**MAJOR PROJECT REPORT**

On

**Credit Card Fraud Detection System**

*Submitted in partial fulfillment of the requirements for*

*The award of the degree of*

**Bachelor of Technology**

In

**ELECTRONICS & COMMUNICATION ENGINEERING**



**HMR INSTITUTE OF TECHNOLOGY & MANAGEMENT**

**HAMIDPUR, DELHI – 110036**

**Affiliated to**

**GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY**

**Sector - 16C, Dwarka, Delhi - 110075, India**

**2016-2020**

|  |  |
| --- | --- |
| Guided By: | Submitted By: |
| Mr. Pawan Bhutani | Ira Sharma (47013302816) |
|  | Mudita Sharma (48513302816) |
|  | Harshita Sharma (35813302816) |

**DECLARATION**

We, students of B.Tech (ECE) hereby declare that the Major Project Entitled **“Credit Card Fraud Detection System”** which is submitted to Department of Electronics & Communication Engineering, HMR Institute of Technology & Management, Hamidpur Delhi, affiliated to Guru Gobind Singh Indraprastha University, Dwarka, New Delhi in partial fulfillment of requirement for the award of the degree of Bachelor of Technology in Electronics & Communication Engineering, has not been previously formed the basis for the award of any degree, diploma or other similar title or recognition. The list of member(s) involved in the project is listed below: -

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Student Name** | **Enrollment Number** | **Student Signature** |
| 1. | IRA SHARMA | 47013302816 |  |
| 2. | MUDITA SHARMA | 48513302816 |  |
| 3. | HARSHITA SHARMA | 35813302816 |  |

This is to certify that the above statement made by the candidate(s) is correct to the best of my knowledge.

|  |  |
| --- | --- |
| **New Delhi:** | Signature of Guide |
| **Date:** | Mr. Pawan Bhutani |
|  | Assistant Professor  Electronics & Communication Department |

**Dr. Avadesh Kumar Sharma**

Head of the Department

Electronics & Communication Department

HMRITM Hamidpur, New Delhi

**ACKNOWLEDGEMENT**

We express our deep gratitude to **Mr. Pawan Bhutani, Assistant Professor**, Department of Electronics & Communication Engineering, for his valuable guidance and suggestions throughout our project work. We are thankful for his constant support and motivation which helped us to perform to the best of our abilities.

We would like to extend our sincere thanks to Head of the Department, **Prof. (Dr.) A. K. SHARMA** for his time to time suggestions to complete our project work. We are also thankful towards our parents, teachers and friends for their kind co-operation and encouragement which helped us in completion of this project.

Ira Sharma (47013302816)

Mudita Sharma (485133028160)

Harshita Sharma (35813302816)

**ABSTRACT**

The Project Entitled “Credit Card Fraud Detection System” works on one of the most important algorithms of “Local Outlier Factor, Isolation Factor, and Support Vector Machine”. We have prepared a working code of this project using Jupyter Notebook, Python basis and Windows 10 which is very helpful.

This report starts with an introduction to the basic concept of the Credit Card Transaction System. It mainly focuses on the main algorithm used in this project which is Local Outlier Factor, Isolation Factor and Support Vector Machine.

It explains the Dataset and its usage in the code. It also informs us about the extraction of dataset from the official websites.

The code is explained step by step, including all the functions, libraries and algorithms used. Screenshots of the output makes the Project Report more interactive and easy to understand.

We’ve used many histograms and pictures in the code, which is the best way to explain the users about various aspects and factors of our code.

We have given its application and how will it be implemented in the future and then finally concluded the report with a briefly explained summary. We ended the report with bibliography which contains all the useful resources or websites that helped us while making the project.

**CONTENTS**

|  |  |
| --- | --- |
| Declaration | ii |
| Acknowledgement | iii |
| Abstract | iv |
| Contents | v-vi |
|  |  |
| **Chapter 1 - Introduction** | **1-3** |
| 1.1 General Introduction | 1 |
| 1.2 Machine Learning | 2 |
|  |  |
| **Chapter 2 - Local Outlier Factor** | **4-9** |
| 2.1 Local Outlier Factor | 4 |
| 2.2 k- Distance | 4 |
| 2.3 Reachability Distance | 5 |
| 2.4 Local Reachability Density | 5 |
| 2.5 LOF | 6 |
| 2.6 Applications of Local Outlier Factor | 7 |
| 2.7 Advantages of Local Outlier Factor | 8 |
| 2.8 Disadvantages of Local Outlier Factor | 9 |
|  |  |
| **Chapter 3 - Isolation Forest** | **10-16** |
| 3.1 Introduction | 10 |
| 3.2 Visualizing Anomalies | 10 |
| 3.3 Anomaly Score | 12 |
| 3.4 Properties of Isolation Factor | 13 |
| 3.5 Isolation Forest Applications | 14 |
| 3.6 Advantages of Isolation Factor | 14 |
| 3.7 Disadvantages of Isolation Factor | 15 |
|  |  |
| **Chapter 4 - Support Vector Machine** | **17-28** |
| 4.1 Support Vector Machine | 26 |
| 4.2 Advantages of SVM | 27 |
| 4.3 Disadvantages of SVM | 27 |
|  |  |
| **Chapter 5 - DATASET** | **29-30** |
| 5.1 Dataset Explanation | 29 |
| 5.2 Sample Dataset | 30 |
|  |  |
| **Chapter 6 - Methodology** | **31-35** |
| 6.1 Methodology Chart | 31 |
| 6.2 Code and its Explanation | 32 |
| **Chapter 7 - Results** | **36-37** |
| 7.1 Amount per transaction by Class | 36 |
| 7.2 Time of Transaction Vs. Amount of Class | 36 |
| 7.3 Heat map | 37 |
| 7.4 Final Prediction | 38 |
|  |  |
| **Chapter 8 - Conclusions And Future Scopes** | 39 |
|  |  |
| **Chapter 9 - Bibliography** | 40 |

**Chapter 1 - Introduction**

* 1. **General Introduction**

Biometrics is a part of the cutting edge of technology. Put simply, biometrics are any metrics related to human features. The most common examples of a biometric recognition system are the iPhone’s fingerprint, facial recognition, and finger knuckle recognition technology. As an emerging technology, biometric systems can add great convenience by replacing passwords and helping law enforcement catch criminals. Biometric identifiers also act as access control in secure environments, both physical and digital.

Finger-knuckle-print is one of the emerging biometric traits. The region of interest is the area where the maximum information is centred, for a finger knuckle it is the area surrounding the knuckle region. The finger knuckle print refers to the inherent skin patterns that are formed at the joints in the finger back surface. Recently it has been found that the finger knuckle print is highly rich in textures and can be used to uniquely identify a person.

Automated security is one of the major concerns of modern times. Secure and reliable authentication systems are in great demand. A biometric trait like Finger Knuckle Print (FKP) of a person is unique and secure. In the recent years, hand based biometrics is extensively used for personal recognition. Finger Knuckle has unique bending and makes a distinctive biometric identifier.

The determination is based on Deep Learning and Convolutional Neural Network (CNN). Here we have used a Training Dataset of about 500 images out of which we have a total of 100 classes each containing 5 images. Using these images we have trained our model. Finally the testing is done on a set of 5 images on which final prediction is made.

In order to use the best technique of Machine learning, we have compared the three algorithms i.e. Convolutional Neural Network, Support Vector Machine, Isolation Forest and Random Forest. On comparison, we found that CNN works out the best for Image Preprocessing and its prediction.

This project can be very useful as this can help in accurate recognition of faces which is helpful for security.

**1.2 Machine Learning**

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. **Machine learning focuses on the development of computer programs** that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. **The primary aim is to allow the computers learn automatically** without human intervention or assistance and adjust actions accordingly.

**Some machine learning methods:**

Machine learning algorithms are often categorized as supervised or unsupervised:

* **Supervised machine learning algorithms**can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.
* **Unsupervised machine learning algorithms**are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn’t figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.
* **Semi-supervised machine learning algorithms** fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring unlabeled data generally doesn’t require additional resources.
* **Reinforcement machine learning algorithms**is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial anderror search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.

Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly. Combining machine learning with AI and cognitive technologies can make it even more effective in processing large volumes of information.

**Chapter 2 - Local Outlier Factor**

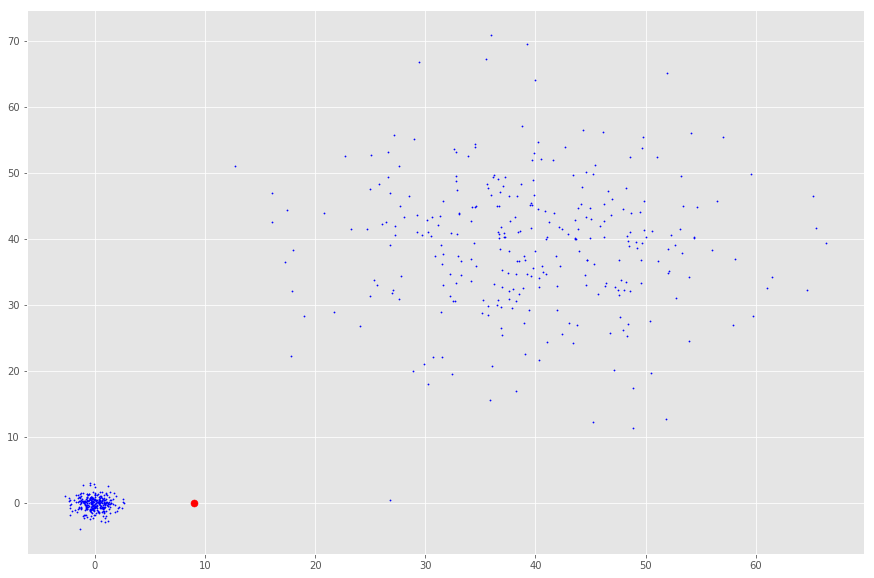
**2.1 Local Outlier Factor**

Local Outlier Factor (LOF) is a score that tells how likely a certain data point is an outlier/anomaly.

LOF≈ 1⇒No Outlier

LOF≫1⇒Outlier

First, we introduce a parameter *k* which is the number of neighbors, the LOF calculation is considering. The LOF is a calculation that looks at the neighbors of a certain point to find out its density and compare this to the density of other points later on. Using a right number *k*isn’t straight forward. While a small *k*has a more local focus, i.e. looks only at nearby points, it is more erroneous when having much noise in the data. A large *k*, however, can miss local outliers.

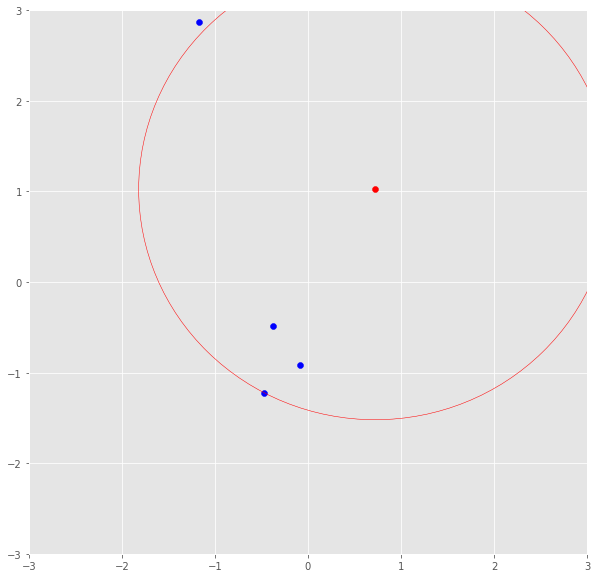


**FIG 2.1:**The density of the red point to its nearest neighbors is not different from the density to the cloud in the upper right corner. However, it is probably an outlier compared to the nearest neighbors’ density.

**2.2 k-Distance**

With this *k*defined, we can introduce the *k-distance* which is the distance of a point to its *kth*neighbor.

If *k* was 3, the *k-distance* would be the distance of a point to the third closest point.

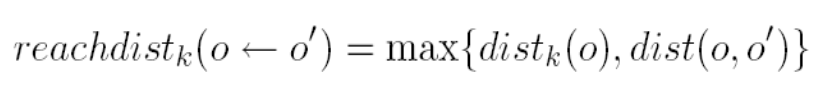


**FIG 2.2:** The red point’s k-distance is illustrated by the red line if k=3.

**2.3 Reachability Distance**

The *k-distance*is now used to calculate the reachability distance. This distance measure is simply the maximum of the distance of two points and the *k-distance* of the second point.

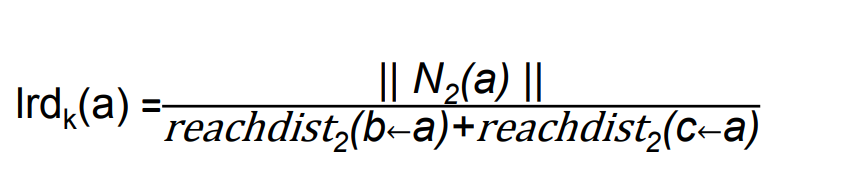
**reach-dist (a, b) = max{ k-distance(b), dist (a,b) }**



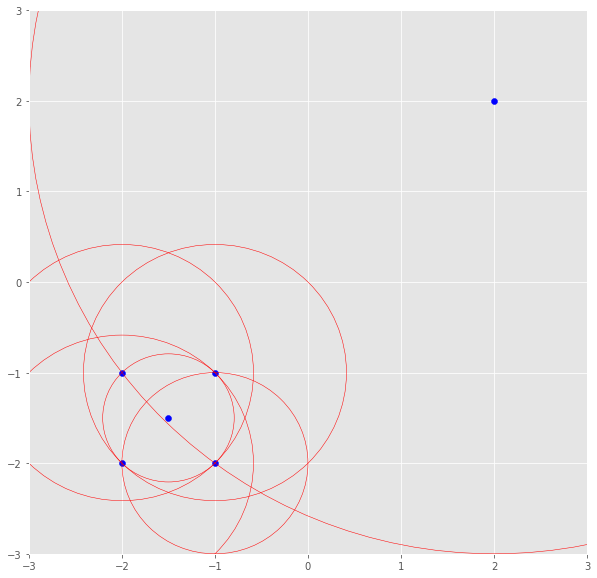
Basically, if point *a* is within the *k* neighbors of point *b*, the *reach-dist(a,b)* will be the *k-distance* of *b*. Otherwise, it will be the real distance of *a*and*b.*This is just a “smoothing factor”. For simplicity, consider this the usual distance between two points.

**2.4 Local Reachability Density**

The *reach-dist* is then used to calculate still another concept — the Local Reachability Density (lrd). To get the lrd for a point *a*, we will first calculate the reachability distance of *a,* to all of its *k*nearest neighbors and take the average of that number. The lrd is then simply the inverse of that average. Remember that we are talking about densities and, therefore, the longer the distance to the next neighbors, the sparser the area the respective point is located in.

**lrd(a) = 1/(sum(reach-dist(a,n))/k)**

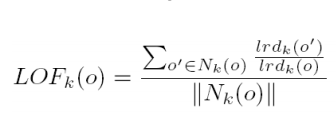
By intuition the local reachability density tells how far we have to travel from our point to reach the next point or cluster of points. The lower it is, the less dense it is, and the longer we have to travel.



**FIG 2.3:**The lrd of the upper right point is the average reachability distance to its nearest neighbors which are points (-1, -1), (-1.5, -1.5) and (-1, -2). These neighbors, however, have other lrds as their nearest neighbors don’t include the upper right point.

**2.5 LOF**

The lrd of each point will then be compared to the lrd of their *k* neighbors. More specifically, *k* ratios of the lrd of each point to its neighboring points will be calculated and averaged. The LOF is basically the average ratio of the lrds of the neighbors of *a,* to the lrd of *a*. If the ratio is greater than *1*, the density of point *a,* is on average smaller than the density of its neighbors and, thus, from point *a*, we have to travel longer distances to get to the next point or cluster of points than from *a*’s neighbors to their next neighbors. Keep in mind, the neighbors of a point *a*may don’t consider *a,*a neighbor as they have points in their reach which are way closer.



LOF(k) ~ 1 means **Similar density as neighbors,**

LOF(k) < 1 means **Higher density than neighbours (Inlier),**

LOF(k) > 1 means **Lower density than neighbours (Outlier)**

In conclusion, the LOF of a point tells the density of this point compared to the density of its neighbors. If the density of a point is much smaller than the densities of its neighbors (LOF ≫1), the point is far from dense areas and, hence, an outlier.

**2.6 Applications of Local Outlier Factor**

Outlier detection is used in various domains of applications. It can easily be used with data, image, and software. Basically anomaly detection and misuse is used for removing the noisy data and producing accurate data set. Various applications of outlier detection are enumerated below:

1. **Intrusion Detection:** Intrusion detection identifies all of the suspicious patterns that may indicate a network or system attack from someone attempting to break into or compromise a system.
2. **Fraud Detection:** Fraud detection is at alarming rate and hence becomes a great threaten for the institution and banks using a credit card transactions Fraud is reported under crime activities that includes banks, mobile phones fraud detection, commercial etc. Outlier is basically used to detect a noisy data that is being presented in the original data.
3. **Medical and public health outlier detection:** The patient data is to be collected from the various features of patient like blood test, height, weight, patient age. The various methods of outlier detection is used in medical diagnoses which helps to detect critical diseases at early stage for preventing it to become a severe and life-taking disease. Outlier detection plays a major role in detecting various kinds of cancers.
4. **Image Detection:** Images can be of any type main aim of outlier detection is to detect an abnormal behavior of the images. Each data consist of the various features of the image that includes color, brightness, image coordinates and texture.
5. **Text data Detection:** Noisy data is present in the pile of contents that is to be detected through the outlier techniques. The data can be spatial or can be a temporal means spatial related to the geographical conditions and temporal related to the time aspects. The main aim of outlier detection is to handle the noisy data that is presented in the pile of text.
6. **Sensor Networks:** Now a days sensor networks are used in the various applications of day to day life activities. A sensor network is a grouping of specialized transducers with ability of communication which helps to monitor and record conditions like humidity, pressure, vibrations, intensity of sound, level of pollution and concentration of chemicals etc. at different locations. A sensor network is a communication system which intends to record conditions and monitor at various locations. A sensor network have multiple detection station called sensor node. Each node is portable, less weighted and very small in size. Basically through the outlier detection techniques faulty sensor networks are detected so that communication level is to be increased. Reliability in wireless sensor networks is affected by the various causes like environment condition, using low quality sensors etc. that leads to corrupted data generations by sensor containing missing values.

**2.7 Advantages of Local Outlier**

1. Due to the local approach, LOF is able to identify outliers in a data set that would not be outliers in another area of the data set.
2. For example, a point at a "small" distance to a very dense cluster is an outlier, while a point within a sparse cluster might exhibit similar distances to its neighbors.
3. While the geometric intuition of LOF is only applicable to low-dimensional vector spaces, the algorithm can be applied in any context dissimilarity function can be defined.
4. It has experimentally been shown to work very well in numerous setups, often outperforming the competitors, for example in network intrusion detection and on processed classification benchmark data.
5. The LOF family of methods can be easily generalized and then applied to various other problems, such as detecting outliers in geographic data, video streams or authorship networks.

**2.8 Disadvantages of Local Outlier Factor**

The resulting values are quotient-values and hard to interpret. A value of 1 or even less indicates a clear inlier, but there is no clear rule for when a point is an outlier. In one data set, a value of 1.1 may already be an outlier, in another dataset and parameterization (with strong local fluctuations) a value of 2 could still be an inlier. These differences can also occur within a dataset due to the locality of the method.

There exist extensions of LOF that try to improve over LOF in these aspects:

1. **Feature Bagging for Outlier Detection** runs LOF on multiple projections and combines the results for improved detection qualities in high dimensions. This is the first ensemble learning approach to outlier detection, for other variants see ref.
2. **Local Outlier Probability**(LoOP) is a method derived from LOF but using inexpensive local statistics to become less sensitive to the choice of the parameter k. In addition, the resulting values are scaled to a value range of {\displaystyle [0:1]}[0:1].
3. **Interpreting and Unifying Outlier Scores** proposes a normalization of the LOF outlier scores to the interval {\displaystyle [0:1]}[0:1] using statistical scaling to increase usability and can be seen an improved version of the LoOP ideas.
4. **On Evaluation of Outlier Rankings and Outlier Scores** proposes methods for measuring similarity and diversity of methods for building advanced outlier detection ensembles using LOF variants and other algorithms and improving on the Feature Bagging approach discussed above.
5. **Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection** discusses the general pattern in various local outlier detection methods (including e.g. LOF, a simplified version of LOF and LoOP) and abstracts from this into a general framework. This framework is then applied e.g. to detecting outliers in geographic data, video streams and authorship networks.

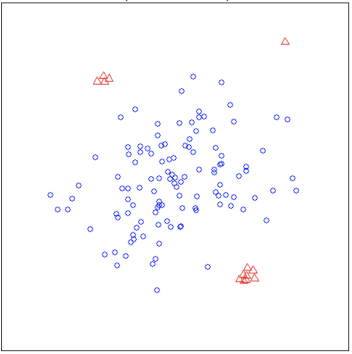
**Chapter 3 - Isolation Forest**

**3.1 Introduction**

Isolation forest detects anomalies by randomly partitioning the domain space. Yeah, you’re heard me right- It works similar to Decision trees algorithm, where we start with a root node and keep on partitioning the space. In Isolation forest we partition randomly, unlike Decision trees where the partition is based on Information gain.

Partitions are created by randomly selecting a feature and then randomly creating a split value between the maximum and the minimum value of the feature. We keep on creating the partitions until we isolate all the points (in most cases we also set a limit on number of partitions/height of the tree).

Now let us visualize how a normal point will differ from an anomalous point in our feature space.



**FIG 3.1:** The red points denote the anomalous points whereas the blue ones denote the normal points.

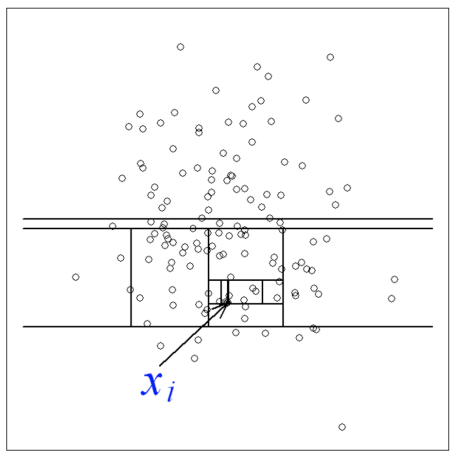
**3.2 Visualizing Anomalies**

In the **FIG 3.1**, the red points denote the anomalous points whereas the blue ones denote the normal points.

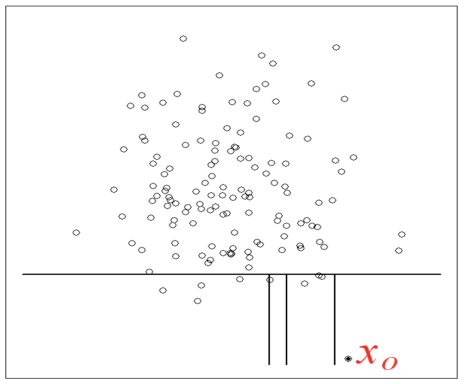
Now clearly, a normal point will be more clustered while an anomalous point will be far away from other points. Thus while randomly partitioning the domain space; the anomaly will be detected in smaller number of partitions than a normal point. Smaller number of partitions means lesser is the distance from the root node (this means lesser number of edges needs to be travelled from root node to the terminal node).

The above discussed concept will be well evident with the images given below.

1. **Isolating a normal point**



1. **Isolating an anomalous point**



**The anomaly will be detected in smaller number of partitions than a normal point.**

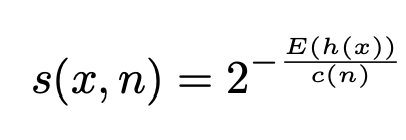
So clearly the path length indicates whether a point is a normal or an anomalous point.

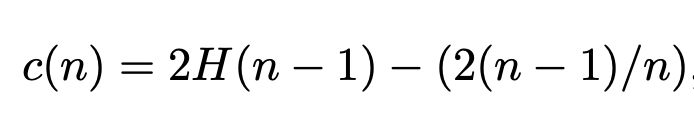
(**Path length-**of a point x is measured by the number of edges x traverses an Isolation tree from the root node until the traversal is terminated at an external node)

Isolation forest is an ensemble method. So we create multiple Isolation trees (generally 100 trees will suffice) and we take the average of all the path lengths. This average path length will then decide whether a point is anomalous or not.

**3.3 Anomaly Score**

Anomaly score is given by the following formula-





Where,

**n -**Number of data points

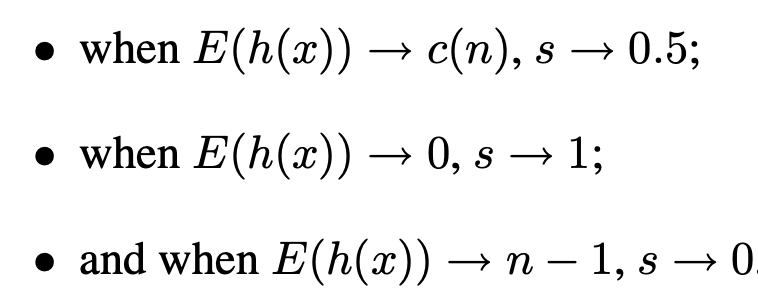
**c(n)** - It is the average path length of unsuccessful search in a Binary search tree

Now observe, we grow an isolation tree by randomly choosing a feature and randomly partitioning. This is very similar to the Binary Search tree. Thus we can approximate the average path length of for a node termination with the unsuccessful search in a Binary Search tree. Thus we use c(n) as the reference.

**If you’re having a hard time figuring out about Binary search trees, c(n) is just a reference metric. It normalizes the score between 0 & 1.**

Note that it is always better to represent score between 0 **&** 1 because the score can now be interpreted as a probability. For example, say for a data point if we get the anomaly score as 0.8, then we can interpret such that the point has a probability of 80% to be an anomalous point.

**E(h(x))** - Average of path lengths from the Isolation forest



* As score is closer to 1, then it is an anomalous point.
* As the score is closer to 0, it a normal observation.
* A score near 0.5 indicates it doesn’t have much distinction from normal observations.

**3.4 Properties of Isolation Forest**

* **Sub-sampling:** Since Isolation Forest does not need to isolate all of normal instances, it can frequently ignore the big majority of the training sample. As a consequence, isolation Forest works very well when the sampling size is kept small, a property that is in contrast with the great majority of existing methods, where large sampling size is usually desirable.
* **Swamping**: When normal instances are too close to anomalies, the number of partitions required to separate anomalies increases, a phenomenon known as *swamping*, which makes it more difficult for Isolation Forest to discriminate between anomalies and normal points. One of the main reasons for swamping is the presence of too many data for the purpose of anomaly detection, which implies one possible solution to the problem is sub-sampling. Since Isolation Forest respond very well to sub-sampling in terms of performance, the reduction of the number of points in the sample is also a good way to reduce the effect of swamping.
* **Masking**: When the number of anomalies is high it is possible that some of those aggregate in a dense and large cluster, making it more difficult to separate the single anomalies and, in turn, to detect such points as anomalous. Similarly to swamping, this phenomenon (known as “*masking*”) is also more likely when the number of points in the sample is big, and can be alleviated through sub-sampling.
* **High Dimensional Data**: One of the main limitations to standard, distance-based methods is their inefficiency in dealing with high dimensional datasets. The main reason for that is, in a high dimensional space every point is equally sparse, so using a distance-based measure of separation is pretty ineffective. Unfortunately, high-dimensional data also affects the detection performance of Isolation Forest, but the performance can be vastly improved by adding a features selection test like Kurtosis to reduce the dimensionality of the sample space.
* **Normal Instances Only**: Isolation Forest performs well even if the training set does not contain any anomalous point,[[3]](https://en.wikipedia.org/wiki/Isolation_forest#cite_note-:2-3) the reason being that Isolation Forest describes data distributions in such a way that high values of the path length h(xi) correspond to the presence of data points. As a consequence, the presence of anomalies is pretty irrelevant to Isolation Forest's detection performance.

**3.5 Isolation Forest Applications**

1. Mobile Phones Fraud Detection

2. Speech Recognition

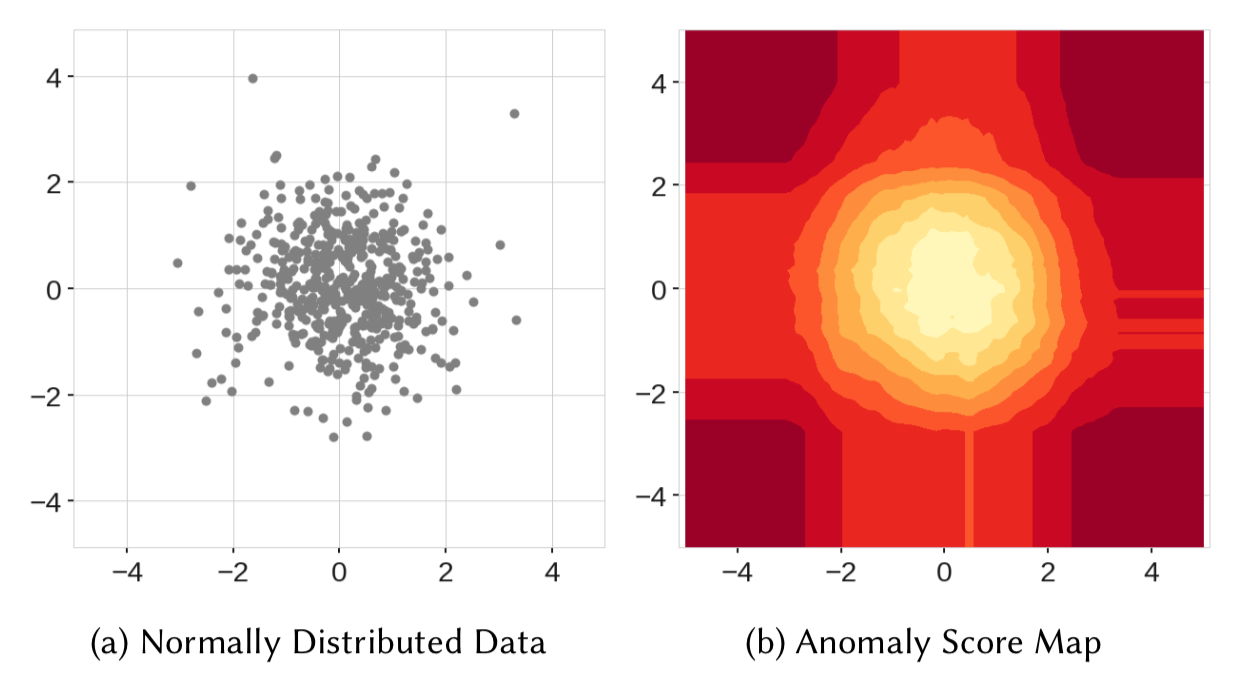
3. Industrial damage Fraud Detection

4. Traffic Monitoring

5. Detecting web faults

**3.6 Isolation Forest Advantages**

1. This method is highly useful and is fundamentally different from all existing methods.
2. It introduces the use of isolation as a more effective and efficient means to detect anomalies than the commonly used basic distance and density measures.
3. Moreover, this method is an algorithm with a low linear time complexity.
4. It has small memory requirement.
5. It builds a good performing model with a small number of trees using small sub-samples of fixed size, regardless of the size of a data set.

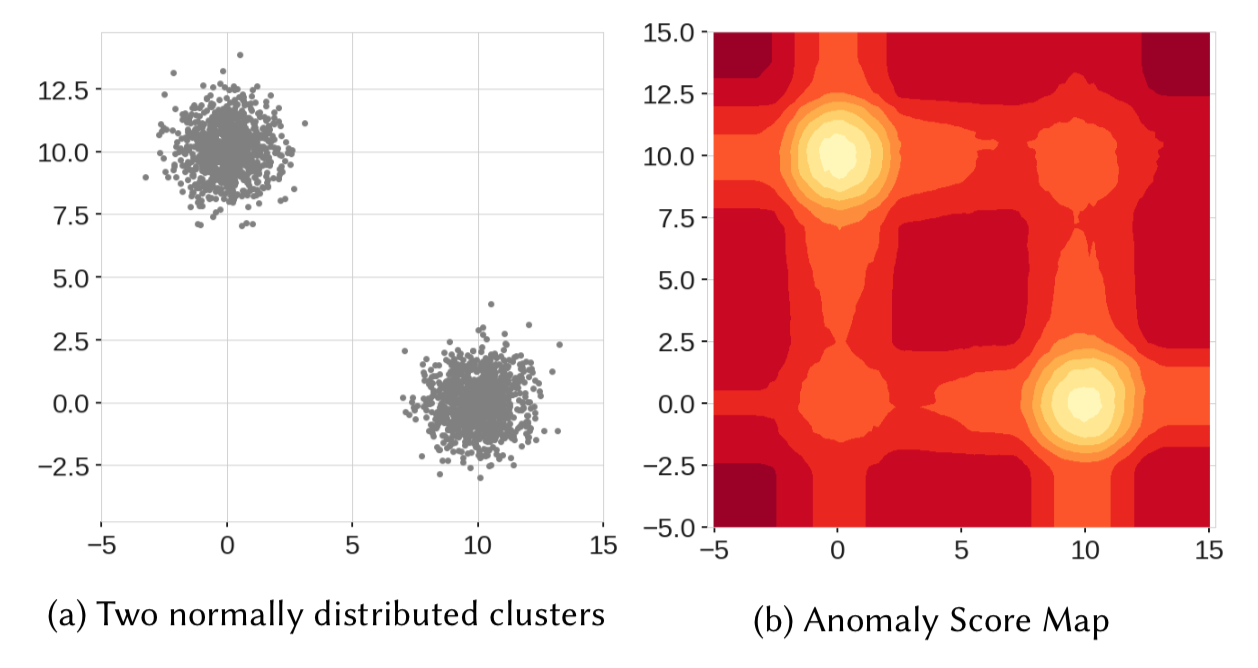
**3.7 Isolation Forest Disadvantages/Problem**

**FIG 3.2: (a)** Normally Distributed Data**, (b)** Anomaly Score Map

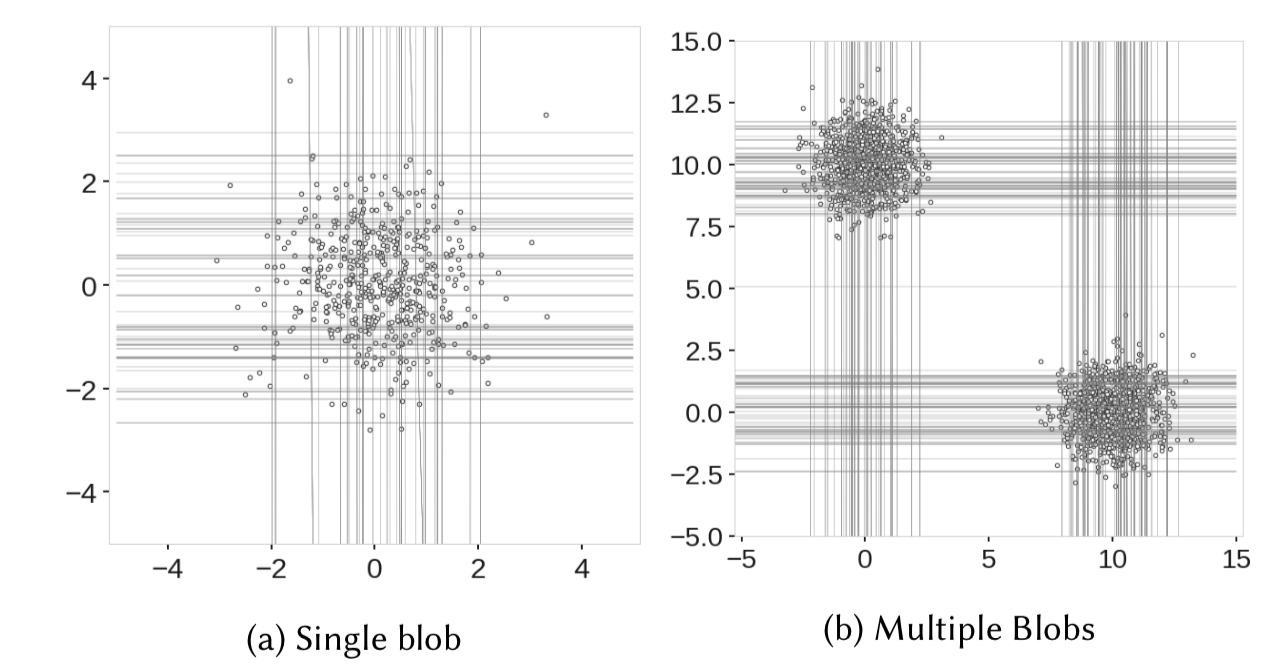
The best way to understand the issue is to see it as an example.

In **FIG 3.2 (a)**, we can see data sampled from the multivariate normal distribution. Intuitively, we would assume that the anomaly score assigned to the observations would increase radially from the central point of the distribution [0, 0]. However, this is clearly not the case, as seen in the **FIG 3.2 (b)**,. What is more, there are also rectangular artifacts of a lower score, such as the vertical one between point 0 and 1 on the x-axis.

Let’s move on to the second example. Here we see two blobs centered at points [0, 10] and [10, 0]. By inspecting the right figure we see not only the artifacts that were present before, but also two ghost clusters (approximately at [0, 0] and [10, 10]).



**FIG 3.3: (a)**Normally Distributed Data**, (b)** Anomaly Score Map

The reason for this peculiar behavior originates from the fact that the decision boundaries of the Isolation Forest are either vertical or horizontal (random value of a random feature), as seen in the picture below, where the authors plot branch cuts generated by the Isolation Forest during the training phase. We see that the branches tend to cluster where the majority of the points are located. But as the lines can only be parallel to the axes, there are regions that contain many branch cuts and only a few or single observations, which results in improper anomaly scores for some of the observations.An example might be points around [3, 0] (many branch cuts) and [3, 3] (few cuts).

**FIG 3.4: (a)**Single Blob**, (b)**Multiple Blobs

**Chapter 4 - Support Vector Machine**

“Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification and regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well (look at the below figure).

**FIG 4.1:** The hyper-plane that differentiate the two classes

Now as we have understood what an SVM is let us now understand how to make different classes or we can say hoe to form different hyper-plane.

* **(Scenario-1): Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.**

You need to remember a thumb rule to identify the right hyper-plane: “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.



**FIG 4.2:** “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.

* **(Scenario-2): Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, how can we identify the right hyper-plane?**

**FIG 4.3:** Here, we have three hyper-planes (A, B and C) and all are segregating the classes well.

Here, maximizing the distances between nearest data points (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**. Let’s look at the below snapshot:

**FIG 4.4:** The margin for hyper-plane C is high as compared to both A and B.

Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.

* **(Scenario-3): Hint: Use the rules as discussed in previous section to identify the right hyper-plane.**



**FIG 4.5:** Here, hyper-plane B has a classification error and A has classified all correctly.

Some of you may have selected the hyper-plane **B**as it has higher margin compared to **A.**But, here is the catch; SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is **A.**

* **(Scenario-4): Below, we are unable to segregate the two classes using a straight line, as one of star lies in the territory of other (circle) class as an outlier.**



**FIG 4.6:**One of star lies in the territory of other (circle) class as an outlier.

One star at other end is like an outlier for star class. SVM has a feature to ignore outliers and find the hyper-plane that has maximum margin. Hence, we can say, SVM is robust to outliers.



**FIG 4.7:** One star at other end is like an outlier for star class.

* **(Scenario-5): In the scenario below, we can’t have linear hyper-plane between the two classes.**



**FIG 4.8:** Here, one cannot have linear hyper-plane between the two classes.

SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature z=x2+y2. Now, let’s plot the data points on axis x and z:



**FIG 4.9:** Solution of the above mentioned problem.

* In above plot, points to consider are:
* All values for z would be positive always because z is the squared sum of both x and y
* In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

In SVM, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, SVM has a technique called the **kernel trick**. These are functions which takes low dimensional input space and transform it to a higher dimensional space i.e. it converts not separable problem to separable problem, these functions are called kernels. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then find out the process to separate the data based on the labels or outputs you’ve defined.

When we look at the hyper-plane in original input space it looks like a circle:



**FIG 4.10:** The hyper-plane in original input space which looks like a circle.

# Kernel

The learning of the hyper-plane in linear SVM is done by transforming the problem using some linear algebra. This is where the kernel plays role.

For **linear kernel** the equation for prediction for a new input using the dot product between the input (x) and each support vector (xi) is calculated as follows:

**F (x) = B (0) + sum (ai \* (x, xi))**

This is an equation that involves calculating the inner products of a new input vector (x) with all support vectors in training data. The coefficients B0 and ai (for each input) must be estimated from the training data by the learning algorithm.

The **Polynomial Kernel** can be written as

**K(x, xi) = 1 + sum(x \* xi) d**

**& Exponential** as

**K(x, xi) = exponent (-γ \* sum(x — xi²))**

Polynomial and exponential kernels calculates separation line in higher dimension. This is called **kernel trick.**

**There are other different types of kernels.**

**For example:**

### Gaussian kernel

It is a general-purpose kernel; used when there is no prior knowledge about the data. Equation is:

Description: Gaussian kernel equation

### Gaussian radial basis function (RBF)

It is a general-purpose kernel; used when there is no prior knowledge about the data.  
Equation is:

Description: Gaussian radial basis function (RBF)

### Laplace RBF kernel

Description: Laplace RBF kernel equationIt is general-purpose kernel; used when there is no prior knowledge about the data.  
Equation is:

### Sigmoid kernel

We can use it as the proxy for neural networks.

Description:  Sigmoid kernel equationEquation is:

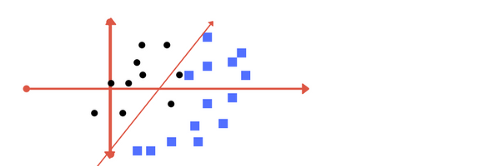
# Regularization

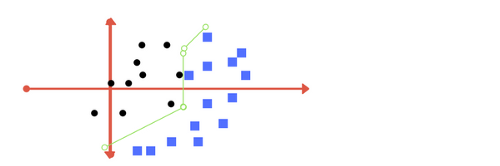
The Regularization parameter (often termed as C parameter in python’s sklearn library) tells the SVM optimization how much you want to avoid misclassifying each training example.

For large values of C, the optimization will choose a smaller-margin hyper-plane if that hyper-plane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyper-plane, even if that hyper-plane misclassifies more points.

The images below are examples of two different regularization parameters. Left one has some misclassification due to lower regularization value. Higher value leads to results like right one.

**FIG 4.7:** Images are examples of two different regularization parameters.

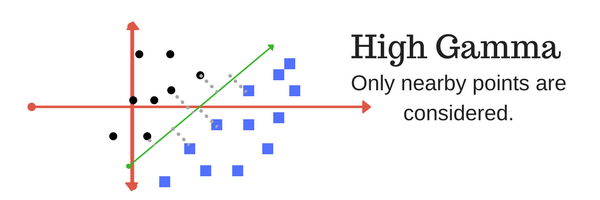


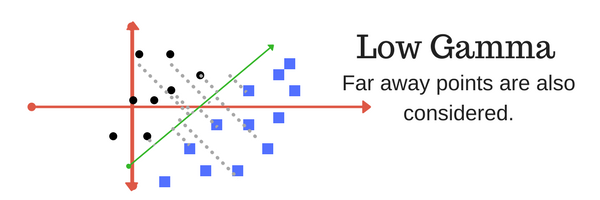
**FIG 4.11:****(a)** Lower Regularization Value

**FIG 4.11:(b)** High Regularization Value

# Gamma

The gamma parameter defines how far the influence of a single training example reaches, with low values meaning ‘far’ and high values meaning ‘close’. In other words, with low gamma, points far away from plausible separation line are considered in calculation for the separation line. Whereas high gamma means the points close to plausible line are considered in calculation.

**FIG 4.12: (a)** High Gamma

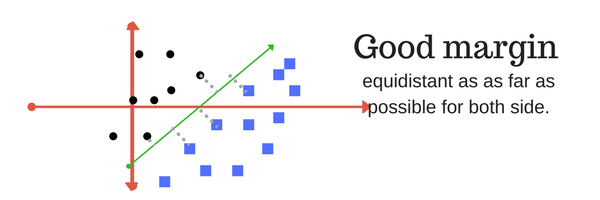
**FIG 4.12: (b)** Low Gamma

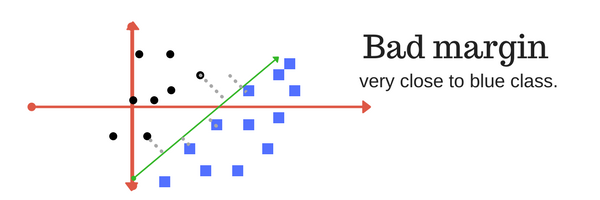
# Margin

And finally last but very important characteristic of SVM classifier. SVM to core tries to achieve a good margin.

**A margin is a separation of line to the closest class points.**

A **good margin** is one where this separation is larger for both the classes. Images below give two visual example of good and bad margin. A good margin allows the points to be in their respective classes without crossing to other class.



**FIG 4.13: (a)** Good Margin

**FIG 4.13: (a)** Bad Margin

**4.1 Support Vector Machine - Regression (SVR)**

Support Vector Machine can also be used as a regression method, maintaining all the main features that characterize the algorithm (maximal margin). The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences. First of all, because output is a real number it becomes very difficult to predict the information at hand, which has infinite possibilities. In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already requested from the problem. But besides this fact, there is also a more complicated reason; the algorithm is more complicated therefore to be taken in consideration. However, the main idea is always the same: to minimize error, individualizing the hyper-plane which maximizes the margin, keeping in mind that part of the error is tolerated.

#### 4.2 Implementation of SVM

The implementation of SVM we define the dependent and independent variable naming them x and y splitting them into test and train and then fitting them into the SVM model, here “x” are in dependent features based on which the prediction will be made and “y” is the variable which is predicted. In SVM for regression we use SVR and for classification we use SVC. The SVC function only take one variable w which is kernel variable then after the fitting the prediction is made.

#### 4.3 Advantages of Support Vector Machine

* SVM’s are very good when we have no idea on the data.
* Works well with even unstructured and semi structured data like text, Images and trees.
* The kernel trick is real strength of SVM. With an appropriate kernel function, we can solve any complex problem.
* Unlike in neural networks, SVM is not solved for local optima.
* It scales relatively well to high dimensional data.
* SVM models have generalization in practice; the risk of over-fitting is less in SVM.
* SVM is always compared with ANN. When compared to ANN models, SVMs give better results.

#### 4.4 Disadvantages of Support Vector Machine

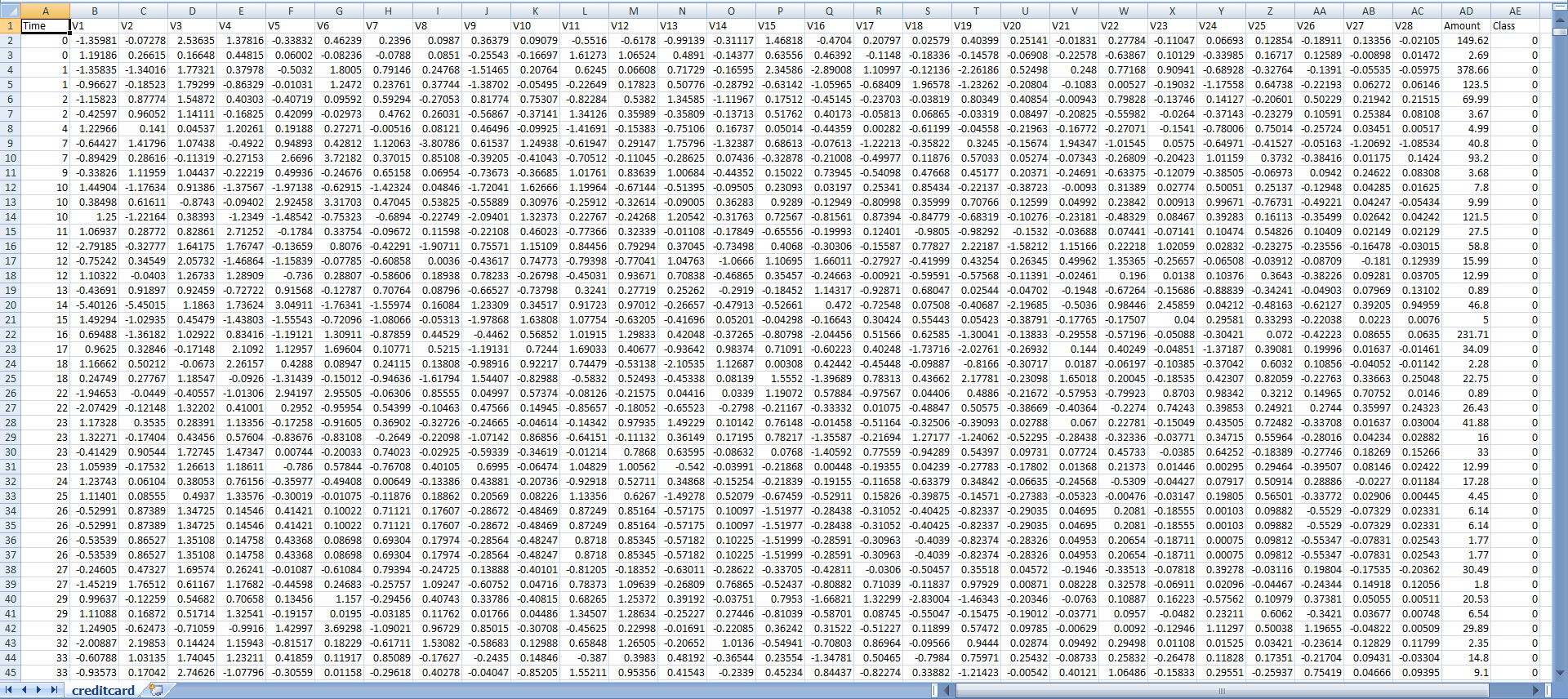
* Choosing a “good” kernel function is not easy.
* Long training time for large datasets.
* Difficult to understand and interpret the final model, variable weights and individual impact.
* Since the final model is not so easy to see, we cannot do small calibrations to the model hence it’s tough to incorporate our business logic.

The SVM hyper parameters are Cost -C and gamma. It is not that easy to fine-tune these hyper-parameters. It is hard to visualize their impact.

**Chapter-5 Dataset**

* 1. **Dataset Explanation**
* The datasets contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 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, we cannot provide the original features and more background information about the data. 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.
  1. **Sample Dataset**

Our dataset looks like:



**Chapter-6 Methodology**

**6.1 Methodology Chart**

Credit Card Fraud Dataset

**6.2 Code & its Explanation**

END

Performance Analysis

Classifier Section

K Fold Cross Validation Selection

Data Cleanup

& Visualization

1. **import numpy as np**

**import pandas as pd**

**importsklearn**

**importscipy**

**importmatplotlib.pyplot as plt**

**importseaborn as sns**

**fromsklearn.metrics import classification\_report,accuracy\_score**

**fromsklearn.ensemble import IsolationForest**

**fromsklearn.neighbors import LocalOutlierFactor**

**fromsklearn.svm import OneClassSVM**

**frompylab import rcParams**

**rcParams['figure.figsize'] = 14, 8**

**RANDOM\_SEED = 42**

**LABELS = ["Normal", "Fraud"]**

First we imported all the major python libraries and packages such as:

**Numpy -** Numerical python. It is an array-processing package.

**Pandas -** It provides data structures and many inbuilt methods forgrouping, combining data, and filtering, as well as time-seriesfunctionality.

**Sklearn -** It is used for working with complex data. Its main feature isCross-Validation.

**Scipy -**SciPy provides all the efficient numerical routines likeoptimization, numerical integration, and many others using its specificsubmodules.

**Matplotlib -**Matplotlib is a comprehensive library for creating static,animated, and interactive visualizations in Python.

Then we have imported some main packages which will be useful inpredictions when we’ll use algorithms.

1. **data = pd.read\_csv('creditcard.csv',sep=',')**

**data.head()**

This command is used for showing the top five entries of thedataset.

1. **data.info()**

It is used to tell how much memory the data is taking and the typeof data.

1. **data.isnull().values.any()**

It is used to check that there shouldn’t be any null value present.

1. **count\_classes = pd.value\_counts(data['Class'], sort = True)**

**count\_classes.plot(kind = 'bar', rot=0)**

**plt.title("Transaction Class Distribution")**

**plt.xticks(range(2), LABELS)**

**plt.xlabel("Class")**

**plt.ylabel("Frequency")**

This command is used to count the total number of normal andfraud transactions and give a plot of it in the form of histogram.

1. **fraud = data[data['Class']==1]**

**normal = data[data['Class']==0]**

This is used to assign class-1 to fraud transaction and class-0 tonormal transaction.

1. **print(fraud.shape,normal.shape)**

This command is used to count the total number of normal and fraud transactions with respect tp the number of features i.e 31.

1. **fraud.Amount.describe()**

This is used to describe the fraudulent data by counting it, calculating its mean, standard deviation and describes what type of data it is.

1. **normal.Amount.describe()**

This is used to describe the normal data by counting it, calculating its mean, standard deviation and describes what type of data it is.

1. **f, (ax1, ax2) = plt.subplots(2, 1, sharex=True)  
   f.suptitle('Amount per transaction by class')  
   bins = 50  
   ax1.hist(fraud.Amount, bins = bins)  
   ax1.set\_title('Fraud')  
   ax2.hist(normal.Amount, bins = bins)**

**ax2.set\_title('Normal')**

**plt.xlabel('Amount ($)')**

**plt.ylabel('Number of Transactions')**

**plt.xlim((0, 20000))**

**plt.yscale('log')**

**plt.show();**

This command gives two histogram plots, i.e. Amount per transaction by Fraud and Amount per transaction by Normal. It has Amount on x-axis and Number of Transactions on y-axis.

1. **f, (ax1, ax2) = plt.subplots(2, 1, sharex=True)**

**f.suptitle('Time of transaction vs Amount by class')**

**ax1.scatter(fraud.Time, fraud.Amount)**

**ax1.set\_title('Fraud')**

**ax2.scatter(normal.Time, normal.Amount)**

**ax2.set\_title('Normal')**

**plt.xlabel('Time (in Seconds)')**

**plt.ylabel('Amount')**

**plt.show();**

With this command We Will check if fraudulent transactions occur more often during certain time frame through a scatter plot with time in seconds on x-axis and Amount on y-axis.

1. **data1= data.sample(frac = 0.1,random\_state=1)**

**data1.shape**

We take a sample of data.

1. **data.shape**

We determine the shape of data.

1. **Fraud = data1[data1['Class']==1]**

**Valid = data1[data1['Class']==0]**

**outlier\_fraction = len(Fraud)/float(len(Valid))**

We determine the number of fraud and valid transactions in the sample dataset.

1. **print(outlier\_fraction)**

**print("Fraud Cases : {}".format(len(Fraud)))**

**print("Valid Cases : {}".format(len(Valid)))**

We print the Fraud and Normal cases from the dataset.

1. **import seaborn as sns**

**corrmat = data1.corr()**

**top\_corr\_features = corrmat.index**

**plt.figure(figsize=(20,20))**

**g=sns.heatmap(data[top\_corr\_features].corr(),annot=True,cmap="RdYlGn")**

We make a heat map using all the 31 features on both axis. A heat map is data analysis software that uses color the way a bar graph uses height and width: as a data visualization tool.

1. **columns = data1.columns.tolist()**

**columns = [c for c in columns if c not in ["Class"]]**

**target = "Class"**

**state = np.random.RandomState(42)**

**X = data1[columns]**

**Y = data1[target]**

**X\_outliers = state.uniform(low=0, high=1, size=(X.shape[0], X.shape[1]))**

**print(X.shape)**

**print(Y.shape)**

Here we create some dependent and independent variables suvh as column and its class and store its result in variables X and Y and finally predict their shape.

1. **classifiers = {**

**"Isolation Forest":IsolationForest(n\_estimators=100, max\_samples=len(X), contamination=outlier\_fraction,random\_state=state, verbose=0),**

**"Local Outlier Factor":LocalOutlierFactor(n\_neighbors=20, algorithm='auto', leaf\_size=30, metric='minkowski',p=2, metric\_params=None, contamination=outlier\_fraction),**

**"Support Vector Machine":OneClassSVM(kernel='rbf', degree=3, gamma=0.1,nu=0.05, max\_iter=-1, random\_state=state)**

**}  
type(classifiers)**

This command defines the various method of outlier detection i.e. Local outlier factor, Isolation forest and Support Vector Machine. The type of classifiers is “dictionary”.

1. **n\_outliers = len(Fraud)**

**for i, (clf\_name,clf) in enumerate(classifiers.items()):**

**ifclf\_name == "Local Outlier Factor":**

**y\_pred = clf.fit\_predict(X)**

**scores\_prediction = clf.negative\_outlier\_factor\_**

**elifclf\_name == "Support Vector Machine":**

**clf.fit(X)**

**y\_pred = clf.predict(X)**

**else:**

**clf.fit(X)**

**scores\_prediction = clf.decision\_function(X)**

**y\_pred = clf.predict(X)**

**y\_pred[y\_pred == 1] = 0**

**y\_pred[y\_pred == -1] = 1**

**n\_errors = (y\_pred != Y).sum()**

**print("{}: {}".format(clf\_name,n\_errors))**

**print("Accuracy Score :")**

**print(accuracy\_score(Y,y\_pred))**

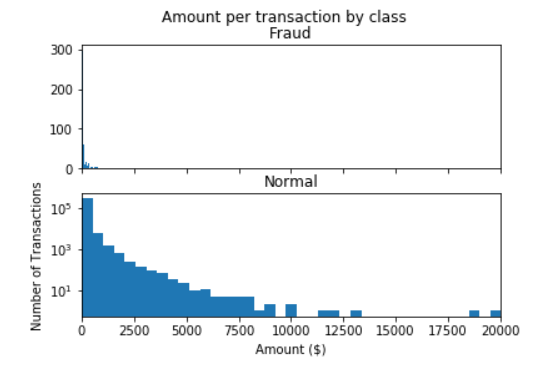
**print("Classification Report :")**

**print(classification\_report(Y,y\_pred))**

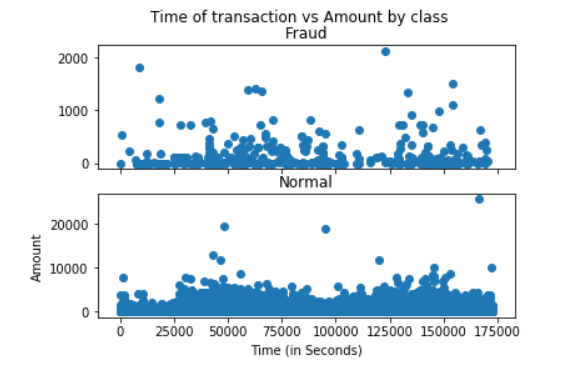
This command is used to make the final prediction. Class-1 represents fraud and Class-0 represents Normal transactions. It prints the accuracy of each method and its classification report.

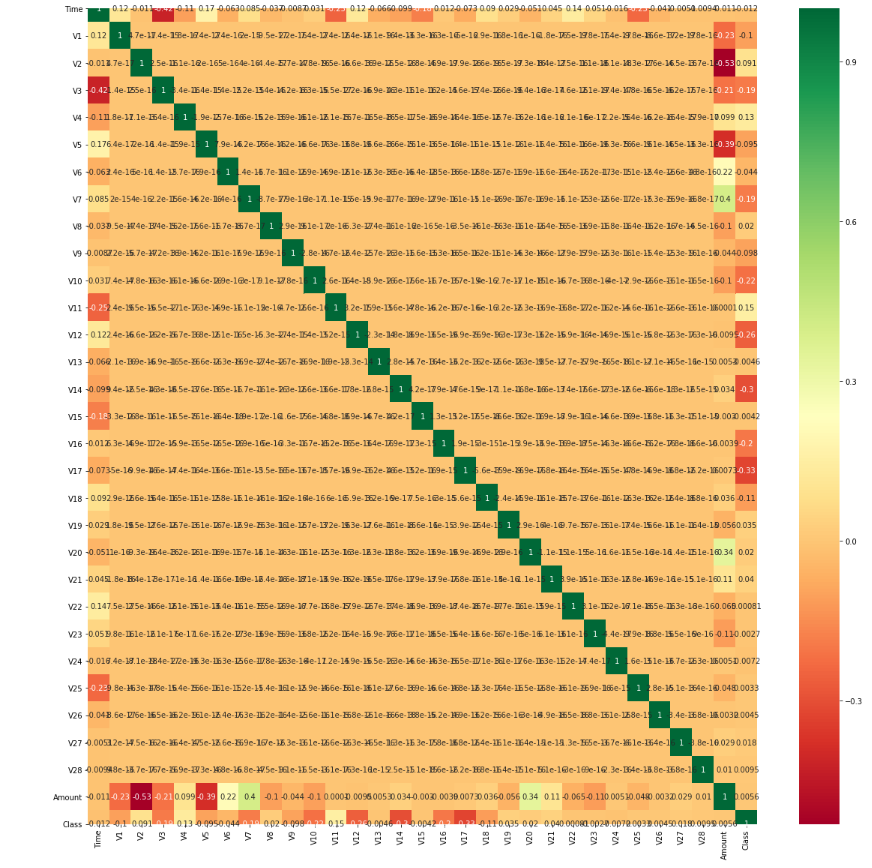
**Chapter-7 Results**

**7.1 Amount per Transaction by class**

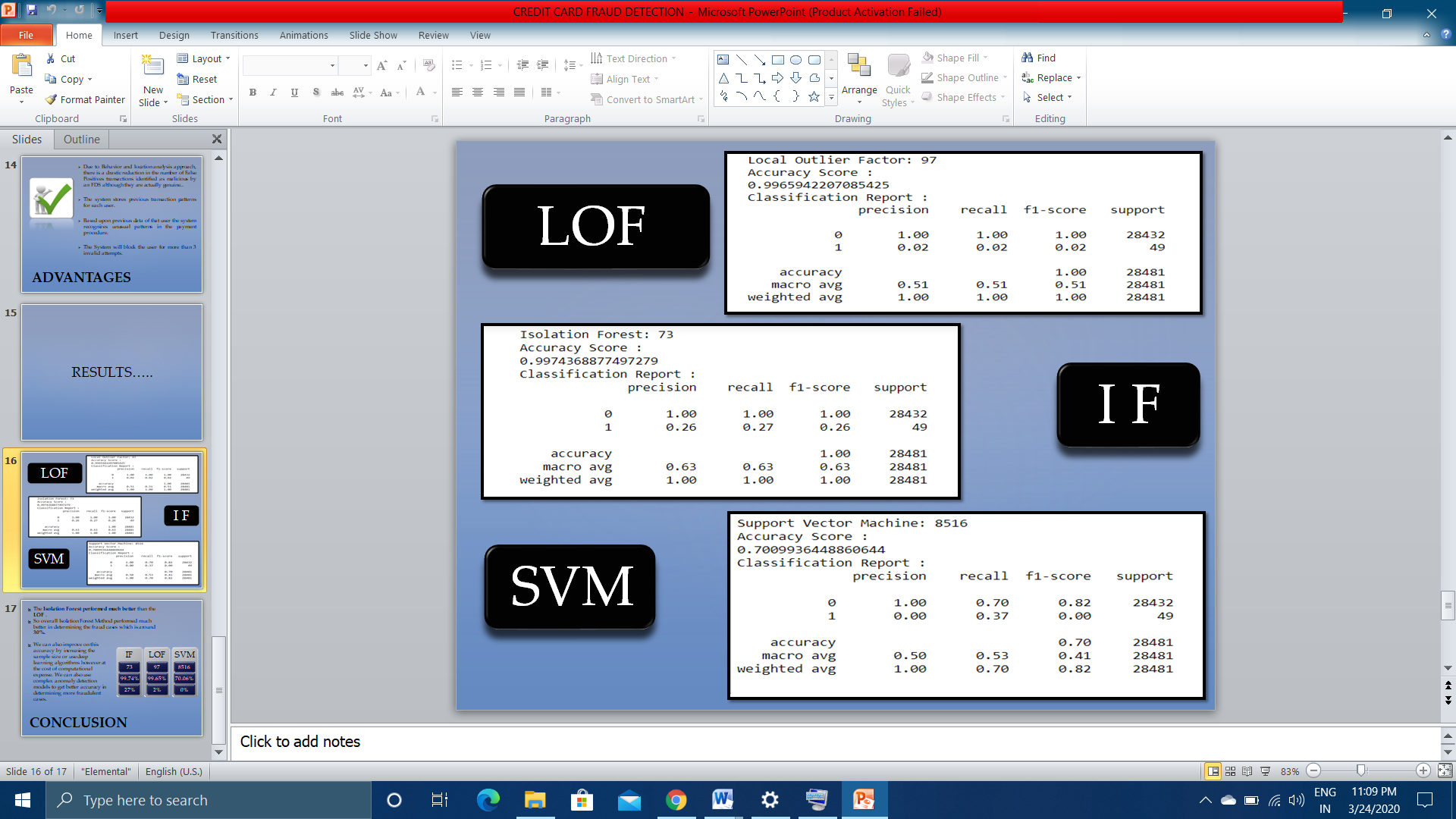
****

**7.2 Time of Transaction Vs. Amount by Class**

****

**7.3 Heat map**

**7.4 Final Prediction**



**Chapter-8 Conclusion & Future Scope**

* The **Isolation Forest performed much better** than the **LOF**.
* So overall Isolation Forest Method performed much better in determining the fraud cases which is around **30%.**
* We can also improve on this accuracy by increasing the sample size or use deep learning algorithms however at the cost of computational expense. We can also use complex anomaly detection models to get better accuracy in determining more fraudulent cases.

We can try to make this project in the form of an application, hence giving it a proper GUI or Graphical User Interface so that it becomes easy for organizations to use it.

**Chapter-9 References**

1. <https://medium.com/@arunm8489/local-outlier-factor-13784dc1992a>
2. <https://towardsdatascience.com/local-outlier-factor-for-anomaly-detection-cc0c770d2ebe>
3. <https://www.ijert.org/research/outlier-detection-for-different-applications-review-IJERTV2IS3508.pdf>
4. [https://towardsdatascience.com/outlier-detection-with-extended-isolation-forest 1e248a3fe97b](https://towardsdatascience.com/outlier-detection-with-extended-isolation-forest%201e248a3fe97b)
5. <https://medium.com/@doedotdev/local-outlier-factor-example-by-hand-b57cedb10bd1>
6. <https://en.wikipedia.org/wiki/Local_outlier_factor>
7. <https://towardsdatascience.com/local-outlier-factor-for-anomaly-detection-cc0c770d2ebe>
8. <https://quantdare.com/isolation-forest-algorithm/>
9. <https://blog.easysol.net/using-isolation-forests-anamoly-detection/>
10. <https://turi.com/learn/userguide/anomaly_detection/local_outlier_factor.html>
11. <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>
12. <https://mlg.ulb.ac.be/wordpress/>
13. <https://www.kaggle.com/>