CSC 730 Assignment 05
AD Bench

Mangesh Sakordekar

Overview

The aim of this assignment is to:

- Install ADBench.
- Generate labels for the skewed-MNIST dataset with two least represented classes marked as anomalies.
- Select 2 supervised algorithms that are built into ADBench, and apply them to the data.
- Characterize the resulting performance.

Installing ADBench

ADBench was installed following the instructions provided on the git repo. I did require help from my classmates and the internet to get around the errors to get it installed properly.

Data Processing

The data was read in from the skewed-MNIST dataset. Labels were generated for this data to classify two lowest represented classes, which are 1 and 7, as anomalies. Then the data was shuffled and split into training and testing dataset. 80% of the data was used for training and the rest 20% was used for testing.

```
lbls = np.asarray([1 if i == 1 or i == 7 else 0 for i in y])
```

```
X_train = X[:9795, :]
y_train = lbls[:9795]
X_test = X[9795:, :]
y_test = lbls[9795:]
```

Figure 1: Creating Labels and Splitting Data

Algorithm 1 : XG Boost

I selected XG Boost as one of the two algorithms. Using ADBench the model was initialised and trained to fit the dataset created. The model predicted with an accuracy of around 99.6%. The implementation and results of run can be seen below.

```
model = supervised(seed=42, model_name='XGB')  # initialization
model.fit(X_train, y_train)  # fit
score = model.predict_score(X_test)  # predict
```

```
preds_XGB = [1 if i > 0.5 else 0 for i in score]
```

```
cm = confusion_matrix(y_test, preds_XGB)
cm
```

```
array([[2439,    0],
       [   8,    2]], dtype=int64)
```

```
cm.trace() / np.sum(cm)
```

```
0.9967333605553287
```

Figure 2: XG Boost

Algorithm 1 : Cat Boost

Cat Boost was the other algorithm selected. Using ADBench the model was initialised and trained to fit the dataset created. The model predicted with an accuracy of around 99.6% as well. The implementation and results of run can be seen below.

```
model = supervised(seed=42, model_name='CatB')  # initialization
model.fit(X_train, y_train)  # fit
score = model.predict_score(X_test)  # predict
```

```
141:    learn: 0.0054092        total: 8.67s    remaining: 52.4s
142:    learn: 0.0053814        total: 8.72s    remaining: 52.3s
143:    learn: 0.0053724        total: 8.78s    remaining: 52.2s
144:    learn: 0.0053142        total: 8.84s    remaining: 52.1s
145:    learn: 0.0052952        total: 8.9s     remaining: 52s
146:    learn: 0.0052199        total: 8.95s    remaining: 51.9s
147:    learn: 0.0051307        total: 9.01s    remaining: 51.9s
148:    learn: 0.0051074        total: 9.08s    remaining: 51.8s
149:    learn: 0.0050846        total: 9.13s    remaining: 51.8s
150:    learn: 0.0050693        total: 9.19s    remaining: 51.7s
151:    learn: 0.0049394        total: 9.25s    remaining: 51.6s
152:    learn: 0.0049149        total: 9.31s    remaining: 51.5s
153:    learn: 0.0048942        total: 9.36s    remaining: 51.4s
154:    learn: 0.0047809        total: 9.42s    remaining: 51.4s
155:    learn: 0.0047702        total: 9.48s    remaining: 51.3s
156:    learn: 0.0046719        total: 9.54s    remaining: 51.2s
157:    learn: 0.0045557        total: 9.6s     remaining: 51.2s
158:    learn: 0.0045202        total: 9.66s    remaining: 51.1s
159:    learn: 0.0045038        total: 9.71s    remaining: 51s
```

```
preds_CatB = [1 if i > 0.5 else 0 for i in score]
```

```
cm = confusion_matrix(y_test, preds_CatB)
cm
```

```
array([[2439,    0],
       [   8,    2]], dtype=int64)
```

```
cm.trace() / np.sum(cm)
```

```
0.9967333605553287
```

Figure 3: Cat Boost

Conclusion

Both the models end up with a pretty high accuracy, but looking at the confusion matrices we can see that the models tend to predict everything as not an anomaly as doing so will lead to a higher accuracy.