**Language:** Python3.8
**Framework:** PyTorch
**Libraries:** Numpy, Pandas, PyTorch, Matplotlib, Sklearn
**Environment:** Jupyter Notebook

**Task:**
Reduce dimensionality of Malware Data from 63 to 15 using **Autoencoders**.

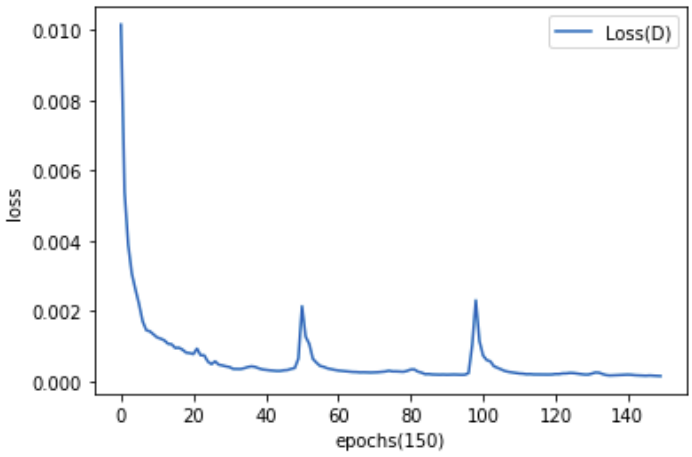
**Outcomes:**
Dimensionality was reduced from 63 features to 15.
MLP AUC: 98.43%

**Models:**

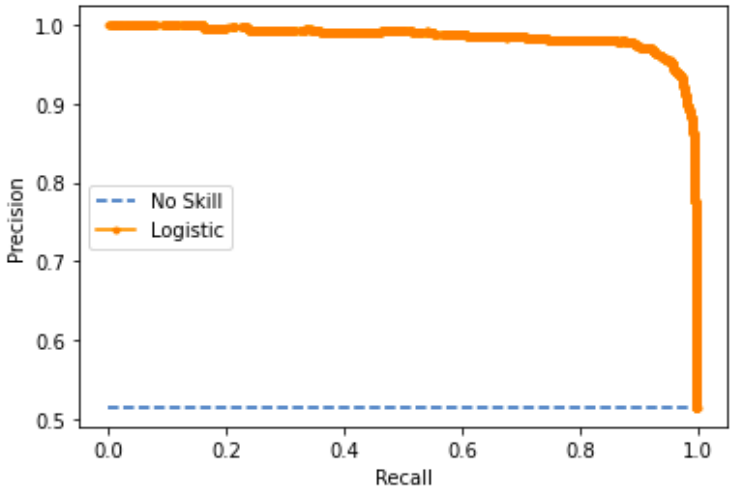
```
encoded_dims = 15
current_dims = 63

class Encoder(nn.Module):
  def __init__(self):
    super().__init__()
    self.encoder = nn.Sequential(
        #N, 63
        nn.Linear(current_dims, current_dims*2),
        nn.BatchNorm1d(current_dims*2),
        nn.LeakyReLU(),
        nn.Linear(current_dims*2, current_dims),
        nn.BatchNorm1d(current_dims),
        nn.LeakyReLU(),
        nn.Linear(current_dims,encoded_dims),
    )

  def forward(self, x):
    encoded = self.encoder(x)
    return encoded

class Decoder(nn.Module):
  def __init__(self):
    super().__init__()
    self.decoder = nn.Sequential(
        nn.Linear(encoded_dims, current_dims),
        nn.BatchNorm1d(current_dims),
        nn.LeakyReLU(),
        nn.Linear(current_dims, current_dims*2),
        nn.BatchNorm1d(current_dims*2),
        nn.LeakyReLU(),
        nn.Linear(current_dims*2,current_dims),
```

```
        nn.Sigmoid()
    )
  def forward(self, x):
    decoded = self.decoder(x)
    return decoded
```

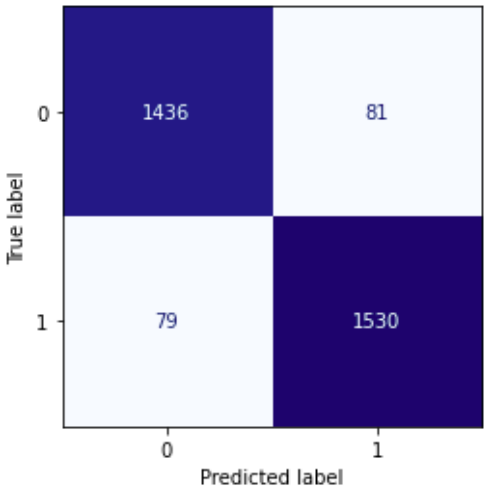
## Training Loss:




## MLP on Encoded Data:




```
F1: 0.9521765787860208
AUC: 0.9843605109253212
Accuracy 0.9500959692898272
```


1436
81
79
1530
0
1
True label
0
1
Predicted label