
 14  su_attempted                 25192 non-null  int64
 15  num_root                     25192 non-null  int64
 16  num_file_creations           25192 non-null  int64
 17  num_shells                   25192 non-null  int64
```

```
 18  num_access_files             25192 non-null  int64
 19  num_outbound_cmds            25192 non-null  int64
 20  is_host_login                25192 non-null  int64
 21  is_guest_login               25192 non-null  int64
 22  count                        25192 non-null  int64
 23  srv_count                    25192 non-null  int64
 24  serror_rate                  25192 non-null  float64
 25  srv_serror_rate              25192 non-null  float64
 26  rerror_rate                  25192 non-null  float64
 27  srv_rerror_rate              25192 non-null  float64
 28  same_srv_rate                25192 non-null  float64
 29  diff_srv_rate                25192 non-null  float64
 30  srv_diff_host_rate           25192 non-null  float64
 31  dst_host_count               25192 non-null  int64
 32  dst_host_srv_count           25192 non-null  int64
 33  dst_host_same_srv_rate       25192 non-null  float64
 34  dst_host_diff_srv_rate       25192 non-null  float64
 35  dst_host_same_src_port_rate  25192 non-null  float64
 36  dst_host_srv_diff_host_rate  25192 non-null  float64
 37  dst_host_serror_rate         25192 non-null  float64
 38  dst_host_srv_serror_rate     25192 non-null  float64
 39  dst_host_rerror_rate         25192 non-null  float64
 40  dst_host_srv_rerror_rate     25192 non-null  float64
 41  class                        25192 non-null  object
dtypes: float64(15), int64(23), object(4)
memory usage: 8.1+ MB
```

```
train.describe()
```

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in
count	25192.000000	2.519200e+04	2.519200e+04	25192.000000	25192.000000	25192.00000	25192.000000	25192.000000	25192.000000
mean	305.054104	2.433063e+04	3.491847e+03	0.000079	0.023738	0.00004	0.198039	0.001191	0.394768
std	2686.555640	2.410805e+06	8.883072e+04	0.008910	0.260221	0.00630	2.154202	0.045418	0.488811
min	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000
25%	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000
50%	0.000000	4.400000e+01	0.000000e+00	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000
75%	0.000000	2.790000e+02	5.302500e+02	0.000000	0.000000	0.00000	0.000000	0.000000	1.000000
max	42862.000000	3.817091e+08	5.151385e+06	1.000000	3.000000	1.00000	77.000000	4.000000	1.000000

8 rows × 38 columns

```
train.describe(include='object')
```

	protocol_type	service	flag	class
count	25192	25192	25192	25192
unique	3	66	11	2
top	tcp	http	SF	normal
freq	20526	8003	14973	13449

```
train.shape
```

```
(25192, 42)
```

```
train.isnull().sum()
```

```
duration                     0
protocol_type                0
service                      0
flag                         0
src_bytes                    0
dst_bytes                    0
land                         0
wrong_fragment               0
urgent                       0
hot                          0
num_failed_logins            0
logged_in                    0
num_compromised              0
```

```
root_shell                    0
su_attempted                  0
num_root                      0
num_file_creations            0
num_shells                    0
num_access_files              0
num_outbound_cmds             0
is_host_login                 0
is_guest_login                0
count                         0
srv_count                     0
serror_rate                   0
srv_serror_rate               0
rerror_rate                   0
srv_rerror_rate               0
same_srv_rate                 0
diff_srv_rate                 0
srv_diff_host_rate            0
dst_host_count                0
dst_host_srv_count            0
dst_host_same_srv_rate        0
dst_host_diff_srv_rate        0
dst_host_same_src_port_rate   0
dst_host_srv_diff_host_rate   0
dst_host_serror_rate          0
dst_host_srv_serror_rate      0
dst_host_rerror_rate          0
dst_host_srv_rerror_rate      0
class                         0
dtype: int64
```

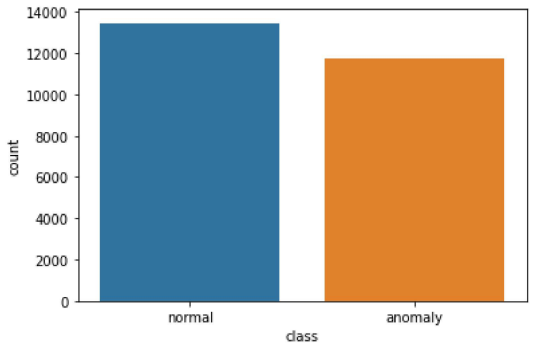
```
total = train.shape[0]
missing_columns = [col for col in train.columns if train[col].isnull().sum() > 0]
for col in missing_columns:
    null_count = train[col].isnull().sum()
    per = (null_count/total) * 100
    print(f"{col}: {null_count} ({round(per, 3)}%)")
```

```
print(f"Number of duplicate rows: {train.duplicated().sum()}")
```

```
Number of duplicate rows: 0
```

```
sns.countplot(x=train['class'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f392d3f55d0>
```




```
print('Class distribution Training set:')
print(train['class'].value_counts())
```

```
Class distribution Training set:
normal     13449
anomaly    11743
Name: class, dtype: int64
```

```
def le(df):
    for col in df.columns:
        if df[col].dtype == 'object':
                label_encoder = LabelEncoder()
                df[col] = label_encoder.fit_transform(df[col])

le(train)
le(test)
```

```
train.drop(['num_outbound_cmds'], axis=1, inplace=True)
test.drop(['num_outbound_cmds'], axis=1, inplace=True)
```

```
train.head()
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	u
0	0	1	19	9	491	0	0	0	
1	0	2	41	9	146	0	0	0	
2	0	1	46	5	0	0	0	0	
3	0	1	22	9	232	8153	0	0	
4	0	1	22	9	199	420	0	0	

5 rows × 41 columns

```
X_train = train.drop(['class'], axis=1)
Y_train = train['class']
```

```
rfc = RandomForestClassifier()

rfe = RFE(rfc, n_features_to_select=10)
rfe = rfe.fit(X_train, Y_train)

feature_map = [(i, v) for i, v in itertools.zip_longest(rfe.get_support(), X_train.columns)]
selected_features = [v for i, v in feature_map if i==True]

selected_features
```

```
['protocol_type',
 'service',
 'flag',
 'src_bytes',
 'dst_bytes',
 'count',
 'same_srv_rate',
 'dst_host_srv_count',
 'dst_host_same_srv_rate',
 'dst_host_same_src_port_rate']
```

```
X_train = X_train[selected_features]
```

```
scale = StandardScaler()
X_train = scale.fit_transform(X_train)
test = scale.fit_transform(test)
```

```
x_train, x_test, y_train, y_test = train_test_split(X_train, Y_train, train_size=0.70, random_state=2)
```

```
x_train.shape
```

```
(17634, 10)
```

```
x_test.shape
```

```
(7558, 10)
```

```
y_train.shape
```

```
(17634,)
```

```
y_test.shape
```

```
(7558,)
```

```
import time


from sklearn.linear_model import LogisticRegression

clfl = LogisticRegression(max_iter = 1200000)
start_time = time.time()
clfl.fit(x_train, y_train.values.ravel())
end_time = time.time()
print("Training time: ", end_time-start_time)
```

```
Training time:  0.09832596778869629
```

```
start_time = time.time()
y_test_pred = clfl.predict(x_train)
end_time = time.time()
print("Testing time: ", end_time-start_time)
```

```
Testing time:  0.004812717437744141
```

```
lg_model = LogisticRegression(random_state = 42)
lg_model.fit(x_train, y_train)
```

```
LogisticRegression(random_state=42)
```

```
lg_train, lg_test = lg_model.score(x_train , y_train), lg_model.score(x_test , y_test)

print(f"Training Score: {lg_train}")
print(f"Test Score: {lg_test}")
```

```
Training Score: 0.9351253260746285
Test Score: 0.9323895210373114
```

```
pip install optuna
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting optuna
  Downloading optuna-3.0.3-py3-none-any.whl (348 kB)
     |████████████████████████████████| 348 kB 5.3 MB/s
Collecting cmaes>=0.8.2
  Downloading cmaes-0.8.2-py3-none-any.whl (15 kB)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.7/dist-packages (from optuna) (6.0)
Requirement already satisfied: sqlalchemy>=1.3.0 in /usr/local/lib/python3.7/dist-packages (from optuna) (1.4.42)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from optuna) (4.64.1)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-packages (from optuna) (21.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from optuna) (1.21.6)
Collecting alembic>=1.5.0
  Downloading alembic-1.8.1-py3-none-any.whl (209 kB)
     |████████████████████████████████| 209 kB 47.9 MB/s
Collecting cliff
  Downloading cliff-3.10.1-py3-none-any.whl (81 kB)
     |████████████████████████████████| 81 kB 8.4 MB/s
Requirement already satisfied: scipy<1.9.0,>=1.7.0 in /usr/local/lib/python3.7/dist-packages (from optuna) (1.7.3)
Collecting colorlog
  Downloading colorlog-6.7.0-py2.py3-none-any.whl (11 kB)
Requirement already satisfied: importlib-metadata<5.0.0 in /usr/local/lib/python3.7/dist-packages (from optuna) (4.13.0)
Collecting Mako
  Downloading Mako-1.2.3-py3-none-any.whl (78 kB)
     |████████████████████████████████| 78 kB 6.8 MB/s
Requirement already satisfied: importlib-resources in /usr/local/lib/python3.7/dist-packages (from alembic>=1.5.0->optuna) (5.10.0)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata<5.0.0->optuna) (3.10.0)
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata<5.0.0->optuna
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging>=20.0->optuna) (3.0.9)
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.7/dist-packages (from sqlalchemy>=1.3.0->optuna) (1.1.3.post0)
Collecting cmd2>=1.0.0
  Downloading cmd2-2.4.2-py3-none-any.whl (147 kB)
     |████████████████████████████████| 147 kB 65.6 MB/s
Collecting stevedore>=2.0.1
  Downloading stevedore-3.5.2-py3-none-any.whl (50 kB)
     |████████████████████████████████| 50 kB 5.6 MB/s
Collecting autopage>=0.4.0
```

```
  Downloading autopage-0.5.1-py3-none-any.whl (29 kB)
Collecting pbr!=2.1.0,>=2.0.0
  Downloading pbr-5.11.0-py2.py3-none-any.whl (112 kB)
     |████████████████████████████████| 112 kB 51.7 MB/s
Requirement already satisfied: PrettyTable>=0.7.2 in /usr/local/lib/python3.7/dist-packages (from cliff->optuna) (3.4.1)
Collecting pyperclip>=1.6
  Downloading pyperclip-1.8.2.tar.gz (20 kB)
Requirement already satisfied: attrs>=16.3.0 in /usr/local/lib/python3.7/dist-packages (from cmd2>=1.0.0->cliff->optuna) (22.1.0)
Requirement already satisfied: wcwidth>=0.1.7 in /usr/local/lib/python3.7/dist-packages (from cmd2>=1.0.0->cliff->optuna) (0.2.5)
Requirement already satisfied: MarkupSafe>=0.9.2 in /usr/local/lib/python3.7/dist-packages (from Mako->alembic>=1.5.0->optuna) (2.0.1)
Building wheels for collected packages: pyperclip
  Building wheel for pyperclip (setup.py) ... done
  Created wheel for pyperclip: filename=pyperclip-1.8.2-py3-none-any.whl size=11137 sha256=b59e109a4af9dbd8b2bafde6c050524fa1efb03179b50
  Stored in directory: /root/.cache/pip/wheels/9f/18/84/8f69f8b08169c7bae2dde6bd7daf0c19fca8c8e500ee620a28
Successfully built pyperclip
Installing collected packages: pyperclip, pbr, stevedore, Mako, cmd2, autopage, colorlog, cmaes, cliff, alembic, optuna
Successfully installed Mako-1.2.3 alembic-1.8.1 autopage-0.5.1 cliff-3.10.1 cmaes-0.8.2 cmd2-2.4.2 colorlog-6.7.0 optuna-3.0.3 pbr-5.11.
```

```
import optuna
optuna.logging.set_verbosity(optuna.logging.WARNING)
```

```
def objective(trial):
    n_neighbors = trial.suggest_int('KNN_n_neighbors', 2, 16, log=False)
    classifier_obj = KNeighborsClassifier(n_neighbors=n_neighbors)
    classifier_obj.fit(x_train, y_train)
    accuracy = classifier_obj.score(x_test, y_test)
    return accuracy


study_KNN = optuna.create_study(direction='maximize')
study_KNN.optimize(objective, n_trials=1)
print(study_KNN.best_trial)
```

```
FrozenTrial(number=0, values=[0.9826673723207198], datetime_start=datetime.datetime(2022, 11, 6, 6, 45, 38, 801815), datetime_complete=d
```

```
KNN_model = KNeighborsClassifier(n_neighbors=study_KNN.best_trial.params['KNN_n_neighbors'])
KNN_model.fit(x_train, y_train)

KNN_train, KNN_test = KNN_model.score(x_train, y_train), KNN_model.score(x_test, y_test)

print(f"Train Score: {KNN_train}")
print(f"Test Score: {KNN_test}")
```

```
Train Score: 0.9895089032550755
Test Score: 0.9826673723207198
```

```
from sklearn.tree import DecisionTreeClassifier

clfd = DecisionTreeClassifier(criterion ="entropy", max_depth = 4)
start_time = time.time()
clfd.fit(x_train, y_train.values.ravel())
end_time = time.time()
print("Training time: ", end_time-start_time)
```

```
Training time:  0.07346153259277344
```

```
start_time = time.time()
y_test_pred = clfd.predict(x_train)
end_time = time.time()
print("Testing time: ", end_time-start_time)
```

```
Testing time:  0.009153604507446289
```

```
def objective(trial):
    dt_max_depth = trial.suggest_int('dt_max_depth', 2, 32, log=False)
    dt_max_features = trial.suggest_int('dt_max_features', 2, 10, log=False)
    classifier_obj = DecisionTreeClassifier(max_features = dt_max_features, max_depth = dt_max_depth)
    classifier_obj.fit(x_train, y_train)
    accuracy = classifier_obj.score(x_test, y_test)
    return accuracy


study_dt = optuna.create_study(direction='maximize')
study_dt.optimize(objective, n_trials=30)
print(study_dt.best_trial)
```

```
FrozenTrial(number=5, values=[0.9952368351415718], datetime_start=datetime.datetime(2022, 11, 6, 6, 45, 55, 147415), datetime_complete=d
```

```
dt = DecisionTreeClassifier(max_features = study_dt.best_trial.params['dt_max_features'], max_depth = study_dt.best_trial.params['dt_max_deptl
dt.fit(x_train, y_train)

dt_train, dt_test = dt.score(x_train, y_train), dt.score(x_test, y_test)

print(f"Train Score: {dt_train}")
print(f"Test Score: {dt_test}")
```

```
Train Score: 1.0
Test Score: 0.9943106641968775
```

```
data = [["KNN", KNN_train, KNN_test],
        ["Logistic Regression", lg_train, lg_test],
        ["Decision Tree", dt_train, dt_test]]

col_names = ["Model", "Train Score", "Test Score"]
print(tabulate(data, headers=col_names, tablefmt="fancy_grid"))
```

```
╒═══════════════════╤═══════════════╤══════════════╕
│ Model             │   Train Score │   Test Score │
╞═══════════════════╪═══════════════╪══════════════╡
│ KNN               │      0.989509 │     0.982667 │
├───────────────────┼───────────────┼──────────────┤
│ Logistic Regression │      0.935125 │     0.93239  │
├───────────────────┼───────────────┼──────────────┤
│ Decision Tree     │      1        │     0.994311 │
╘═══════════════════╧═══════════════╧══════════════╛
```

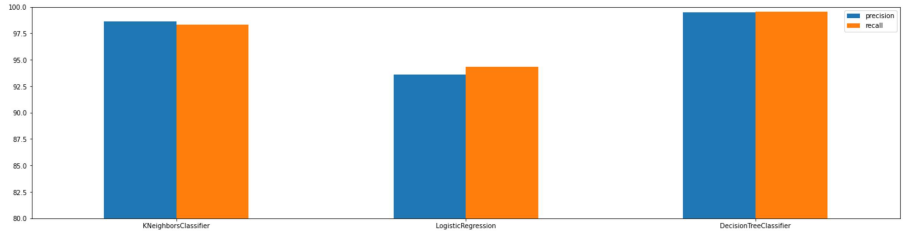
```
SEED = 42

# Decision Tree Model
dtc = DecisionTreeClassifier()

# KNN
knn = KNeighborsClassifier()

# LOGISTIC REGRESSION MODEL

lr = LogisticRegression()


from sklearn.model_selection import cross_val_score
models = {}
models['KNeighborsClassifier']= knn
models['LogisticRegression']= lr
models['DecisionTreeClassifier']= dtc
```

```
scores = {}
for name in models:
  scores[name]={}
  for scorer in ['precision','recall']:
    scores[name][scorer] = cross_val_score(models[name], x_train, y_train, cv=10, scoring=scorer)


def line(name):
  return '*'*(25-len(name)//2)

for name in models:
  print(line(name), name, 'Model Validation', line(name))

  for scorer in ['precision','recall']:
    mean = round(np.mean(scores[name][scorer])*100,2)
    stdev = round(np.std(scores[name][scorer])*100,2)
    print ("Mean {}:".format(scorer),"\n", mean,"%", "+-",stdev)
    print()
```

```
********** KNeighborsClassifier Model Validation **********
Mean precision:
 98.64 % +- 0.5

Mean recall:
 98.31 % +- 0.6

********** LogisticRegression Model Validation **********
Mean precision:
 93.57 % +- 0.71

Mean recall:
 94.31 % +- 0.61

********** DecisionTreeClassifier Model Validation **********
Mean precision:
 99.51 % +- 0.25

Mean recall:
 99.57 % +- 0.28
```

```
for name in models:
    for scorer in ['precision','recall']:
        scores[name][scorer] = scores[name][scorer].mean()
scores=pd.DataFrame(scores).swapaxes("index", "columns")*100
scores.plot(kind = "bar",  ylim=[80,100], figsize=(24,6), rot=0)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3928c19210>
```




```
models = {}
models['KNeighborsClassifier']= knn
models['LogisticRegression']= lr
models['DecisionTreeClassifier']= dtc
```

```
preds={}
for name in models:
    models[name].fit(x_train, y_train)
    preds[name] = models[name].predict(x_test)
print("Predictions complete.")
```

```
 Predictions complete.
```

```
from sklearn.metrics import confusion_matrix, classification_report, f1_score
def line(name,sym="*"):
    return sym*(25-len(name)//2)
target_names=["normal","anamoly"]
for name in models:
    print(line(name), name, 'Model Testing', line(name))
    print(confusion_matrix(y_test, preds[name]))
    print(line(name,'-'))
    print(classification_report(y_test, preds[name], target_names=target_names))
```

```
********** KNeighborsClassifier Model Testing **********
[[3447   51]
 [  66 3994]]
--------------
              precision    recall  f1-score   support

      normal       0.98      0.99      0.98      3498
     anamoly       0.99      0.98      0.99      4060

    accuracy                           0.98      7558
   macro avg       0.98      0.98      0.98      7558
weighted avg       0.98      0.98      0.98      7558

*************** LogisticRegression Model Testing ***************
[[3228  270]
 [ 241 3819]]
----------------
              precision    recall  f1-score   support

      normal       0.93      0.92      0.93      3498
     anamoly       0.93      0.94      0.94      4060

    accuracy                           0.93      7558
   macro avg       0.93      0.93      0.93      7558
weighted avg       0.93      0.93      0.93      7558

************** DecisionTreeClassifier Model Testing **************
[[3479   19]
 [  25 4035]]
--------------
              precision    recall  f1-score   support

      normal       0.99      0.99      0.99      3498
     anamoly       1.00      0.99      0.99      4060

    accuracy                           0.99      7558
   macro avg       0.99      0.99      0.99      7558
weighted avg       0.99      0.99      0.99      7558
```

```
f1s = {}
for name in models:
    f1s[name]=f1_score(y_test, preds[name])
f1s=pd.DataFrame(f1s.values(),index=f1s.keys(),columns=["F1-score"])*100
f1s.plot(kind = "bar",  ylim=[80,100], figsize=(10,6), rot=0)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3928bc4450>
```