Boosting Network Intrusion Detection: Two-Level Ensemble Learning and Stacked Knowledge Distillation Approaches

```
!pip install catboost
```

```
Collecting catboost
  Downloading catboost-1.2.5-cp310-cp310-manylinux2014_x86_64.whl.metadata (1.2 kB)
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (from catboost) (0.20.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from catboost) (3.7.1)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dist-packages (from catboost) (1.26.4)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist-packages (from catboost) (2.1.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from catboost) (1.13.1)
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (from catboost) (5.15.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from catboost) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2.8
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (3.1.4)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly->catboost) (9.0.0)
Downloading catboost-1.2.5-cp310-cp310-manylinux2014_x86_64.whl (98.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.2/98.2 MB 7.3 MB/s eta 0:00:00
Installing collected packages: catboost
Successfully installed catboost-1.2.5
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn.svm import SVC
import statsmodels.api as sm
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import RobustScaler
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree  import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import tree
from sklearn import svm
from sklearn.svm import SVC
from sklearn.calibration import CalibratedClassifierCV
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import RocCurveDisplay
from itertools import cycle
from sklearn.model_selection import GridSearchCV
from mlxtend.plotting import plot_confusion_matrix
from sklearn.ensemble import GradientBoostingClassifier
import xgboost as xgb
import lightgbm as lgb
from catboost import CatBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.linear_model import RidgeClassifier
from sklearn.calibration import CalibratedClassifierCV
from sklearn.linear_model import RidgeClassifier
```

```
# Turn off the warnings.
warnings.filterwarnings(action='ignore')
%matplotlib inline
```

```
# prompt: GIVE ME CODE FOR GOOGLE DRIVE MOUNT
```

```
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).
```

Reading Data

```
Trained_Data = pd.read_csv("/content/drive/MyDrive/KDDTrain+.txt" , sep = "," , encoding = 'utf-8')
Tested_Data  = pd.read_csv("/content/drive/MyDrive/KDDTest+.txt" , sep = "," , encoding = 'utf-8')
```

Exploring Data

```
Trained_Data
```

	0	tcp	ftp_data	SF	491	0.1	0.2	0.3	0.4	0.5	...	0.17	0.03	0.17.1	0.00.6	0.00.7	0.00.8	0.05	0.00.9	normal
0	0	udp	other	SF	146	0	0	0	0	0	...	0.00	0.60	0.88	0.00	0.00	0.00	0.00	0.00	normal
1	0	tcp	private	S0	0	0	0	0	0	0	...	0.10	0.05	0.00	0.00	1.00	1.00	0.00	0.00	neptune
2	0	tcp	http	SF	232	8153	0	0	0	0	...	1.00	0.00	0.03	0.04	0.03	0.01	0.00	0.01	normal
3	0	tcp	http	SF	199	420	0	0	0	0	...	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	normal
4	0	tcp	private	REJ	0	0	0	0	0	0	...	0.07	0.07	0.00	0.00	0.00	0.00	1.00	1.00	neptune
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
125967	0	tcp	private	S0	0	0	0	0	0	0	...	0.10	0.06	0.00	0.00	1.00	1.00	0.00	0.00	neptune
125968	8	udp	private	SF	105	145	0	0	0	0	...	0.96	0.01	0.01	0.00	0.00	0.00	0.00	0.00	normal
125969	0	tcp	smtp	SF	2231	384	0	0	0	0	...	0.12	0.06	0.00	0.00	0.72	0.00	0.01	0.00	normal
125970	0	tcp	klogin	S0	0	0	0	0	0	0	...	0.03	0.05	0.00	0.00	1.00	1.00	0.00	0.00	neptune
125971	0	tcp	ftp_data	SF	151	0	0	0	0	0	...	0.30	0.03	0.30	0.00	0.00	0.00	0.00	0.00	normal

125972 rows × 43 columns

```
Tested_Data
```

	0	tcp	private	REJ	0.1	0.2	0.3	0.4	0.5	0.6	...	0.04.1	0.06.1	0.00.3	0.00.4	0.00.5	0.00.6	1.00.2	1.00.3
0	0	tcp	private	REJ	0	0	0	0	0	0	...	0.00	0.06	0.00	0.00	0.00	0.0	1.00	1.0(
1	2	tcp	ftp_data	SF	12983	0	0	0	0	0	...	0.61	0.04	0.61	0.02	0.00	0.0	0.00	0.0(
2	0	icmp	eco_i	SF	20	0	0	0	0	0	...	1.00	0.00	1.00	0.28	0.00	0.0	0.00	0.0(
3	1	tcp	telnet	RSTO	0	15	0	0	0	0	...	0.31	0.17	0.03	0.02	0.00	0.0	0.83	0.7
4	0	tcp	http	SF	267	14515	0	0	0	0	...	1.00	0.00	0.01	0.03	0.01	0.0	0.00	0.0(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
22538	0	tcp	smtp	SF	794	333	0	0	0	0	...	0.72	0.06	0.01	0.01	0.01	0.0	0.00	0.0(
22539	0	tcp	http	SF	317	938	0	0	0	0	...	1.00	0.00	0.01	0.01	0.01	0.0	0.00	0.0(
22540	0	tcp	http	SF	54540	8314	0	0	0	2	...	1.00	0.00	0.00	0.00	0.00	0.0	0.07	0.0
22541	0	udp	domain_u	SF	42	42	0	0	0	0	...	0.99	0.01	0.00	0.00	0.00	0.0	0.00	0.0(
22542	0	tcp	sunrpc	REJ	0	0	0	0	0	0	...	0.08	0.03	0.00	0.00	0.00	0.0	0.44	1.0(

22543 rows × 43 columns

Columns Modification

```
Columns = (['duration','protocol_type','service','flag','src_bytes','dst_bytes','land','wrong_fragment','urgent','hot',
            'num_failed_logins','logged_in','num_compromised','root_shell','su_attempted','num_root','num_file_creations',
            'num_shells','num_access_files','num_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','attack','level'])


Trained_Data.columns = Columns
Tested_Data.columns  = Columns
```

```
Trained_Data.head(10)
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_same_srv_rate	ds
0	0	udp	other	SF	146	0	0	0	0	0	...	0.00	
1	0	tcp	private	S0	0	0	0	0	0	0	...	0.10	
2	0	tcp	http	SF	232	8153	0	0	0	0	...	1.00	
3	0	tcp	http	SF	199	420	0	0	0	0	...	1.00	
4	0	tcp	private	REJ	0	0	0	0	0	0	...	0.07	
5	0	tcp	private	S0	0	0	0	0	0	0	...	0.04	
6	0	tcp	private	S0	0	0	0	0	0	0	...	0.06	
7	0	tcp	remote_job	S0	0	0	0	0	0	0	...	0.09	
8	0	tcp	private	S0	0	0	0	0	0	0	...	0.05	
9	0	tcp	private	REJ	0	0	0	0	0	0	...	0.05	

10 rows × 43 columns

```
Tested_Data.head(10)
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_same_srv_rate	dst
0	0	tcp	private	REJ	0	0	0	0	0	0	...	0.00	
1	2	tcp	ftp_data	SF	12983	0	0	0	0	0	...	0.61	
2	0	icmp	eco_i	SF	20	0	0	0	0	0	...	1.00	
3	1	tcp	telnet	RSTO	0	15	0	0	0	0	...	0.31	
4	0	tcp	http	SF	267	14515	0	0	0	0	...	1.00	
5	0	tcp	smtp	SF	1022	387	0	0	0	0	...	0.11	
6	0	tcp	telnet	SF	129	174	0	0	0	0	...	1.00	
7	0	tcp	http	SF	327	467	0	0	0	0	...	1.00	
8	0	tcp	ftp	SF	26	157	0	0	0	0	...	0.50	
9	0	tcp	telnet	SF	0	0	0	0	0	0	...	0.50	

10 rows × 43 columns

Data Description

```
Trained_Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 125972 entries, 0 to 125971
Data columns (total 43 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   duration                     125972 non-null  int64
 1   protocol_type                125972 non-null  object
 2   service                      125972 non-null  object
 3   flag                         125972 non-null  object
 4   src_bytes                    125972 non-null  int64
 5   dst_bytes                    125972 non-null  int64
 6   land                         125972 non-null  int64
 7   wrong_fragment               125972 non-null  int64
 8   urgent                       125972 non-null  int64
 9   hot                          125972 non-null  int64
 10  num_failed_logins            125972 non-null  int64
 11  logged_in                    125972 non-null  int64
 12  num_compromised              125972 non-null  int64
 13  root_shell                   125972 non-null  int64
 14  su_attempted                 125972 non-null  int64
 15  num_root                     125972 non-null  int64
 16  num_file_creations           125972 non-null  int64
 17  num_shells                   125972 non-null  int64
 18  num_access_files             125972 non-null  int64
 19  num_outbound_cmds            125972 non-null  int64
 20  is_host_login                125972 non-null  int64
 21  is_guest_login               125972 non-null  int64
```

```
 22  count                        125972 non-null  int64
 23  srv_count                    125972 non-null  int64
 24  serror_rate                  125972 non-null  float64
 25  srv_serror_rate              125972 non-null  float64
 26  rerror_rate                  125972 non-null  float64
 27  srv_rerror_rate              125972 non-null  float64
 28  same_srv_rate                125972 non-null  float64
 29  diff_srv_rate                125972 non-null  float64
 30  srv_diff_host_rate           125972 non-null  float64
 31  dst_host_count               125972 non-null  int64
 32  dst_host_srv_count           125972 non-null  int64
 33  dst_host_same_srv_rate       125972 non-null  float64
 34  dst_host_diff_srv_rate       125972 non-null  float64
 35  dst_host_same_src_port_rate  125972 non-null  float64
 36  dst_host_srv_diff_host_rate  125972 non-null  float64
 37  dst_host_serror_rate         125972 non-null  float64
 38  dst_host_srv_serror_rate     125972 non-null  float64
 39  dst_host_rerror_rate         125972 non-null  float64
 40  dst_host_srv_rerror_rate     125972 non-null  float64
 41  attack                       125972 non-null  object
 42  level                        125972 non-null  int64
dtypes: float64(15), int64(24), object(4)
memory usage: 41.3+ MB
```

```
Tested_Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22543 entries, 0 to 22542
Data columns (total 43 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   duration                     22543 non-null  int64
 1   protocol_type                22543 non-null  object
 2   service                      22543 non-null  object
 3   flag                         22543 non-null  object
 4   src_bytes                    22543 non-null  int64
 5   dst_bytes                    22543 non-null  int64
 6   land                         22543 non-null  int64
 7   wrong_fragment               22543 non-null  int64
 8   urgent                       22543 non-null  int64
 9   hot                          22543 non-null  int64
 10  num_failed_logins            22543 non-null  int64
 11  logged_in                    22543 non-null  int64
 12  num_compromised              22543 non-null  int64
 13  root_shell                   22543 non-null  int64
 14  su_attempted                 22543 non-null  int64
 15  num_root                     22543 non-null  int64
 16  num_file_creations           22543 non-null  int64
 17  num_shells                   22543 non-null  int64
 18  num_access_files             22543 non-null  int64
 19  num_outbound_cmds            22543 non-null  int64
 20  is_host_login                22543 non-null  int64
 21  is_guest_login               22543 non-null  int64
 22  count                        22543 non-null  int64
 23  srv_count                    22543 non-null  int64
 24  serror_rate                  22543 non-null  float64
 25  srv_serror_rate              22543 non-null  float64
 26  rerror_rate                  22543 non-null  float64
 27  srv_rerror_rate              22543 non-null  float64
 28  same_srv_rate                22543 non-null  float64
 29  diff_srv_rate                22543 non-null  float64
 30  srv_diff_host_rate           22543 non-null  float64
 31  dst_host_count               22543 non-null  int64
 32  dst_host_srv_count           22543 non-null  int64
 33  dst_host_same_srv_rate       22543 non-null  float64
 34  dst_host_diff_srv_rate       22543 non-null  float64
 35  dst_host_same_src_port_rate  22543 non-null  float64
 36  dst_host_srv_diff_host_rate  22543 non-null  float64
 37  dst_host_serror_rate         22543 non-null  float64
 38  dst_host_srv_serror_rate     22543 non-null  float64
 39  dst_host_rerror_rate         22543 non-null  float64
 40  dst_host_srv_rerror_rate     22543 non-null  float64
 41  attack                       22543 non-null  object
 42  level                        22543 non-null  int64
dtypes: float64(15), int64(24), object(4)
memory usage: 7.4+ MB
```

```
Trained_Data.describe()
```

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	l
count	125972.000000	1.259720e+05	1.259720e+05	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	1259
mean	287.146929	4.556710e+04	1.977927e+04	0.000198	0.022688	0.000111	0.204411	0.001222	
std	2604.525522	5.870354e+06	4.021285e+06	0.014086	0.253531	0.014366	2.149977	0.045239	
min	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	4.400000e+01	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	
75%	0.000000	2.760000e+02	5.160000e+02	0.000000	0.000000	0.000000	0.000000	0.000000	
max	42908.000000	1.379964e+09	1.309937e+09	1.000000	3.000000	3.000000	77.000000	5.000000	

8 rows × 39 columns

```
Tested_Data.describe()
```

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged
count	22543.000000	2.254300e+04	2.254300e+04	22543.000000	22543.000000	22543.000000	22543.000000	22543.000000	22543.000
mean	218.868784	1.039591e+04	2.056110e+03	0.000311	0.008428	0.000710	0.105399	0.021648	0.442
std	1407.207069	4.727969e+05	2.121976e+04	0.017619	0.142602	0.036474	0.928448	0.150331	0.496
min	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
25%	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
50%	0.000000	5.400000e+01	4.600000e+01	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
75%	0.000000	2.870000e+02	6.010000e+02	0.000000	0.000000	0.000000	0.000000	0.000000	1.000
max	57715.000000	6.282565e+07	1.345927e+06	1.000000	3.000000	3.000000	101.000000	4.000000	1.000

8 rows × 39 columns

Exploring Responses

```
Results = set(Trained_Data['attack'].values)
print(Results,end=" ")
```

```
{'neptune', 'ipsweep', 'buffer_overflow', 'multihop', 'loadmodule', 'back', 'spy', 'warezclient', 'normal', 'phf', 'land', 'smurf',
```

Classifying The Attack Results

```
# changing attack labels to their respective attack class
def change_label(df):
 df.attack.replace(['apache2','back','land','neptune','mailbomb','pod','processtable','smurf','teardrop','udpstorm','worm'],'Dos',inpla
 df.attack.replace(['ftp_write','guess_passwd','httptunnel','imap','multihop','named','phf','sendmail','snmpgetattack','snmpguess','spy
 df.attack.replace(['ipsweep','mscan','nmap','portsweep','saint','satan'],'Probe',inplace=True)
 df.attack.replace(['buffer_overflow','loadmodule','perl','ps','rootkit','sqlattack','xterm'],'U2R',inplace=True)


change_label(Trained_Data)
change_label(Tested_Data)


# label encoding (0,1,2,3,4) multi-class labels (Dos,normal,Probe,R2L,U2R)
LE = LabelEncoder()
attack_LE= LabelEncoder()
Trained_Data['attack_state'] = attack_LE.fit_transform(Trained_Data["attack"])
Tested_Data['attack_state'] = attack_LE.fit_transform(Tested_Data["attack"])


Trained_Data.head(10)
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_diff_srv_rate	ds
0	0	udp	other	SF	146	0	0	0	0	0	...	0.60	
1	0	tcp	private	S0	0	0	0	0	0	0	...	0.05	
2	0	tcp	http	SF	232	8153	0	0	0	0	...	0.00	
3	0	tcp	http	SF	199	420	0	0	0	0	...	0.00	
4	0	tcp	private	REJ	0	0	0	0	0	0	...	0.07	
5	0	tcp	private	S0	0	0	0	0	0	0	...	0.05	
6	0	tcp	private	S0	0	0	0	0	0	0	...	0.07	
7	0	tcp	remote_job	S0	0	0	0	0	0	0	...	0.05	
8	0	tcp	private	S0	0	0	0	0	0	0	...	0.06	
9	0	tcp	private	REJ	0	0	0	0	0	0	...	0.07	

10 rows × 44 columns

```
Trained_Data.attack.value_counts()
```

attack	count
normal	67342
Dos	45927
Probe	11656
R2L	995
U2R	52

dtype: int64

```
Tested_Data.attack.value_counts()
```

attack	count
normal	9711
Dos	7459
R2L	2885
Probe	2421
U2R	67

dtype: int64

```
Trained_Data.head(10)
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_diff_srv_rate	ds
0	0	udp	other	SF	146	0	0	0	0	0	...	0.60	
1	0	tcp	private	S0	0	0	0	0	0	0	...	0.05	
2	0	tcp	http	SF	232	8153	0	0	0	0	...	0.00	
3	0	tcp	http	SF	199	420	0	0	0	0	...	0.00	
4	0	tcp	private	REJ	0	0	0	0	0	0	...	0.07	
5	0	tcp	private	S0	0	0	0	0	0	0	...	0.05	
6	0	tcp	private	S0	0	0	0	0	0	0	...	0.07	
7	0	tcp	remote_job	S0	0	0	0	0	0	0	...	0.05	
8	0	tcp	private	S0	0	0	0	0	0	0	...	0.06	
9	0	tcp	private	REJ	0	0	0	0	0	0	...	0.07	

10 rows × 44 columns

```
Tested_Data.head(10)
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_diff_srv_rate	dst
0	0	tcp	private	REJ	0	0	0	0	0	0	...	0.06	
1	2	tcp	ftp_data	SF	12983	0	0	0	0	0	...	0.04	
2	0	icmp	eco_i	SF	20	0	0	0	0	0	...	0.00	
3	1	tcp	telnet	RSTO	0	15	0	0	0	0	...	0.17	
4	0	tcp	http	SF	267	14515	0	0	0	0	...	0.00	
5	0	tcp	smtp	SF	1022	387	0	0	0	0	...	0.72	
6	0	tcp	telnet	SF	129	174	0	0	0	0	...	0.00	
7	0	tcp	http	SF	327	467	0	0	0	0	...	0.00	
8	0	tcp	ftp	SF	26	157	0	0	0	0	...	0.08	
9	0	tcp	telnet	SF	0	0	0	0	0	0	...	0.01	

10 rows × 44 columns

Data preprocessing

Chechking for Missing Data

```
Trained_Data.isnull().sum()
```

	0
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
attack	0
level	0
attack_state	0

dtype: int64

```
Tested_Data.isnull().sum()
```

	0
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
attack	0
level	0
attack_state	0

dtype: int64

There is no missing data

Checking for Duplicates

```
Trained_Data.duplicated().sum()
```

```
9
```

```
Trained_Data.drop_duplicates(subset=None, keep="first", inplace=True)
Trained_Data.duplicated().sum()
```

```
0
```

```
Tested_Data.duplicated().sum()
```

```
3
```

```
Tested_Data.drop_duplicates(subset=None, keep="first", inplace=True)
Tested_Data.duplicated().sum()
```

```
0
```

There is no duplicated data

Data Encoding

```
Trained_Data = pd.get_dummies(Trained_Data,columns=['protocol_type','service','flag'],prefix="",prefix_sep="")
```

```
Tested_Data = pd.get_dummies(Tested_Data,columns=['protocol_type','service','flag'],prefix="",prefix_sep="")
```

```
LE = LabelEncoder()
attack_LE= LabelEncoder()
Trained_Data['attack'] = attack_LE.fit_transform(Trained_Data["attack"])
Tested_Data['attack'] = attack_LE.fit_transform(Tested_Data["attack"])
```

```
Trained_Data['attack']
```

	attack
0	4
1	0
2	4
3	4
4	0
...	...
125967	0
125968	4
125969	4
125970	0
125971	4

125963 rows × 1 columns

dtype: int64

Data Splitting

```
X_train = Trained_Data.drop('attack', axis = 1)
X_train = Trained_Data.drop('level', axis = 1)
X_train = Trained_Data.drop('attack_state', axis = 1)

X_test = Tested_Data.drop('attack', axis = 1)
X_test = Tested_Data.drop('level', axis = 1)
X_test = Tested_Data.drop('attack_state', axis = 1)
```

```
Y_train = Trained_Data['attack_state']
Y_test = Tested_Data['attack_state']


X_train_train,X_test_train ,Y_train_train,Y_test_train = train_test_split(X_train, Y_train, test_size= 0.25 , random_state=42)
X_train_test,X_test_test,Y_train_test,Y_test_test = train_test_split(X_test, Y_test, test_size= 0.25 , random_state=42)
```

Data Scaling

```
Ro_scaler = RobustScaler()
X_train_train = Ro_scaler.fit_transform(X_train_train)
X_test_train= Ro_scaler.transform(X_test_train)
X_train_test = Ro_scaler.fit_transform(X_train_test)
X_test_test= Ro_scaler.transform(X_test_test)


X_train_train.shape, Y_train_train.shape
```

```
((94472, 124), (94472,))
```

```
X_test_train.shape, Y_test_train.shape
```

```
((31491, 124), (31491,))
```

```
X_train_test.shape, Y_train_test.shape
```

```
((16905, 118), (16905,))
```

```
X_test_test.shape, Y_test_test.shape
```

```
((5635, 118), (5635,))
```

Working on Trained Data

(VIF)Variance Inflation Factor (VIF) measures the intercorrelation among independent variables in a multiple regression model..

```
A = sm.add_constant(X_train.astype(float))
Est1 = sm.GLM(Y_train.astype(float), A)
Est2 = Est1.fit()
Est2.summary()
```

Generalized Linear Model Regression Results

Dep. Variable:	attack_state	**No. Observations:**	125963
Model:	GLM	**Df Residuals:**	125843
Model Family:	Gaussian	**Df Model:**	119
Link Function:	Identity	**Scale:**	3.2303e-18
Method:	IRLS	**Log-Likelihood:**	2.7502e+06
Date:	Fri, 30 Aug 2024	**Deviance:**	8.0097e-16
Time:	01:32:12	**Pearson chi2:**	8.01e-16
No. Iterations:	3	**Pseudo R-squ. (CS):**	1.000
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.667e-11	8.59e-11	-0.194	0.846	-1.85e-10	1.52e-10
duration	1.724e-16	2.43e-15	0.071	0.944	-4.6e-15	4.94e-15
src_bytes	-8.883e-19	8.83e-19	-1.006	0.314	-2.62e-18	8.42e-19
dst_bytes	4.417e-18	1.26e-18	3.504	0.000	1.95e-18	6.89e-18
land	1.344e-10	3.66e-10	0.367	0.714	-5.84e-10	8.53e-10
wrong_fragment	9.476e-12	2.47e-11	0.384	0.701	-3.89e-11	5.79e-11
urgent	-3.059e-11	3.58e-10	-0.085	0.932	-7.32e-10	6.71e-10
hot	1.224e-11	4.71e-12	2.601	0.009	3.02e-12	2.15e-11
num_failed_logins	2.275e-10	1.16e-10	1.967	0.049	7.57e-13	4.54e-10
logged_in	-9.304e-12	4.06e-11	-0.229	0.819	-8.88e-11	7.02e-11
num_compromised	-1.142e-11	6.37e-12	-1.792	0.073	-2.39e-11	1.07e-12
root_shell	-3.13e-10	1.78e-10	-1.760	0.078	-6.62e-10	3.56e-11
su_attempted	1.354e-10	1.83e-10	0.740	0.459	-2.23e-10	4.94e-10
num_root	1.147e-11	6.35e-12	1.807	0.071	-9.68e-13	2.39e-11
num_file_creations	2.177e-12	1.09e-11	0.201	0.841	-1.91e-11	2.35e-11
num_shells	-2.819e-10	2.31e-10	-1.218	0.223	-7.36e-10	1.72e-10
num_access_files	-2.06e-10	7.02e-11	-2.935	0.003	-3.44e-10	-6.84e-11
num_outbound_cmds	-1.096e-17	2.67e-23	-4.1e+05	0.000	-1.1e-17	-1.1e-17
is_host_login	1.574e-11	1.8e-09	0.009	0.993	-3.51e-09	3.54e-09
is_guest_login	-3.487e-10	1.32e-10	-2.641	0.008	-6.07e-10	-8.99e-11
count	-8.78e-14	1.07e-13	-0.821	0.412	-2.97e-13	1.22e-13
srv_count	6.702e-14	1.59e-13	0.421	0.674	-2.45e-13	3.79e-13
serror_rate	-4.121e-10	1.27e-10	-3.235	0.001	-6.62e-10	-1.62e-10
srv_serror_rate	2.26e-10	1.39e-10	1.625	0.104	-4.65e-11	4.99e-10
rerror_rate	-3.357e-10	1.36e-10	-2.465	0.014	-6.03e-10	-6.87e-11
srv_rerror_rate	2.371e-10	1.59e-10	1.488	0.137	-7.53e-11	5.49e-10
same_srv_rate	-4.056e-11	4.74e-11	-0.856	0.392	-1.33e-10	5.23e-11
diff_srv_rate	-2.192e-11	4.42e-11	-0.496	0.620	-1.09e-10	6.47e-11
srv_diff_host_rate	1.567e-11	2.54e-11	0.618	0.537	-3.41e-11	6.54e-11
dst_host_count	1.328e-14	7.98e-14	0.166	0.868	-1.43e-13	1.7e-13
dst_host_srv_count	-3.832e-14	1.44e-13	-0.267	0.790	-3.2e-13	2.43e-13
dst_host_same_srv_rate	4.97e-11	4.31e-11	1.153	0.249	-3.48e-11	1.34e-10
dst_host_diff_srv_rate	9.967e-11	5e-11	1.992	0.046	1.62e-12	1.98e-10
dst_host_same_src_port_rate	-9.888e-12	3.14e-11	-0.315	0.752	-7.13e-11	5.16e-11
dst_host_srv_diff_host_rate	-4.658e-11	7.01e-11	-0.664	0.507	-1.84e-10	9.09e-11
dst_host_serror_rate	2.803e-10	7.87e-11	3.559	0.000	1.26e-10	4.35e-10

Data Modeling

Evaluating Function

level	-7.982e-13	3.21e-12	-0.249	0.803	-7.09e-12	5.49e-12

```
def Evaluate(Model_Name, Model_Abb, X_test, Y_test):

    Pred_Value = Model_Abb.predict(X_test)

    Accuracy = metrics.accuracy_score(Y_test, Pred_Value)
    Precision = metrics.precision_score(Y_test, Pred_Value, average='weighted')
    Recall = metrics.recall_score(Y_test, Pred_Value, average='weighted')
    F1_score = metrics.f1_score(Y_test, Pred_Value, average='weighted')

    print('-----------------------------------------------------\n')
    print('The {} Model Accuracy   = {}\n'.format(Model_Name, np.round(Accuracy,2)))
    print('The {} Model Precision  = {}\n'.format(Model_Name, np.round(Precision,2)))
    print('The {} Model Recall     = {}\n'.format(Model_Name, np.round(Recall,2)))
    print('The {} Model F1 Score   = {}\n'.format(Model_Name, np.round(F1_score,2)))
    print('-----------------------------------------------------\n')

    Confusion_Matrix = metrics.confusion_matrix(Y_test, Pred_Value)
    plot_confusion_matrix(Confusion_Matrix, class_names=['normal','Dos','R2L','Probe','U2R'], figsize=(5.55,5), colorbar= "blue")

    # Compute ROC curve and ROC AUC for each class
    fpr = dict()
    tpr = dict()
    roc_auc = dict()
```

```
    n_classes = len(np.unique(Y_test))
    y_score = Model_Abb.predict_proba(X_test)
    y_onehot_test = pd.get_dummies(Y_test)

    for i in range(n_classes):
        fpr[i], tpr[i], _ = metrics.roc_curve(y_onehot_test.iloc[:, i], y_score[:, i])
        roc_auc[i] = metrics.auc(fpr[i], tpr[i])

    # Compute micro-average ROC curve and ROC AUC
    fpr["micro"], tpr["micro"], _ = metrics.roc_curve(y_onehot_test.values.ravel(), y_score.ravel())
    roc_auc["micro"] = metrics.auc(fpr["micro"], tpr["micro"])

    # Compute macro-average ROC curve and ROC AUC
    all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))
    mean_tpr = np.zeros_like(all_fpr)
    for i in range(n_classes):
        mean_tpr += np.interp(all_fpr, fpr[i], tpr[i])
    mean_tpr /= n_classes
    fpr["macro"] = all_fpr
    tpr["macro"] = mean_tpr
    roc_auc["macro"] = metrics.auc(fpr["macro"], tpr["macro"])

    # Plot ROC curve
    fig, ax = plt.subplots(figsize=(6, 6))
    plt.plot(fpr["micro"], tpr["micro"], label=f"micro-average ROC curve (AUC = {roc_auc['micro']:.2f})", color="deeppink", linestyle="
    plt.plot(fpr["macro"], tpr["macro"], label=f"macro-average ROC curve (AUC = {roc_auc['macro']:.2f})", color="navy", linestyle=":", l
    colors = cycle(["aqua", "darkorange", "cornflowerblue"])
    for class_id, color in zip(range(n_classes), colors):
        plt.plot(fpr[class_id], tpr[class_id], color=color, lw=2, label=f"ROC curve for class {class_id} (AUC = {roc_auc[class_id]:.2f}

    plt.plot([0, 1], [0, 1], 'k--', lw=2)
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(Model_Name)
    plt.legend(loc="lower right")
    plt.show()
```

Grid Search Function

```
def GridSearch(Model_Abb, Parameters, X_train, Y_train):
    Grid = GridSearchCV(estimator=Model_Abb, param_grid= Parameters, cv = 3, n_jobs=-1)
    Grid_Result = Grid.fit(X_train, Y_train)
    Model_Name = Grid_Result.best_estimator_

    return (Model_Name)
```

1. Logistic Regression

```
LR= LogisticRegression()
LR.fit(X_train_train , Y_train_train)
```

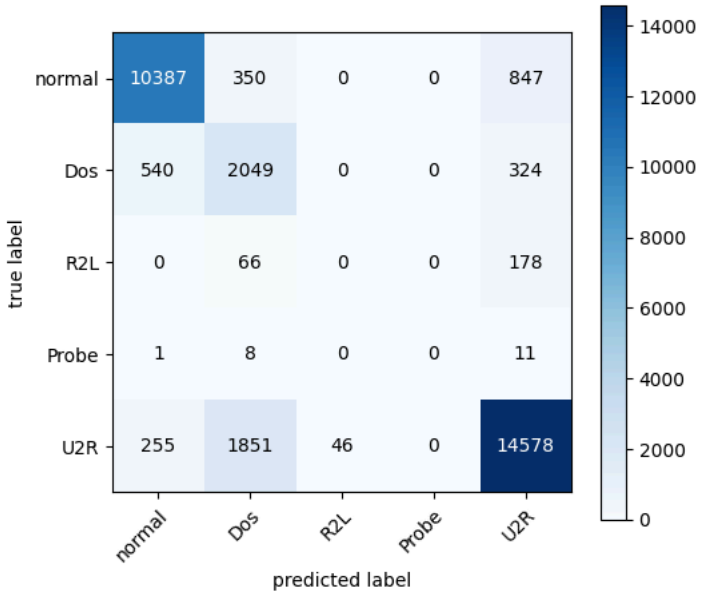
```
LogisticRegression()
```

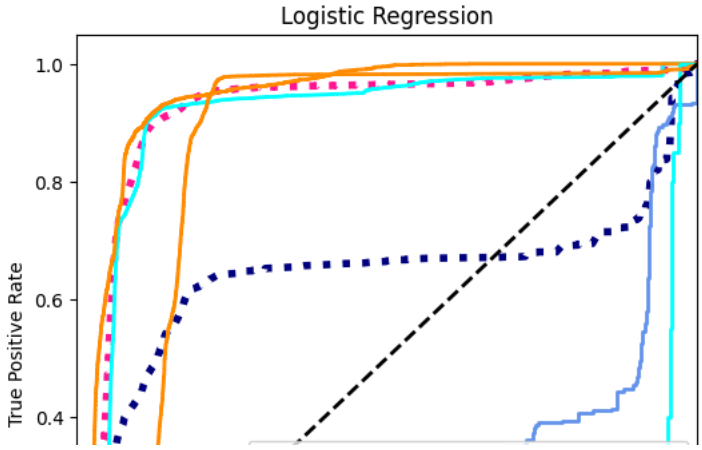
```
LR.score(X_train_train, Y_train_train), LR.score(X_test_train, Y_test_train)
```

```
(0.8589740875603353, 0.8578323965577467)
```

```
Evaluate('Logistic Regression', LR, X_test_train, Y_test_train)
```

```
--------------------------------------------------

The Logistic Regression Model Accuracy   = 0.86

The Logistic Regression Model Precision  = 0.87

The Logistic Regression Model Recall     = 0.86

The Logistic Regression Model F1 Score   = 0.86

--------------------------------------------------
```





2. Decision Tree Classifier


```
DT =DecisionTreeClassifier(max_features=6, max_depth=4)
DT.fit(X_train_train, Y_train_train)
```

```
DecisionTreeClassifier(max_depth=4, max_features=6)
```

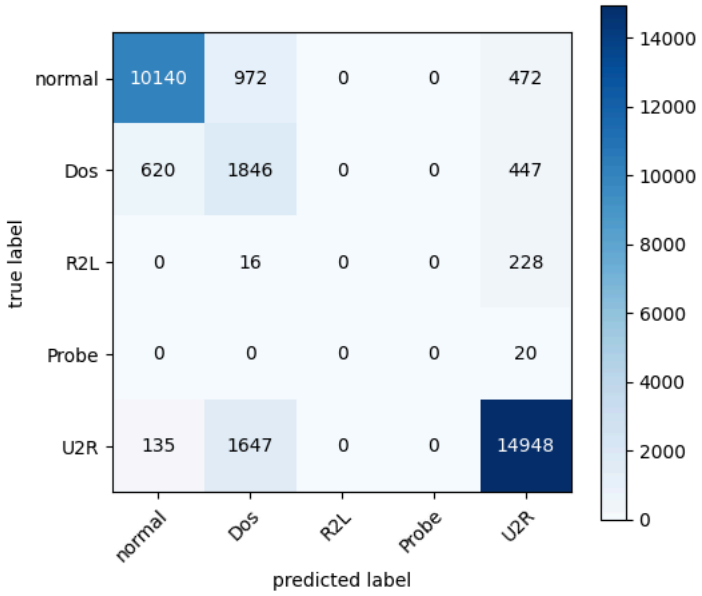
False Positive Rate

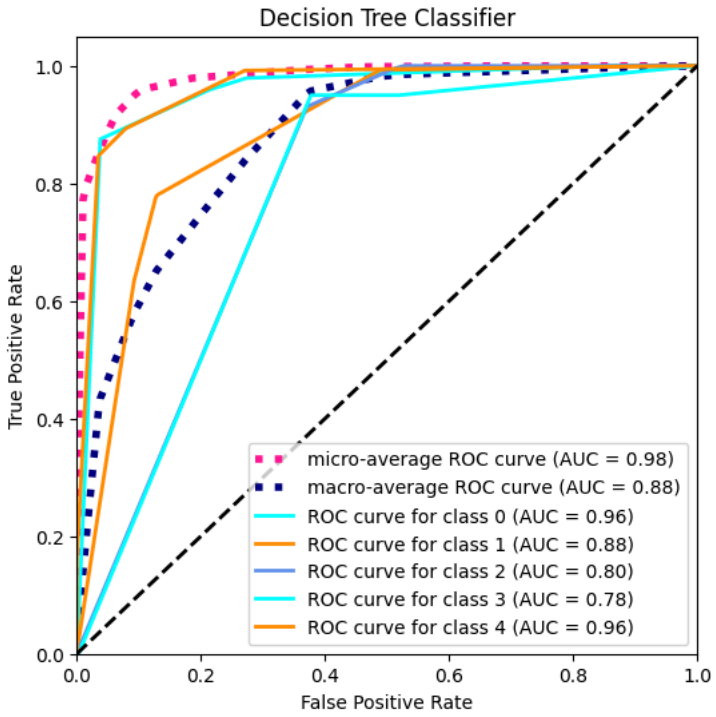
```
DT.score(X_train_train, Y_train_train), DT.score(X_test_train, Y_test_train)
```

```
(0.8563489711237192, 0.8552919881871011)
```

```
Evaluate('Decision Tree Classifier', DT, X_test_train, Y_test_train)
```

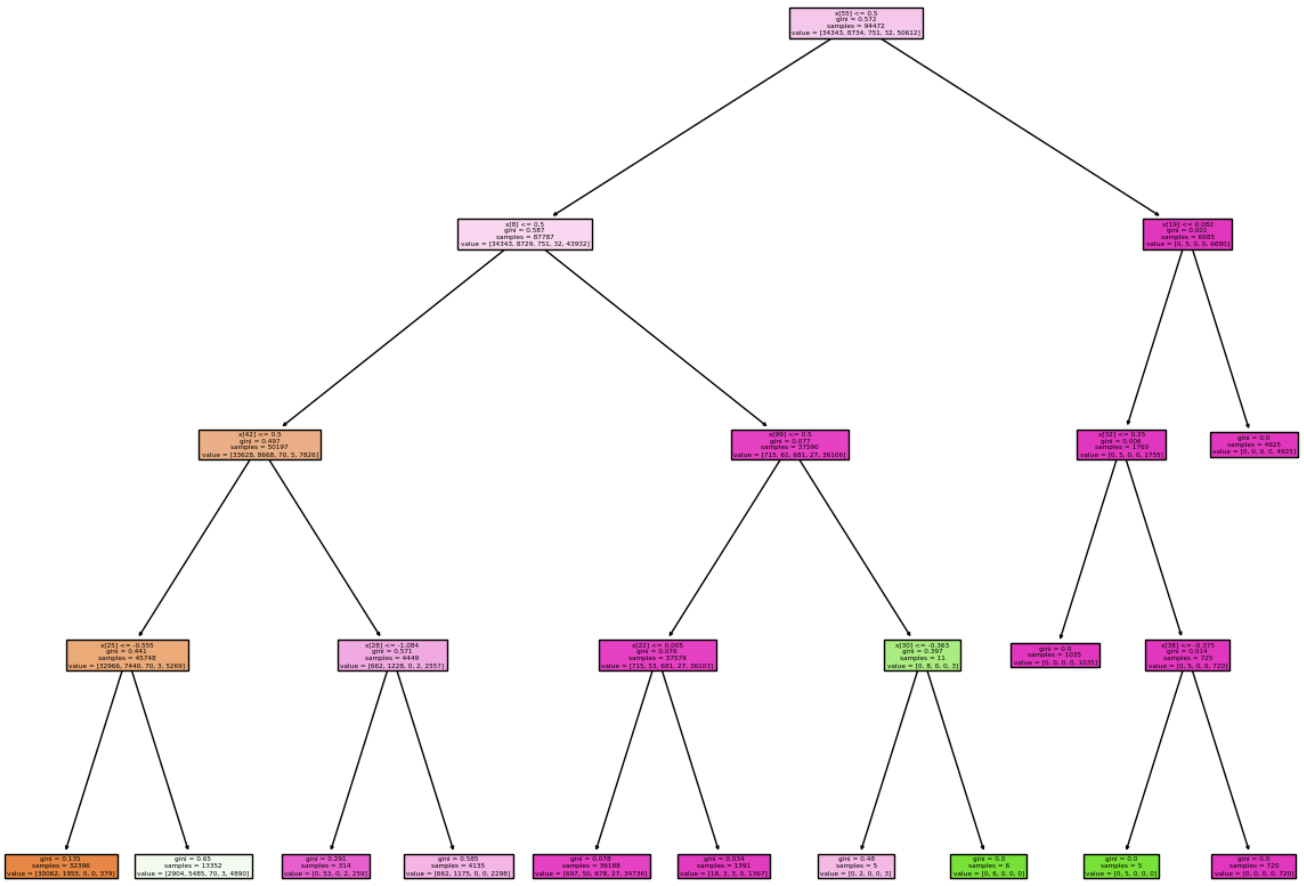
```
--------------------------------------------------

The Decision Tree Classifier Model Accuracy   = 0.86

The Decision Tree Classifier Model Precision  = 0.87

The Decision Tree Classifier Model Recall     = 0.86

The Decision Tree Classifier Model F1 Score   = 0.86

--------------------------------------------------
```






```
fig = plt.figure(figsize=(15,12))
tree.plot_tree(DT, filled=True)
```

```
[Text(0.65, 0.9, 'x[55] <= 0.5\ngini = 0.572\nsamples = 94472\nvalue = [34343, 8734, 751, 32, 50612]'),
 Text(0.4, 0.7, 'x[8] <= 0.5\ngini = 0.587\nsamples = 87787\nvalue = [34343, 8729, 751, 32, 43932]'),
 Text(0.2, 0.5, 'x[42] <= 0.5\ngini = 0.497\nsamples = 50197\nvalue = [33628, 8668, 70, 5, 7826]'),
 Text(0.1, 0.3, 'x[25] <= -0.555\ngini = 0.441\nsamples = 45748\nvalue = [32966, 7440, 70, 3, 5269]'),
 Text(0.05, 0.1, 'gini = 0.135\nsamples = 32396\nvalue = [30062, 1955, 0, 0, 379]'),
 Text(0.15, 0.1, 'gini = 0.65\nsamples = 13352\nvalue = [2904, 5485, 70, 3, 4890]'),
 Text(0.3, 0.3, 'x[28] <= -1.084\ngini = 0.571\nsamples = 4449\nvalue = [662, 1228, 0, 2, 2557]'),
 Text(0.25, 0.1, 'gini = 0.291\nsamples = 314\nvalue = [0, 53, 0, 2, 259]'),
 Text(0.35, 0.1, 'gini = 0.585\nsamples = 4135\nvalue = [662, 1175, 0, 0, 2298]'),
 Text(0.6, 0.5, 'x[99] <= 0.5\ngini = 0.077\nsamples = 37590\nvalue = [715, 61, 681, 27, 36106]'),
 Text(0.5, 0.3, 'x[22] <= 0.005\ngini = 0.076\nsamples = 37579\nvalue = [715, 53, 681, 27, 36103]'),
 Text(0.45, 0.1, 'gini = 0.078\nsamples = 36188\nvalue = [697, 50, 678, 27, 34736]'),
 Text(0.55, 0.1, 'gini = 0.034\nsamples = 1391\nvalue = [18, 3, 3, 0, 1367]'),
 Text(0.7, 0.3, 'x[30] <= -0.363\ngini = 0.397\nsamples = 11\nvalue = [0, 8, 0, 0, 3]'),
 Text(0.65, 0.1, 'gini = 0.48\nsamples = 5\nvalue = [0, 2, 0, 0, 3]'),
 Text(0.75, 0.1, 'gini = 0.0\nsamples = 6\nvalue = [0, 6, 0, 0, 0]'),
 Text(0.9, 0.7, 'x[19] <= 0.082\ngini = 0.001\nsamples = 6685\nvalue = [0, 5, 0, 0, 6680]'),
 Text(0.85, 0.5, 'x[32] <= 0.25\ngini = 0.006\nsamples = 1760\nvalue = [0, 5, 0, 0, 1755]'),
 Text(0.8, 0.3, 'gini = 0.0\nsamples = 1035\nvalue = [0, 0, 0, 0, 1035]'),
 Text(0.9, 0.3, 'x[38] <= -0.375\ngini = 0.014\nsamples = 725\nvalue = [0, 5, 0, 0, 720]'),
 Text(0.85, 0.1, 'gini = 0.0\nsamples = 5\nvalue = [0, 5, 0, 0, 0]'),
 Text(0.95, 0.1, 'gini = 0.0\nsamples = 720\nvalue = [0, 0, 0, 0, 720]'),
 Text(0.95, 0.5, 'gini = 0.0\nsamples = 4925\nvalue = [0, 0, 0, 0, 4925]')]
```



3. Random Forest Classifier

```
max_depth= [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

Parameters={ 'max_depth': max_depth}
```

```
RF= RandomForestClassifier()
GridSearch(RF, Parameters, X_train_train, Y_train_train)
```

```
RandomForestClassifier
RandomForestClassifier(max_depth=11)
```

```
RF.fit(X_train_train, Y_train_train)
```

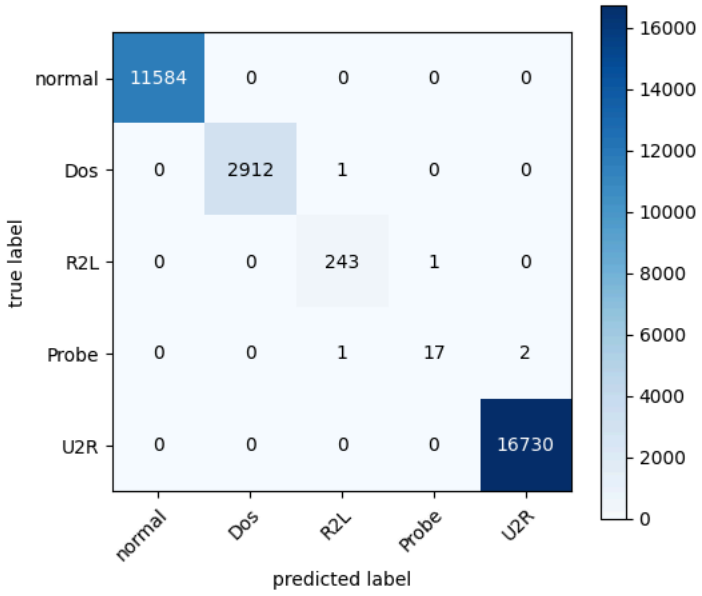
```
RandomForestClassifier
RandomForestClassifier()
```

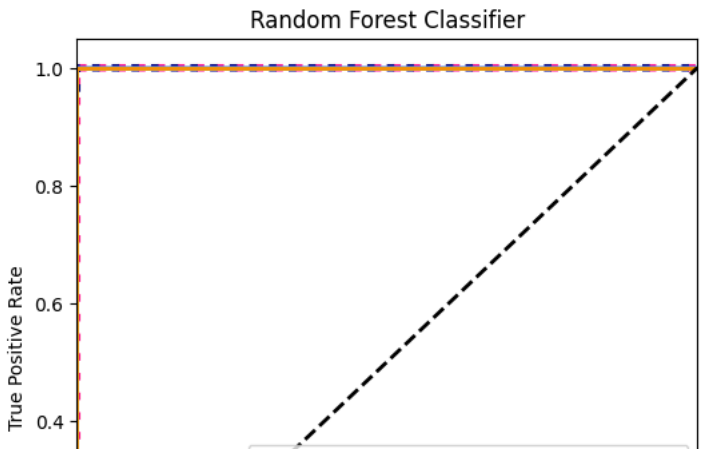
```
RF.score(X_train_train, Y_train_train), RF.score(X_test_train, Y_test_train)
```

```
(1.0, 0.9998412244768347)
```

```
Evaluate('Random Forest Classifier', RF, X_test_train, Y_test_train)
```

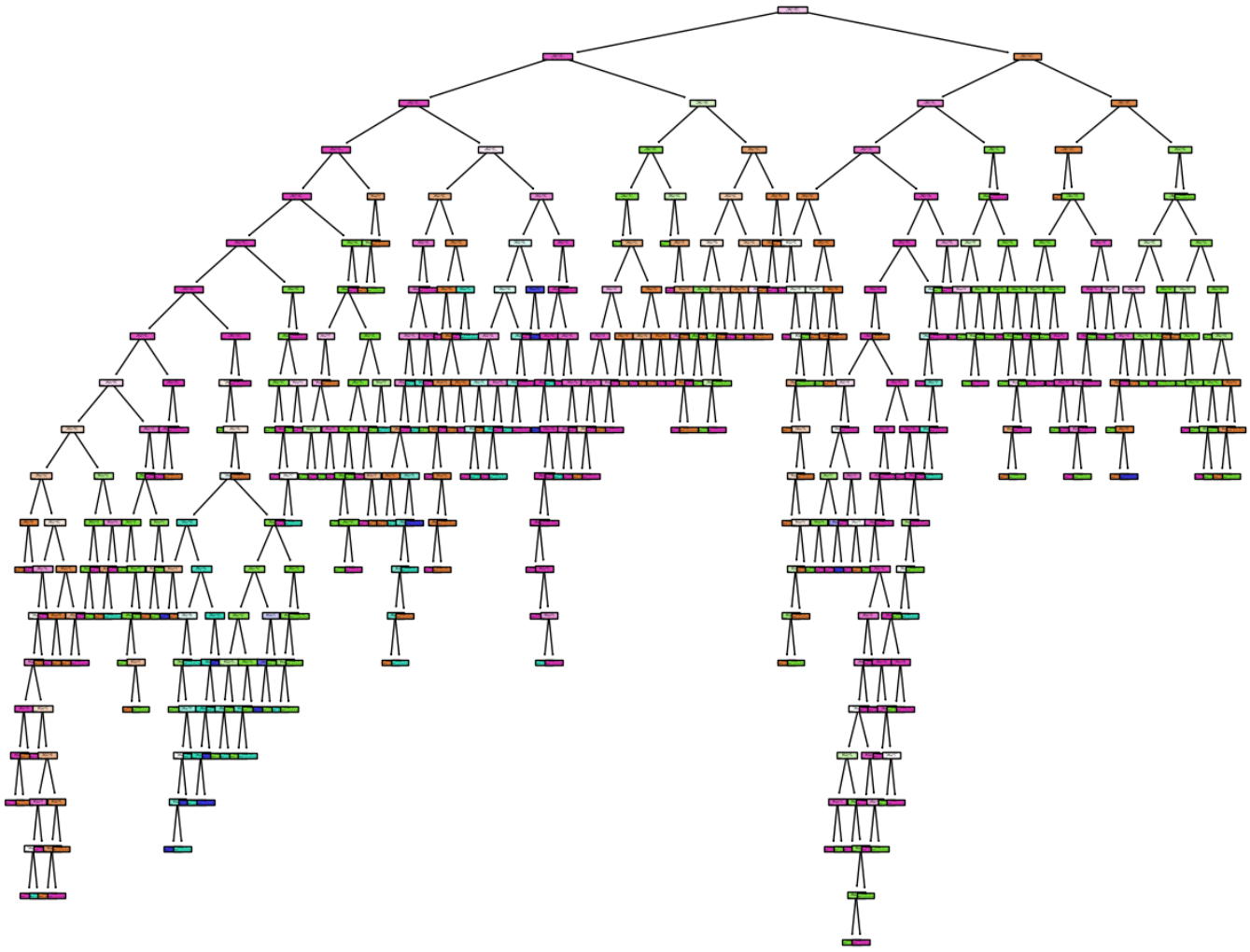
```
--------------------------------------------------
The Random Forest Classifier Model Accuracy   = 1.0
The Random Forest Classifier Model Precision  = 1.0
The Random Forest Classifier Model Recall     = 1.0
The Random Forest Classifier Model F1 Score   = 1.0
--------------------------------------------------
```






```
# prompt: generate a diagram for the random forest implementation archecture in the above model

from sklearn import tree
import matplotlib.pyplot as plt
```

```
# Assuming 'RF' is your trained RandomForestClassifier
estimator = RF.estimators_[0]  # Extract one of the decision trees from the forest

fig = plt.figure(figsize=(15, 12))
tree.plot_tree(estimator, filled=True)
plt.show()
```



4. KNN-Model

```
KNN= KNeighborsClassifier(n_neighbors=6)
KNN.fit(X_train_train, Y_train_train)
```

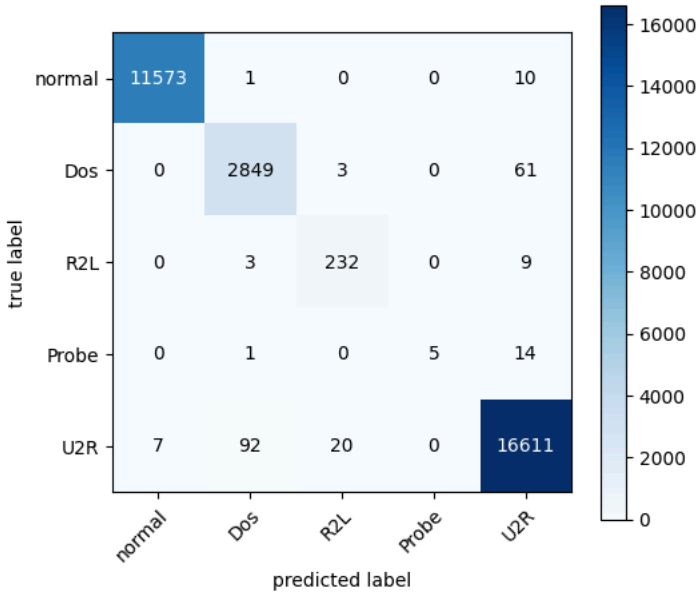
```
KNeighborsClassifier(n_neighbors=6)
```

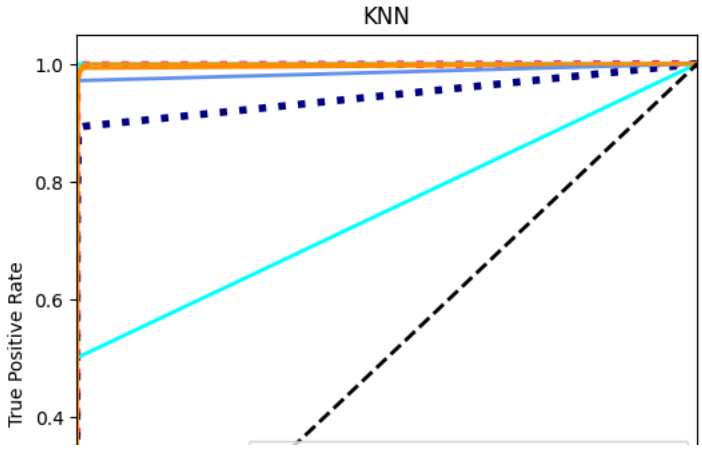
```
KNN.score(X_train_train, Y_train_train), KNN.score(X_test_train, Y_test_train)
```

```
(0.9949402997713608, 0.9929821218760916)
```

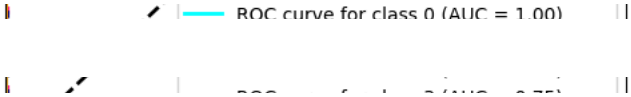
```
Evaluate('KNN', KNN, X_test_train, Y_test_train)
```

```
--------------------------------------------------

The KNN Model Accuracy   = 0.99

The KNN Model Precision  = 0.99

The KNN Model Recall     = 0.99

The KNN Model F1 Score   = 0.99

--------------------------------------------------
```



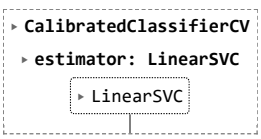


5. SVM Classifier



1st Kerne

```
# Create LinearSVC classifier without probability estimation
LinearSVC_classifier = LinearSVC()

# Wrap LinearSVC inside CalibratedClassifierCV with method='sigmoid' for Platt scaling
Platt_SVC = CalibratedClassifierCV(LinearSVC_classifier, method='sigmoid')

# Fit the classifier
Platt_SVC.fit(X_train_train, Y_train_train)
```

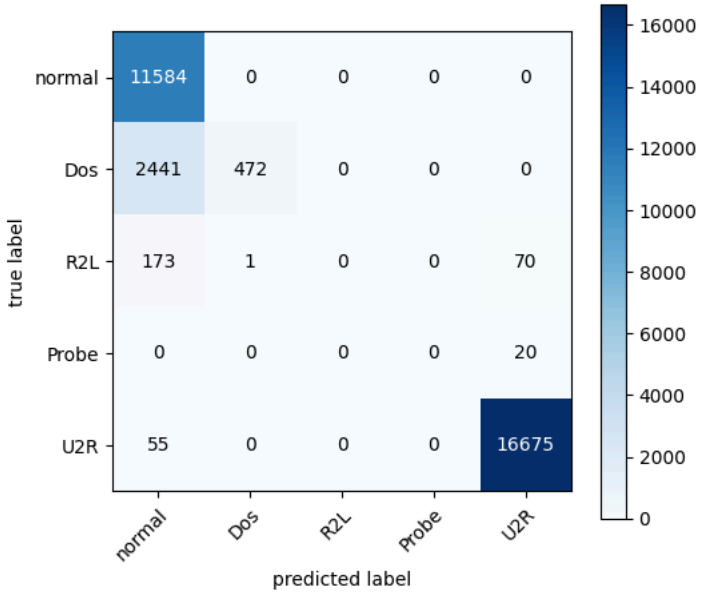


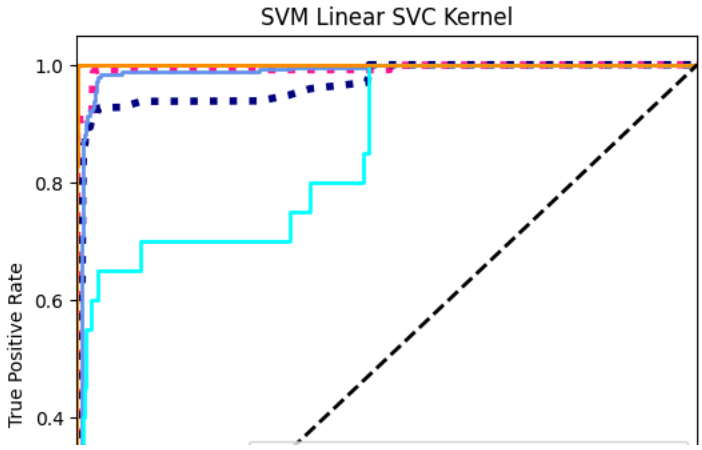

```
Platt_SVC.score(X_train_train, Y_train_train), Platt_SVC.score(X_test_train, Y_test_train)
```

```
(0.9123761537810144, 0.9123559112127274)
```

```
Evaluate('SVM Linear SVC Kernel', Platt_SVC,X_test_train, Y_test_train)
```

```
--------------------------------------------------

The SVM Linear SVC Kernel Model Accuracy   = 0.91

The SVM Linear SVC Kernel Model Precision  = 0.92

The SVM Linear SVC Kernel Model Recall     = 0.91

The SVM Linear SVC Kernel Model F1 Score   = 0.88

--------------------------------------------------
```





MLP MODEL

ROC curve for class 0 (AUC = 1.00)

```
# prompt: give me the code for the mlp model  for the above data

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout

# Define the model
model = Sequential()
model.add(Dense(128, activation='relu', input_shape=(X_train_train.shape[1],)))
model.add(Dropout(0.2))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(5, activation='softmax'))  # 5 output classes

# Compile the model
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
model.fit(X_train_train, Y_train_train, epochs=10, batch_size=32, validation_data=(X_test_train, Y_test_train))

# Evaluate the model
loss, accuracy = model.evaluate(X_test_train, Y_test_train)
print("Loss:", loss)
```

```
print("Accuracy:", accuracy)
```

```
Epoch 1/10
2953/2953 ━━━━━━━━━━━━━━━━━━━━ 11s 3ms/step - accuracy: 0.9247 - loss: 11.7564 - val_accuracy: 0.9785 - val_loss: 17.4485
Epoch 2/10
2953/2953 ━━━━━━━━━━━━━━━━━━━━ 7s 2ms/step - accuracy: 0.9808 - loss: 19.4482 - val_accuracy: 0.9772 - val_loss: 1.1395
Epoch 3/10
2953/2953 ━━━━━━━━━━━━━━━━━━━━ 12s 3ms/step - accuracy: 0.9821 - loss: 2.4106 - val_accuracy: 0.9896 - val_loss: 15.8995
Epoch 4/10
2953/2953 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - accuracy: 0.9864 - loss: 1.1530 - val_accuracy: 0.9900 - val_loss: 6.2870
Epoch 5/10
2953/2953 ━━━━━━━━━━━━━━━━━━━━ 8s 2ms/step - accuracy: 0.9865 - loss: 3.2645 - val_accuracy: 0.9858 - val_loss: 6.6217
Epoch 6/10
2953/2953 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.9881 - loss: 2.3588 - val_accuracy: 0.9896 - val_loss: 73.8383
Epoch 7/10
2953/2953 ━━━━━━━━━━━━━━━━━━━━ 7s 2ms/step - accuracy: 0.9882 - loss: 46.1470 - val_accuracy: 0.9899 - val_loss: 48.7512
Epoch 8/10
2953/2953 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.9866 - loss: 84.6154 - val_accuracy: 0.9869 - val_loss: 1.7253
Epoch 9/10
2953/2953 ━━━━━━━━━━━━━━━━━━━━ 8s 2ms/step - accuracy: 0.9857 - loss: 9.2379 - val_accuracy: 0.9900 - val_loss: 116.1652
Epoch 10/10
2953/2953 ━━━━━━━━━━━━━━━━━━━━ 10s 2ms/step - accuracy: 0.9871 - loss: 41.8824 - val_accuracy: 0.9896 - val_loss: 3.0144
985/985 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.9897 - loss: 2.6379
Loss: 3.014374256134033
Accuracy: 0.9895843267440796
```

```
# prompt: generate the validation code alsofor the mlp

# ... (preceding code)
from sklearn.metrics import classification_report, confusion_matrix # Import necessary functions

# Evaluate the model on the validation set
loss, accuracy = model.evaluate(X_test_train, Y_test_train, verbose=0)
print("Validation Loss:", loss)
print("Validation Accuracy:", accuracy)

# Make predictions on the validation set
y_pred_probs = model.predict(X_test_train)
y_pred = np.argmax(y_pred_probs, axis=1)

# Generate confusion matrix and classification report for validation set
conf_matrix = confusion_matrix(Y_test_train, y_pred)
class_report = classification_report(Y_test_train, y_pred)

print("Validation Confusion Matrix:\n", conf_matrix)
print("\nValidation Classification Report:\n", class_report)
```

```
Validation Loss: 3.014374256134033
Validation Accuracy: 0.9895843267440796
985/985 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
Validation Confusion Matrix:
 [[11582     2     0     0     0]
 [    0  2646     1     0   266]
 [    0     0   208     0    36]
 [    0     1     1     0    18]
 [    1     1     1     0 16727]]

Validation Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00     11584
           1       1.00      0.91      0.95      2913
           2       0.99      0.85      0.91       244
           3       0.00      0.00      0.00        20
           4       0.98      1.00      0.99     16730

    accuracy                           0.99     31491
   macro avg       0.79      0.75      0.77     31491
weighted avg       0.99      0.99      0.99     31491
```

6. Gradient Boosting Classifier

```
# Instantiate and fit the Gradient Boosting Classifier
GB = GradientBoostingClassifier()
GB.fit(X_train_train, Y_train_train)
```

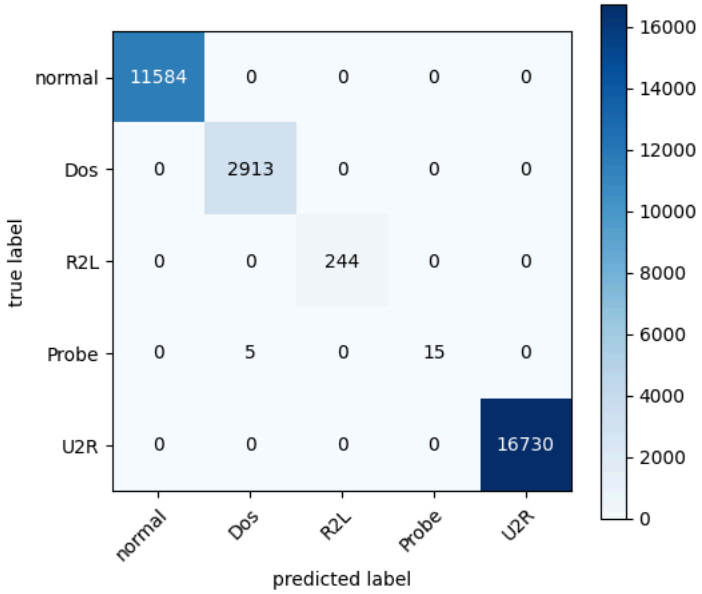
```
▾ GradientBoostingClassifier
GradientBoostingClassifier()
```

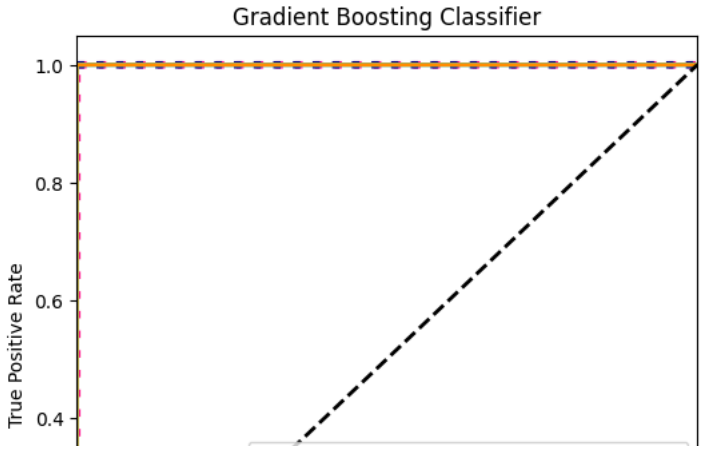
```
# Print the training and testing scores
print("Gradient Boosting Classifier Training Score:", GB.score(X_train_train, Y_train_train))
print("Gradient Boosting Classifier Testing Score:", GB.score(X_test_train, Y_test_train))
```

```
Gradient Boosting Classifier Training Score: 1.0
Gradient Boosting Classifier Testing Score: 0.9998412244768347
```

```
# Evaluate the Gradient Boosting Classifier
Evaluate('Gradient Boosting Classifier', GB, X_test_train, Y_test_train)
```

```
-----------------------------------------------------

The Gradient Boosting Classifier Model Accuracy   = 1.0

The Gradient Boosting Classifier Model Precision  = 1.0

The Gradient Boosting Classifier Model Recall     = 1.0

The Gradient Boosting Classifier Model F1 Score   = 1.0

-----------------------------------------------------
```

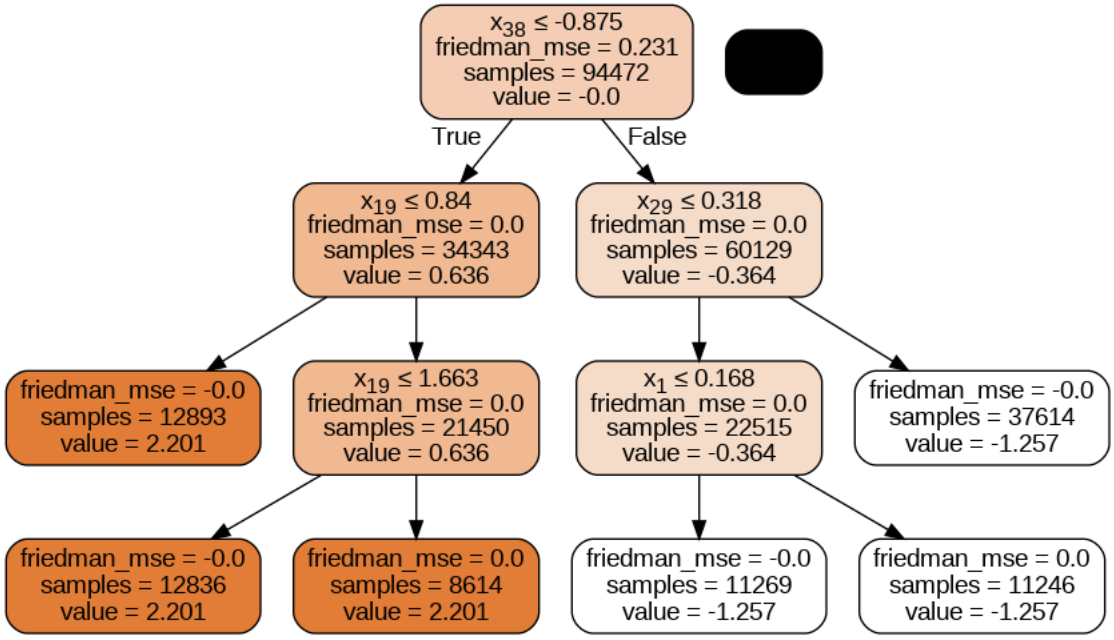





```
# prompt: generate a diagram for the gradient boosting implementation archecture in the above model

!pip install graphviz
!pip install pydotplus

from sklearn.tree import export_graphviz
import pydotplus
from IPython.display import Image

# Assuming 'GB' is your trained GradientBoostingClassifier
estimator = GB.estimators_[0][0]  # Extract one of the decision trees from the first iteration

dot_data = export_graphviz(estimator,
                           filled=True,
                           rounded=True,
                           special_characters=True)

graph = pydotplus.graph_from_dot_data(dot_data)
```

```
Image(graph.create_png())
```

```
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (0.20.3)
Requirement already satisfied: pydotplus in /usr/local/lib/python3.10/dist-packages (2.0.2)
Requirement already satisfied: pyparsing>=2.0.1 in /usr/local/lib/python3.10/dist-packages (from pydotplus) (3.1.4)
```



7. Extreme Gradient Boosting (XGBoost) Classifier

```
# Instantiate and fit the XGBoost Classifier
XGB = xgb.XGBClassifier(n_estimators=100, random_state=42)
XGB.fit(X_train_train, Y_train_train)
```

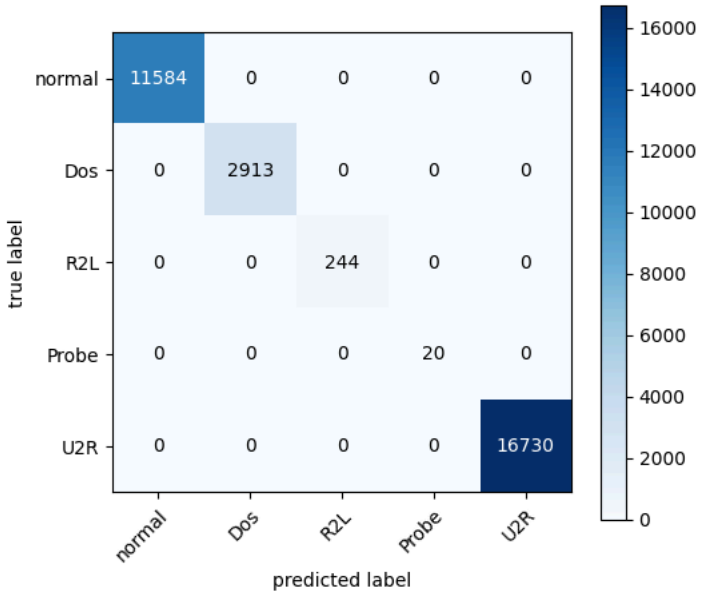
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=100, n_jobs=None,
              num_parallel_tree=None, objective='multi:softprob', ...)
```

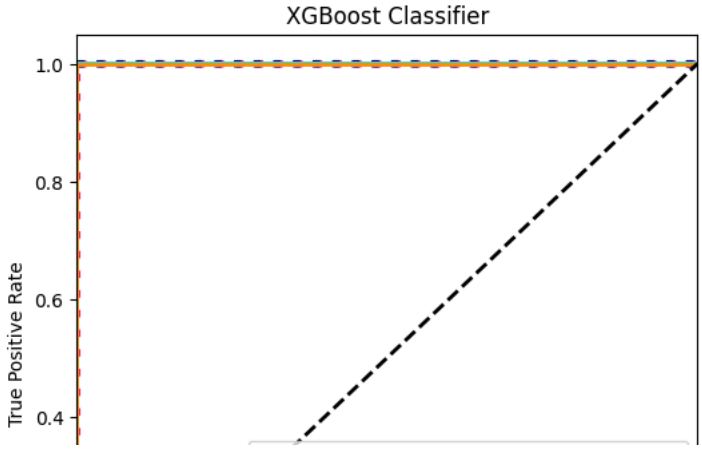
```
# Print the training and testing scores
print("XGBoost Classifier Training Score:", XGB.score(X_train_train, Y_train_train))
print("XGBoost Classifier Testing Score:", XGB.score(X_test_train, Y_test_train))
```

```
XGBoost Classifier Training Score: 1.0
XGBoost Classifier Testing Score: 1.0
```

```
# Evaluate the XGBoost Classifier
Evaluate('XGBoost Classifier', XGB, X_test_train, Y_test_train)
```

```
--------------------------------------------------

The XGBoost Classifier Model Accuracy   = 1.0

The XGBoost Classifier Model Precision  = 1.0

The XGBoost Classifier Model Recall     = 1.0

The XGBoost Classifier Model F1 Score   = 1.0

--------------------------------------------------
```





8. Light Gradient Boosting Machine (LGBM) Classifie


```
# Instantiate and fit the LGBM Classifier
LGBM = lgb.LGBMClassifier()
LGBM.fit(X_train_train, Y_train_train)
```

```
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.055062 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 3327
[LightGBM] [Info] Number of data points in the train set: 94472, number of used features: 112
[LightGBM] [Info] Start training from score -1.011905
[LightGBM] [Info] Start training from score -2.381080
[LightGBM] [Info] Start training from score -4.834653
[LightGBM] [Info] Start training from score -7.990323
[LightGBM] [Info] Start training from score -0.624115
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
```

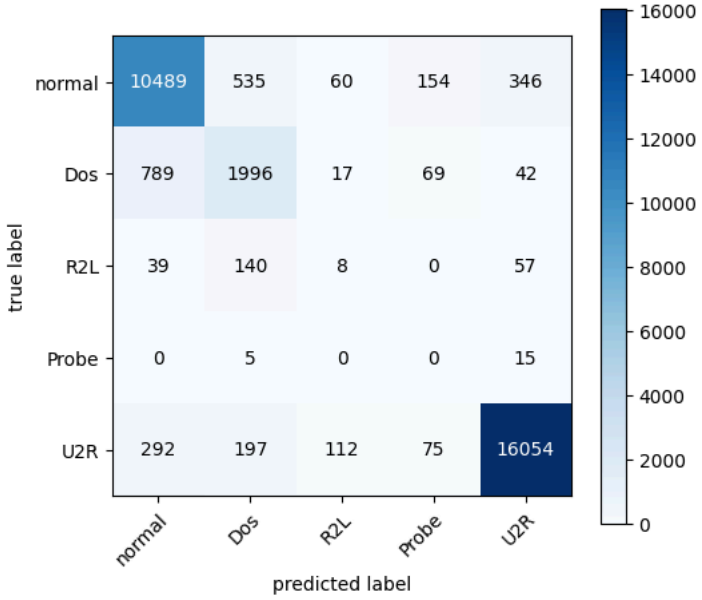
```
▾ LGBMClassifier
LGBMClassifier()
```

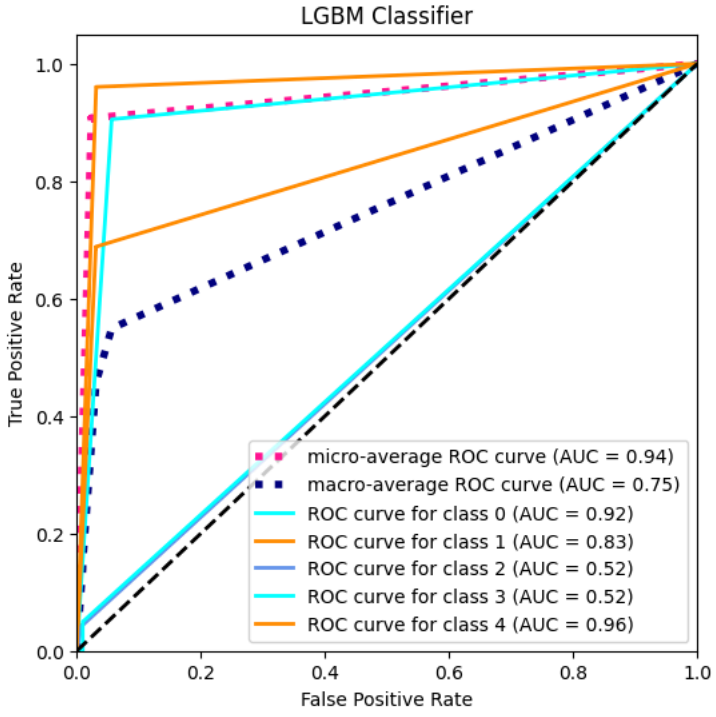
```
# Print the training and testing scores
print("LGBM Classifier Training Score:", LGBM.score(X_train_train, Y_train_train))
print("LGBM Classifier Testing Score:", LGBM.score(X_test_train, Y_test_train))
```

```
LGBM Classifier Training Score: 0.909909814548226
LGBM Classifier Testing Score: 0.9065129719602426
```

```
# Evaluate the LGBM Classifier
Evaluate('LGBM Classifier', LGBM, X_test_train, Y_test_train)
```

```
--------------------------------------------------

The LGBM Classifier Model Accuracy   = 0.91

The LGBM Classifier Model Precision  = 0.91

The LGBM Classifier Model Recall     = 0.91

The LGBM Classifier Model F1 Score   = 0.91

--------------------------------------------------
```





9. CatBoost Classifier

```
# Instantiate and fit the CatBoost Classifier
CatBoost = CatBoostClassifier(learning_rate=0.1,depth=6,n_estimators=100,l2_leaf_reg=3,random_state=42)
CatBoost.fit(X_train_train, Y_train_train)
```

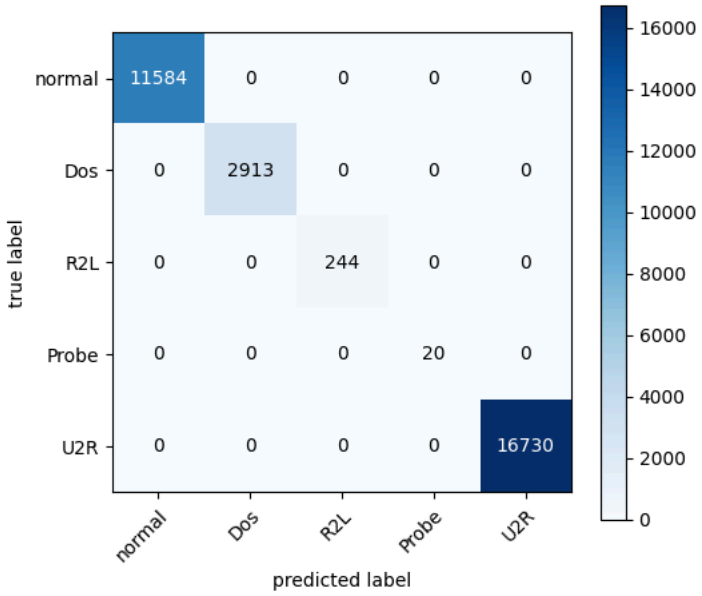
```
0:      learn: 1.2327535        total: 627ms    remaining: 1m 2s
1:      learn: 1.0030130        total: 915ms    remaining: 44.8s
2:      learn: 0.8402456        total: 1.39s    remaining: 44.9s
3:      learn: 0.7160126        total: 1.83s    remaining: 43.9s
4:      learn: 0.6169513        total: 2.22s    remaining: 42.2s
5:      learn: 0.5360977        total: 2.67s    remaining: 41.8s
6:      learn: 0.4689003        total: 3.02s    remaining: 40.1s
7:      learn: 0.4122002        total: 3.36s    remaining: 38.6s
8:      learn: 0.3636956        total: 3.79s    remaining: 38.3s
9:      learn: 0.3222869        total: 4.1s     remaining: 36.9s
10:     learn: 0.2861390        total: 4.35s    remaining: 35.2s
11:     learn: 0.2547386        total: 4.66s    remaining: 34.2s
12:     learn: 0.2272971        total: 4.94s    remaining: 33s
13:     learn: 0.2032578        total: 5.28s    remaining: 32.4s
14:     learn: 0.1818616        total: 5.57s    remaining: 31.6s
15:     learn: 0.1630264        total: 5.82s    remaining: 30.6s
16:     learn: 0.1462372        total: 6.11s    remaining: 29.9s
17:     learn: 0.1312873        total: 6.31s    remaining: 28.7s
18:     learn: 0.1180131        total: 6.56s    remaining: 28s
19:     learn: 0.1061130        total: 6.85s    remaining: 27.4s
20:     learn: 0.0955440        total: 7.17s    remaining: 27s
21:     learn: 0.0860237        total: 7.5s     remaining: 26.6s
22:     learn: 0.0775082        total: 7.69s    remaining: 25.7s
23:     learn: 0.0698499        total: 7.98s    remaining: 25.3s
24:     learn: 0.0630016        total: 8.26s    remaining: 24.8s
25:     learn: 0.0568354        total: 8.52s    remaining: 24.3s
26:     learn: 0.0513090        total: 8.82s    remaining: 23.8s
27:     learn: 0.0463282        total: 9.08s    remaining: 23.3s
28:     learn: 0.0418554        total: 9.36s    remaining: 22.9s
29:     learn: 0.0378111        total: 9.62s    remaining: 22.4s
30:     learn: 0.0341986        total: 9.91s    remaining: 22.1s
31:     learn: 0.0309210        total: 10.3s    remaining: 21.8s
32:     learn: 0.0279540        total: 10.5s    remaining: 21.3s
33:     learn: 0.0252841        total: 10.7s    remaining: 20.8s
34:     learn: 0.0228686        total: 11s      remaining: 20.4s
35:     learn: 0.0207259        total: 11.2s    remaining: 19.9s
36:     learn: 0.0187581        total: 11.4s    remaining: 19.5s
37:     learn: 0.0169866        total: 11.7s    remaining: 19.1s
38:     learn: 0.0153755        total: 12s      remaining: 18.8s
39:     learn: 0.0139245        total: 12.4s    remaining: 18.6s
40:     learn: 0.0126097        total: 12.7s    remaining: 18.3s
41:     learn: 0.0114223        total: 13.1s    remaining: 18.1s
42:     learn: 0.0103558        total: 13.4s    remaining: 17.7s
43:     learn: 0.0093917        total: 13.6s    remaining: 17.4s
44:     learn: 0.0085134        total: 14.1s    remaining: 17.2s
45:     learn: 0.0077171        total: 14.6s    remaining: 17.2s
46:     learn: 0.0070058        total: 15.1s    remaining: 17s
47:     learn: 0.0063535        total: 15.6s    remaining: 16.9s
48:     learn: 0.0057652        total: 15.9s    remaining: 16.5s
49:     learn: 0.0052422        total: 16.3s    remaining: 16.3s
50:     learn: 0.0047596        total: 16.7s    remaining: 16.1s
51:     learn: 0.0043258        total: 17.3s    remaining: 15.9s
52:     learn: 0.0039289        total: 17.7s    remaining: 15.7s
53:     learn: 0.0035712        total: 18.1s    remaining: 15.4s
54:     learn: 0.0032548        total: 18.6s    remaining: 15.2s
55:     learn: 0.0029618        total: 19.1s    remaining: 15s
56:     learn: 0.0026981        total: 19.5s    remaining: 14.7s
57:     learn: 0.0024600        total: 19.9s    remaining: 14.4s
```

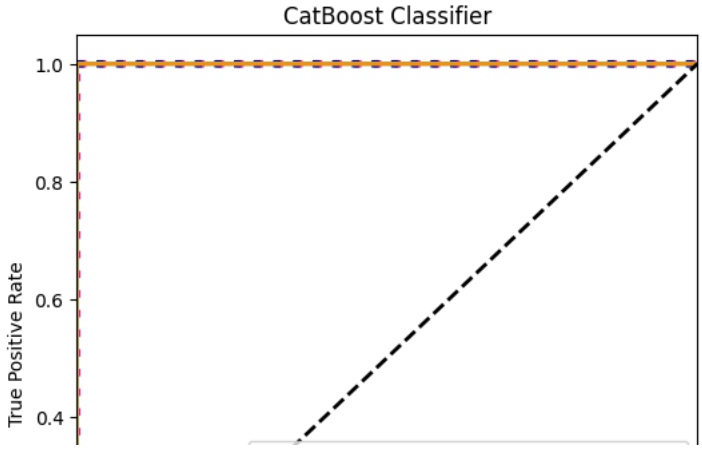
```
# Print the training and testing scores
print("CatBoost Classifier Training Score:", CatBoost.score(X_train_train, Y_train_train))
print("CatBoost Classifier Testing Score:", CatBoost.score(X_test_train, Y_test_train))
```

```
CatBoost Classifier Training Score: 1.0
CatBoost Classifier Testing Score: 1.0
```

```
# Evaluate the CatBoost Classifier
Evaluate('CatBoost Classifier', CatBoost, X_test_train, Y_test_train)
```

```
--------------------------------------------------

The CatBoost Classifier Model Accuracy   = 1.0

The CatBoost Classifier Model Precision  = 1.0

The CatBoost Classifier Model Recall     = 1.0

The CatBoost Classifier Model F1 Score   = 1.0

--------------------------------------------------
```





10. Naive Bayes Classifier

```
# Instantiate and fit the Naive Bayes Classifier
NB = GaussianNB(var_smoothing=1e-9)
NB.fit(X_train_train, Y_train_train)
```

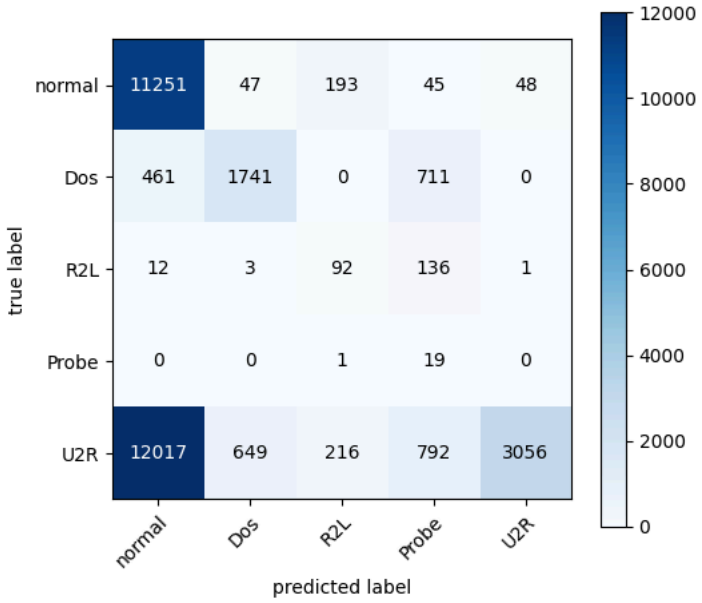
```
GaussianNB()
```

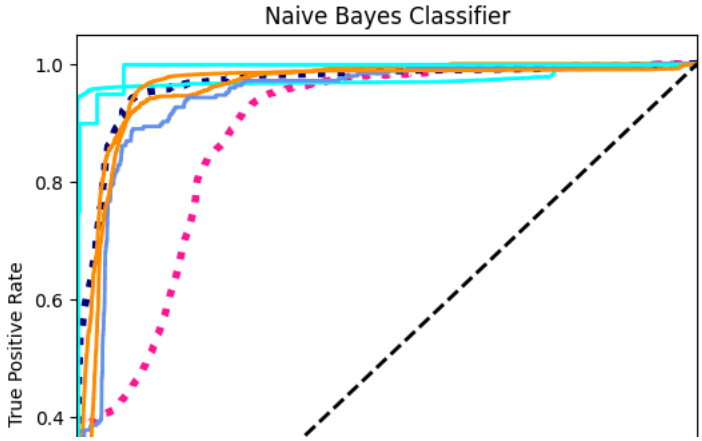
```
# Print the training and testing scores
print("Naive Bayes Classifier Training Score:", NB.score(X_train_train, Y_train_train))
print("Naive Bayes Classifier Testing Score:", NB.score(X_test_train, Y_test_train))
```

```
Naive Bayes Classifier Training Score: 0.5128397832161911
Naive Bayes Classifier Testing Score: 0.5131307357657744
```

```
# Evaluate the Naive Bayes Classifier
Evaluate('Naive Bayes Classifier', NB, X_test_train, Y_test_train)
```

```
--------------------------------------------------

The Naive Bayes Classifier Model Accuracy   = 0.51

The Naive Bayes Classifier Model Precision  = 0.76

The Naive Bayes Classifier Model Recall     = 0.51

The Naive Bayes Classifier Model F1 Score   = 0.46

--------------------------------------------------
```





11. Linear Discriminant Analysis (LDA)

ROC curve for class 0 (AUC = 0.97)

```
# Instantiate the LDA model
LDA = LinearDiscriminantAnalysis(solver='lsqr')
LDA.fit(X_train_train, Y_train_train)
```

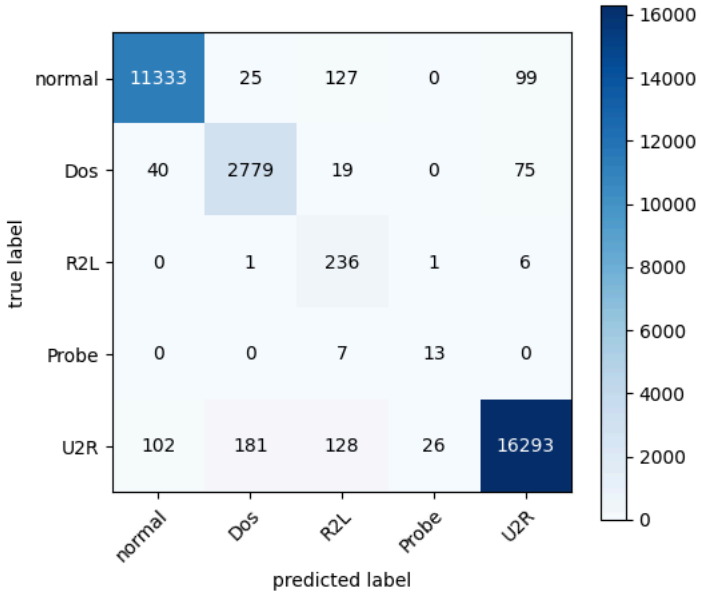
```
LinearDiscriminantAnalysis
LinearDiscriminantAnalysis(solver='lsqr')
```

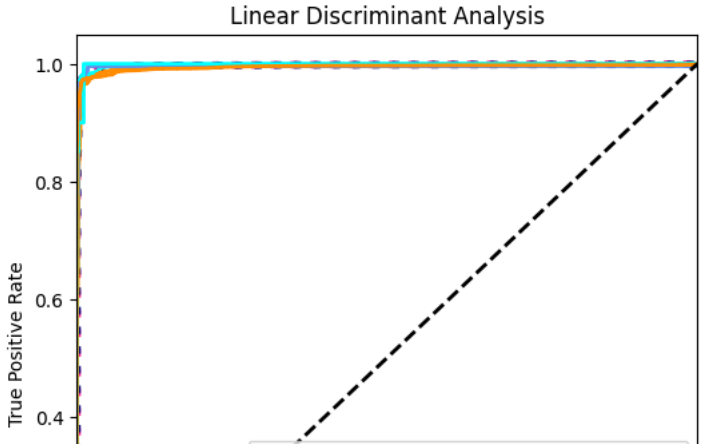
```
# Print the training and testing scores
print("Linear Discriminant Analysis Training Score:", LDA.score(X_train_train, Y_train_train))
print("Linear Discriminant Analysis Testing Score:", LDA.score(X_test_train, Y_test_train))
```

```
Linear Discriminant Analysis Training Score: 0.97444745533068
Linear Discriminant Analysis Testing Score: 0.9734209774221206
```

```
# Evaluate the LDA model
Evaluate('Linear Discriminant Analysis', LDA, X_test_train, Y_test_train)
```

```
--------------------------------------------------

The Linear Discriminant Analysis Model Accuracy   = 0.97

The Linear Discriminant Analysis Model Precision  = 0.98

The Linear Discriminant Analysis Model Recall     = 0.97

The Linear Discriminant Analysis Model F1 Score   = 0.98

--------------------------------------------------
```





12. Quadratic Discriminant Analysis (QDA)

```
# Instantiate the QDA model
QDA = QuadraticDiscriminantAnalysis(reg_param=0.1)
QDA.fit(X_train_train, Y_train_train)
```

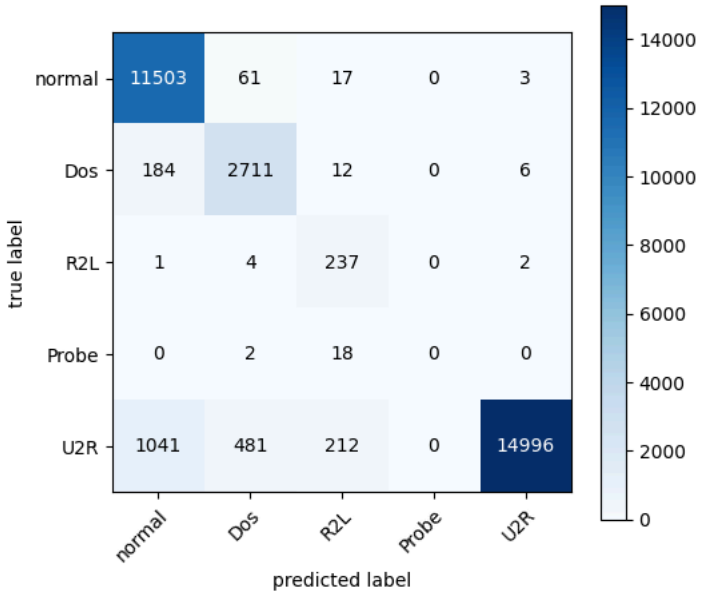
```
QuadraticDiscriminantAnalysis(reg_param=0.1)
```

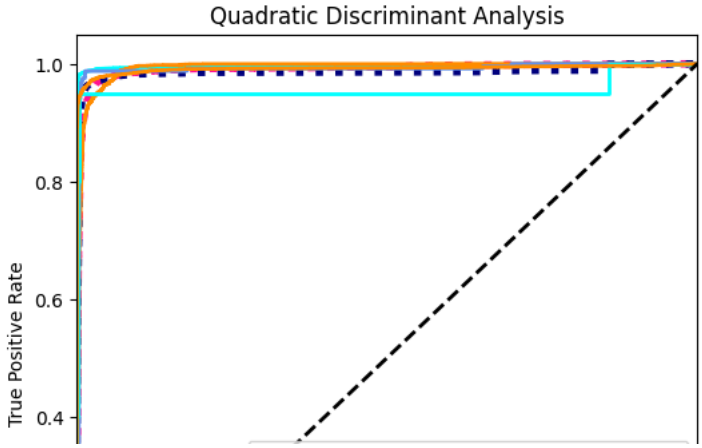
```
# Print the training and testing scores
print("Quadratic Discriminant Analysis Training Score:", QDA.score(X_train_train, Y_train_train))
print("Quadratic Discriminant Analysis Testing Score:", QDA.score(X_test_train, Y_test_train))
```

```
Quadratic Discriminant Analysis Training Score: 0.9376958252180541
Quadratic Discriminant Analysis Testing Score: 0.9350925661300054
```

```
# Evaluate the QDA model
Evaluate('Quadratic Discriminant Analysis', QDA, X_test_train, Y_test_train)
```

```
--------------------------------------------------

The Quadratic Discriminant Analysis Model Accuracy   = 0.94

The Quadratic Discriminant Analysis Model Precision  = 0.94

The Quadratic Discriminant Analysis Model Recall     = 0.94

The Quadratic Discriminant Analysis Model F1 Score   = 0.94

--------------------------------------------------
```





13. Passive Aggressive Classifier

```
# Create PassiveAggressiveClassifier
PAC_classifier = PassiveAggressiveClassifier(C=0.1, max_iter=1000)

# Wrap PassiveAggressiveClassifier inside CalibratedClassifierCV with method='sigmoid' for Platt scaling
Platt_PAC = CalibratedClassifierCV(PAC_classifier, method='sigmoid')

# Fit the classifier
Platt_PAC.fit(X_train_train, Y_train_train)
```

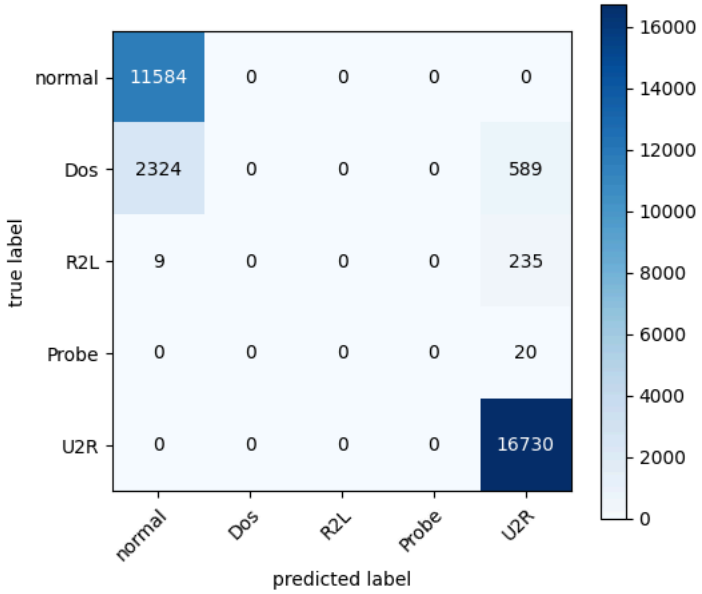
```
CalibratedClassifierCV
estimator: PassiveAggressiveClassifier
PassiveAggressiveClassifier
```

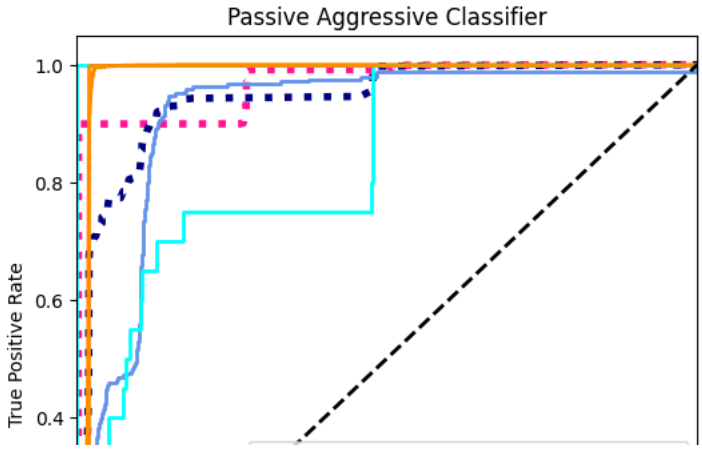
```
# Print the training and testing scores
print("Passive Aggressive Classifier Training Score:", Platt_PAC.score(X_train_train, Y_train_train))
print("Passive Aggressive Classifier Testing Score:", Platt_PAC.score(X_test_train, Y_test_train))
```

```
Passive Aggressive Classifier Training Score: 0.8992505715979338
Passive Aggressive Classifier Testing Score: 0.8991140325807373
```

```
# Evaluate the Passive Aggressive Classifier
Evaluate('Passive Aggressive Classifier', Platt_PAC, X_test_train, Y_test_train)
```

```
--------------------------------------------------

The Passive Aggressive Classifier Model Accuracy   = 0.9

The Passive Aggressive Classifier Model Precision  = 0.81

The Passive Aggressive Classifier Model Recall     = 0.9

The Passive Aggressive Classifier Model F1 Score   = 0.85

--------------------------------------------------
```





14. AdaBoost Classifier

ROC curve for class 0 (AUC = 1.00)

```
# Instantiate the base estimator (DecisionTreeClassifier)
base_estimator = DecisionTreeClassifier(max_features=6, max_depth=4)

# Instantiate the AdaBoost Classifier with adjusted parameters
AdaBoost = AdaBoostClassifier(base_estimator=base_estimator, n_estimators=100, learning_rate=1.0)
AdaBoost.fit(X_train_train, Y_train_train)
```

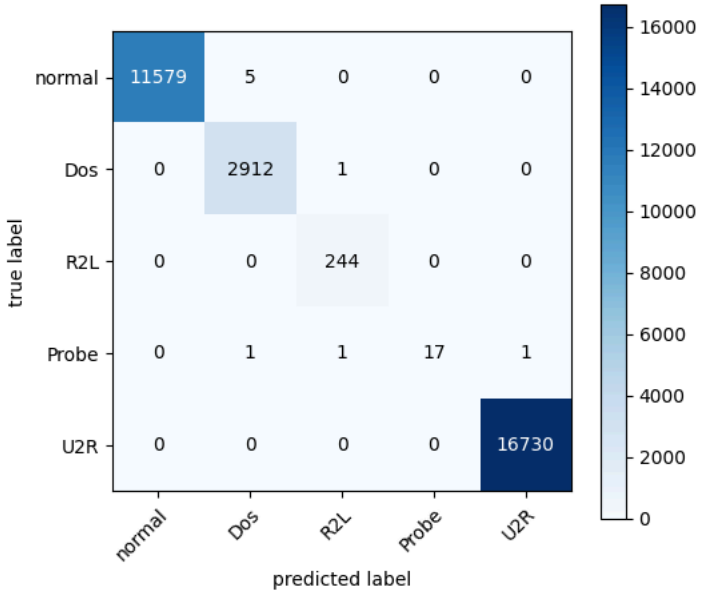
```
AdaBoostClassifier
base_estimator: DecisionTreeClassifier
DecisionTreeClassifier
```

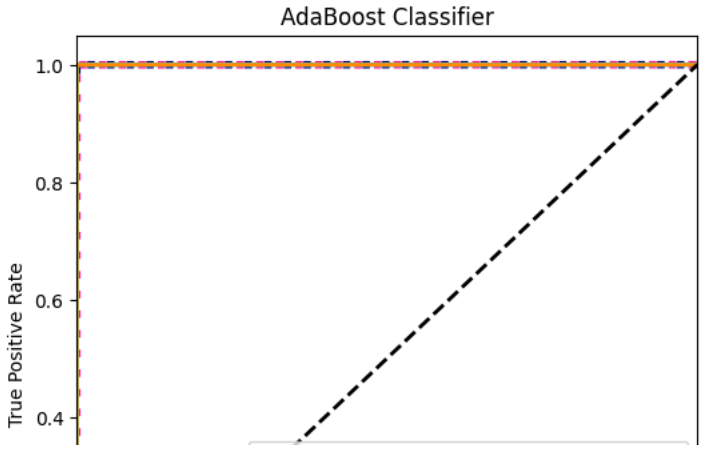
```
# Print the training and testing scores
print("AdaBoost Classifier Training Score:", AdaBoost.score(X_train_train, Y_train_train))
print("AdaBoost Classifier Testing Score:", AdaBoost.score(X_test_train, Y_test_train))
```

```
AdaBoost Classifier Training Score: 0.9999259039715471
AdaBoost Classifier Testing Score: 0.9997142040583024
```

```
# Evaluate the AdaBoost Classifier
Evaluate('AdaBoost Classifier', AdaBoost, X_test_train, Y_test_train)
```

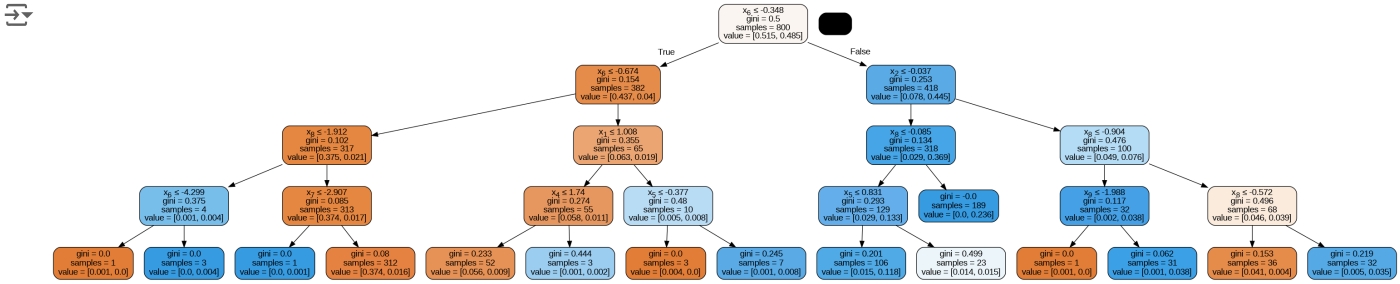
```
--------------------------------------------------

The AdaBoost Classifier Model Accuracy   = 1.0

The AdaBoost Classifier Model Precision  = 1.0

The AdaBoost Classifier Model Recall     = 1.0

The AdaBoost Classifier Model F1 Score   = 1.0

--------------------------------------------------
```





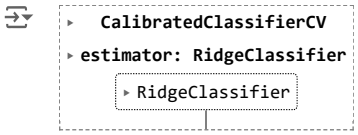

```
# prompt: generate a diagram for the adaboost implementation archecture in the above model

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from IPython.display import Image, display
import pydotplus
from sklearn import tree

# Sample data (replace with your actual data)
X, y = make_classification(n_samples=1000, n_features=10, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Instantiate the base estimator (DecisionTreeClassifier)
base_estimator = DecisionTreeClassifier(max_features=6, max_depth=4)

# Instantiate the AdaBoost Classifier with adjusted parameters
AdaBoost = AdaBoostClassifier(base_estimator=base_estimator, n_estimators=100, learning_rate=1.0)
AdaBoost.fit(X_train, y_train)

# Visualize the first decision tree in the AdaBoost ensemble
estimator = AdaBoost.estimators_[0]
dot_data = tree.export_graphviz(estimator,
                                filled=True,
                                rounded=True,
```

```
                                special_characters=True)

graph = pydotplus.graph_from_dot_data(dot_data)
display(Image(graph.create_png()))
```



15. Ridge Classifier

```
# Create RidgeClassifier
ridge_classifier = RidgeClassifier(alpha=1.0)

# Wrap RidgeClassifier inside CalibratedClassifierCV with method='sigmoid' for Platt scaling
Platt_ridge = CalibratedClassifierCV(ridge_classifier, method='sigmoid')

# Fit the classifier
Platt_ridge.fit(X_train_train, Y_train_train)
```

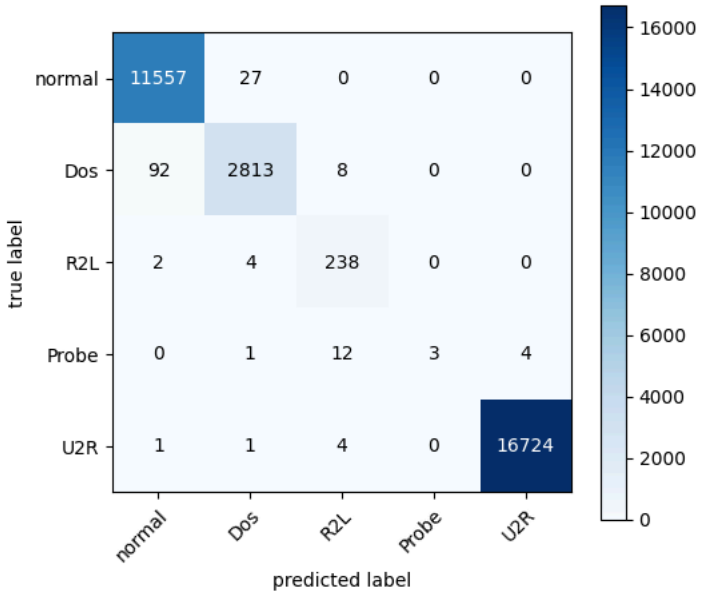


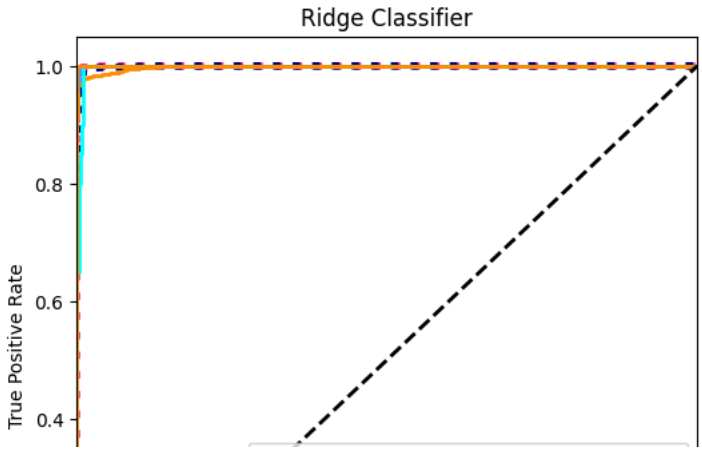

```
# Print the training and testing scores
print("Ridge Classifier Training Score:", Platt_ridge.score(X_train_train, Y_train_train))
print("Ridge Classifier Testing Score:", Platt_ridge.score(X_test_train, Y_test_train))
```

```
Ridge Classifier Training Score: 0.995585993733593
Ridge Classifier Testing Score: 0.9950462036772412
```

```
# Evaluate the Ridge Classifier
Evaluate('Ridge Classifier', Platt_ridge, X_test_train, Y_test_train)
```

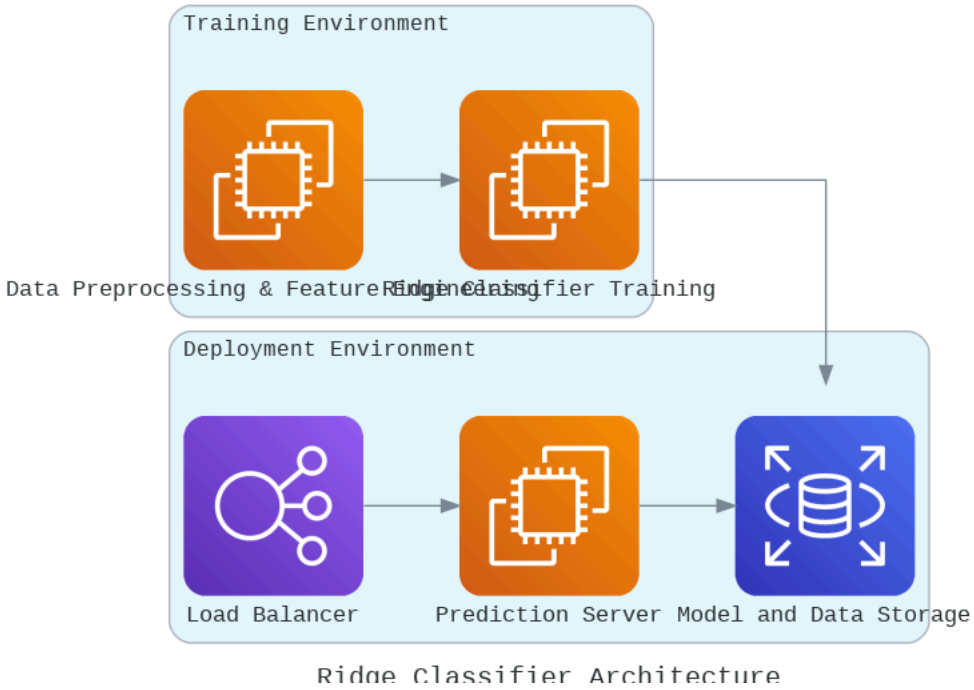
```
--------------------------------------------------

The Ridge Classifier Model Accuracy   = 1.0

The Ridge Classifier Model Precision  = 1.0

The Ridge Classifier Model Recall     = 1.0

The Ridge Classifier Model F1 Score   = 0.99

--------------------------------------------------
```






```
# prompt: generate a diagram for the ridge classifier implementation archecture in the above model

!pip install diagrams

from diagrams import Diagram, Cluster
from diagrams.aws.compute import EC2
from diagrams.aws.database import RDS
from diagrams.aws.network import ELB

with Diagram("Ridge Classifier Architecture", show=False) as diag:
    with Cluster("Training Environment"):
        data_prep = EC2("Data Preprocessing & Feature Engineering")
        model_training = EC2("Ridge Classifier Training")

    with Cluster("Deployment Environment"):
        load_balancer = ELB("Load Balancer")
        prediction_server = EC2("Prediction Server")
        database = RDS("Model and Data Storage")

    data_prep >> model_training >> database
    load_balancer >> prediction_server >> database

diag
```

```
Collecting diagrams
  Downloading diagrams-0.23.4-py3-none-any.whl.metadata (7.0 kB)
Requirement already satisfied: graphviz<0.21.0,>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from diagrams) (0.20.3)
Requirement already satisfied: jinja2<4.0,>=2.10 in /usr/local/lib/python3.10/dist-packages (from diagrams) (3.1.4)
Collecting typed-ast<2.0.0,>=1.5.4 (from diagrams)
  Downloading typed_ast-1.5.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (1.7 kB)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2<4.0,>=2.10->diagrams) (2.1
Downloading diagrams-0.23.4-py3-none-any.whl (24.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 53.1 MB/s eta 0:00:00
Downloading typed_ast-1.5.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (824 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 824.7/824.7 kB 38.2 MB/s eta 0:00:00
Installing collected packages: typed-ast, diagrams
Successfully installed diagrams-0.23.4 typed-ast-1.5.5
Warning: node '270c3c0508e3421a90ced18e96417c62', graph 'Ridge Classifier Architecture' size too small for label
Warning: node '438dd7af075f4650a2a21d890ed17de6', graph 'Ridge Classifier Architecture' size too small for label
Warning: node '31831ce3b5394320adf3fa6cfcf233ec', graph 'Ridge Classifier Architecture' size too small for label
Warning: node '33ffd0a022c54060b5e7030ec7526c50', graph 'Ridge Classifier Architecture' size too small for label
```



Ridge Classifier Architecture

***Stacking ***

```
# Stacking Model

# Import necessary libraries
from sklearn.ensemble import RandomForestClassifier, StackingClassifier
from sklearn.linear_model import LogisticRegression
import xgboost as xgb
import lightgbm as lgb
from catboost import CatBoostClassifier

# Base Models (Use the best performing models from your previous analysis)
models = [
    ('rf', RandomForestClassifier(max_depth=10)),
    ('xgb', xgb.XGBClassifier(n_estimators=100, random_state=42)),
    ('lgbm', lgb.LGBMClassifier()),
    ('catboost', CatBoostClassifier(learning_rate=0.1, depth=6, n_estimators=100, l2_leaf_reg=3, random_state=42))
]

# Meta Model
meta_model = LogisticRegression()

# Stacking Classifier
stacking_clf = StackingClassifier(estimators=models, final_estimator=meta_model, cv=5) # Now StackingClassifier is defined
stacking_clf.fit(X_train_train, Y_train_train)
```

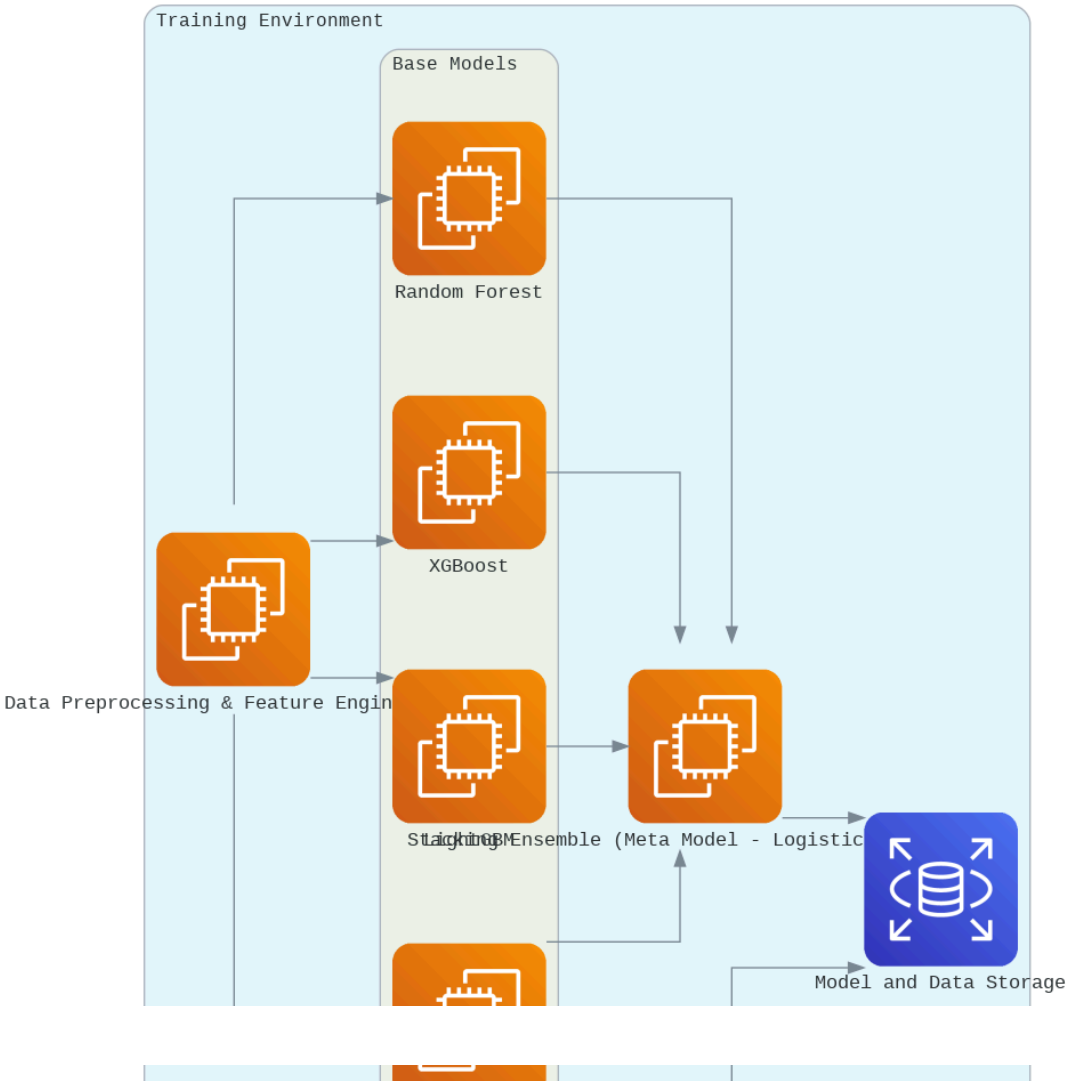
```
# Evaluate Stacking Model
Evaluate('Stacking Classifier', stacking_clf, X_test_train, Y_test_train)
```

```
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.052928 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 3327
[LightGBM] [Info] Number of data points in the train set: 94472, number of used features: 112
[LightGBM] [Info] Start training from score -1.011905
[LightGBM] [Info] Start training from score -2.381080
[LightGBM] [Info] Start training from score -4.834653
[LightGBM] [Info] Start training from score -7.990323
[LightGBM] [Info] Start training from score -0.624115
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
# prompt: generate a diagram for the stacking  implementation archecture in the above model

from diagrams import Diagram, Cluster
from diagrams.aws.compute import EC2
from diagrams.aws.database import RDS
from diagrams.aws.network import ELB

with Diagram("Stacking Model Architecture", show=False) as diag:
    with Cluster("Training Environment"):
        data_prep = EC2("Data Preprocessing & Feature Engineering")

        with Cluster("Base Models"):
            rf = EC2("Random Forest")
            xgb = EC2("XGBoost")
            lgbm = EC2("LightGBM")
            catboost = EC2("CatBoost")

        stacking = EC2("Stacking Ensemble (Meta Model - Logistic Regression)")
        model_storage = RDS("Model and Data Storage")

    with Cluster("Deployment Environment"):
        load_balancer = ELB("Load Balancer")
        prediction_server = EC2("Prediction Server")

    data_prep >> [rf, xgb, lgbm, catboost] >> stacking >> model_storage
    load_balancer >> prediction_server >> model_storage
```

```
diag
```

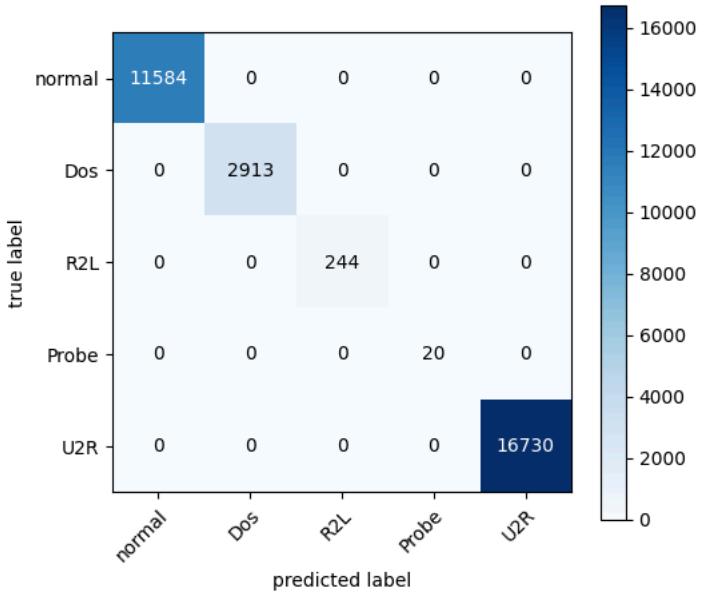
```
Warning: node '7c6478f84c5e48d28d5dbef3a82c6087', graph 'Stacking Model Architecture' size too small for label
Warning: node 'ac71a4fc6abd4b6a80778ed6876d655f', graph 'Stacking Model Architecture' size too small for label
Warning: node 'e850087e4ccb40edb02a1c46f47d818b', graph 'Stacking Model Architecture' size too small for label
Warning: node '8fd744ab6a054c5894c182f57d1fde2a', graph 'Stacking Model Architecture' size too small for label
```

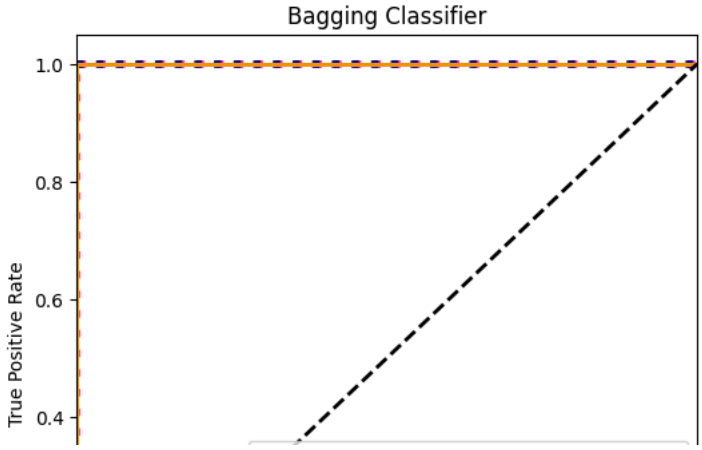


Bagging

```
# prompt: generate a code for bagging model for the above data

from sklearn.ensemble import BaggingClassifier

# Instantiate the base estimator (e.g., DecisionTreeClassifier)
base_estimator = DecisionTreeClassifier(max_depth=10)

# Instantiate the Bagging Classifier
bagging_clf = BaggingClassifier(base_estimator=base_estimator, n_estimators=100, random_state=42)

# Fit the Bagging Classifier
bagging_clf.fit(X_train_train, Y_train_train)

# Evaluate Bagging Model
Evaluate('Bagging Classifier', bagging_clf, X_test_train, Y_test_train)
```

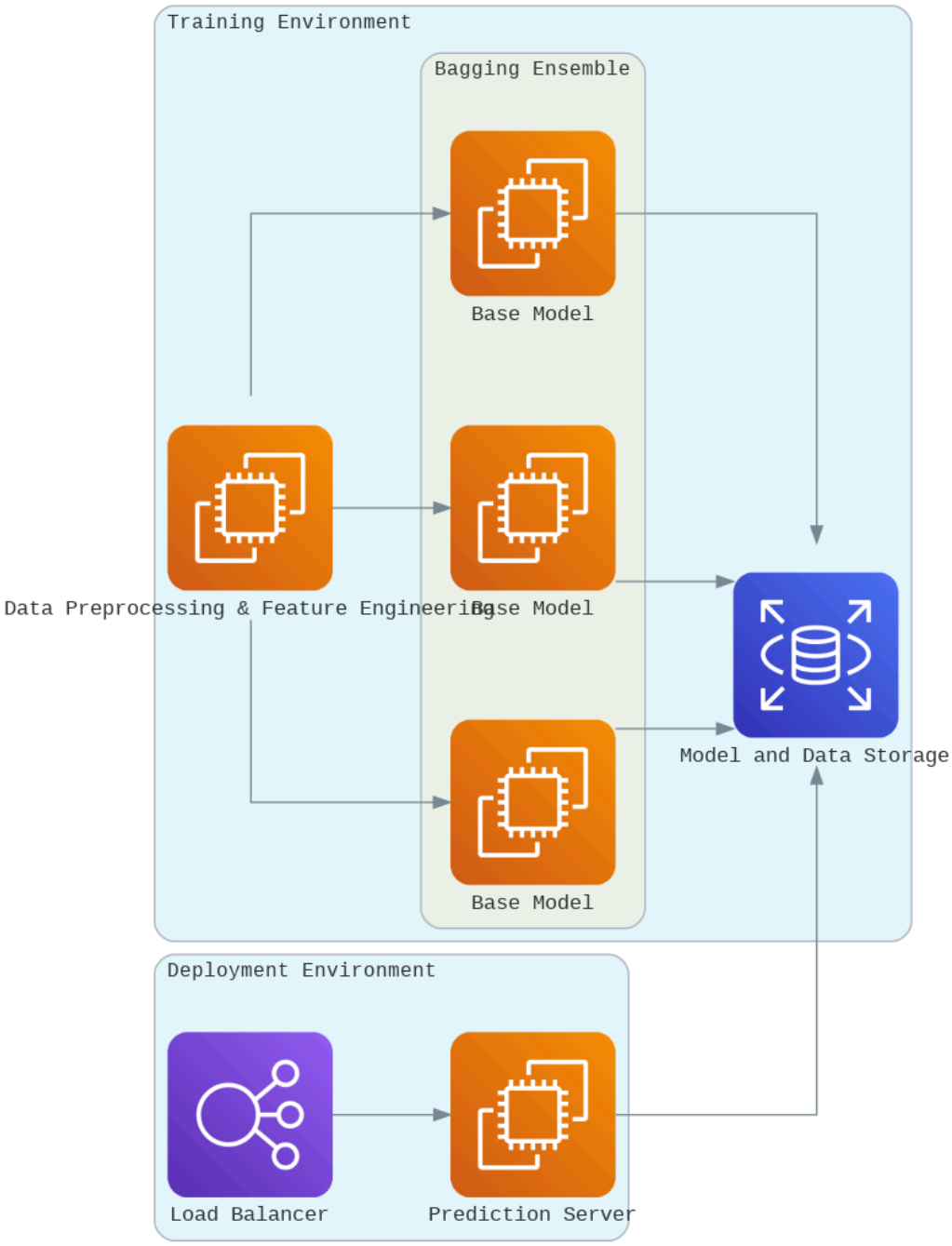
```
--------------------------------------------------

The Bagging Classifier Model Accuracy   = 1.0

The Bagging Classifier Model Precision  = 1.0

The Bagging Classifier Model Recall     = 1.0

The Bagging Classifier Model F1 Score   = 1.0

--------------------------------------------------
```






```
# prompt: generate a diagram for the bagging implementation archecture in the above model

from diagrams import Diagram, Cluster
from diagrams.aws.compute import EC2
from diagrams.aws.network import ELB
from diagrams.aws.database import RDS

with Diagram("Bagging Model Architecture", show=False) as diag:
    with Cluster("Training Environment"):
        data_prep = EC2("Data Preprocessing & Feature Engineering")

        with Cluster("Bagging Ensemble"):
            multiple_base_models = [EC2("Base Model") for _ in range(3)] # Representing multiple base models

        model_storage = RDS("Model and Data Storage")

    with Cluster("Deployment Environment"):
        load_balancer = ELB("Load Balancer")
        prediction_server = EC2("Prediction Server")

    data_prep >> multiple_base_models >> model_storage
    load_balancer >> prediction_server >> model_storage

diag
```

```
Warning: node '1fdf54f7956e498884fa6dfcc6ea0ffd', graph 'Bagging Model Architecture' size too small for label
Warning: node 'dd475a0d50c84c23a420da814b5a900b', graph 'Bagging Model Architecture' size too small for label
Warning: node '558333bdb63744b283de983ca9441edc', graph 'Bagging Model Architecture' size too small for label
```



Bagging Model Architecture

[LightGBM] [Warning] No further splits with positive gain, best gain: -inf

```
# prompt: Genarate the code for the stacked Knowledge Distillation model for the above training data

import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import xgboost as xgb
import lightgbm as lgb
from catboost import CatBoostClassifier
from sklearn.ensemble import StackingClassifier
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam

# Assuming 'X_train_train', 'Y_train_train', 'X_test_train', 'Y_test_train' are defined from your preceding code.


# Define the base models
base_models = [
    ('rf', RandomForestClassifier(max_depth=10)),
    ('xgb', xgb.XGBClassifier(n_estimators=100, random_state=42)),
    ('lgbm', lgb.LGBMClassifier()),
    ('catboost', CatBoostClassifier(learning_rate=0.1, depth=6, n_estimators=100, l2_leaf_reg=3, random_state=42))
]

# Train the base models
base_models_predictions = []
for name, model in base_models:
    model.fit(X_train_train, Y_train_train)
    predictions = model.predict_proba(X_train_train)
    base_models_predictions.append(predictions)

# Combine predictions from base models
combined_predictions = np.concatenate(base_models_predictions, axis=1)

# Define the knowledge distillation model
distillation_model = Sequential()
distillation_model.add(Dense(128, activation='relu', input_shape=(combined_predictions.shape[1],)))
distillation_model.add(Dropout(0.2))
distillation_model.add(Dense(64, activation='relu'))
distillation_model.add(Dropout(0.2))
distillation_model.add(Dense(5, activation='softmax'))  # 5 output classes
```