Project:

Comparison of various Anomaly Detection Algorithms and Techniques

Prakhar Dogra

Email: pdogra@gmu.edu

G Number: G01009586

Introduction

Main goal of the project is to apply different kinds of anomaly detection algorithms and techniques and compare the results. Area under the curve, Recall (Detection Rate) and False (Alarm) Positive Rate have been used as the evaluation measures. Accuracy is not a good measure since the dataset is highly unbalanced.

Data Set Used

The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where there were 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, the original features and more background information about the data was not available. Features V1, V2, ... V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependent cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

File WebLink: https://www.kaggle.com/dalpozz/creditcardfraud/downloads/creditcard.csv.zip

Download the zip file. Extract it and keep it in the same folder as the source file.

Method and Results

For the purpose of this project, different kinds of algorithms have been selected. Following are the different types of Anomaly Detection algorithms and techniques that were used for this project:

1. Classification Based – One Class SVM, Random Forest Classifier, Isolation Forest
2. Nearest Neighbor Based – KNN and Local Outlier Factor (LOF)
3. Statistical Based – using Mahalanobis distance and Log Likelihood approach
4. Transduction Based – StrOUD Algorithm using KNN and LOF as strangeness function
5. Graph Based – Convex Hull method
6. Resampling the Data

Since the original data set is sorted by the 'Time' attribute, for the purpose of this project, the data set was shuffled (randomly) and saved as a separate CSV file. Once the dataset is shuffled, 80% of data was used for training and cross validation (4-fold cross validation) and remaining 20% of the data was used for testing.

For the purpose of this project (6), resampling of data followed by testing on popular algorithms (like KNN, Naïve Bayes, Decision Trees, Neural Networks and SVM) has been done in Weka 3.8. And the rest of the project (1 to 5), all the different type of algorithms and techniques, have been implemented using Python 2.7. Popular libraries like numpy, sklearn, scipy, matplotlib, time, csv have been used.

Initially all the algorithms and techniques were tested on 10% of data (randomly chosen data points but also keeping in mind of ratio the anomalous to non-anomalous data points i.e, 0.001728 or 0.1728%). Once that was done, algorithms were run on complete (original) data set. Except Convex Hull and Chi Square Test method, all the implementations gave almost the same results. In some case, even better results than before.

The source code files contain code for training and cross validation followed by testing.

1. CLASSIFICATION BASED

These type of Anomaly Detection algorithms can be supervised as well as unsupervised. But since number of anomaly class in test set is unbounded and training might lead to overfitting. General classification methods:

- One Class SVM
- Neural Networks Based – Auto Encoders
- Forest Based Classifiers – Random Forest Classifier and Isolation Forest Classifier
- Bayesian Network Approaches

For this project, One Class SVM, Random Forest Classifier and Isolation Forest were chosen as the Classification Based Anomaly Detection algorithm.

ONE-CLASS SVM

Main idea is to separate the entire set of training data from the origin, i.e. to find a small region where most of the data lies and label data points in this region as one class. Following results are obtained by applying the algorithm on test data set. Parameter 'nu' was set to 0.05 and RBF kernel was used.

Area Under Curve	False Positive Rate	Recall	Run-Time
0.907	0.048	0.86	799.235 seconds ~ 13.32 minutes

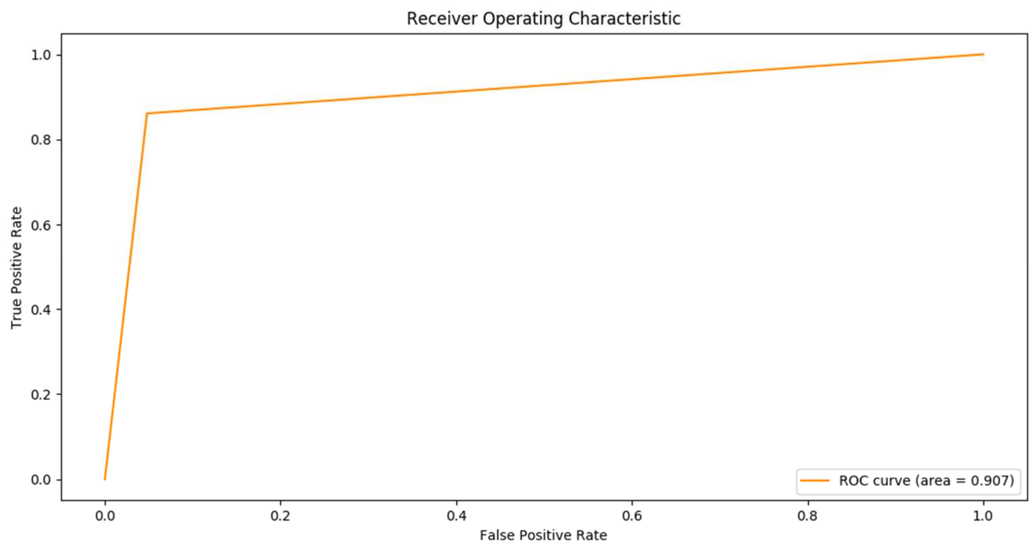

Figure 1: ROC for One Class SVM

RANDOM FOREST CLASSIFIER

A random forest classifier fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. This classifier is very popular to handle unbalanced classes. Following results are obtained by applying the algorithm on test data set. All the features (maximum features = 30) were used for this method.

Area Under Curve	False Positive Rate	Recall	Run-Time
0.8935	0.0001	0.787	131.498 seconds ~ 2.2 minutes

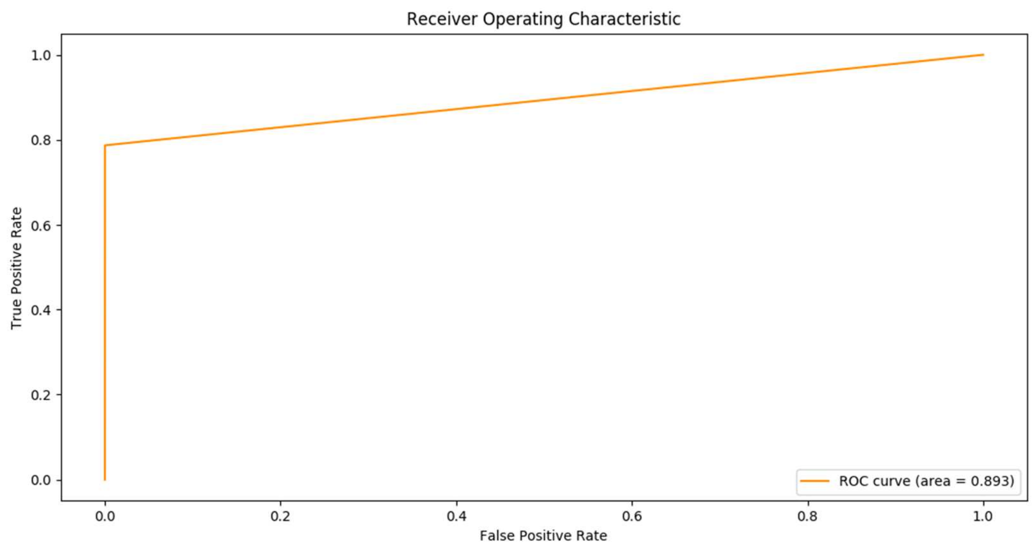

Figure 2: ROC for Random Forest Classifier

ISOLATION FOREST

This algorithm is based on the fact that anomalies are data points that are few and different. As a result of these properties, anomalies are susceptible to a mechanism called isolation. Following results are obtained by applying the algorithm on test data set. Parameter 'contamination' was set to 0.1 for this method.

Area Under Curve	False Positive Rate	Recall	Run-Time
0.896	0.098	0.889	26.44 seconds

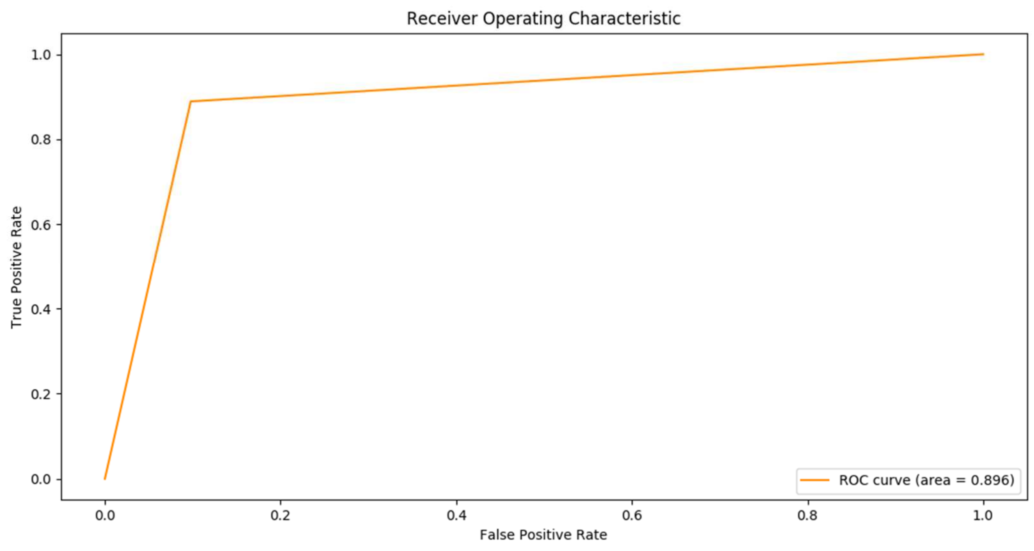

Figure 3: ROC for Isolation Forest

2. NEAREST NEIGHBOR BASED

These types of Anomaly Detection Algorithm techniques have a simple two-step approach:

- Compute neighborhood for each data record
- Analyze the neighborhood to determine whether data record is anomaly or not.

They can be further classified as:

- Distance Based – KNN algorithm
- Density Based – LOF, COF, LOCI, etc.

For this project, KNN and LOF were chosen as Nearest Neighborhood Based Anomaly Detection algorithms.

K-NEAREST NEIGHBOUR BASED APPROACH

It's slightly different from the KNN classification approach. It can be explained in the following points:

- For each data point, compute the distance to the Kth nearest neighbor.
- Sort all data points according to the calculated distance.
- Outliers are points that have the largest calculated distance and therefore are located in the more sparse neighborhoods.

Following results are obtained by applying the algorithm on test data set. K = 10 for determining the neighborhood.

Area Under Curve	False Positive Rate	Recall	Run-Time
0.9015	0.067	0.87	53.335 seconds ~ 1 minute

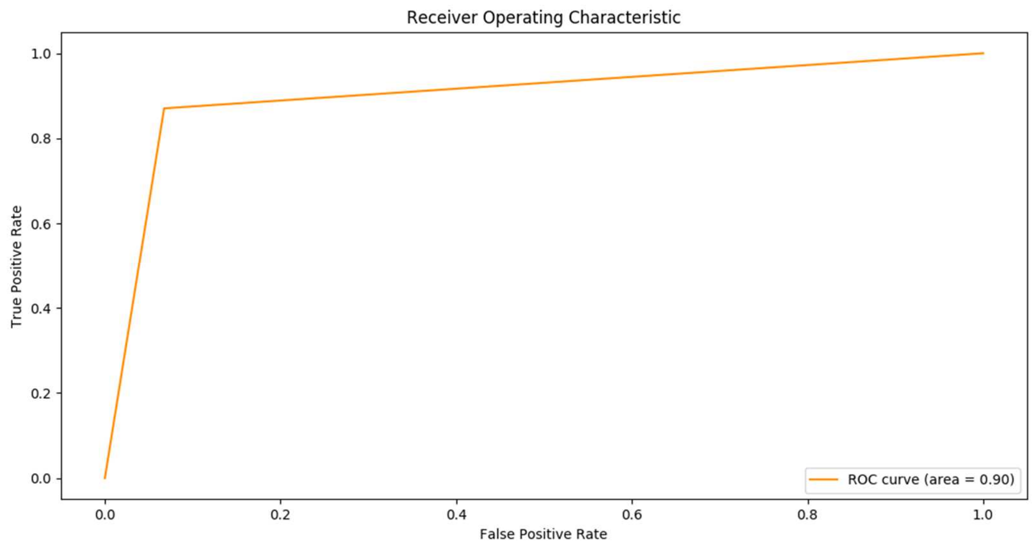

Figure 4: ROC for KNN based Anomaly Detection

LOCAL OUTLIER FACTOR BASED APPROACH

LOF scores can be used to detect outliers. Following points give a brief of how to do it:

- Find the k-nearest-neighbors
- For each instance, compute the local density
- For each instance compute the ratio of local densities
- Normal examples have scores close to 1.0
- Anomalies have scores > (1.2 ... 2.0)

Following results are obtained by applying the algorithm on test data set. K = 5 was used for determining the neighborhood.

Area Under Curve	False Positive Rate	Recall	Run-Time
0.911	0.09	0.901	7837.593 seconds ~ 2.18 hrs

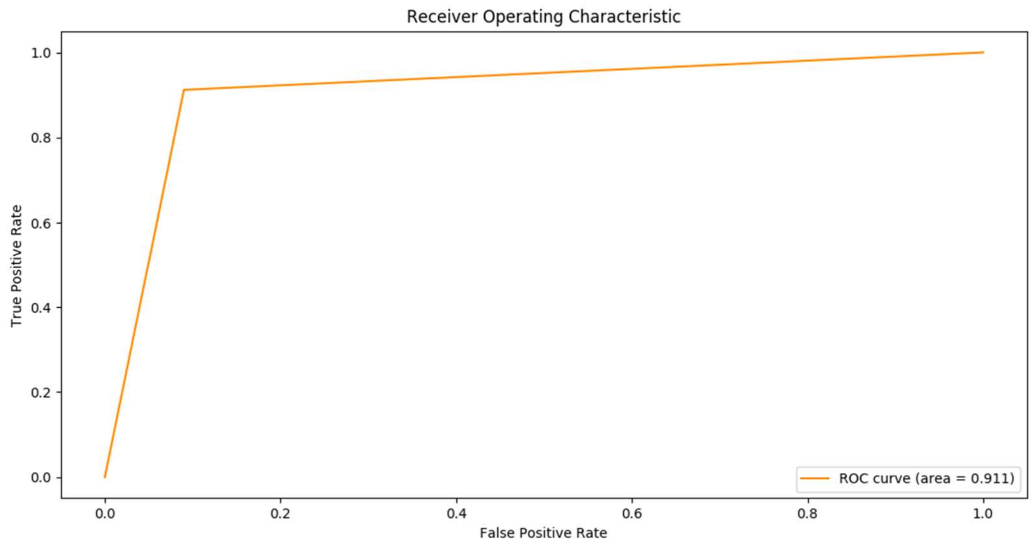

Figure 4: ROC for LOF based Anomaly Detection

3. STATISTICAL BASED APPROACH

In these types of Anomaly Detection approaches, data points are modeled using stochastic distribution and points are determined to be outliers depending on their relationship with this model.

For the purpose of this project two approaches have been used:

- Mahalonobis distance from the centroid can be used to find outliers
- Log Likelihood measure can also be used to detect outliers by checking if the log likelihood of the dataset changes significantly by removing a data point.

MAHALONOBIS DISTANCE

The fact that outliers have a higher mahalanobis distance than inliers, can be used to detect anomalies. Since the mahanalobis distances can be huge, a logarithm of base 10 was used. And for each data point whenever that exceeded the value of 1.838 (determined using cross validation), that point was declared as an outlier. Following results are obtained by applying the algorithm on test data set.

Area Under Curve	False Positive Rate	Recall	Run-Time
0.9124	0.037	0.862	7.811 seconds

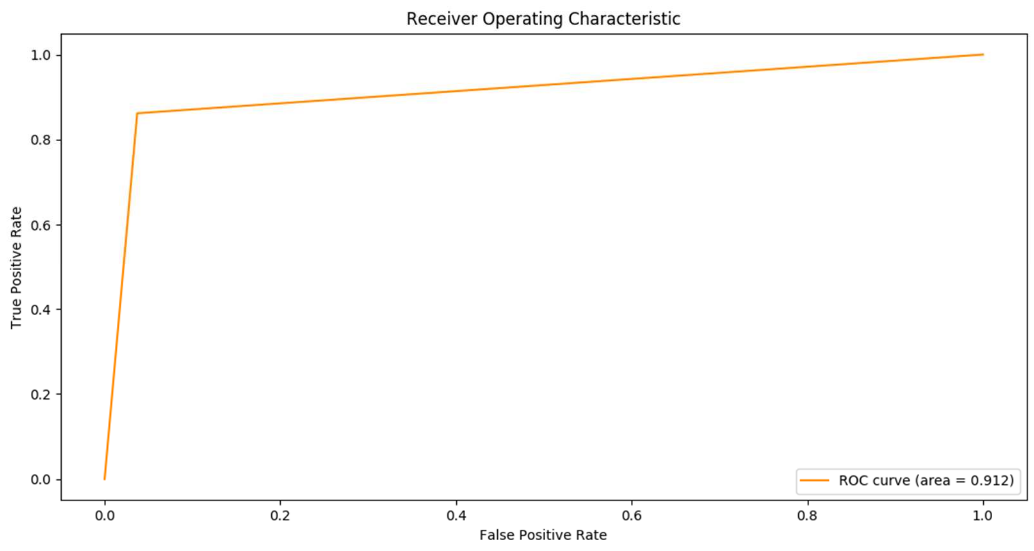

Figure 6: ROC for Mahalanobis Distance based approach

LOG LIKELIHOOD APPROACH

We can use the fact that if an anomaly is removed from the data set there is a significant change in log likelihood of the entire data set. Following algorithms explains how we can exploit that fact:

- Let **M** be the set of normal data points and **A** be the set of anomalous points.
- Initially assume all points to be in **M** (**A** is empty) and find Loglikelihood of **M**.
- Remove a point from **M** and find the new Loglikelihood of **M**.
- If the Loglikelihood changes significantly (change in likelihood is greater than some threshold) then move it to **A**
- Otherwise keep it in **M**

Following results are obtained by applying the algorithm on test data set.

Area Under Curve	False Positive Rate	Recall	Run-Time
0.9215	0.038	0.882	2.6 seconds

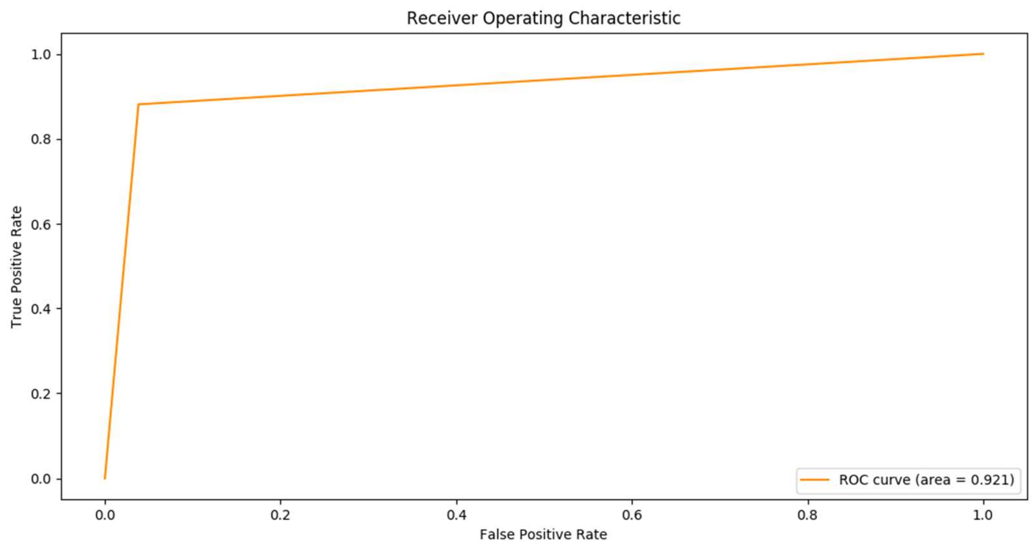

Figure 7: ROC for Log Likelihood Approach

4. TRANSDUCTION BASED

Transduction is the procedure that reasons from specific cases (training) to specific cases (test). A strangeness function measures how strange an item is. Given a distribution of strangeness values for the general population, compute the likelihood of being an outlier for a given point. It avoids creating a model by making only decisions about individual points at a time.

For the purpose of this project, the StrOUD algorithm has been implemented using KNN, LOF and Chi-Square as the strangeness functions.

STROUD ALGORITHM

The algorithm can be briefly described in the following points:

- Sample the data set. Make sure only normal data points are selected.
- Calculate strangeness value of each point. This is called the baseline.
- Sort the baseline (strangeness).
- For each test point, compute strangeness value with respect to previous data points.
- Find how many of those previous points had strangeness more than or equal to the calculated strangeness of the test point. Let's call it b.

- Compute p-value = (b+1)/(N+1) where N is the number of previous data points.
- If p-value < (1-confidence) then the test point is an outlier otherwise it is not.

The p-value returned by the StrOUD algorithm is tested against a confidence interval of 0.9 to 1.0. Confidence value for best results is determined during cross validation and used during testing.

KNN AS STRANGENESS FUNCTION

Strangeness value is calculated by first finding out the distances to K nearest neighbour of each point and then adding them up. Following results are obtained by applying the algorithm on test data set. Confidence of 91% is used for this method. K = 9 was used for determining the neighborhood.

Area Under Curve	False Positive Rate	Recall	Run-Time
0.864	0.032	0.759	1041.336 seconds ~ 17.4 minutes

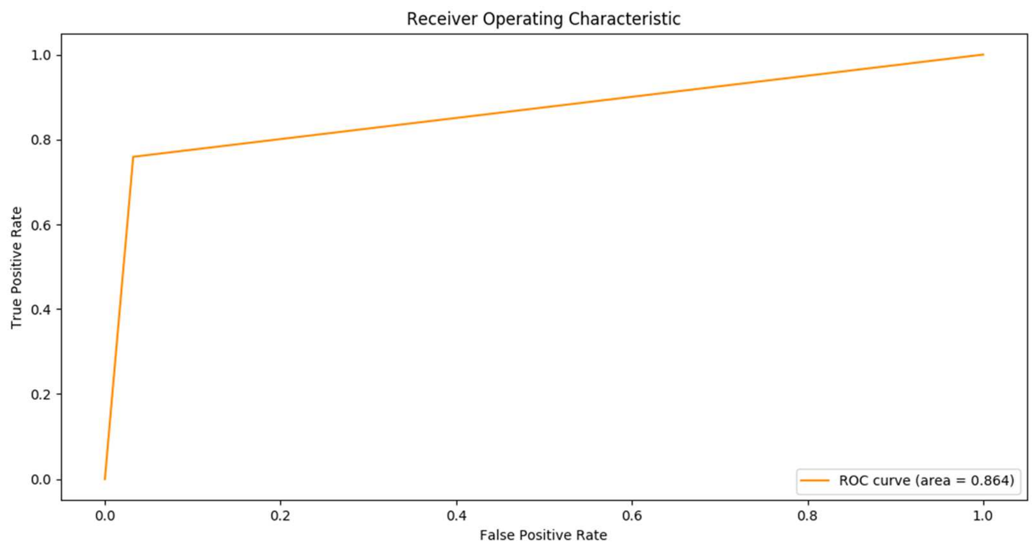

Figure 8: ROC for StrOUD using KNN as strangeness function

LOF AS STRANGENESS FUNCTION

Strangeness value is calculated by finding out the Local Outlier Factor of each point. Following results are obtained by applying the algorithm on test data set. Confidence of 97% is used for this method. K = 5 was used for determining the neighborhood.

Area Under Curve	False Positive Rate	Recall	Run-Time
0.89	0.032	0.854	8412.44 seconds ~ 2.34 hrs

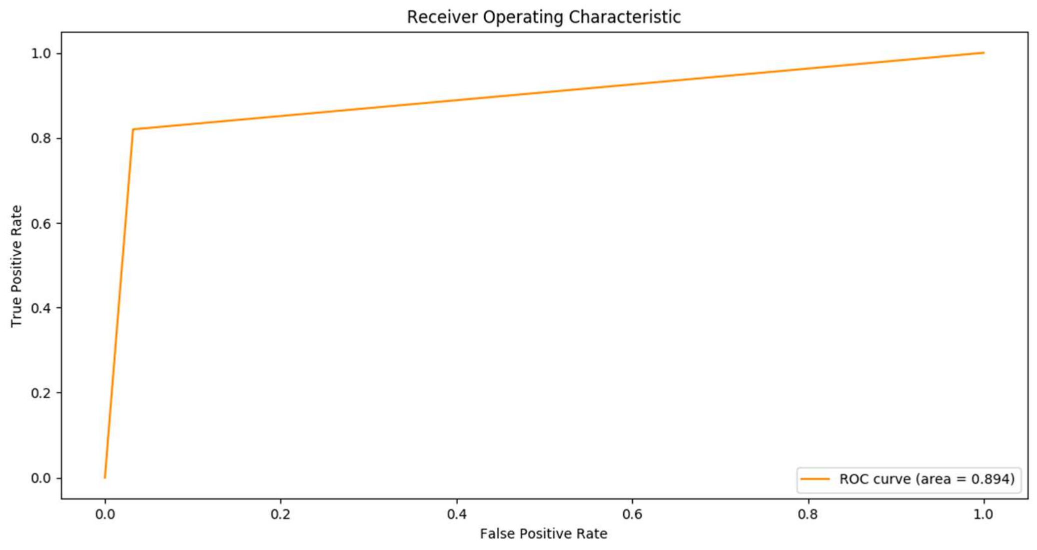

Figure 8: ROC for StrOUD using LOF as strangeness function

PVALUE FROM CHI-SQUARE TEST

The p-value returned from the Chi-Square Test can also be used in the StrOUD algorithm to detect outliers. Initially I ran the algorithm on 10% data and that gave me decent results (Area Under Curve of 0.865 for a confidence of 92%). But when the same method is applied on the complete data set, results were pretty bad (less than 0.7 of Area Under Curve). So the following algorithm hasn't been compared with others.

5. GRAPH BASED APPROACH

We can use visualization tools to observe the data in order to find anomalies. Graph based approaches provide alternate views of data for manual inspection.

For this project, Convex Hull method has been implemented to show the use of a Graph Based Anomaly Detection method.

CONVEX HULL METHOD

In this method, extreme points are assumed to be outliers. Following are the results when the method was run on 10% of data set. For complete data set, the results were pretty bad (less than 0.7 of Area Under the Curve). Number of PCA components in this method were kept to 11.

Area Under Curve	False Positive Rate	Recall	Run-Time
0.855	0.09	0.783	8133.81 seconds ~ 2.26 hrs

6. RESAMPLING THE DATA

When the data is highly unbalanced then resampling the data can be done in order to make the number of instances of each class equal. Following are the methods that can be used:

- Undersampling
- Oversampling
- SMOTE

For the purpose of this project, resampling techniques have been done in Weka. Moreover, the classification algorithms that have been used for evaluation are using Weka as well.

UNDERSAMPLING

It is basically downsizing (sampling) the majority class (normal class).

Algorithms	Accuracy	AUC	FPR	Recall
J48 Decision Tree	88.475	0.919	0.093	0.885
KNN (K=11)	91.19	0.953	0.097	0.912
SVM	92.20	0.919	0.085	0.922
Neural Networks	**93.56**	**0.973**	**0.067**	**0.936**
Naïve Bayes	91.86	0.956	0.088	0.919

Table 1: Results of various algorithms after undersampling

After undersampling, several popular algorithms are applied after undersampling and the results are compared.As can be seen from the above table, accuracy obviously decreases due to the loss of information (downsizing the majority class). Area under the curve is relatively higher but the fact that the data set is around 0.35% of the original data set's size (majority class has been downsized) may lead to underfitting.

OVERSAMPLING

It is basically making duplicates of the rare class (anomalous class) examples. For the purpose of this project, the oversampling technique has been performed and evaluated only on 10% of the data (anomalous vs non-anomalous example ratio is consistent with the original data set). Reason is because using complete data set was giving a memory error.

Algorithms	Accuracy	AUC	FPR	Recall
J48 Decision Tree	99.93	0.999	0.001	0.999
KNN (K=1)	**99.997**	**1.000**	**0.000**	**1.000**
SVM	99.95	0.999	0.001	0.999
Neural Networks	97.98	0.993	0.020	0.980
Naïve Bayes	91.37	0.977	0.086	0.914

Table 2: Results of various algorithms after oversampling

After oversampling, several popular algorithms are applied and the results were compared. As can be seen from the above table, the accuracy and the Area under the curve for most algorithms is very high. It is due to the fact that oversampling creates duplicates examples from the original data set which might be leading to overfitting.

SMOTE

It is basically used to generate artificial anomalies (not duplicates). For the given dataset, SMOTE percentage used in Weka was 54570%. For the purpose of this project, the oversampling technique has been performed and evaluated only on 10% of the data (anomalous vs non-anomalous example ratio is consistent with the original data set). Reason is because using complete data set was giving a memory error.

Algorithms	Accuracy	AUC	FPR	Recall
J48 Decision Tree	99.82	0.998	0.002	0.998
KNN (K=1)	**99.93**	**0.999**	**0.001**	**0.999**
SVM	99.64	0.996	0.004	0.996
Neural Networks	99.13	0.999	0.009	0.991
Naïve Bayes	90.89	0.981	0.091	0.909

Table 3: Results of various algorithms after SMOTE

After applying SMOTE, several popular algorithms were applied and the results were compared.

As can be seen from the above table, the accuracy and the Area under the curve for most algorithms is very high. It is due to the fact that SMOTE creates synthetic examples from the original data set which might be leading to overfitting.

Comparing the results

Following table shows a comparison of all the algorithms and techniques discussed in the previous section:

Algorithms	**Area Under Curve**	**False Positive Rate**	**Recall**	**Run-Time (minutes)**
One Class SVM	0.907	0.048	0.86	13.32
Random Forest Classifier	0.8935	**0.0001**	0.787	2.2
Isolation Forest Method	0.896	0.098	**0.889**	0.44
K Nearest Neighbour	0.9015	0.067	0.87	0.89
Local Outlier Factor	0.911	0.09	0.87	130.63
Mahalanobis Distance	0.9124	0.037	0.862	0.13
Log Likelihood	**0.9215**	0.038	0.882	**0.043**
StrOUD using KNN	0.864	0.032	0.759	17.4
StrOUD using LOF	0.89	0.032	0.854	140.21

Table 3: Comparison of all the methods

Resampling of data set hasn’t been included since it is simply a method to balance the data set and cannot be compared with the above algorithms that detect outliers.

Conclusions

From the above section that compares all the algorithms and techniques, we can say that not one algorithm is the best. On one hand Log Likelihood gives the best Area Under Curve and runs in shortest amount of time, Random Forest Classifier has the least False Positive Rate and Isolation Forest has the best Recall.

We can conclude from the above statement that a single algorithm isn't the best one. But we can find the best one relative to the use. For example, we can clearly see among the Classifier Based Anomaly Detection methods, Isolation Forest is the fastest and has the best recall. But One Class SVM has the best Area Under Curve and Random Forest Classifier has the least False Positive Rate. So, from this, we can say that if the cost of falsely identifying a valid credit card transaction is high then Random Forest Classifier is the best for this case. But if computation(detection) needs to be fast then Isolation Forest is the one to choose. And if it comes down to choosing the one with the best accuracy (area under the curve) then One Class SVM algorithm should be your choice.

Other types of Anomaly Detection techniques can be compared in a similar way.

As for the 'resampling of data set' technique, we can say that oversampling and SMOTE causes overfitting and undersampling causes underfitting due to extremely low ratio of anomalous to non-anomalous data points (0.1728%).

References

1. For doubts and queries:
    - http://www.stackoverflow.com/
    - https://www.researchgate.net/
    - https://www.stats.stackexchange.com/
2. Documentation of scikit-learn to implement the algorithms:

   http://www.scikit-learn.org/stable/user_guide.html

3. Course Lecture slides for theory and idea of algorithms:
   http://cs.gmu.edu/~dbarbara/CS584/CS584L11.pdf