Anomaly Detection
Project Overview

What is anomaly detection?

Anomaly detection (aka outlier analysis) is a step in data mining that identifies data points, events, and/or observations that deviate from a dataset's normal behaviour. Anomalous data can indicate critical incidents, such as a technical glitch, or potential opportunities, for instance a change in consumer behaviour.

Applications of Anomaly detection

- Banking, Financial Services, and Insurance (BFSI) – In the banking sector, some of the use cases for anomaly detection are to flag abnormally high transactions, fraudulent activity, and phishing attacks.
- Retail – In Retail, anomaly detection is used for processing large volumes of financial transactions to identify fraudulent behaviors, such as identity theft and fraudulent credit card usage.
- Manufacturing – In Manufacturing, anomaly detection can be used in a number of important ways, such as identifying machines and tools that are underperforming, which can take months to find without anomaly detection technology.
- IT and Telecom – In IT and Telecommunications, anomaly detection is increasingly valuable to detect and act on personal threats to users, financial threats to service providers, or other types of unexpected threats.
- Defence and Government – In Defence and Government setting, anomaly detection is best used for identifying excessive and fraudulent government spending, budgeting, and audits. This can save governments an immense amount of money.
- Healthcare – In Health Care, anomaly detection is used for its application in a crucial management task that can improve the quality of the health services and avoid loss of huge amounts of money. In terms of identifying fraudulent claims from hospitals and on the side of the insurance providers.

Tech Stack

➔ Language used : R
➔ Machine Learning interface : H2O
➔ Other packages used : caret, e1071, ROCR and many more

Dataset Overview

In this project, we will be using a credit card fraud dataset which represents the fraudulent and legal transactions over a certain period. The data is available in a .csv format.
We can observe in the dataset that most of the column names (V1 to V28) are not mentioned explicitly. This is because PCA (Principal Component Analysis) transformation has been performed on the original dataset in order to maintain the confidentiality of the data. Apart from these variables, we have a few explicit variables as follows

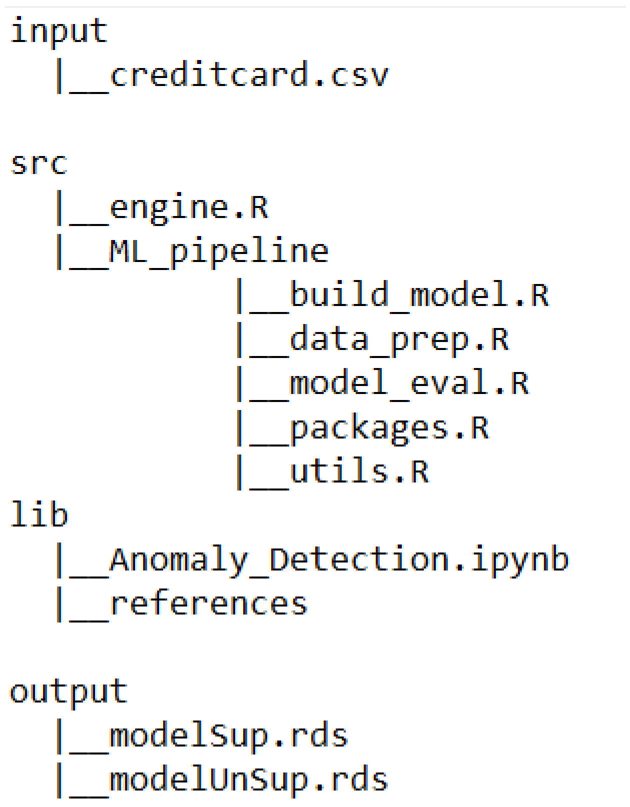
1. Time - Difference in seconds between each transaction and its previous transaction
2. Amount - Transaction Amount
3. Class
    a. 0 - Non fraudulent transaction
    b. 1 - Fraudulent Transaction

Approach

1. Business context and objective
2. Translating into DS approach
    a. What, why, where Anomaly Detection ?
    b. Why we are using fraud dataset for this problem
    c. Algorithms used to solve this problem
3. Data importing and Data Understanding
4. Data Pre processing
    a. Creating time variable
5. EDA
6. Preparing data for modelling
7. Understanding neural networks and deep neural networks
8. Understanding auto encoders
9. Unsupervised Learning using h2o
    a. Building model and Model Details
10. Evaluation parameters understanding
    a. Evaluating based on Reconstructed MSE
11. Supervised Learning using h2o
    a. Building and tuning supervised learning model using H2O
12. Transfer learning
    a. Supervised Learning using Pre trained model and evaluation
13. Precision-recall curve
    a. Try different threshold to improve accuracy
14. Making production ready code

Modular code overview

The ipython notebook is modularized into different functions so that the user can use those functions instantly whenever needed. The modularized code folder is structured in the following way.



```
input
  |__creditcard.csv

src
  |__engine.R
  |__ML_pipeline
           |__build_model.R
           |__data_prep.R
           |__model_eval.R
           |__packages.R
           |__utils.R
lib
  |__Anomaly_Detection.ipynb
  |__references

output
  |__modelSup.rds
  |__modelUnSup.rds
```

Once you unzip the modular_code.zip file you can find the following folders within it.

1. input
2. src
3. output
4. lib

1. The input folder contains all the data that we have for analysis. In our case, it will contain two tsv files which are
   a. train.tsv
   b. test.tsv
2. The src folder is the heart of the project. This folder contains all the modularized code for all the above steps in a modularized manner. It further contains the following.
   a. ML_pipeline
   b. engine.R

The ML_pipeline is a folder that contains all the functions put into different R files which are appropriately named. These python functions are then called inside the engine.R file

3. The output folder contains all the models that we trained for this data saved as .rds files. These models can be easily loaded and used for future use and the user need not have to train all the models from the beginning.
4. The lib folder is a reference folder. It contains the original notebook that we saw in the videos.

Project Takeaways

- Understanding the problem statement
- Importing the dataset and importing libraries
- Performing basic EDA
- Data cleaning Imputing the null values filling them using appropriate method
- Using ggplot to visualize the Dataset
- Importing h2o library and initializing an h2o cluster
- Splitting Dataset into Train and Test
- Defining parameters for training a Neural Network
- Training the neural network
- Understanding what is difference between Artificial Neural Networks and Autoencoders
- How does an Autoencoder work
- Loading the pre-trained Neural Network
- How to Autoencode a pre-trained Neural Networks
- Visualizing the effectiveness of an Autoencoded model and a Neural Networks using ggplot
- Making predictions using the trained model