A Report On

One-Class Classification

Prepared For:

Prof. Navneet Goyal

Instructor-in-Charge

(BITS F464: Machine Learning)

Prepared By:

Aaditya Kulkarni (2021A7PS0426P)

Aditya Deshpande (2021A7PS2681P)

Rahul Srivastava (2021A2PS2630P)

TABLE OF CONTENTS

Introduction

In the field of machine learning, one-class classification, also called unary classification or class modelling, aims to identify specific objects within a larger dataset. Unlike typical classification tasks that involve distinguishing between multiple classes using a diverse training set, one-class classification relies on learning primarily from a training set consisting exclusively of objects from the target class. Though some variants of one-class classifiers incorporate counter-examples to refine the classification boundary, the core objective remains to detect objects within a particular class. This presents a distinct challenge compared to traditional classification because it requires identifying objects of interest within a broader dataset without contrasting classes for comparison. One-class classification finds applications in various domains, such as monitoring helicopter components, predicting motor failures, and assessing the normal operational status of nuclear plants. In these cases, the training data often lacks examples of extreme situations, making it necessary to rely solely on information from normal operations.

While many one-class classification approaches focus on identifying and removing a few outliers or anomalies, an alternative approach involves learning a specific subset of data where the single class covers a small yet coherent portion. This can be achieved through an information bottleneck method, which extracts essential features while discarding unnecessary information, resulting in a more focused representation of the target class within the dataset. Top of Form

In this report, we aim to explore One-Class Classification, its applications, and how it differs from binary classification problems. We will identify several real-world applications of one-class classification and highlight the fundamental differences between one-class classification and binary classification.

Applications

One-Class Classification has found applications in numerous domains:

Fraud Detection

In the world of financial transactions, One-Class Classification is used to carefully look at patterns in normal transactions to understand how they usually work. With this knowledge, it becomes really good at spotting any transactions that don't fit the usual pattern. These unusual transactions could be signs of fraud like someone trying to take money without permission or using someone else's identity. By quickly identifying these odd transactions, One-Class Classification can help banks and financial institutions keep their customers' money safe and protect the integrity of the financial system.

Network Anomaly Detection

In the world of cybersecurity, keeping our digital systems safe from attacks is more important than ever. That's where One-Class Classification comes in. It's like a frontline guard against cyber threats. By looking at things like network traffic and system logs, One-Class Classification keeps a close eye out for anything unusual that might signal a cyberattack. Whether it's a sudden increase in data transfer, strange access attempts, or unusual behaviour in the system, One-Class Classification is always on alert, ready to raise the alarm if it detects something suspicious. With One-Class Classification on duty, organizations can strengthen their digital defences, prevent cyberattacks, and protect their sensitive information from hackers.

Healthcare

In healthcare, One-Class Classification is like a reliable helper for doctors and medical staff. It uses lots of patient information to find unusual things in medical records that might indicate

health problems or rare diseases. This could be anything from unusual vital signs to abnormality in ECG data. One-Class Classification can help doctors make better diagnoses.

Quality Control

In manufacturing, it's super important to make sure products are top-notch to keep customers happy and maintain a good reputation. One-Class Classification helps with this by carefully checking how things are made and what they should look like. It looks for any problems or mistakes that might pop up during production. Whether it's finding issues with machines, spotting mistakes in product sizes, or noticing when things aren't being made the usual way, One-Class Classification is like a quality control superhero. It helps manufacturers make sure their products are of good quality so that customers are happy and keep coming back for more.

Environmental Monitoring

As we become increasingly concerned about the state of our environment and the effects of climate change, the importance of effective environmental monitoring solutions is becoming clearer. One-Class Classification has stepped up as a valuable tool in this effort. By carefully analyzing vast amounts of environmental data, it helps pinpoint anomalies that could signal environmental hazards or disruptions to ecosystems. Whether it's spotting pollution spikes, irregular weather patterns, or changes in ecosystem dynamics, One-Class Classification is vital for maintaining environmental health. Its proactive monitoring empowers decision-makers to take prompt action to mitigate risks and conserve natural resources for future generations.

Differences from Binary Classification

One-Class Classification and Binary Classification are two fundamental methodologies in machine learning, each offering unique strategies and applications. One-Class Classification distinguishes itself by its focused approach to categorizing instances into a singular class, typically identified as the 'normal' class. In this method, instances from other classes referred to as outliers or anomalies, are disregarded. This specialized approach finds widespread utility in anomaly detection tasks, where the primary objective is to uncover rare or unusual occurrences within datasets. Conversely, Binary Classification operates on the principle of distinguishing between two distinct classes, often labelled as 'positive' and 'negative', using labelled data for training. In the context of One-Class Classification, the training dataset exclusively comprises instances from the target class, enabling the model to discern and encapsulate its unique characteristics more effectively. In contrast, Binary Classification leverages instances from both classes to better understand their inherent differences and establish an optimal decision boundary. This fundamental dichotomy underscores the diverse methodologies and contexts in which these classification techniques are applied, highlighting their versatility and importance in addressing a wide range of practical challenges across various industries and domains. Through their specialized approaches, One-Class Classification and Binary Classification play crucial roles in advancing machine learning solutions and driving innovation in diverse fields.

Imbalance Datasets

Here are some reasons why addressing imbalanced class datasets is important:

1. Real-world applicability: Imbalanced datasets are common in many real-world scenarios, such as fraud detection, anomaly detection, and rare event prediction.

2. Performance evaluation: Imbalanced datasets can significantly affect the performance evaluation of one-class classification models. Traditional performance metrics such as accuracy can be misleading in the presence of imbalanced classes. Including discussions on appropriate evaluation metrics like precision, recall, F1-score, and area under the ROC curve (AUC-ROC) will provide a comprehensive understanding of model performance.

3. Model bias: Imbalanced datasets can lead to biased models that favour the majority class. It's essential to explore techniques such as resampling methods (oversampling, undersampling), algorithmic approaches (cost-sensitive learning), and ensemble methods to mitigate bias and improve the model's ability to detect instances of the minority class.

4. Feature importance: Imbalanced datasets may cause models to prioritize features that distinguish the majority class, neglecting important characteristics of the minority class. Analyzing feature importance and understanding how features contribute to the model's decision-making process can provide insights into potential biases and help improve model fairness.

5. Data preprocessing: Preprocessing techniques such as data resampling, feature engineering, and data augmentation are often employed to address class imbalance.

Algorithms

Several algorithms have been proposed for One-Class Classification. Here are a few of them:

One-Class Support Vector Machine

One-Class SVM is a specialized algorithm in the field of machine learning primarily utilized for anomaly detection. Unlike traditional Support Vector Machines (SVMs) that are designed for binary classification tasks, One-Class SVM is explicitly tailored to handle scenarios where data from the anomalous class is scarce or difficult to obtain.

The primary objective of One-Class SVM is to delineate a boundary around the 'normal' instances within a single class in the feature space. This boundary encapsulates as many normal instances as possible while excluding outliers. During the training process, One-Class SVM learns to separate the normal instances from the origin in the feature space, aiming to maximize the margin around them. In practice, when presented with new instances, One-Class SVM classifies them as either belonging to the normal class (inlier) or as outliers (anomalies) based on their proximity to the learned boundary.

K-Nearest Neighbour

K-Nearest Neighbors (KNN) is a versatile algorithm used in both classification and regression tasks across various domains. Its simplicity and effectiveness make it a popular choice for many machine learning applications. At its core, KNN operates on the principle of similarity: similar data points are expected to belong to the same class or exhibit similar characteristics. When applied to anomaly detection, KNN considers each data point's proximity to its neighbors in the feature space. By analyzing the labels of its 'k' nearest neighbors, KNN classifies each data point based on the majority class label among its neighbors. The decision boundary in KNN is

implicitly defined by the distribution of the training data. Data points that fall outside the majority class regions or exhibit unusual patterns relative to their neighbors are more likely to be classified as anomalies. Therefore, KNN is particularly effective in detecting local anomalies, where anomalies are distinct from the surrounding normal instances.

Isolation Forest

Isolation Forest is an ensemble learning method designed for anomaly detection, particularly well-suited for handling imbalanced datasets where the positive class represents rare instances or anomalies. The algorithm operates by isolating the instances of the positive class (anomalies) through a process of recursively partitioning the feature space using random splits.

Unlike traditional decision tree algorithms that aim to classify instances into specific classes, Isolation Forest focuses on isolating anomalies by randomly selecting feature splits and partitioning the data space. This random partitioning process allows Isolation Forest to efficiently separate anomalies from normal instances, as anomalies are more likely to be isolated in smaller partitions due to their rarity. By leveraging the principle of isolation, Isolation Forest can detect anomalies with fewer partitions, making it computationally efficient and effective even on high-dimensional datasets.

Local Outlier Factor

Local Outlier Factor (LOF) is a popular algorithm for detecting outliers or anomalies in one-class classification scenarios. Unlike global outlier detection methods that consider the entire dataset, LOF focuses on identifying outliers based on their local neighbourhood density. In one-class classification, LOF works by assessing the density of each data point relative to its neighbouring points. It calculates a score for each data point based on how isolated or "outlying" it is compared to its local neighbourhood. Points with significantly lower density compared to their

neighbours are considered potential outliers. LOF operates under the assumption that outliers have lower density compared to their neighbours, making them stand out as anomalies in the dataset. By assigning an outlier score to each data point, LOF ranks them based on their degree of outlierness, allowing for the identification of the most anomalous instances in the dataset. One of the key advantages of LOF is its ability to detect outliers in high-dimensional spaces and handle datasets with varying densities effectively.

DATASET 1: Network Anomaly Detection

Local Outlier Factor:

```
import numpy as np
import pandas as pd
from sklearn.neighbors import LocalOutlierFactor
from sklearn.metrics import classification_report, confusion_matrix
data = pd.read_csv('http.csv')
majority_class = data[data['attack'] == 0]
minority_class = data[data['attack'] == 1]
X_train = majority_class.drop('attack', axis=1)
lof = LocalOutlierFactor(n_neighbors=20, contamination='auto')
lof.fit(X_train)
X_test = data.drop('attack', axis=1)
y_pred = lof.fit_predict(X_test)
y_pred[y_pred == 1] = 0
y_pred[y_pred == -1] = 1
print(confusion_matrix(data['attack'], y_pred))
print(classification_report(data['attack'], y_pred))
```

✓ 19.1s

```
[[516377  48910]
 [  2145     66]]
              precision    recall  f1-score   support

           0       1.00      0.91      0.95    565287
           1       0.00      0.03      0.00      2211

    accuracy                           0.91    567498
   macro avg       0.50      0.47      0.48    567498
weighted avg       0.99      0.91      0.95    567498
```

neighbours=20 and contamination=’auto’ was giving better results.

-

K-Nearest Neighbours:

```
data=pd.read_csv('http.csv')
df=pd.DataFrame(data)
df['attack'].value_counts()
df_majority = df[df.attack==0]
df_minority = df[df.attack==1]
X_train = df_majority.drop('attack', axis=1)
y_train = np.zeros(len(X_train))
k = 5
knn = NearestNeighbors(n_neighbors=k)
knn.fit(X_train)
X_test = df.drop('attack', axis=1)
y_test = df['attack']
distances, _ = knn.kneighbors(X_test)
anomaly_score = distances.mean(axis=1)
print(anomaly_score.mean())
y_pred = [1 if x > 0.2 else 0 for x in anomaly_score]

print(classification_report(y_test, y_pred))
```

✓ 6.9s

```
0.0080818430814220 64
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    565287
           1       0.89      1.00      0.94      2211

    accuracy                           1.00    567498
   macro avg       0.94      1.00      0.97    567498
weighted avg       1.00      1.00      1.00    567498
```

Isolation Forests:

```
# Import necessary libraries
import numpy as np
import pandas as pd
from sklearn.ensemble import IsolationForest
from sklearn.metrics import confusion_matrix, classification_report
```

✓ 1.4s

```
df=pd.read_csv('http.csv')
df_majority = df[df['attack'] == 0]
model = IsolationForest(contamination=0.0104, random_state=42)
X_train = df_majority.drop(['attack'], axis=1)
y_train = df_majority['attack']
model.fit(X_train)
y_pred = model.predict(df.drop(['attack'], axis=1))
y_pred = [1 if x == -1 else 0 for x in y_pred]
print(classification_report(df['attack'], y_pred))
```

✓ 6.2s

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99    565287
           1       0.27      0.98      0.42      2211

    accuracy                           0.99    567498
   macro avg       0.63      0.98      0.71    567498
weighted avg       1.00      0.99      0.99    567498
```

DATASET 2: Detection of Financial Frauds

Local Outlier Factor

```
import numpy as np
import pandas as pd
from sklearn.neighbors import LocalOutlierFactor
from sklearn.metrics import classification_report, confusion_matrix

data = pd.read_csv('creditcard.csv')

majority_class = data[data['Class'] == 0]
minority_class = data[data['Class'] == 1]

X_train = majority_class.drop('Class', axis=1)

lof = LocalOutlierFactor(n_neighbors=20, contamination='auto')
lof.fit(X_train)

X_test = data.drop('Class', axis=1)
y_pred = lof.fit_predict(X_test)

y_pred[y_pred == 1] = 0
y_pred[y_pred == -1] = 1

print(confusion_matrix(data['Class'], y_pred))
print(classification_report(data['Class'], y_pred))
```

```
[[280108   4207]
 [   393     99]]
              precision    recall  f1-score   support

           0       1.00      0.99      0.99    284315
           1       0.02      0.20      0.04       492

    accuracy                           0.98    284807
   macro avg       0.51      0.59      0.52    284807
weighted avg       1.00      0.98      0.99    284807
```

K-Nearest Neighbours

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import NearestNeighbors
from sklearn.metrics import classification_report, confusion_matrix

data = pd.read_csv('creditcard.csv')

majority_class = data[data['Class'] == 0]
minority_class = data[data['Class'] == 1]

X_train = majority_class.drop('Class', axis=1)
y_train = np.zeros(len(X_train))

k = 5
knn = NearestNeighbors(n_neighbors=k)
knn.fit(X_train)

X_test = data.drop('Class', axis=1)
y_test = data['Class']

distances, _ = knn.kneighbors(X_test)
anomaly_score = distances.mean(axis=1)
print(anomaly_score.mean())

y_pred = [1 if x > 50 else 0 for x in anomaly_score]
print(classification_report(y_test, y_pred))
```

```
13.613461550505574
              precision    recall  f1-score   support

           0       1.00      0.97      0.98    284315
           1       0.01      0.24      0.02       492

    accuracy                           0.97    284807
   macro avg       0.51      0.60      0.50    284807
weighted avg       1.00      0.97      0.98    284807
```

Isolation Forests

```python
import numpy as np
import pandas as pd
from sklearn.ensemble import IsolationForest
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns

df=pd.read_csv('creditcard.csv')
df_majority = df[df['Class'] == 0]

model = IsolationForest(contamination=0.02, random_state=42)

model.fit(df_majority.drop('Class', axis=1))

predictions = model.predict(df.drop('Class', axis=1))

predictions[predictions == 1] = 0
predictions[predictions == -1] = 1

print("Confusion Matrix:")
print(confusion_matrix(df['Class'], predictions))
print("\nClassification Report:")
print(classification_report(df['Class'], predictions))
```

```
Confusion Matrix:
[[278628   5687]
 [   145    347]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99    284315
           1       0.06      0.71      0.11       492

    accuracy                           0.98    284807
   macro avg       0.53      0.84      0.55    284807
weighted avg       1.00      0.98      0.99    284807
```

Conclusion

Quantitative Comparison:

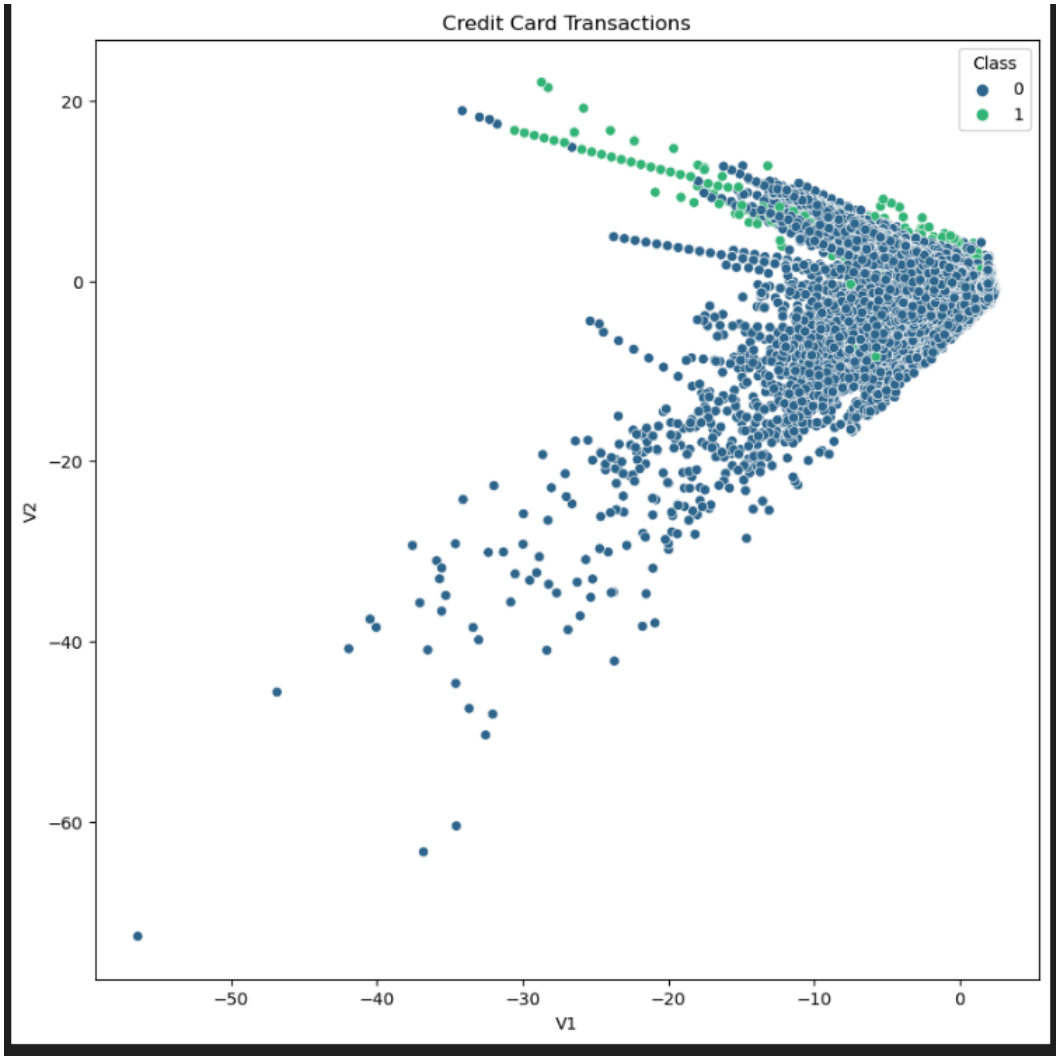
- Quantitative Comparison metrics for one class classification problem are **Precision**, **F1 score** and **Recall** value.
- Recall measures the ability of a model to correctly identify all relevant instances of the positive class (anomalies) out of all actual positive instances
- Precision measures the ability of a model to correctly identify only the relevant instances of the positive class (anomalies) out of all instances it has classified as positive.
- The F1 score is the harmonic mean of precision and recall. It provides a single metric that balances both precision and recall.
- We have implemented 3 algorithms namely One class SVM, KNN and Isolation Forests, on comparison we can see that all the three metrics discussed above are the best in case of **KNN**.
- Low Recall values in Dataset2 are due to the outliers not being separated so that the anomalies cannot be detected accurately.

- ***KNN should be best for one class classification problem on this dataset1 and Isolation forests must be chosen for dataset2.***

Qualitative Comparison:

- Local Outlier Factor : complexity depends on the number of data points and the chosen value of 'n_neighbors'. It can be computationally expensive for large datasets, especially when 'n_neighbors' is large.It can be sensitive to the choice of parameters, particularly 'n_neighbors' and 'contamination'
- KNN: anomalies are located in sparse regions of the feature space or when the density of normal instances varies across different regions. It can adapt well to local data density variations. KNN's performance can degrade with high-dimensional data due to the curse of dimensionality, and it may require careful tuning of parameters such as the number of neighbours ('k') and the distance metric.
- Isolation Forests: anomalies are rare and can be isolated into small partitions in the feature space. They are efficient, scalable to high-dimensional data, and robust to outliers.They can handle mixed data types and are less sensitive to the choice of hyperparameters compared to other methods like One-Class SVM
- In dataset1, we only have 4 columns, so it is not very high dimensional data also the data is not very cluttered so application of SVM is not very suitable for this.
- Dataset1 was cleaner, and there was a clear demarcation between its outliers and normal data but the dataset2 was more cluttered and its outliers were not separated well.

- ***KNN should be more suitable among the three in Dataset1 but Isolation forest in dataset2.***


Credit Card Transactions
Class
0
1
20
0
−20
−40
−60
V2
−50
−40
−30
−20
−10
0
V1

References

One-Class Classification Algorithms for Imbalanced Datasets**:** machinelearningmastery.com

One-Class Classification: Wikipedia

A literature review on one-class classification and its potential applications in big data

One class classification: A survey

https://aran.library.nuigalway.ie/bitstream/10379/1472/1/camera_ready_occ_lnai.pdf

Datasets used:

ECG data: https://www.kaggle.com/datasets/devavratatripathy/ecg-dataset

HTTP data for network anomalies: