```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier, VotingClassifier


# Import the dataset
df = pd.read_csv('/content/Kddcup_99_csv.csv')
df.columns
```

```
    Index(['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
           'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
           'num_failed_logins', 'logged_in', 'lnum_compromised', 'lroot_shell',
           'lsu_attempted', 'lnum_root', 'lnum_file_creations', 'lnum_shells',
           'lnum_access_files', 'lnum_outbound_cmds', 'is_host_login',
           'is_guest_login', 'count', 'srv_count', 'serror_rate',
           'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate', 'same_srv_rate',
           'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
           'dst_host_srv_count', 'dst_host_same_srv_rate',
           'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
           'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
           'dst_host_srv_serror_rate', 'dst_host_rerror_rate',
           'dst_host_srv_rerror_rate', 'label'],
          dtype='object')
```

```
# Splitting dataset into features and label
X = df[['duration','src_bytes','dst_bytes','src_bytes','num_failed_logins','is_hos
       'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate', 'same_srv_rate',
       'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
       'dst_host_srv_count', 'dst_host_same_srv_rate',
       'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
       'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
       'dst_host_srv_serror_rate', 'dst_host_rerror_rate',
       'dst_host_srv_rerror_rate']]
y = df['label']



#Splitting the dataset into the training set and the test set
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st

# Feature scaling (or standardization)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


Model1 = LogisticRegression(multi_class='multinomial', random_state=1)
Model2 = RandomForestClassifier(n_estimators=50, random_state=1)
Model3 = GaussianNB()


eclf1 = VotingClassifier(estimators=[('lr', Model1), ('rf', Model2), ('gnb', Model
```

```
eclf1 = VotingClassifier(estimators=[('lr', Model1), ('rf', Model2), ('gnb', Model
eclf1 = eclf1.fit(X_train, y_train)


y_pred1=(eclf1.predict(X_test))
print(y_pred1)
```

```
['normal' 'normal' 'normal' 'smurf' 'smurf' 'normal' 'neptune' 'neptune'
 'normal' 'normal' 'neptune' 'normal' 'neptune' 'neptune' 'smurf' 'smurf'
 'normal' 'normal' 'neptune' 'neptune' 'neptune' 'normal' 'smurf' 'normal'
 'neptune' 'neptune' 'smurf' 'neptune' 'neptune' 'neptune' 'neptune'
 'normal' 'neptune' 'neptune' 'neptune' 'smurf' 'neptune' 'smurf'
 'neptune' 'neptune' 'normal' 'normal' 'neptune' 'neptune' 'normal'
 'normal' 'neptune' 'smurf' 'smurf' 'smurf' 'smurf' 'smurf' 'neptune'
 'normal' 'smurf' 'normal' 'smurf' 'neptune' 'neptune' 'neptune' 'neptune'
 'neptune' 'neptune' 'neptune' 'smurf' 'smurf' 'neptune' 'normal'
 'neptune' 'neptune' 'normal' 'normal' 'neptune' 'smurf' 'smurf' 'neptune'
 'neptune' 'normal' 'smurf' 'normal' 'smurf' 'normal' 'neptune' 'normal'
 'normal' 'neptune' 'smurf' 'smurf' 'normal' 'neptune' 'normal' 'neptune'
 'neptune' 'normal' 'neptune' 'smurf' 'neptune']
```

```
eclf2 = VotingClassifier(estimators=[('lr', Model1), ('rf', Model2), ('gnb', Model
eclf2 = eclf1.fit(X_train, y_train)


y_pred2=(eclf2.predict(X_test))
print(y_pred2)
```

```
['normal' 'normal' 'normal' 'smurf' 'smurf' 'normal' 'neptune' 'neptune'
 'normal' 'normal' 'neptune' 'normal' 'neptune' 'neptune' 'smurf' 'smurf'
 'normal' 'normal' 'neptune' 'neptune' 'neptune' 'normal' 'smurf' 'normal'
 'neptune' 'neptune' 'smurf' 'neptune' 'neptune' 'neptune' 'neptune'
 'normal' 'neptune' 'neptune' 'neptune' 'smurf' 'neptune' 'smurf'
 'neptune' 'neptune' 'normal' 'normal' 'neptune' 'neptune' 'normal'
 'normal' 'neptune' 'smurf' 'smurf' 'smurf' 'smurf' 'smurf' 'neptune'
 'normal' 'smurf' 'normal' 'smurf' 'neptune' 'neptune' 'neptune' 'neptune'
 'neptune' 'neptune' 'neptune' 'smurf' 'smurf' 'neptune' 'normal'
 'neptune' 'neptune' 'normal' 'normal' 'neptune' 'smurf' 'smurf' 'neptune'
 'neptune' 'normal' 'smurf' 'normal' 'smurf' 'normal' 'neptune' 'normal'
 'normal' 'neptune' 'smurf' 'smurf' 'normal' 'neptune' 'normal' 'neptune'
 'neptune' 'normal' 'neptune' 'smurf' 'neptune']
```

```
from sklearn.metrics import accuracy_score
acc = accuracy_score(y_test, y_pred1)
print(acc)
```

```
1.0
```

```
from sklearn.metrics import accuracy_score
acc = accuracy_score(y_test, y_pred2)
print(acc)
```

```
1.0
```

✓ 0s completed at 13:14