Test: Machine Learning (Predicting Equipment Failure)

Submission : Abhijith Kumar
9740055714

Problem Statement:

The data is representative of a telecom equipment. Predicting the equipment's failure helps the business plan to service the equipment before it fails. You are given this challenge to be the superstar who can save millions of dollars for the business.

The target variable is **Packet Drop.** The target variable is available as both continuous and categorical. Use your discretion backed by Statistical reasoning to choose the right target variable and build a model to predict **Packet Drop.** The target variables to consider are **PacketDrop_Current_Value** or **PacketDrop_Severity**.

The data dictionary is not provided for a reason. All you need to know is Packet Drop is the target variable.

___

The initial step is to understand the variables that affect our packet drop target variable, this comes down to network latency, congestion , server health , memory etc.
So we would be giving these variables if they do come up in our analysis.

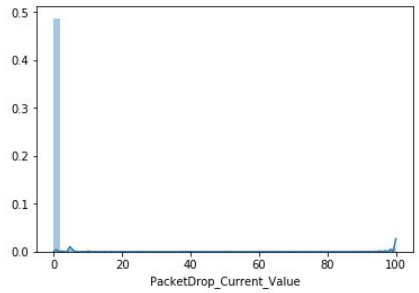
Selecting the target variable

This would be a business decision as the business would usually be giving us the variable they want to predict or which has a higher importance to them. Since we are given the choice here, the approach to selecting the variable would involve looking at the distribution of two.i.e we would be performing a univariate analysis on the two variables **PacketDrop_Current_Value** or **PacketDrop_Severity**.

We would be using distplot from seaborn library as it comines histogram with kde,rug & PDE in a nice graph. Also we would be looking at the boxplot and summary of the data.
Looking at the distplot of the two variables it is very clear that they are highly skewed.
Note: The categorical variable has been label encoded. It is a ordinal categorical variable so we would be giving 'High' the value 4.

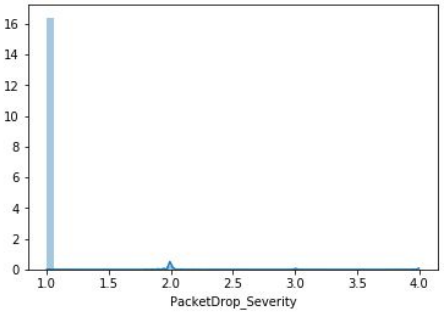
```
sns.distplot(df['PacketDrop_Current_Value'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb637aa6780>
```




```
sns.distplot(df.PacketDrop_Severity)
```

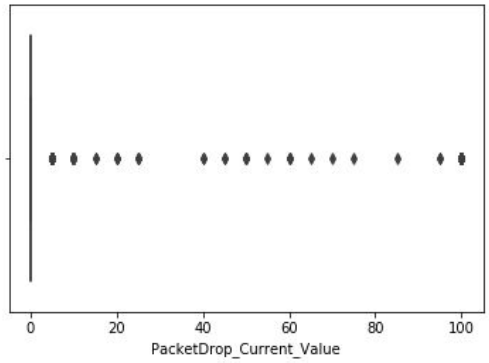
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f90
```



Boxplot: The boxplot of the two variables is as shown below.

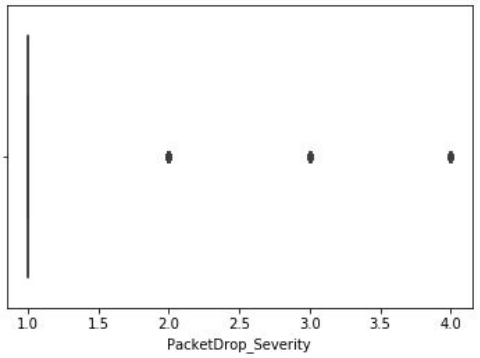
```
sns.boxplot(df.PacketDrop_Current_Value)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x
```




```
sns.boxplot(df.PacketDrop_Severity)
```

```
<matplotlib.axes._subplots.AxesSubplot at
```



We can see that the variable **PacketDrop_Current_value** has a mean at 0 and most of the data is distributed along it.
Summary : Looking at the summary of the two variables we can see that 75th percentile of **PacketDrop_Current_value** is 0 and that majority class in **PacketDrop_Severity** is **up**.

```
df.PacketDrop_Current_Value.describe()
```

```
count    14832.000000
mean         1.432039
std         11.493648
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max        100.000000
Name: PacketDrop_Current_Value, dtype: float64
```

```
encode = le.fit_transform(df.PacketDrop_Severity)
df.PacketDrop_Severity.value_counts()
```

```
up          14603
down          191
critical       21
warning        17
Name: PacketDrop_Severity, dtype: int64
```

From the above we can conclude that the variable **PacketDrop_Severity** would be a better variable choice as it captures more information about the data and taking the variable **PacketDrop_Current_value** would bring high bias to the model.

Approach

From the previous section we had selected a categorical variable as our target variable so the problem becomes a classification problem. The data is highly skewed so we can do the following to balance out the model:

1. Under/Over sampling
2. Selecting appropriate classification threshold
3. Assigning higher weightage to the minority class
4. Using an anomaly detection algorithm

Due to the time constraint we would be applying only a few of the approaches in the final submission.

Cleaning

The most crucial step in the modelling is preparing the data. The importance can be explained by a common saying “Junk in Junk out”. Later we would be formatting the variables to appropriate continuous and categorical types.
The check for missing values is carried out.
There are 44 rows in total that have missing values so they have been dropped considering the fact that the data is sparse for them.
The columns with standard deviation 0 have been removed.

After the above step 20 redundant columns were removed. So we have 51 columns from 71 , including OneHotEncoded columns.

Outlier removal is also carried out taking out all the values that are not within 2 standard deviations from the distribution.

Feature scaling was done using Fastai library as it would bring the model to convergence quicker.

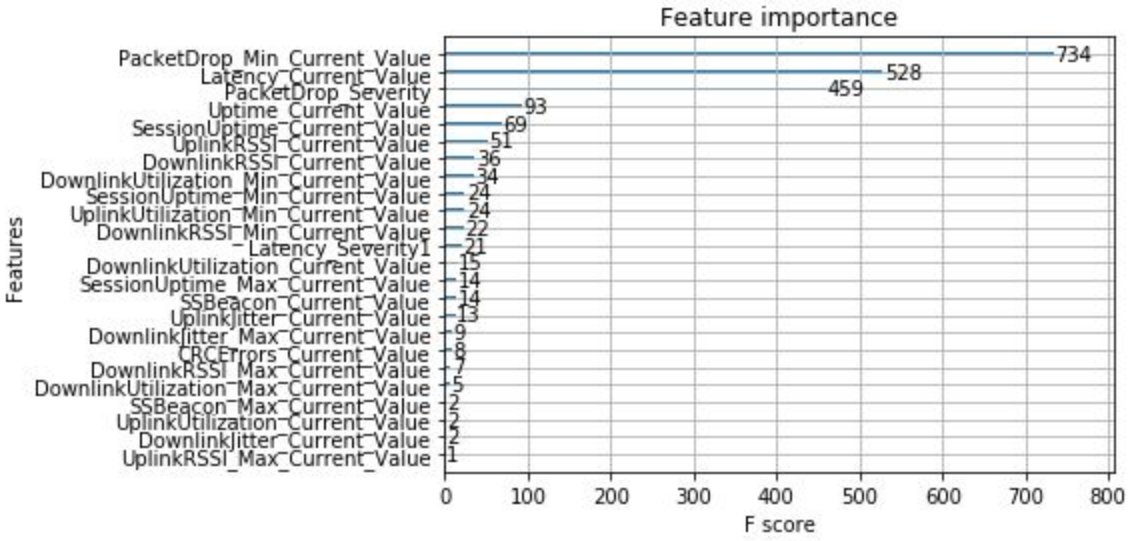
Feature Engineering

We would be splitting the Time field into date and time.

I have created a function called ohe that does one hot encoding for the categorical variables and drops one column to overcome dummy variable trap and then merges it with the input dataframe.

Feature Importance

Below is the feature importance of the data.
PacketDrop_min_current_vakue has the highest importance.



Model Selection

Based on the principle of no free lunch the model was fit for all the models like SVC, XGboost, RandomForset , Lightgbm, logistic regression etc and the metric used for scoring was the F1 score.
Neural Networks are not used due to its lack of explainability.
F1 score was selected as accuracy would not be a good measure for models with unbalanced class distribution. It would score high even when the model was performing well in reality.
For example if it is a binary classifier and we only have 0.01 negative class in our test data , if the model was only predicting true for the negative class it would be having a very high accuracy, close to 99%.
So the F1 is a better indicator and it is a measure that encompasses the precision and recall score of a model.
Higher the value better(closer to 1) is the result.

Our base model without hyperparameter tuning gave the F1 score of 0.45.
After optimizing it through grid search we get.

In [35]:
```
xg = XGBClassifier(learning_rate=0.01,max_depth=1,gamma=0.01,reg_alpha=0.1,reg_lambda=0.1)
xg.fit(X_train,y_train)
y_pred = xg.predict(X_test)
#xgbmdl_f.fit(X_train,y_train)
```

In [36]:
```
f1_score(y_test,y_pred,average='macro')
```

Out[36]:
```
0.5401098901098901
```

In [37]:
```
accuracy_score(y_test,y_pred)
```

Out[37]:
```
0.9988726042841037
```