```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import IsolationForest
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.impute import SimpleImputer
import numpy as np
```

```
data = pd.read_csv('RT_IOT2022.csv')
```

```
data
```

	Unnamed: 0	id.orig_p	id.resp_p	proto	service	flow_duration	fwd_pkts_tot
0	0	38667	1883	tcp	mqtt	32.011598	9
1	1	51143	1883	tcp	mqtt	31.883584	9
2	2	44761	1883	tcp	mqtt	32.124053	9
3	3	60893	1883	tcp	mqtt	31.961063	9
4	4	51087	1883	tcp	mqtt	31.902362	9
...	...	...	...	...	...	...	...
123112	2005	59247	63331	tcp	-	0.000006	1
123113	2006	59247	64623	tcp	-	0.000007	1
123114	2007	59247	64680	tcp	-	0.000006	1
123115	2008	59247	65000	tcp	-	0.000006	1
123116	2009	59247	65129	tcp	-	0.000006	1

123117 rows × 85 columns

```
print(data.info())
```

```
 44  flow_pkts_payload.std     123117 non-null  float64
 45  fwd_iat.min               123117 non-null  float64
 46  fwd_iat.max               123117 non-null  float64
 47  fwd_iat.tot               123117 non-null  float64
 48  fwd_iat.avg               123117 non-null  float64
 49  fwd_iat.std               123117 non-null  float64
 50  bwd_iat.min               123117 non-null  float64
 51  bwd_iat.max               123117 non-null  float64
 52  bwd_iat.tot               123117 non-null  float64
 53  bwd_iat.avg               123117 non-null  float64
 54  bwd_iat.std               123117 non-null  float64
 55  flow_iat.min              123117 non-null  float64
 56  flow_iat.max              123117 non-null  float64
 57  flow_iat.tot              123117 non-null  float64
 58  flow_iat.avg              123117 non-null  float64
 59  flow_iat.std              123117 non-null  float64
 60  payload_bytes_per_second  123117 non-null  float64
 61  fwd_subflow_pkts          123117 non-null  float64
 62  bwd_subflow_pkts          123117 non-null  float64
 63  fwd_subflow_bytes         123117 non-null  float64
 64  bwd_subflow_bytes         123117 non-null  float64
 65  fwd_bulk_bytes            123117 non-null  float64
 66  bwd_bulk_bytes            123117 non-null  float64
 67  fwd_bulk_packets          123117 non-null  float64
 68  bwd_bulk_packets          123117 non-null  float64
 69  fwd_bulk_rate             123117 non-null  float64
 70  bwd_bulk_rate             123117 non-null  float64
 71  active.min                123117 non-null  float64
 72  active.max                123117 non-null  float64
 73  active.tot                123117 non-null  float64
 74  active.avg                123117 non-null  float64
 75  active.std                123117 non-null  float64
 76  idle.min                  123117 non-null  float64
 77  idle.max                  123117 non-null  float64
 78  idle.tot                  123117 non-null  float64
 79  idle.avg                  123117 non-null  float64
 80  idle.std                  123117 non-null  float64
 81  fwd_init_window_size      123117 non-null  int64
 82  bwd_init_window_size      123117 non-null  int64
 83  fwd_last_window_size      123117 non-null  int64
 84  Attack_type               123117 non-null  object
dtypes: float64(56), int64(26), object(3)
memory usage: 79.8+ MB
None
```

```
data.isnull().sum()
```

	0
Unnamed: 0	0
id.orig_p	0
id.resp_p	0
proto	0
service	0
...	...
idle.std	0
fwd_init_window_size	0
bwd_init_window_size	0
fwd_last_window_size	0
Attack_type	0

85 rows × 1 columns

dtype: int64

```
data.columns
```

```
Index(['Unnamed: 0', 'id.orig_p', 'id.resp_p', 'proto', 'service',
       'flow_duration', 'fwd_pkts_tot', 'bwd_pkts_tot', 'fwd_data_pkts_tot',
       'bwd_data_pkts_tot', 'fwd_pkts_per_sec', 'bwd_pkts_per_sec',
       'flow_pkts_per_sec', 'down_up_ratio', 'fwd_header_size_tot',
       'fwd_header_size_min', 'fwd_header_size_max', 'bwd_header_size_tot',
       'bwd_header_size_min', 'bwd_header_size_max', 'flow_FIN_flag_count',
       'flow_SYN_flag_count', 'flow_RST_flag_count', 'fwd_PSH_flag_count',
       'bwd_PSH_flag_count', 'flow_ACK_flag_count', 'fwd_URG_flag_count',
       'bwd_URG_flag_count', 'flow_CWR_flag_count', 'flow_ECE_flag_count',
       'fwd_pkts_payload.min', 'fwd_pkts_payload.max', 'fwd_pkts_payload.tot',
       'fwd_pkts_payload.avg', 'fwd_pkts_payload.std', 'bwd_pkts_payload.min',
       'bwd_pkts_payload.max', 'bwd_pkts_payload.tot', 'bwd_pkts_payload.avg',
       'bwd_pkts_payload.std', 'flow_pkts_payload.min',
       'flow_pkts_payload.max', 'flow_pkts_payload.tot',
       'flow_pkts_payload.avg', 'flow_pkts_payload.std', 'fwd_iat.min',
       'fwd_iat.max', 'fwd_iat.tot', 'fwd_iat.avg', 'fwd_iat.std',
       'bwd_iat.min', 'bwd_iat.max', 'bwd_iat.tot', 'bwd_iat.avg',
       'bwd_iat.std', 'flow_iat.min', 'flow_iat.max', 'flow_iat.tot',
       'flow_iat.avg', 'flow_iat.std', 'payload_bytes_per_second',
       'fwd_subflow_pkts', 'bwd_subflow_pkts', 'fwd_subflow_bytes',
       'bwd_subflow_bytes', 'fwd_bulk_bytes', 'bwd_bulk_bytes',
       'fwd_bulk_packets', 'bwd_bulk_packets', 'fwd_bulk_rate',
       'bwd_bulk_rate', 'active.min', 'active.max', 'active.tot', 'active.avg',
       'active.std', 'idle.min', 'idle.max', 'idle.tot', 'idle.avg',
       'idle.std', 'fwd_init_window_size', 'bwd_init_window_size',
       'fwd_last_window_size', 'Attack_type'],
      dtype='object')
```

```
le = LabelEncoder()
categorical_columns = ['proto', 'service', 'Attack_type']
for col in categorical_columns:
    data[col] = le.fit_transform(data[col])
```

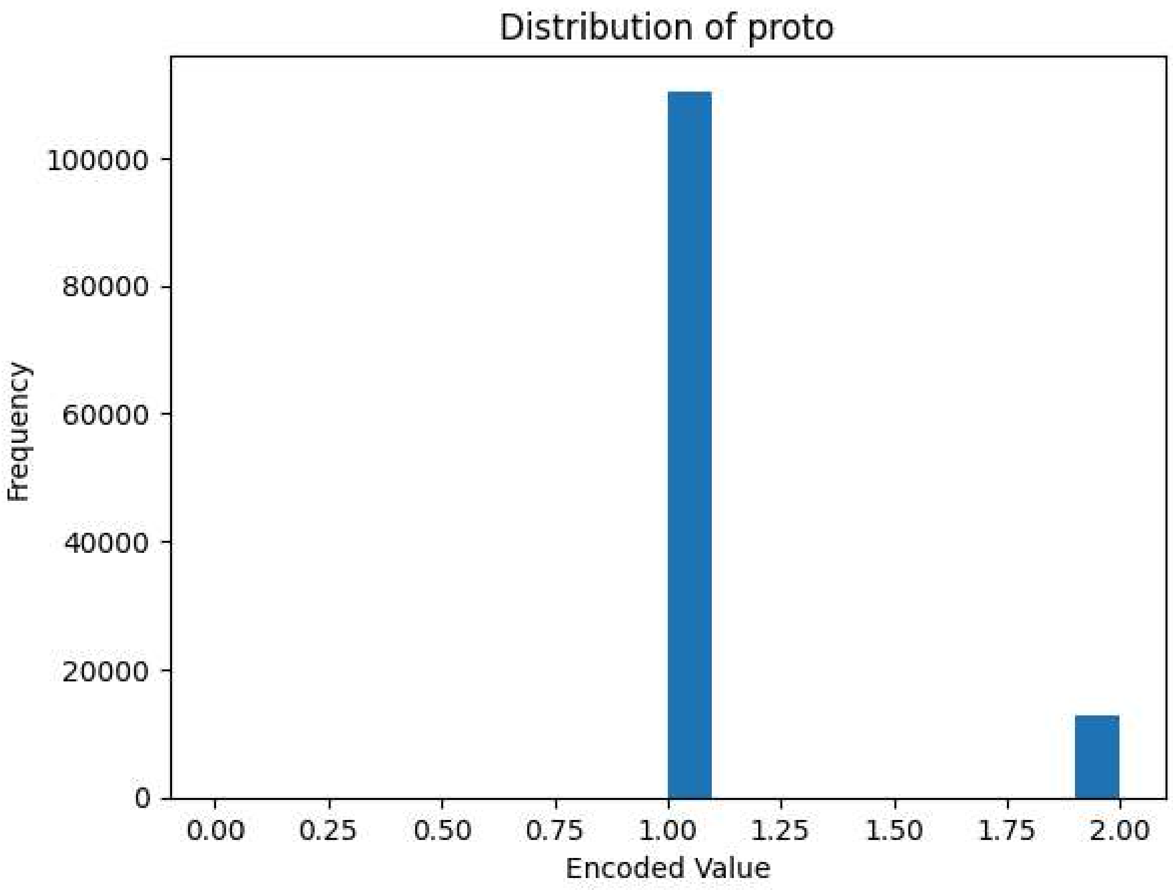
```
x = data.drop(columns=['Unnamed: 0'])
```

```
x
```

	id.orig_p	id.resp_p	proto	service	flow_duration	fwd_pkts_tot	bwd_pkts_t
0	38667	1883	1	5	32.011598	9	
1	51143	1883	1	5	31.883584	9	
2	44761	1883	1	5	32.124053	9	
3	60893	1883	1	5	31.961063	9	
4	51087	1883	1	5	31.902362	9	
...	...	...	...	...	...	...	
123112	59247	63331	1	0	0.000006	1	
123113	59247	64623	1	0	0.000007	1	
123114	59247	64680	1	0	0.000006	1	
123115	59247	65000	1	0	0.000006	1	
123116	59247	65129	1	0	0.000006	1	

123117 rows × 84 columns

```
x['proto'].plot(kind='hist', bins=20)
plt.title('Distribution of proto')
plt.xlabel('Encoded Value')
plt.ylabel('Frequency')
plt.show()
```



```
imputer = SimpleImputer(strategy='mean') # You can change the strategy if needed
x_imputed = imputer.fit_transform(x)
```

```
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)
```

```
x_scaled
```

```
array([[ 0.21120223,  0.16526578, -0.33560483, ...,  2.40619686,
        -0.0395628 ,  0.08276416],
       [ 0.8654049 ,  0.16526578, -0.33560483, ...,  2.40619686,
        -0.0395628 ,  0.08276416],
       [ 0.53075265,  0.16526578, -0.33560483, ...,  2.40619686,
        -0.0395628 ,  0.08276416],
       ...,
       [ 1.29035347, 12.11214761, -0.33560483, ..., -0.27346328,
         0.04316096,  2.51990234],
       [ 1.29035347, 12.17302635, -0.33560483, ..., -0.27346328,
         0.04316096,  2.51990234],
       [ 1.29035347, 12.19756809, -0.33560483, ..., -0.27346328,
         0.04316096,  2.51990234]])
```

```
x_train, x_test = train_test_split(x_scaled, test_size=0.2, random_state=42)
```

Start coding or generate with AI.

```
model = IsolationForest(contamination=0.1, random_state=42)
model.fit(x_train)
```

```
▾ IsolationForest
IsolationForest(contamination=0.1, random_state=42)
```

```
y_pred_test = model.predict(x_test)
```

```
y_pred_test
```

```
array([ 1,  1,  1, ...,  1, -1,  1])
```

```
y_pred_test = [1 if x==1 else 0 for x in y_pred_test]
```

```
y_pred_test
```

```
        0,
        1,
        1,
        1,
        1,
        1,
        0,
        1,
        1,
        0,
        1,
        1,
        1,
        1,
        0,
        1,
        1,
        1,
        1,
        ...]
```

```
y_test = [1]*len(y_pred_test)
```

```
y_test
```

```
        1,
        1,
        1,
        1,
        1,
        1,
        1,
        1,
        1,
        1,
        1,
        1,
        1,
        1,
        1,
        1,
        1,
        1,
        1,
        1,
        1,
        1,
        1,
        1,
        ...]
```

```
print('Classification Report:')
print(classification_report(y_test, y_pred_test))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       1.00      0.90      0.95     24624

    accuracy                           0.90     24624
   macro avg       0.50      0.45      0.47     24624
weighted avg       1.00      0.90      0.95     24624

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: Unde
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: Unde
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: Unde
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
conf_matrix = confusion_matrix(y_test, y_pred_test)
```