```
### Network Slicing Recognition
```

About Dataset

The telecom industry is going through a massive digital transformation with the adoption of ML, AI, feedback-based automation and advanced analytics to handle the next generation applications and services. AI concepts are not new; the algorithms used by Machine Learning and Deep Learning are being currently implemented in various industries and technology verticals. With growing data and immense volume

of information over 5G, the ability to predict data proactively, swiftly and with accuracy, is critically important. Data-driven decision making will be vital in future communication networks due to the traffic explosion and Artificial Intelligence (AI) will accelerate the 5G network performance.

Mobile operators are looking for a programmable solution that will allow them to accommodate multiple independent tenants on the same physical infrastructure and 5G networks allow for end-to-end network resource allocation using the concept of Network Slicing (NS).

Network Slicing will play a vital role in enabling a multitude of 5G applications, use cases, and services. Network slicing functions will provide an end-to-end isolation between slices with an ability to customize each slice based on the service demands (bandwidth, coverage, security, latency, reliability, etc).

Your Task is to build a Machine Learning model that will be able to to proactively detect and eliminate threats based on incoming connections thereby selecting the most appropriate network slice, even in case of a network failure.

Dataset Description
LTE/5g - User Equipment categories or classes to define the performance specifications
Packet Loss Rate - number of packets not received divided by the total number of packets sent.
Packet Delay - The time for a packet to be received.
Slice type - network configuration that allows multiple networks (virtualized and independent)
GBR - Guaranteed Bit Rate
Healthcare - Usage in Healthcare (1 or 0)
Industry 4.0 - Usage in Digital Enterprises(1 or 0)
IoT Devices - Usage
Public Safety - Usage for public welfare and safety purposes (1 or 0)
Smart City & Home - usage in daily household chores
Smart Transportation - usage in public transportation
Smartphone - whether used for smartphone cellular data

```
###!mkdir ~/.kaggle
```

```
###!cp /kaggle.json ~/.kaggle/
```

```
###!chmod 600 ~/.kaggle/kaggle.json
```

```
###! pip install kaggle
```

```
###!kaggle datasets download -d gauravduttakiit/network-slicing-recognition
```

```
###! unzip /content/network-slicing-recognition.zip
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
print('setup Completed^___^')
```

```
setup Completed^___^
```

```
###! pip install --upgrade pandas
```

```
train = pd.read_csv('/content/train_dataset.csv')
train.head(2)
```

LTE/5g Category	Time	Packet Loss Rate	Packet delay	IoT	LTE/5G	GBR	Non-GBR	AR/VR/Gaming	Healthcare	Industry 4.0	IoT Devices	Public Safety	Smart City & Home	Trans

```
test = pd.read_csv('/content/test_dataset.csv')
test.head(2)
```

	LTE/5g Category	Time	Packet Loss Rate	Packet delay	IoT	LTE/5G	GBR	Non-GBR	AR/VR/Gaming	Healthcare	Industry 4.0	IoT Devices	Public Safety	Smart City & Home	Trans
0	15	17	0.001000	100	0	1	1	0	1	0	0	0	0	0	
1	14	18	0.000001	10	1	0	0	1	0	0	0	0	0	0	

```
train["IoT Devices"].value_counts()
```

```
0    29755
1     1828
Name: IoT Devices, dtype: int64
```

```
print(train.columns)
print(test.columns)
```

```
Index(['LTE/5g Category', 'Time', 'Packet Loss Rate', 'Packet delay', 'IoT',
       'LTE/5G', 'GBR', 'Non-GBR', 'AR/VR/Gaming', 'Healthcare',
       'Industry 4.0', 'IoT Devices', 'Public Safety', 'Smart City & Home',
       'Smart Transportation', 'Smartphone', 'slice Type'],
      dtype='object')
Index(['LTE/5g Category', 'Time', 'Packet Loss Rate', 'Packet delay', 'IoT',
       'LTE/5G', 'GBR', 'Non-GBR', 'AR/VR/Gaming', 'Healthcare',
       'Industry 4.0', 'IoT Devices', 'Public Safety', 'Smart City & Home',
       'Smart Transportation', 'Smartphone'],
      dtype='object')
```

```
###! pip install klib
```

```
import klib
```

```
train = klib.data_cleaning(train)
```

```
Shape of cleaned data: (8958, 17) - Remaining NAs: 0


Dropped rows: 22625
     of which 22625 duplicates. (Rows (first 150 shown): [99, 250, 289, 305, 334, 362, 431, 446, 457, 461, 504, 509, 517, 52

Dropped columns: 0
     of which 0 single valued.     Columns: []
Dropped missing values: 0
Reduced memory by at least: 3.92 MB (-95.61%)
```

```
test = klib.data_cleaning(test)
```

```
Shape of cleaned data: (8960, 16) - Remaining NAs: 0


Dropped rows: 22624
     of which 22624 duplicates. (Rows (first 150 shown): [130, 182, 275, 279, 306, 367, 379, 396, 420, 468, 555, 560, 576, 5

Dropped columns: 0
     of which 0 single valued.     Columns: []
Dropped missing values: 0
Reduced memory by at least: 3.69 MB (-95.6%)
```

```
train = klib.convert_datatypes(train)
```

```
test = klib.convert_datatypes(test)
```

```
train.dtypes
```

```
lte_5g_category          int8
time                     int8
packet_loss_rate      float32
```

```
packet_delay              int16
io_t                       int8
lte_5g                     int8
gbr                        int8
non_gbr                    int8
ar_vr_gaming               int8
healthcare                 int8
industry_4_0               int8
io_t_devices               int8
public_safety              int8
smart_city_and_home        int8
smart_transportation       int8
smartphone                 int8
slice_type                 int8
dtype: object
```

```
klib.cat_plot(train)
```

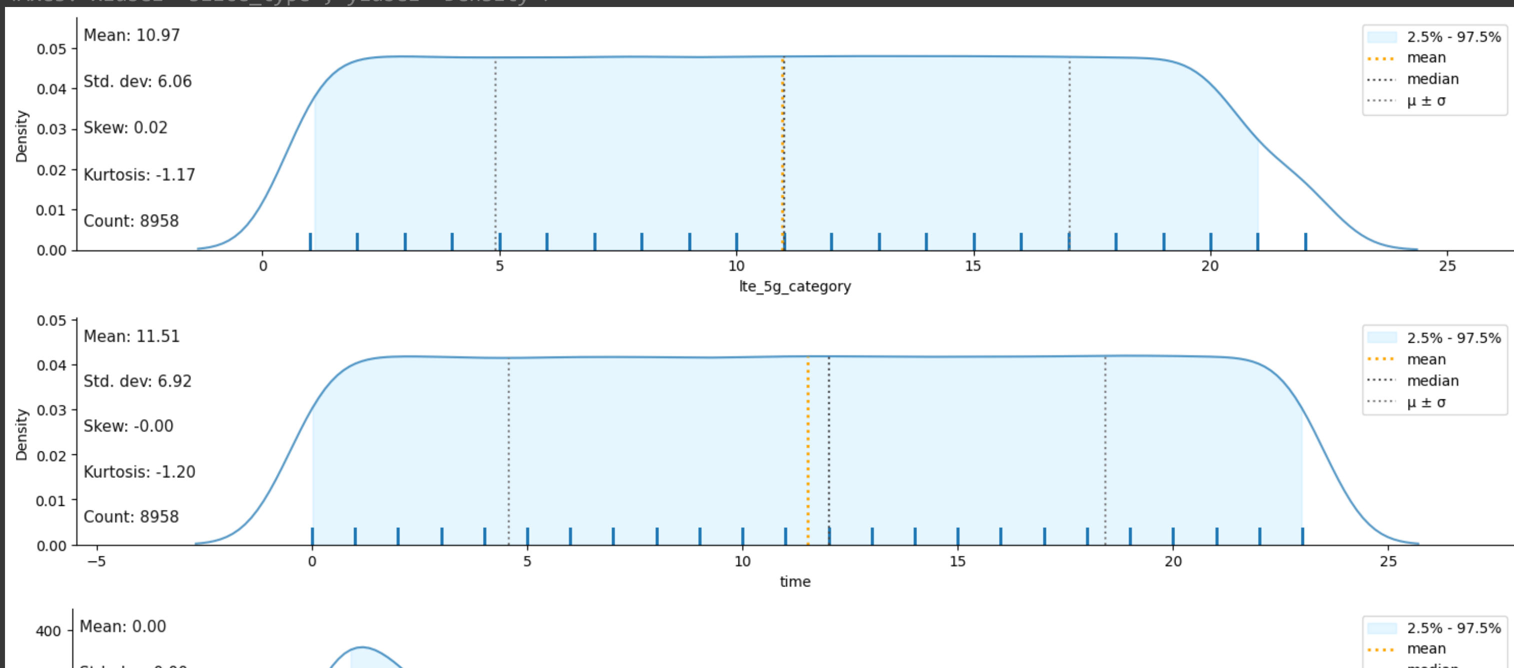
```
No columns with categorical data were detected.
```

```
klib.corr_mat(train)
```

	lte_5g_category	time	packet_loss_rate	packet_delay	io_t	lte_5g	gbr	non_gbr	ar_vr_gaming	healthc
lte_5g_category	1.00	-0.00	0.01	-0.02	0.08	-0.08	-0.01	0.01	-0.03	0
time	-0.00	1.00	0.00	0.00	0.00	-0.00	0.00	-0.00	-0.00	-0
packet_loss_rate	0.01	0.00	1.00	0.31	0.17	-0.17	-0.01	0.01	-0.17	-0
packet_delay	-0.02	0.00	0.31	1.00	-0.19	0.19	0.43	-0.43	-0.13	-0
io_t	0.08	0.00	0.17	-0.19	1.00	-1.00	-0.13	0.13	-0.32	0
lte_5g	-0.08	-0.00	-0.17	0.19	-1.00	1.00	0.13	-0.13	0.32	-0
gbr	-0.01	0.00	-0.01	0.43	-0.13	0.13	1.00	-1.00	0.04	-0
non_gbr	0.01	-0.00	0.01	-0.43	0.13	-0.13	-1.00	1.00	-0.04	0
ar_vr_gaming	-0.03	-0.00	-0.17	-0.13	-0.32	0.32	0.04	-0.04	1.00	-0
healthcare	0.02	-0.00	-0.18	-0.24	0.27	-0.27	-0.22	0.22	-0.09	1
industry_4_0	0.03	0.00	-0.22	-0.29	0.39	-0.39	0.04	-0.04	-0.13	-0
io_t_devices	0.02	0.00	0.39	0.43	0.26	-0.26	0.28	-0.28	-0.09	-0
public_safety	0.02	0.00	-0.18	-0.24	0.27	-0.27	-0.22	0.22	-0.09	-0
smart_city_and_home	0.03	-0.00	0.58	0.21	0.39	-0.39	0.04	-0.04	-0.13	-0
smart_transportation	0.02	0.00	-0.18	-0.25	0.27	-0.27	-0.22	0.22	-0.09	-0
smartphone	-0.07	-0.00	-0.07	0.28	-0.81	0.81	0.10	-0.10	-0.30	-0
slice_type	0.08	0.00	-0.10	-0.40	0.91	-0.91	-0.33	0.33	-0.29	0

```
klib.dist_plot(train)
```

```
<Axes: xlabel='slice_type', ylabel='Density'>
```




```python
klib.missingval_plot(train)
```

```
No missing values found in the dataset.
```

```python
##! pip install mlxtend
```

```python
train.columns
```

```
Index(['lte_5g_category', 'time', 'packet_loss_rate', 'packet_delay', 'io_t',
       'lte_5g', 'gbr', 'non_gbr', 'ar_vr_gaming', 'healthcare',
       'industry_4_0', 'io_t_devices', 'public_safety', 'smart_city_and_home',
       'smart_transportation', 'smartphone', 'slice_type'],
      dtype='object')
```

```python
num_var = [feature for feature in train.columns if train[feature].dtypes != 'O']
discrete_var = [feature for feature in num_var if len(train[feature].unique()) <= 25]
cont_var = [feature for feature in num_var if feature not in discrete_var]
categ_var = [feature for feature in train.columns if feature not in num_var]
```

```python
print("The Numerical Columns :", num_var,
      "The discreate Columns :", discrete_var,
      "The continuous Columns :", cont_var,
      "The categorical Columns :",categ_var)
```

```
The Numerical Columns : ['lte_5g_category', 'time', 'packet_loss_rate', 'packet_delay', 'io_t', 'lte_5g', 'gbr', 'non_gbr',
```

```python
def find_var_type(var):


    if var in discrete_var:
        print("{} is a Numerical Variable, Discrete in nature".format(var))
    elif var in cont_var :
        print("{} is a Numerical Variable, Continuous in nature".format(var))
    else :
        print("{} is a Categorical Variable".format(var))
```

```python
print("The continuous variables are :", cont_var)
print("The categorical variables are :", categ_var)
```

```
The continuous variables are : []
The categorical variables are : []
```

```python
Q1 = train.quantile(0.25)
Q3 = train.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
lte_5g_category         10.000000
time                    12.000000
packet_loss_rate         0.009999
packet_delay           100.000000
io_t                     1.000000
lte_5g                   1.000000
gbr                      1.000000
non_gbr                  1.000000
ar_vr_gaming             0.000000
healthcare               0.000000
industry_4_0             0.000000
io_t_devices             0.000000
```

```
public_safety              0.000000
smart_city_and_home        0.000000
smart_transportation       0.000000
smartphone                 1.000000
slice_type                 1.000000
dtype: float64
```

```
Q1 = train.quantile(0.25)
Q3 = train.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
lte_5g_category           10.000000
time                      12.000000
packet_loss_rate           0.009999
packet_delay             100.000000
io_t                       1.000000
lte_5g                     1.000000
gbr                        1.000000
non_gbr                    1.000000
ar_vr_gaming               0.000000
healthcare                 0.000000
industry_4_0               0.000000
io_t_devices               0.000000
public_safety              0.000000
smart_city_and_home        0.000000
smart_transportation       0.000000
smartphone                 1.000000
slice_type                 1.000000
dtype: float64
```

```
test.columns
```

```
Index(['lte_5g_category', 'time', 'packet_loss_rate', 'packet_delay', 'io_t',
       'lte_5g', 'gbr', 'non_gbr', 'ar_vr_gaming', 'healthcare',
       'industry_4_0', 'io_t_devices', 'public_safety', 'smart_city_and_home',
       'smart_transportation', 'smartphone'],
      dtype='object')
```

```
test["slice_type"] = 0
```

```
train.columns
```

```
Index(['lte_5g_category', 'time', 'packet_loss_rate', 'packet_delay', 'io_t',
       'lte_5g', 'gbr', 'non_gbr', 'ar_vr_gaming', 'healthcare',
       'industry_4_0', 'io_t_devices', 'public_safety', 'smart_city_and_home',
       'smart_transportation', 'smartphone', 'slice_type'],
      dtype='object')
```

```
train["slice_type"].value_counts()
```

```
1    4772
3    2098
2    2088
Name: slice_type, dtype: int64
```

```
X_train = train.drop(columns = "slice_type")
```

```
X_test = test.drop(columns = "slice_type")
```

```
y_train = train["slice_type"]
```

```
y_test = test["slice_type"]
```

```
from sklearn.preprocessing import RobustScaler
scaler=RobustScaler()
X_scaler_train=pd.DataFrame(scaler.fit_transform(X_train),columns=X_train.columns)
X_scaler_train.head()
```

	lte_5g_category	time	packet_loss_rate	packet_delay	io_t	lte_5g	gbr	non_gbr	ar_vr_gaming	healthcare	industry_
0	0.3	-1.000000	-0.09991	-0.65	1.0	-1.0	0.0	0.0	0.0	0.0	
1	0.7	0.666667	0.00000	0.25	0.0	0.0	1.0	-1.0	1.0	0.0	
2	0.6	0.166667	-0.09991	2.25	0.0	0.0	0.0	0.0	0.0	0.0	
3	-0.8	0.416667	0.90009	0.25	0.0	0.0	0.0	0.0	0.0	0.0	
4	-0.2	-0.666667	0.90009	-0.25	1.0	-1.0	0.0	0.0	0.0	0.0	

```
X_scaler_test=pd.DataFrame(scaler.fit_transform(X_test),columns=X_test.columns)
X_scaler_test.head()
```

	lte_5g_category	time	packet_loss_rate	packet_delay	io_t	lte_5g	gbr	non_gbr	ar_vr_gaming	healthcare	industry_
0	0.4	0.408163	0.00000	0.25	0.0	0.0	1.0	-1.0	1.0	0.0	
1	0.3	0.489796	-0.09991	-0.65	1.0	-1.0	0.0	0.0	0.0	0.0	
2	0.0	-0.408163	0.00000	-0.25	1.0	-1.0	1.0	-1.0	0.0	0.0	
3	0.9	0.163265	0.00000	-0.25	1.0	-1.0	1.0	-1.0	0.0	0.0	
4	-0.9	0.816327	0.00000	-0.25	0.0	0.0	0.0	0.0	1.0	0.0	

```
print(X_scaler_train.shape, X_scaler_test.shape)
print(y_train.shape, y_test.shape)
```

```
(8958, 16) (8960, 16)
(8958,) (8960,)
```

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
```

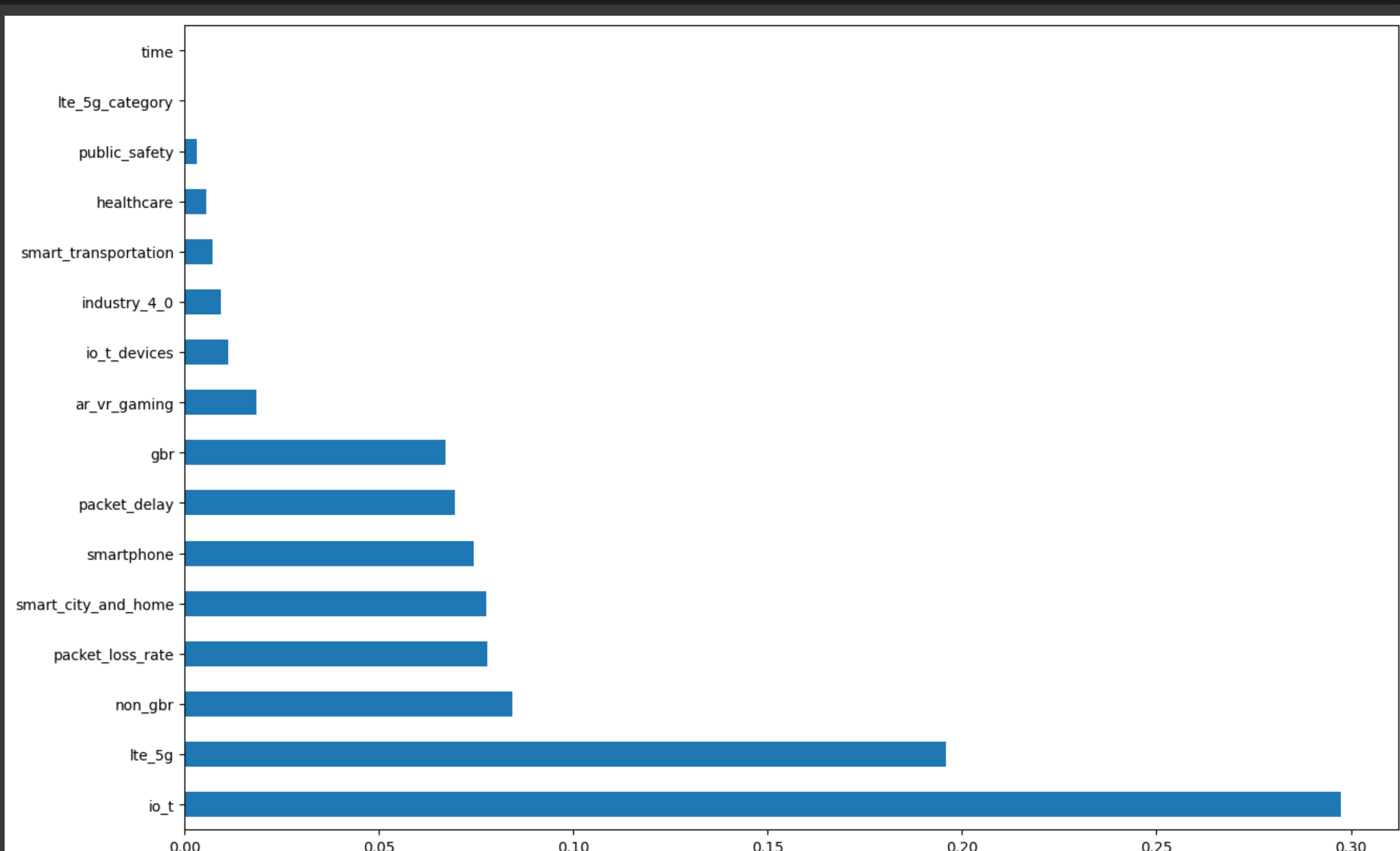
```
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
model=ExtraTreesClassifier()
model.fit(X_train,y_train)
```

```
▾ ExtraTreesClassifier
ExtraTreesClassifier()
```

```
print(model.feature_importances_)
```

```
[2.06758238e-05 3.63599101e-08 7.80159712e-02 6.94973944e-02
 2.97473790e-01 1.95934415e-01 6.70491183e-02 8.44892465e-02
 1.86110388e-02 5.64888748e-03 9.39961409e-03 1.12343831e-02
 3.21413826e-03 7.77708638e-02 7.16988517e-03 7.44705418e-02]
```

```
plt.figure(figsize = [15,10])
ranked_features=pd.Series(model.feature_importances_,index=X_train.columns)
ranked_features.nlargest(20).plot(kind='barh')
plt.show()
```

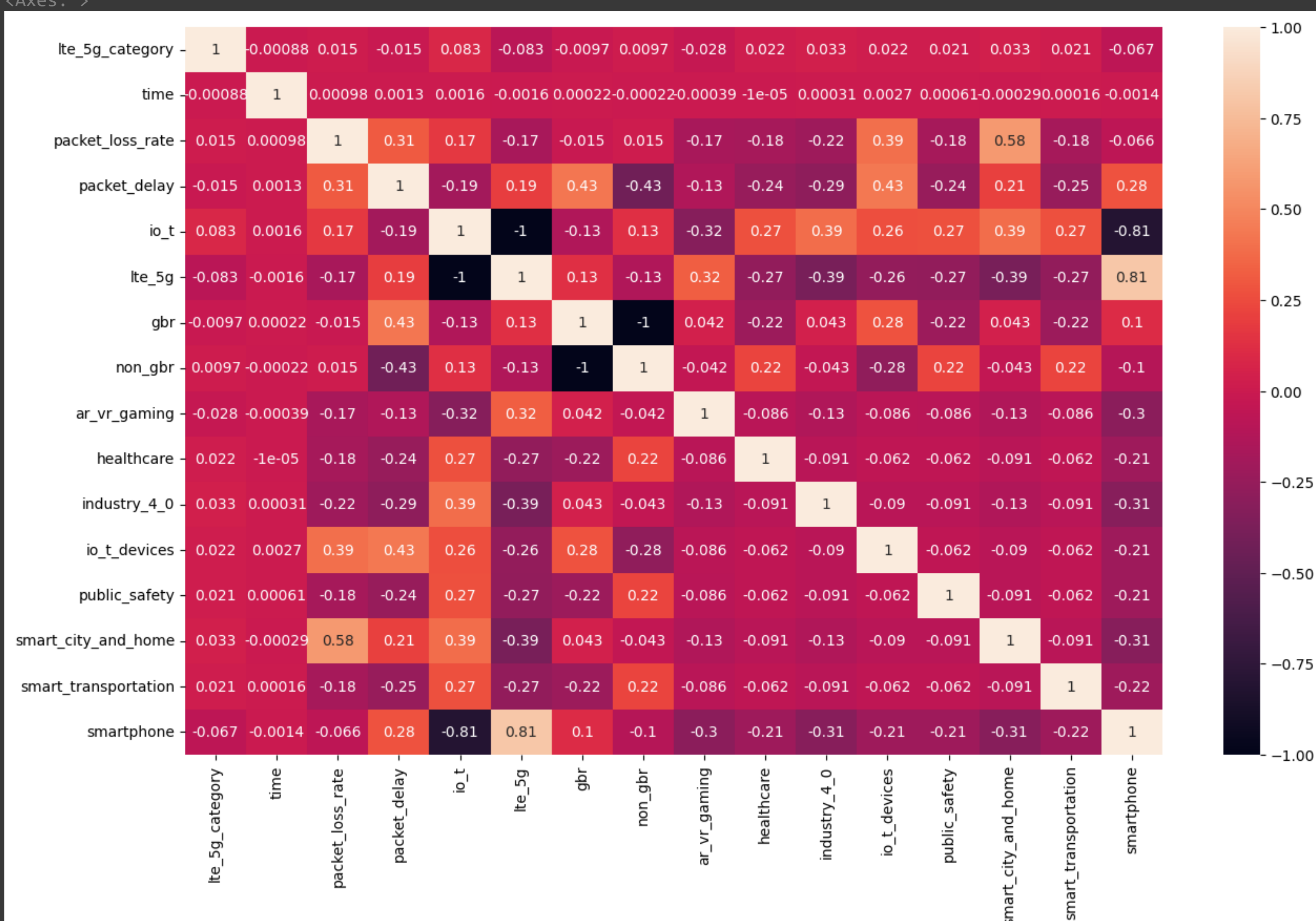


```
X_train.corr()
```

	lte_5g_category	time	packet_loss_rate	packet_delay	io_t	lte_5g	gbr	non_gbr	ar_vr_
lte_5g_category	1.000000	-0.000883	0.014919	-0.015346	0.083488	-0.083488	-0.009673	0.009673	-0
time	-0.000883	1.000000	0.000984	0.001302	0.001612	-0.001612	0.000216	-0.000216	-0
packet_loss_rate	0.014919	0.000984	1.000000	0.310946	0.167889	-0.167889	-0.014765	0.014765	-0
packet_delay	-0.015346	0.001302	0.310946	1.000000	-0.193956	0.193956	0.428456	-0.428456	-0
io_t	0.083488	0.001612	0.167889	-0.193956	1.000000	-1.000000	-0.126712	0.126712	-0
lte_5g	-0.083488	-0.001612	-0.167889	0.193956	-1.000000	1.000000	0.126712	-0.126712	0
gbr	-0.009673	0.000216	-0.014765	0.428456	-0.126712	0.126712	1.000000	-1.000000	0
non_gbr	0.009673	-0.000216	0.014765	-0.428456	0.126712	-0.126712	-1.000000	1.000000	-0
ar_vr_gaming	-0.027743	-0.000386	-0.165734	-0.127318	-0.323727	0.323727	0.041602	-0.041602	1
healthcare	0.021993	-0.000010	-0.176833	-0.244449	0.266133	-0.266133	-0.221519	0.221519	-0
industry_4_0	0.033045	0.000312	-0.216224	-0.288221	0.388426	-0.388426	0.043371	-0.043371	-0
io_t_devices	0.022474	0.002650	0.394824	0.433778	0.264782	-0.264782	0.279047	-0.279047	-0
public_safety	0.021208	0.000608	-0.176833	-0.244449	0.266133	-0.266133	-0.221519	0.221519	-0
smart_city_and_home	0.033184	-0.000290	0.578566	0.208454	0.388006	-0.388006	0.043324	-0.043324	-0
smart_transportation	0.021236	0.000159	-0.177370	-0.245191	0.266942	-0.266942	-0.222191	0.222191	-0
smartphone	-0.066915	-0.001385	-0.065912	0.275193	-0.806840	0.806840	0.101873	-0.101873	-0

```
import seaborn as sns
corr=X_train.corr()
top_features=corr.index
plt.figure(figsize=(15,9))
sns.heatmap(X_train[top_features].corr(),annot=True)
```

<Axes: >




```
threshold=0.8
```

```
# find and remove correlated features
def correlation(dataset, threshold):
    col_corr = set()  # Set of all the names of correlated columns
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold: # we are interested in absolute coeff value
                colname = corr_matrix.columns[i]  # getting the name of column
                col_corr.add(colname)
    return col_corr
```

```
correlation(X_train,threshold)
```

```
{'lte_5g', 'non_gbr', 'smartphone'}
```

```
X_train = X_train.drop(columns = ['lte_5g', 'non_gbr', 'smartphone'])
```

```
X_test = X_test.drop(columns = ['lte_5g', 'non_gbr', 'smartphone'])
```

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
```

```
from sklearn.ensemble import AdaBoostClassifier
```

```
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
(8958, 13) (8958,)
(8960, 13) (8960,)
```

```
logisticr = LogisticRegression()
dtc = DecisionTreeClassifier()
rfc = RandomForestClassifier()
gbc = GradientBoostingClassifier()
xgbc = XGBClassifier()
adac = AdaBoostClassifier()
```

```
logisticr.fit(X_train, y_train)
dtc.fit(X_train, y_train)
rfc.fit(X_train, y_train)
gbc.fit(X_train, y_train)
#xgbc.fit(X_train, y_train)
adac.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

▾ AdaBoostClassifier

AdaBoostClassifier()

```
ypred_logisticr= logisticr.predict(X_test)
ypred_dtc = dtc.predict(X_test)
ypred_rfc = rfc.predict(X_test)
ypred_gbc = gbc.predict(X_test)
ypred_adac = adac.predict(X_test)
```

```
print("The LR train score is ", logisticr.score(X_train, y_train))
print("The LR test score is ", logisticr.score(X_test, ypred_logisticr))
print("The DT train score is ", dtc.score(X_train, y_train))
print("The DT test score is ", dtc.score(X_test, ypred_dtc))
print("The RF train score is ", rfc.score(X_train, y_train))
print("The RF test score is ", rfc.score(X_test, ypred_rfc))
print("The GB train score is ", gbc.score(X_train, y_train))
print("The GB test score is ", gbc.score(X_test, ypred_gbc))
```

```
print("The AB train score is ", adac.score(X_train, y_train))
print("The AB test score is ", adac.score(X_test, ypred_adac))
```

```
The LR train score is  1.0
The LR test score is  1.0
The DT train score is  1.0
The DT test score is  1.0
The RF train score is  1.0
The RF test score is  1.0
The GB train score is  1.0
The GB test score is  1.0
The AB train score is  1.0
The AB test score is  1.0
```

```
###!pip install pycaret
```

```
###! pip install jinja2
```

```
###! pip install markupsafe==2.0.1
```

```
import pycaret
```

```
from pycaret.classification import *
```

```
train.columns
```

```
Index(['lte_5g_category', 'time', 'packet_loss_rate', 'packet_delay', 'io_t',
       'lte_5g', 'gbr', 'non_gbr', 'ar_vr_gaming', 'healthcare',
       'industry_4_0', 'io_t_devices', 'public_safety', 'smart_city_and_home',
       'smart_transportation', 'smartphone', 'slice_type'],
      dtype='object')
```

```
model= setup(data= train, target= 'slice_type')
```

	Description	Value
0	Session id	3583
1	Target	slice_type
2	Target type	Multiclass
3	Target mapping	1: 0, 2: 1, 3: 2
4	Original data shape	(8958, 17)
5	Transformed data shape	(8958, 17)
6	Transformed train set shape	(6270, 17)
7	Transformed test set shape	(2688, 17)
8	Numeric features	16
9	Preprocess	True
10	Imputation type	simple
11	Numeric imputation	mean
12	Categorical imputation	mode
13	Fold Generator	StratifiedKFold
14	Fold Number	10
15	CPU Jobs	-1
16	Use GPU	False
17	Log Experiment	False
18	Experiment Name	clf-default-name
19	USI	8598

```
compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.2050
knn	K Neighbors Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0900
nb	Naive Bayes	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0530
dt	Decision Tree Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0680
ridge	Ridge Classifier	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0850
rf	Random Forest Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.4530
ada	Ada Boost Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.2760
gbc	Gradient Boosting Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.3980
et	Extra Trees Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.4140
xgboost	Extreme Gradient Boosting	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.4530
lightgbm	Light Gradient Boosting Machine	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.2560
svm	SVM - Linear Kernel	0.9337	0.0000	0.9337	0.9059	0.9149	0.8848	0.9013	0.0890
lda	Linear Discriminant Analysis	0.8721	0.9661	0.8721	0.8897	0.8696	0.7775	0.7918	0.0670

```
lrclassifier = create_model('lr')
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
2	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
3	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
5	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
6	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
7	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
8	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
9	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Mean	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Std	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

```
test.columns
```

```
Index(['lte_5g_category', 'time', 'packet_loss_rate', 'packet_delay', 'io_t',
       'lte_5g', 'gbr', 'non_gbr', 'ar_vr_gaming', 'healthcare',
       'industry_4_0', 'io_t_devices', 'public_safety', 'smart_city_and_home',
       'smart_transportation', 'smartphone', 'slice_type'],
      dtype='object')
```

```
X_test = test.drop(columns="slice_type")
```

```
y_test = test["slice_type"]
```

```
pred_holdout = predict_model(lrclassifier, data= X_test)
```

```
pred_holdout
```

	lte_5g_category	time	packet_loss_rate	packet_delay	io_t	lte_5g	gbr	non_gbr	ar_vr_gaming	healthcare	industry_4
0	15	17	0.001000	100	0	1	1	0	1	0	
1	14	18	0.000001	10	1	0	0	1	0	0	
2	11	7	0.001000	50	1	0	1	0	0	0	
3	20	14	0.001000	50	1	0	1	0	0	0	
4	2	22	0.001000	50	0	1	0	1	1	0	

```
pred_holdout.columns
```

```
Index(['lte_5g_category', 'time', 'packet_loss_rate', 'packet_delay', 'io_t',
       'lte_5g', 'gbr', 'non_gbr', 'ar_vr_gaming', 'healthcare',
       'industry_4_0', 'io_t_devices', 'public_safety', 'smart_city_and_home',
       'smart_transportation', 'smartphone', 'prediction_label',
       'prediction_score'],
      dtype='object')
```

```
pred_holdout["prediction_label"].value_counts()
```

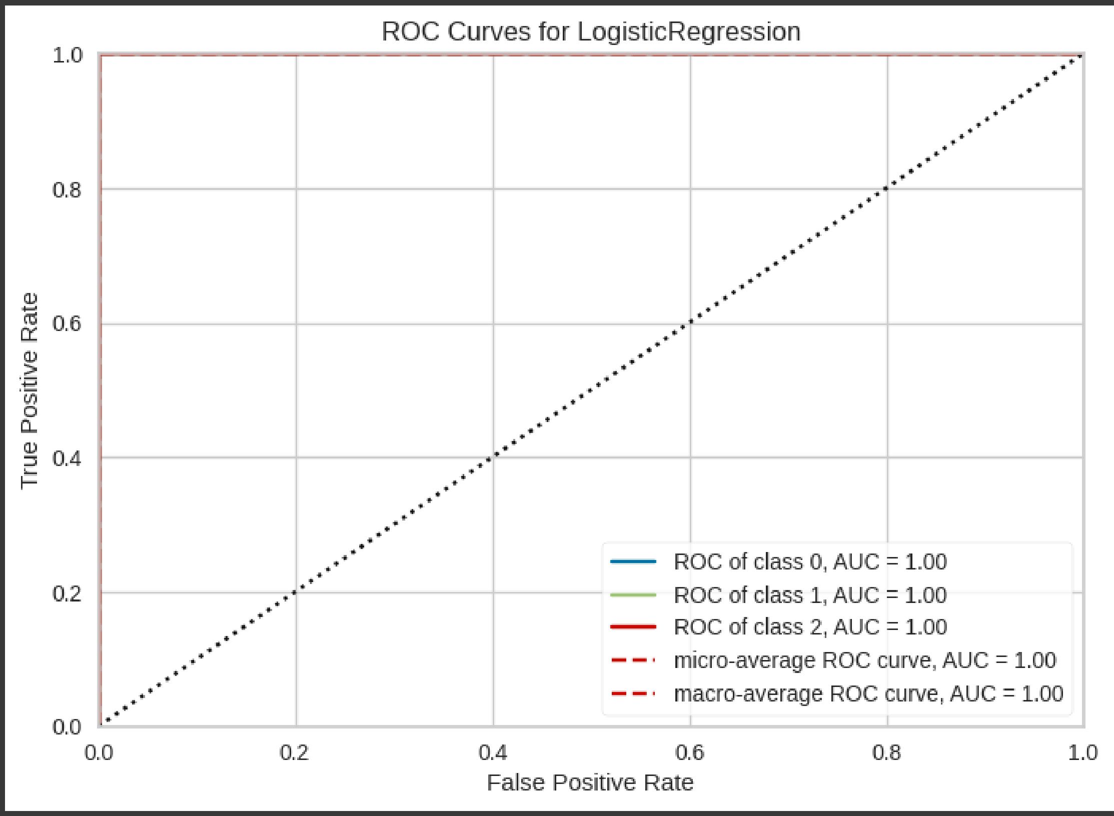
```
1    4763
2    2100
3    2097
Name: prediction_label, dtype: int64
```

```
###! pip install shap
```

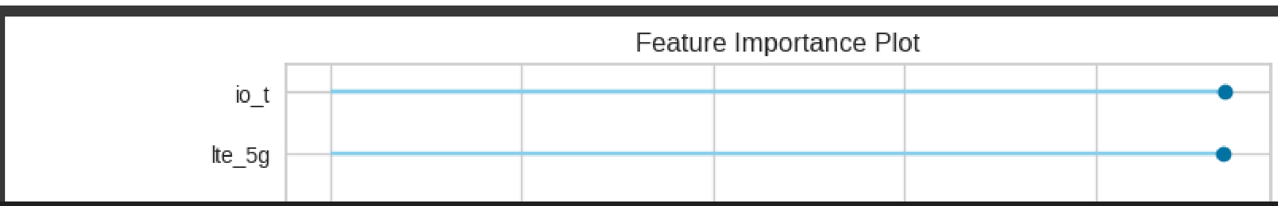
```
Installing collected packages: slicer, shap
Successfully installed shap-0.41.0 slicer-0.0.7
```

```
import shap
```

```
plot_model(lrclassifier)
```




```
plot_model(lrclassifier, plot = 'feature')
```




```
print(lrclassifier)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=1000,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=3583, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
logisticrgr = LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=1000,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=1801, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
###! pip install explainerdashboard
```

```
train.columns
```

```
Index(['lte_5g_category', 'time', 'packet_loss_rate', 'packet_delay', 'io_t',
       'lte_5g', 'gbr', 'non_gbr', 'ar_vr_gaming', 'healthcare',
       'industry_4_0', 'io_t_devices', 'public_safety', 'smart_city_and_home',
       'smart_transportation', 'smartphone', 'slice_type'],
      dtype='object')
```

```
test.columns
```

```
Index(['lte_5g_category', 'time', 'packet_loss_rate', 'packet_delay', 'io_t',
       'lte_5g', 'gbr', 'non_gbr', 'ar_vr_gaming', 'healthcare',
       'industry_4_0', 'io_t_devices', 'public_safety', 'smart_city_and_home',
       'smart_transportation', 'smartphone', 'slice_type'],
      dtype='object')
```

```
X_tr = train.drop(columns = "slice_type")
```

```
x_ts = test.drop(columns = "slice_type")
```

```
y_tr = train["slice_type"]
```

```
y_ts = test["slice_type"]
```

```
logisticrgr.fit(X_tr, y_tr)
```

```
y_pr = logisticrgr.predict(x_ts)
```

```
y_pr[:100]
```

```
array([1, 3, 2, 2, 1, 1, 2, 2, 3, 1, 1, 1, 2, 1, 3, 3, 1, 1, 1, 2, 3, 1,
       2, 3, 1, 2, 2, 1, 2, 3, 2, 1, 1, 1, 1, 2, 1, 1, 1, 3, 2, 3, 1, 1,
       1, 1, 3, 1, 3, 1, 2, 2, 1, 1, 1, 2, 3, 1, 2, 3, 1, 1, 1, 1, 1, 1,
       2, 2, 1, 1, 3, 2, 2, 1, 2, 2, 1, 1, 2, 2, 3, 1, 3, 3, 1, 1, 1, 1,
       2, 1, 1, 3, 2, 2, 1, 1, 3, 1, 1, 1], dtype=int8)
```

```
y_predict = pd.DataFrame(y_pr)
```

```
y_predict.rename(columns = { 0 : "Predict"}, inplace=True)
```

```
y_predict.value_counts()
```

```
Predict
1          4763
2          2100
3          2097
dtype: int64
```

```
###! pip install pickle
```

```
# Save the trained model as a pickle string.
import pickle

saved_model = pickle.dump(logisticrgr, open('/content/telecom.pickle','wb'))
```

0s completed at 3:38 PM