```
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
```

```
train=pd.read_csv('Train_data.csv')
```

```
test=pd.read_csv('Test_data.csv')
```

```
train.head()
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment
0	0	tcp	ftp_data	SF	491	0	0	0
1	0	udp	other	SF	146	0	0	0
2	0	tcp	private	S0	0	0	0	0
3	0	tcp	http	SF	232	8153	0	0
4	0	tcp	http	SF	199	420	0	0

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
```

```
1 train.describe()
```

	duration	src_bytes	dst_bytes	land	wrong_fragment	urg
count	25192.000000	2.519200e+04	2.519200e+04	25192.000000	25192.000000	25192.00
mean	305.054104	2.433063e+04	3.491847e+03	0.000079	0.023738	0.00
std	2686.555640	2.410805e+06	8.883072e+04	0.008910	0.260221	0.00
min	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.00
25%	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.00
50%	0.000000	4.400000e+01	0.000000e+00	0.000000	0.000000	0.00
75%	0.000000	2.790000e+02	5.302500e+02	0.000000	0.000000	0.00
max	42862.000000	3.817091e+08	5.151385e+06	1.000000	3.000000	1.00

8 rows × 38 columns

```
1 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

```
1 train.shape
```

```
(25192, 42)
```

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

```
<bound method NDFrame._add_numeric_operations.<locals>.sum of        duration  protocol_type  service   flag  src_bytes
dst_bytes   land  \
0         False          False    False  False      False      False  False
1         False          False    False  False      False      False  False
2         False          False    False  False      False      False  False
3         False          False    False  False      False      False  False
4         False          False    False  False      False      False  False
...         ...            ...      ...    ...        ...        ...    ...
25187     False          False    False  False      False      False  False
25188     False          False    False  False      False      False  False
25189     False          False    False  False      False      False  False
25190     False          False    False  False      False      False  False
25191     False          False    False  False      False      False  False

       wrong_fragment  urgent    hot  ...  dst_host_srv_count  \
0               False   False  False  ...               False
1               False   False  False  ...               False
2               False   False  False  ...               False
3               False   False  False  ...               False
4               False   False  False  ...               False
...               ...     ...    ...  ...                 ...
25187           False   False  False  ...               False
25188           False   False  False  ...               False
25189           False   False  False  ...               False
25190           False   False  False  ...               False
25191           False   False  False  ...               False

       dst_host_same_srv_rate  dst_host_diff_srv_rate  \
0                       False                   False
1                       False                   False
2                       False                   False
3                       False                   False
4                       False                   False
...                       ...                     ...
25187                   False                   False
```

```
25188                  False                  False
25189                  False                  False
25190                  False                  False
25191                  False                  False

       dst_host_same_src_port_rate  dst_host_srv_diff_host_rate  \
0                            False                       False
1                            False                       False
2                            False                       False
3                            False                       False
4                            False                       False
...                            ...                         ...
25187                        False                       False
25188                        False                       False
25189                        False                       False
25190                        False                       False
25191                        False                       False

       dst_host_serror_rate  dst_host_srv_serror_rate  dst_host_rerror_rate  \
0                     False                     False                 False
1                     False                     False                 False
2                     False                     False                 False
3                     False                     False                 False
```

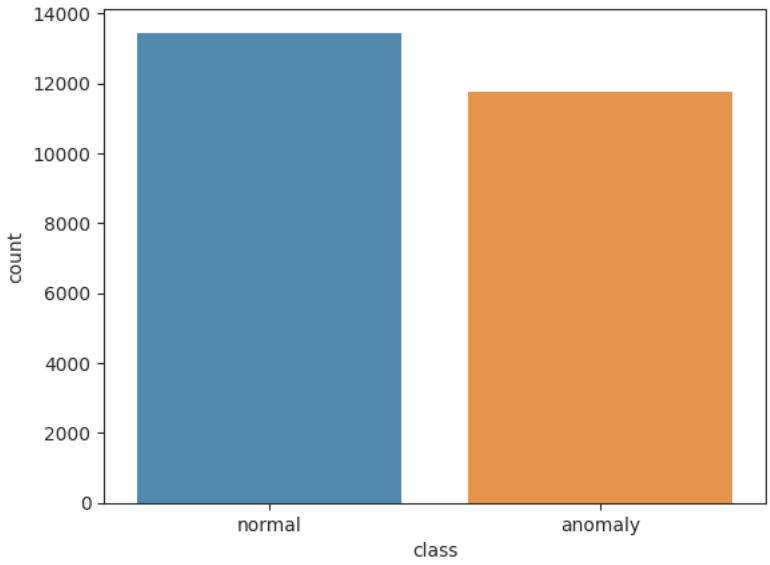
```
total=train.shape[0]
missing_columns=[col for col in train.columns if train[col].isnull().sum()>0]
for col in missing_columns:
    null_count=train[col].isnull().sum()
    per=(null_count/total)* 100
    print(f"{col}:{null_count}({round(per,3)}%)")
```

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

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

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

```
<Axes: xlabel='class', ylabel='count'>
```




```
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)
```

```
1 train.head()
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host
0	0	1	19	9	491	0	0	0	0	0	...	25	
1	0	2	41	9	146	0	0	0	0	0	...	1	
2	0	1	46	5	0	0	0	0	0	0	...	26	
3	0	1	22	9	232	8153	0	0	0	0	...	255	
4	0	1	22	9	199	420	0	0	0	0	...	255	

5 rows × 41 columns

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

```
1 rfc = RandomForestClassifier()
2
3 rfe = RFE(rfc, n_features_to_select=10)
4 rfe = rfe.fit(X_train, Y_train)
5
6 feature_map = [(i, v) for i, v in itertools.zip_longest(rfe.get_support(), X_train.columns)]
7 selected_features = [v for i, v in feature_map if i==True]
8
9 selected_features
```

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

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

```
1 scale = StandardScaler()
2 X_train = scale.fit_transform(X_train)
3 test = scale.fit_transform(test)
4
```

```
1 x_train, x_test, y_train, y_test = train_test_split(X_train, Y_train)
```

```
1 x_train.shape
```

```
(18894, 10)
```

```
1 x_test.shape
```

```
(6298, 10)
```

```
1 y_train.shape
```

```
(18894,)
```

```
1 y_test.shape
```

```
(6298,)
```

```
1 import time
```

```
1 from sklearn.linear_model import LogisticRegression
2
3 clfl = LogisticRegression(max_iter = 1200000)
4 start_time = time.time()
5 clfl.fit(x_train, y_train.values.ravel())
6 end_time = time.time()
7 print("Training time: ", end_time-start_time)
```

```
Training time:  0.36036181449890137
```

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

```
Testing time:  0.002568960189819336
```

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

```
LogisticRegression
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: 0.9431036307822589
Test Score: 0.9406160685932042
```

```
pip install optuna
```

```
Collecting optuna
  Downloading optuna-3.4.0-py3-none-any.whl (409 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 409.6/409.6 kB 7.0 MB/s eta 0:00:00
Collecting alembic>=1.5.0 (from optuna)
  Downloading alembic-1.13.0-py3-none-any.whl (230 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 230.6/230.6 kB 11.7 MB/s eta 0:00:00
Collecting colorlog (from optuna)
  Downloading colorlog-6.8.0-py3-none-any.whl (11 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from optuna) (1.23.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from optuna) (23.2)
Requirement already satisfied: sqlalchemy>=1.3.0 in /usr/local/lib/python3.10/dist-packages (from optuna) (2.0.23)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from optuna) (4.66.1)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.10/dist-packages (from optuna) (6.0.1)
Collecting Mako (from alembic>=1.5.0->optuna)
  Downloading Mako-1.3.0-py3-none-any.whl (78 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 78.6/78.6 kB 7.0 MB/s eta 0:00:00
Requirement already satisfied: typing-extensions>=4 in /usr/local/lib/python3.10/dist-packages (from alembic>=1.5.0->optuna) (4.5.0
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.10/dist-packages (from sqlalchemy>=1.3.0->optuna) (3.0.1)
Requirement already satisfied: MarkupSafe>=0.9.2 in /usr/local/lib/python3.10/dist-packages (from Mako->alembic>=1.5.0->optuna) (2.1
Installing collected packages: Mako, colorlog, alembic, optuna
Successfully installed Mako-1.3.0 alembic-1.13.0 colorlog-6.8.0 optuna-3.4.0
```

```
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, state=TrialState.COMPLETE, values=[0.9791997459510956], datetime_start=datetime.datetime(2023, 12, 3, 8, 21, 7
```

```
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: 0.9837514554885148
Test Score: 0.9791997459510956
```

```
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:  0.04148530960083008
```

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

```
Testing time:  0.003440380096435547
```

```
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=21, state=TrialState.COMPLETE, values=[0.9960304858685297], datetime_start=datetime.datetime(2023, 12, 3, 8, 22,
```

```
dt = DecisionTreeClassifier(max_features = study_dt.best_trial.params['dt_max_features'], max_depth = study_dt.best_trial.params['dt_
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: 0.9998941462898274
Test Score: 0.9938075579549063
```

```
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	0.983751	0.9792
Logistic Regression	0.943104	0.940616
Decision Tree	0.999894	0.993808

```
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
```

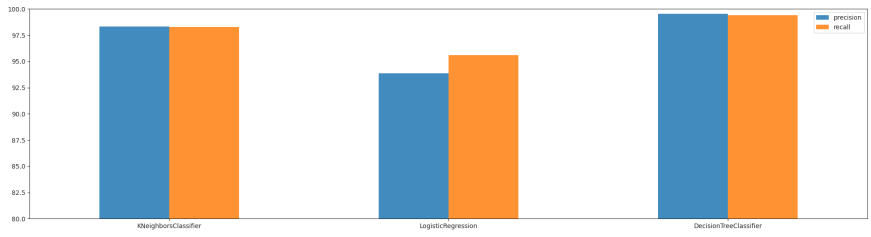
```
scores = {}
for name in models:
  scores[name]={}
  for scorer in ['precision','recall']:
    scores[name][scorer] = cross_val_score(models[name], x_train, y_train, cv=10, scoring=scorer)

```

```
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:
 98.33 % +- 0.33

Mean recall:
 98.28 % +- 0.32

********** LogisticRegression Model Validation **********
Mean precision:
 93.86 % +- 0.54

Mean recall:
 95.59 % +- 0.69

********** DecisionTreeClassifier Model Validation **********
Mean precision:
 99.56 % +- 0.25

Mean recall:
 99.39 % +- 0.17
```

```
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)
```

```
<Axes: >
```




```
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 **********
[[2904   61]
 [  52 3281]]
--------------
              precision    recall  f1-score   support

      normal       0.98      0.98      0.98      2965
     anamoly       0.98      0.98      0.98      3333

    accuracy                           0.98      6298
   macro avg       0.98      0.98      0.98      6298
weighted avg       0.98      0.98      0.98      6298

********** LogisticRegression Model Testing **********
[[2731  234]
 [ 140 3193]]
----------------
              precision    recall  f1-score   support

      normal       0.95      0.92      0.94      2965
     anamoly       0.93      0.96      0.94      3333

    accuracy                           0.94      6298
   macro avg       0.94      0.94      0.94      6298
weighted avg       0.94      0.94      0.94      6298

********** DecisionTreeClassifier Model Testing **********
[[2948   17]
 [  19 3314]]
--------------
              precision    recall  f1-score   support

      normal       0.99      0.99      0.99      2965
     anamoly       0.99      0.99      0.99      3333

    accuracy                           0.99      6298
   macro avg       0.99      0.99      0.99      6298
weighted avg       0.99      0.99      0.99      6298
```

```
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)
```

```
<Axes: >
```