Anomoly detection:

Outlier detection and novelty detection are both used for anomaly detection, where one is interested in detecting abnormal or unusual observations.

Novelty detection:

Novelty detection is the task of classifying test data that differ in some respect from the data that are available during training. The novelty detection approach is typically used when the quantity of available "abnormal" data is insufficient to construct explicit models for non-normal classes. Application includes inference in datasets from critical systems, where the quantity of available normal data is very large, such that "normality" may be accurately modelled. In this we apply novelty detection algorithm that have appeared in the machine learning literature during the last decade to find the optimal solution to the problem.

novelty detection techniques is classified according to the following five general categories:

(i) probabilistic,

(ii) (ii) distance-based

(iii) reconstruction-based

(iv) domain- based

(v) information-theoretic techniques

Approach(i) uses probabilistic methods that often involve a density estimation of the "normal" class. These methods assume that low density areas in the training set indicate that these areas have a low probability of containing "normal" objects. Approach (ii) includes the concepts of nearest- neighbour and clustering analysis that have also been used in classification problems. The assumption here is that "normal" data are tightly clustered, while novel data occur far from their nearest neighbours. Approach (iii) involves training a regression model using the training set. When "abnormal" data are mapped using the trained model, the reconstruction error between the regression target and the actual observed value gives rise to a high novelty score. Neural networks, for example, can be used in this way and can offer many of the same advantages for novelty detection as they do for regular classification problems. Approach (iv) uses domain-based methods to characterise the training data. These methods typically try to describe a domain containing "normal" data by defining a boundary around the "normal" class such that it follows the dis- tribution of the data, but does not explicitly provide a distribution in high-density regions. Approach (v) com- putes the information content in the training data using information-theoretic measures, such as entropy or Kolmogorov complexity. The main concept here is that novel data significantly alter the information content in a dataset.

Outlier Detection:

Outlier detection is similar to novelty detection in the sense that the goal is to separate a core of regular observations from some polluting ones, called *outliers*. Outlier detection is then also known as unsupervised anomaly detection and novelty detection as semi-supervised anomaly detection. In the context of outlier detection, the outliers/anomalies cannot form a dense cluster as available estimators assume that the outliers/anomalies are located in low density regions. On the contrary, in the context of novelty detection, novelties/anomalies can form a dense cluster as long as they are in a low density region of the training data, considered as normal in this context. this approach is analogous to a semi-supervised recognition approach, which they term novelty detection or novelty recognition. outlier detection methods are grouped into “statistical models” and “neural networks” Additionally, the authors suggest another two categories: machine learning and hybrid methods. According to Hodge and Austin , most “statistical” and “neural network” approaches require cardinal or ordinal data to allow distances to be computed between data points. For this reason, the machine learning category was suggested to include multi-type vectors and symbolic attributes, such as rule-based systems and tree-structure based methods. Their “hybrid” category covers systems that incorporate algorithms from at least two of the other three categories.

Three fundamental approaches to the problem of outlier detection are, In the first approach, outliers are determined with no prior knowledge of the data; this is a learning approach analogous to unsupervised clustering. The second approach is analogous to supervised classification and requires labelled data (“normal” or “abnormal”). In this latter type, both normality and abnormality are modelled explicitly. Lastly, the third approach models only normality, this approach is analogous to a semi-supervised recognition approach, which is termed as novelty detection or novelty recognition. outlier detection methods are grouped into “statistical models” and “neural networks”. Most “statistical” and “neural network” approaches require cardinal or ordinal data to allow distances to be computed between data points. For this reason, the machine learning category was suggested to include multi-type vectors and symbolic attributes, such as rule-based systems and tree-structure based methods.

Two important distinctions :

Outlier detection:

The training data contains outliers which are defined as observations that are far from the others. Outlier detection estimators thus try to fit the regions where the training data is the most concentrated, ignoring the deviant observations.

Novelty detection:

The training data is not polluted by outliers and we are interested in detecting whether a **new** observation is an outlier. In this context an outlier is also called a novelty.

ML tools used both for novelty or outlier detection:

1. Isolation Forest
2. Local Outlier Factor
3. Svm One ClassSVM
4. Covariance Elliptic Envelope

For further detection of outliers we chose Local Outlier Factor algorithm, it is efficient way to perform outlier detection on moderately high dimensional datasets.

why local outlier factor?

A comparison of these outlier detection algorithms was done and LOF has been more effective algorithm in majority of aspects.

Robust covariance ,one class svm ,isolation forest and local outlier factor are few algorithms used for novelty or outlier detection.

local outlier factor and isolation forest perform well in detection of outliers .The strength of the LOF algorithm is that it takes both local and global properties of datasets into consideration: it can perform well even in datasets where abnormal samples have different underlying densities. The question is not, how isolated the sample is, but how isolated it is with respect to the surrounding neighborhood. where as other algorithms like one class-SVM it is sensitive to outliers and thus does not perform very well for outlier detection, It also requires the choice of a kernel and a scalar parameter to define a frontier.

Outlier detection in Robust covariance is to assume that the regular data come from a known distribution (e.g. data are Gaussian distributed). From this assumption, “shape” of the data is defined , and can define outlying observations as observations which stand far enough from the fit shape and since our objective is to learns along the detection that the entry made could be a new practice by the user or could an outsider hence robust covariance is not suitable.

local outlier factor

The Local Outlier Factor (LOF) algorithm is an unsupervised anomaly detection method which computes the local density deviation of a given data point with respect to its neighbors. It considers as outliers the samples that have a substantially lower density than their neighbors. This example shows how to use LOF for outlier detection which is the default use case of this estimator in scikit-learn.

(LOF) algorithm computes a score (called local outlier factor) reflecting the degree of abnormality of the observations. It measures the local density deviation of a given data point with respect to its neighbors. The idea is to detect the samples that have a substantially lower density than their neighbors.

In practice the local density is obtained from the k-nearest neighbors. The LOF score of an observation is equal to the ratio of the average local density of his k-nearest neighbors, and its own local density: a normal instance is expected to have a local density similar to that of its neighbors, while abnormal data are expected to have much smaller local density.

The number k of neighbors considered, (alias parameter n_neighbors) is typically chosen

1) greater than the minimum number of objects a cluster has to contain, so that other objects can be local outliers relative to this cluster, and

2) smaller than the maximum number of close by objects that can potentially be local outliers.

In practice, such informations are generally not available, and taking n_neighbors=20 appears to work well in general. When the proportion of outliers is high (i.e. greater than 10 %), n_neighbors should be greater.

Key features of algorithm:

1. LOF Algorithm Detects novelty/outliers with respect to density.
2. It measures the local deviation of density of a given sample with respect to its neighbors.
3. By comparing the local density of a sample to the local densities of its neighbors, one can identify samples that have a substantially lower density than their neighbors. These are considered outliers.
4. Local density score of a point is the ratio of average local density of its neighbors to own local density.
5. Density of a point is the ratio of number of points in the set to distance between points p and x (where x = set of neighbors of point p)
6. If density of a point is less (outlier), its LOF score is high.
7. Density will be less if sum of distances to the neighbors is high.
8. If LOF is close to 1, object is comparable to its neighbors

    If LOF is less or very close to 1, it is considered to be in denser region denser region and hence inlier.

    a. If LOF >>> 1, outlier
    b. Distance could be Euclidean or Manhattan.

Algorithm :

- Data from json file is read and data frame is created .
- All string variables are converted to integers using STN function.(Elaborated below)
- Data is modified for better results (Elaborated below)

- New sub data frame is created " new_dataset " which only has integer values
- Data is segregated according to unique package ID.
- data is scaled using MinMaxScaler.(Elaborated below)
- data featuring is done using PCA.(Elaborated below)
- data is fed to the classifier and is plotted
- Data is plotted using LocalOutlierFactor for each package.

Data Preprocessing

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.

Data goes through a series of steps during preprocessing:

- Data Cleaning: Data is cleansed through processes such as filling in missing values, smoothing the noisy data, or resolving the inconsistencies in the data.
- Data Integration: Data with different representations are put together and conflicts within the data are resolved.
- Data Transformation: Data is normalized, aggregated and generalized.
- Data Reduction: This step aims to present a reduced representation of the data in a data warehouse.
- Data Discretization: Involves the reduction of a number of values of a continuous attribute by dividing the range of attribute intervals.

Feature scaling

Why Scaling

Most of the times, your dataset will contain features highly varying in magnitudes, units and range. But since, most of the machine learning algorithms use Eucledian distance between two data points in their computations, this is a problem.If left alone, these algorithms only take in the magnitude of

features neglecting the units. The results would vary greatly between different units, 5kg and 5000gms. The features with high magnitudes will weigh in a lot more in the distance calculations than features with low magnitudes.

Feature scaling is a method used to standardize the range of independent variables or features of data. It transforms features by scaling each feature to a given range. This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one.so it is required that the datatype of the features are float .

There are four common methods to perform Feature Scaling.

1. Standardisation: Standardisation replaces the values by their Z scores.

   a. $$x' = \frac{x - \bar{x}}{\sigma}$$

   This redistributes the features with their mean μ = 0 and standard deviation σ=1 . sklearn.preprocessing.scale helps us implementing standardization in python.

2. Mean Normalisation:

   i. $$x' = \frac{x - \text{mean}(x)}{\max(x) - \min(x)}$$

   This distribution will have values between -1 and 1with μ=0. Standardization and Mean Normalization can be used for algorithms that assumes zero centric data like Principal Component Analysis(PCA).

3. Min-Max Scaling: This scaling brings the value between 0 and 1.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Feature Extraction

Principal Component Analysis:A more common way of speeding up a machine learning algorithm is by using Principal Component Analysis (PCA). using PCA to speed it up can be a reasonable choice. This is probably the most common application of PCA. Another common application of PCA is for data visualization. Feature Extraction basically helps in reducing the number of features from n to a

desired number of feature, here n_components hold the value of desired number of features which by default is 2.

limitations of PCA:

PCA is not scale invariant: we need to scale our data first. The directions with largest variance are assumed to be of the most interest Only considers orthogonal transformations (rotations) of the original variables PCA is only based on the mean vector and covariance matrix. Some distributions (multivariate normal) are characterized by this, but some are not.

If the variables are correlated, PCA can achieve dimension reduction. If not, PCA just orders them according to their variances.

CODE :

```
y_scaled=minmax.fit_transform(y.values)
pca = PCA(n_components=2, whiten='True')
 x = pca.fit(y_scaled).transform(y_scaled)
```

"y_scaled" has the value after normalizing contaning 19 data features which is reduced to 2 features using PCA and is stored in "x".