Credit Card Fraud Detection

CLinsight

Introduction

- Fraud is a Big problem
- Why is Machine Learning necessary?
- Comprehensive solution?

CLinsight

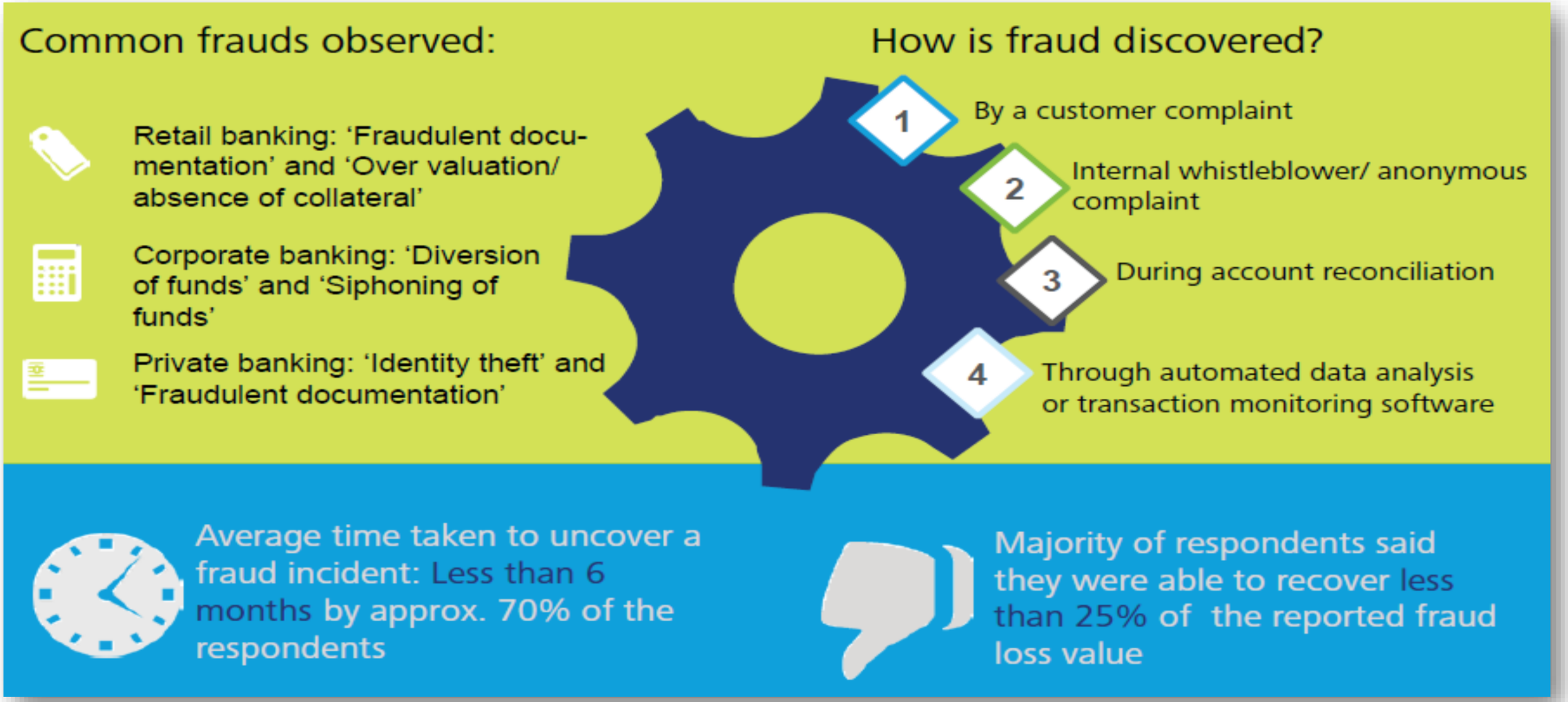
FRAUD IS A BIG PROBLEM

- “More than 23 billion credit card transactions are processed annually in USA”

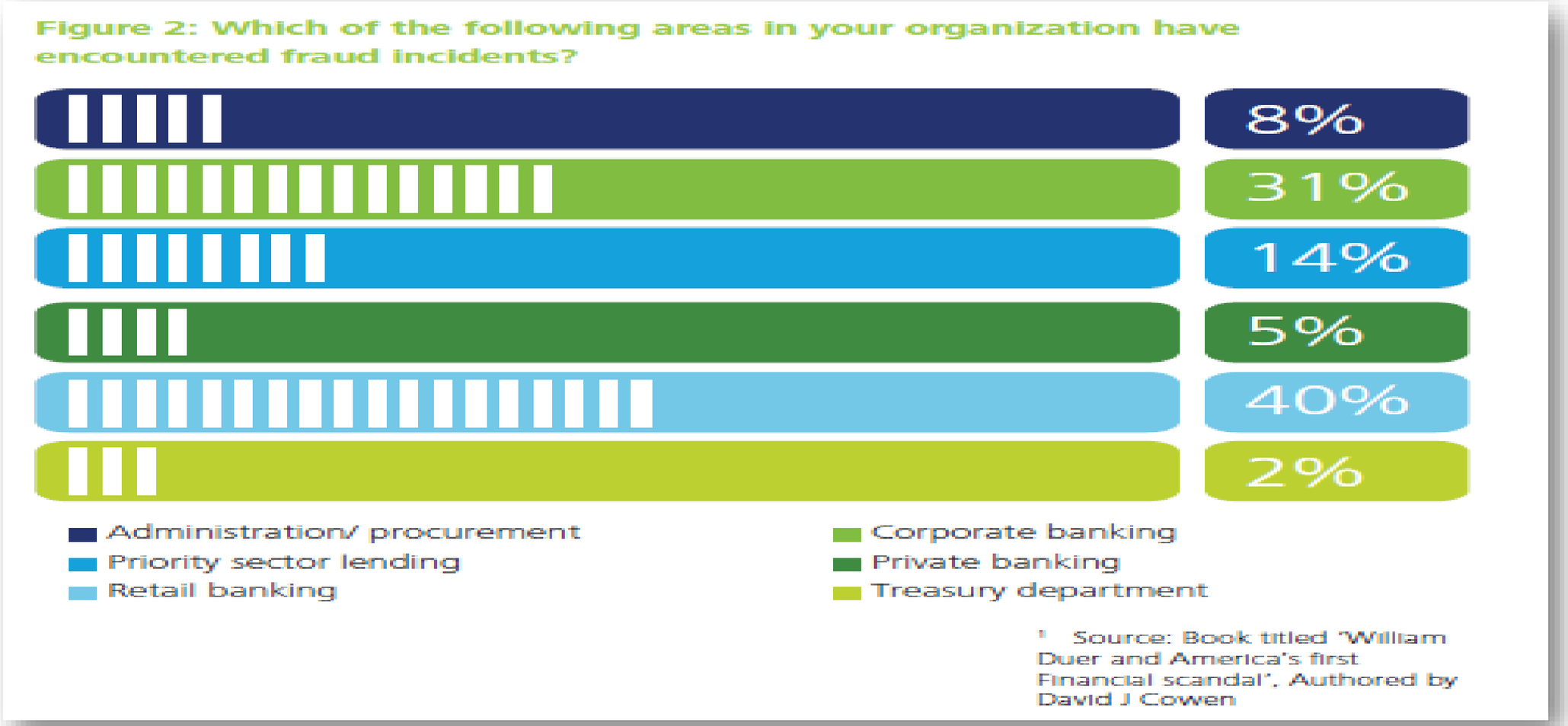
CreditCards.com

- Credit card transaction alone generates multiple Terabytes of data a year

SOME INTERESTING FACTS:



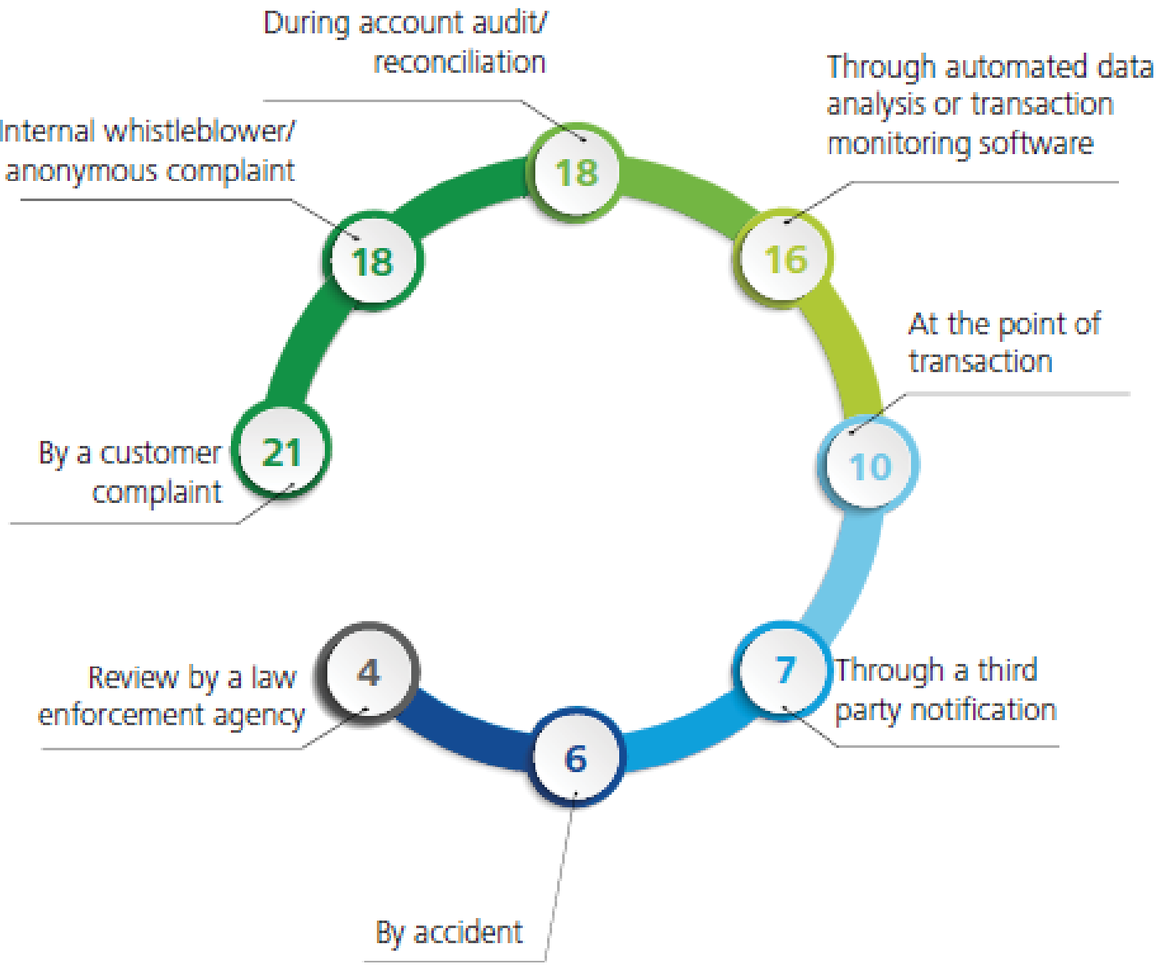
RETAIL BANKING – MOST FRAUD ENCOUNTERED AREA



REASONS FOR INCREASE IN FRAUD DETECTION:

MOST TYPICALLY DETECTED BY - CUSTOMER COMPLAINT

Figure : How is a fraud incident involving your organization typically detected?

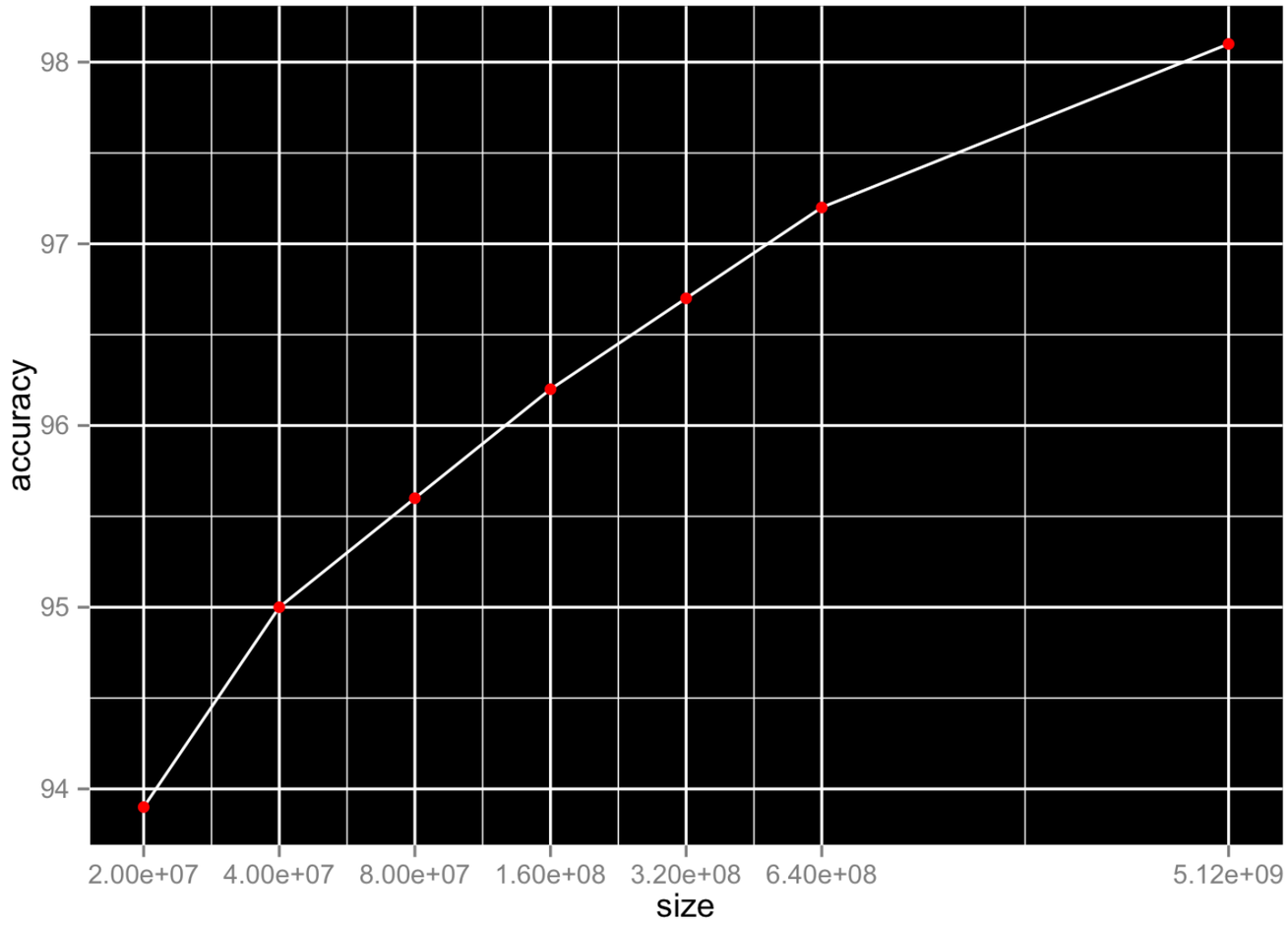


WHY MACHINE LEARNING?

- Traditional ideas of finding patterns does not scale to large datasets
- Machine Learning concerns with algorithms that can learn from data
- Even a tiny increase in accuracy can lead to millions of dollars in savings

BIGGER DATA, BETTER RESULTS ON REAL WORLD DATA

Dataset Size	AUC
20,000,000	93.9%
40,000,000	95.0%
80,000,000	95.6%
160,000,000	96.2%
320,000,000	96.7%
640,000,000	97.2%
5,120,000,000	98.1%



MACHINE LEARNING SOLUTION FOR FINANCIAL SERVICES

Multiple algorithms for higher accuracy

- ANN
- Random Decision Forest
- Logistic Regression
- SOM
- Mixed models (combine supervised and unsupervised models)

Automatic Parameter Selection

- Automatically create best performing model for any algorithm in fewer iteration
- Allow for usage by domain experts (non data scientists)
- Higher Accuracy machine can tune better than humans

Speed and Scalability

- Catch latest trends in fraud
- Improve accuracy
- Iterate over multiple algorithms and parameters
- Faster model creation and model update

Visualization and Optimization

- Visualize model performance
- Provide knobs to choose a model
- Ensure optimality of models without over-feeding
- Visualize models to interpret results

MACHINE LEARNING APPROACH

Supervised Learning: Predict Fraud	Collect historical transactions
	Learn from past examples of fraud
	Predict fraud (in real-time)
Unsupervised Learning: Discover Fraud	Segment transactions
	Investigate potentially new fraud
	Detect Outliers
Mixed Approach: Discover and predict Fraud	Detect “Points of Compromise” to prevent fraud

COMMON ISSUES

- Imbalanced Datasets

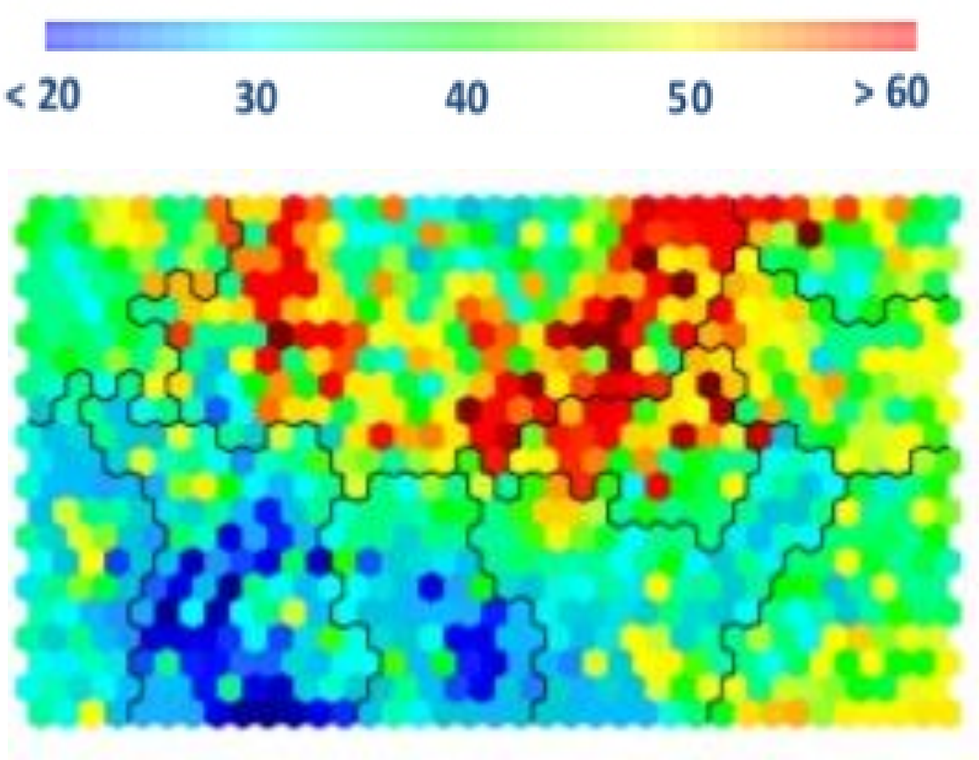
  Too few examples of ‘known’ fraud
- False positives

  Credit card transactions that are actually legit but financial institutions decline due to suspicion of fraud
- What loss function to use?
- How to handle missing values?
- Which algorithm to use?

CURRENT INDUSTRY STANDARD SOLUTION

SOM (Self Organizing Map) algorithm by Kohonen (kohonen map).

- A SOM is a form of unsupervised neural network that produces a low (typically two) dimensional representation of the input space of the set of training samples.
- The result of SOM is a HeatMap.
- Heatmap are used to discover pattern between variables.
- Heatmaps colour the map by chosen variables-Each node coloured using average value of all linked data points.



IMPORTING AND LOADING DATA

In [375]:
```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [376]:
```
dataset = pd.read_csv('Credit_Card_Applications.csv')
```

dataset - DataFrame

Index	CustomerID	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	Class
0	15776156	1	22.08	11.46	2	4	4	1.585	0	0	0	1	2	100	1213	0
1	15739548	0	22.67	7	2	8	4	0.165	0	0	0	0	2	160	1	0
2	15662854	0	29.58	1.75	1	4	4	1.25	0	0	0	1	2	280	1	0
3	15687688	0	21.67	11.5	1	5	3	0	1	1	11	1	2	0	1	1
4	15715750	1	20.17	8.17	2	6	4	1.96	1	1	14	0	2	60	159	1
5	15571121	0	15.83	0.585	2	8	8	1.5	1	1	2	0	2	100	1	1
6	15726466	1	17.42	6.5	2	3	4	0.125	0	0	0	0	2	60	101	0
7	15660390	0	58.67	4.46	2	11	8	3.04	1	1	6	0	2	43	561	1
8	15663942	1	27.83	1	1	2	8	3	0	0	0	0	2	176	538	0
9	15638610	0	55.75	7.08	2	4	8	6.75	1	1	3	1	2	100	51	0
10	15644446	1	33.5	1.75	2	14	8	4.5	1	1	4	1	2	253	858	1
11	15585892	1	41.42	5	2	11	8	5	1	1	6	1	2	470	1	1
12	15609356	1	20.67	1.25	1	8	8	1.375	1	1	3	1	2	140	211	0
13	15803378	1	34.92	5	2	14	8	7.5	1	1	6	1	2	0	1001	1
14	15599440	1	58.58	2.71	2	8	4	2.415	0	0	0	1	2	320	1	0
15	15692408	1	48.08	6.04	2	4	4	0.04	0	0	0	0	2	0	2691	1
16	15683168	1	29.58	4.5	2	9	4	7.5	1	1	2	1	2	330	1	1

CREATING AND SCALING VARIABLE X

```
In [377]: X = dataset.iloc[:, :-1].values
          y = dataset.iloc[:, -1].values

In [378]: from sklearn.preprocessing import MinMaxScaler
          sc = MinMaxScaler(feature_range = (0, 1))

In [379]: X = sc.fit_transform(X)
```

X - NumPy array

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1.57762e+07	1	22.08	11.46	2	4	4	1.585	0	0	0	1	2	100	1213
1	1.57395e+07	0	22.67	7	2	8	4	0.165	0	0	0	0	2	160	1
2	1.56629e+07	0	29.58	1.75	1	4	4	1.25	0	0	0	1	2	280	1
3	1.56877e+07	0	21.67	11.5	1	5	3	0	1	1	11	1	2	0	1
4	1.57158e+07	1	20.17	8.17	2	6	4	1.96	1	1	14	0	2	60	159
5	1.55711e+07	0	15.83	0.585	2	8	8	1.5	1	1	2	0	2	100	1
6	1.57265e+07	1	17.42	6.5	2	3	4	0.125	0	0	0	0	2	60	101
7	1.56604e+07	0	58.67	4.46	2	11	8	3.04	1	1	6	0	2	43	561
8	1.56639e+07	1	27.83	1	1	2	8	3	0	0	0	0	2	176	538
9	1.56386e+07	0	55.75	7.08	2	4	8	6.75	1	1	3	1	2	100	51
10	1.56444e+07	1	33.5	1.75	2	14	8	4.5	1	1	4	1	2	253	858
11	1.55859e+07	1	41.42	5	2	11	8	5	1	1	6	1	2	470	1
12	1.56094e+07	1	20.67	1.25	1	8	8	1.375	1	1	3	1	2	140	211

X - NumPy array

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0.842681	1	0.125263	0.409286	0.5	0.230769	0.375	0.055614	0	0	0	1	0.5	0.05	0.01212
1	0.696091	0	0.134135	0.25	0.5	0.538462	0.375	0.00578947	0	0	0	0	0.5	0.08	0
2	0.388982	0	0.238045	0.0625	0	0.230769	0.375	0.0438596	0	0	0	1	0.5	0.14	0
3	0.488425	0	0.119098	0.410714	0	0.307692	0.25	0	1	1	0.164179	1	0.5	0	0
4	0.600795	1	0.0965414	0.291786	0.5	0.384615	0.375	0.0687719	1	1	0.208955	0	0.5	0.03	0.00158
5	0.0216515	0	0.0312782	0.0208929	0.5	0.538462	0.875	0.0526316	1	1	0.0298507	0	0.5	0.05	0
6	0.643706	1	0.055188	0.232143	0.5	0.153846	0.375	0.00438596	0	0	0	0	0.5	0.03	0.001
7	0.379115	0	0.675489	0.159286	0.5	0.769231	0.875	0.106667	1	1	0.0895522	0	0.5	0.0215	0.0056
8	0.393338	1	0.211729	0.0357143	0	0.0769231	0.875	0.105263	0	0	0	0	0.5	0.088	0.00537
9	0.2919	0	0.631579	0.252857	0.5	0.230769	0.875	0.236842	1	1	0.0447761	1	0.5	0.05	0.0005
10	0.31527	1	0.296992	0.0625	0.5	1	0.875	0.157895	1	1	0.0597015	1	0.5	0.1265	0.00857
11	0.0807996	1	0.41609	0.178571	0.5	0.769231	0.875	0.175439	1	1	0.0895522	1	0.5	0.235	0

UNSUPERVISED LEARNING: SOM

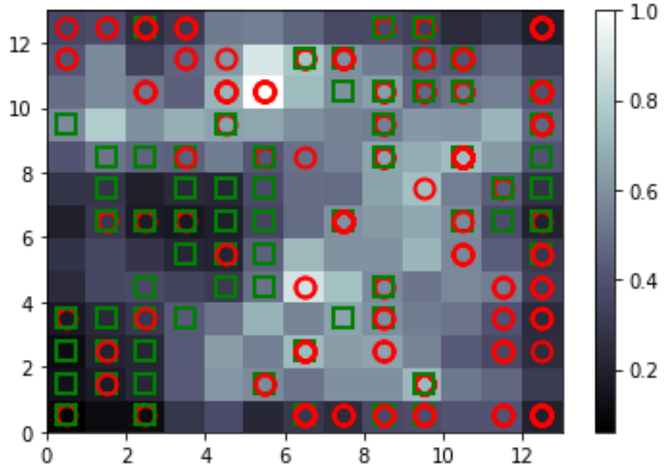
IMPORTING MINISOM CLASS AND CREATING OBJECT

In [380]:
```
from minisom import MiniSom
som = MiniSom(x = 13, y = 13, input_len = 15, sigma = 1.0, learning_rate = 0.5, random_seed=42)
```

In [381]:
```
# randomly initialize the weight vectors to small numbers close to 0
som.random_weights_init(X)
# train som on X, matrix of features and patterns recognized
som.train_random(data = X, num_iteration = 100)
```

INSIGHTS OF SOM

In [382]:
```
from pylab import bone, pcolor, colorbar, plot, show
bone()
pcolor(som.distance_map().T)
colorbar()
markers = ['o', 's']
colors = ['r', 'g']
for i, x in enumerate(X):
    w = som.winner(x)
    plot(w[0] + 0.5,
         w[1] + 0.5,
         markers[y[i]],
         markeredgecolor = colors[y[i]],
         markerfacecolor = 'None',
         markersize = 10,
         markeredgewidth = 2)
show()
```



markers - List (2 elements)

Index	Type	Size	Value
0	str	1	o
1	str	1	s

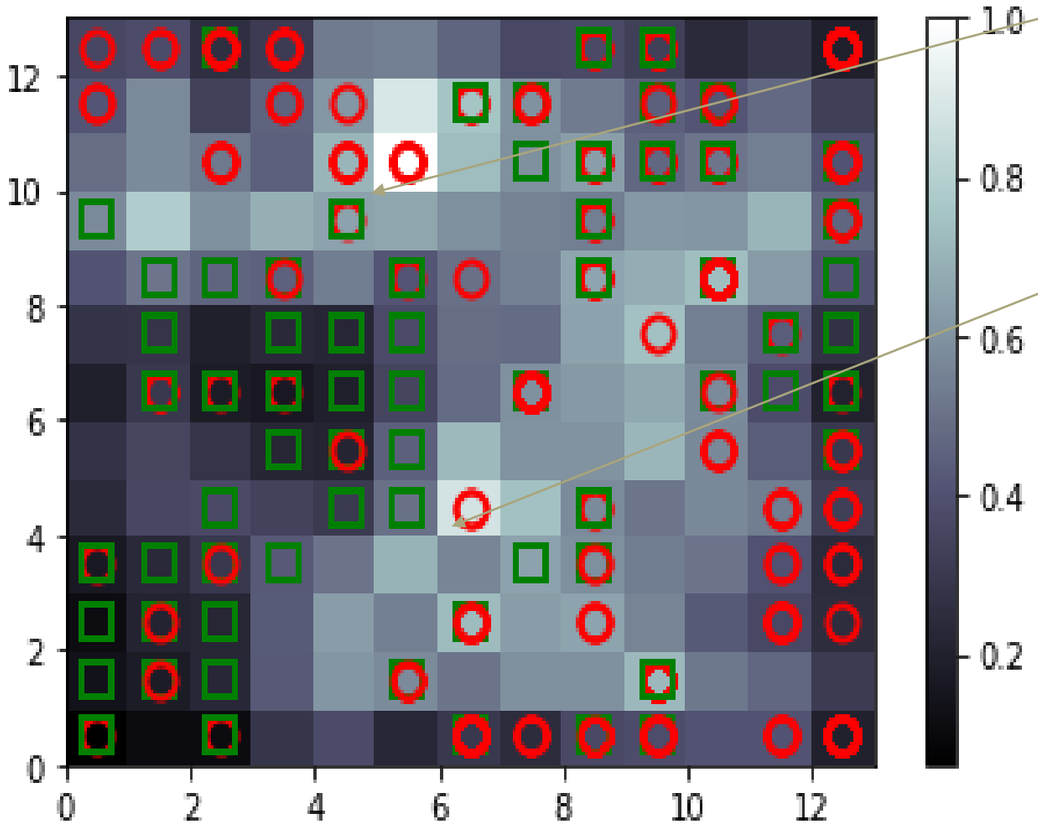
colors - List (2 elements)

Index	Type	Size	Value
0	str	1	r
1	str	1	g

SELECTION OF MOST LIKELY FRAUDS

```
# Finding the frauds
mappings = som.win_map(X)
```

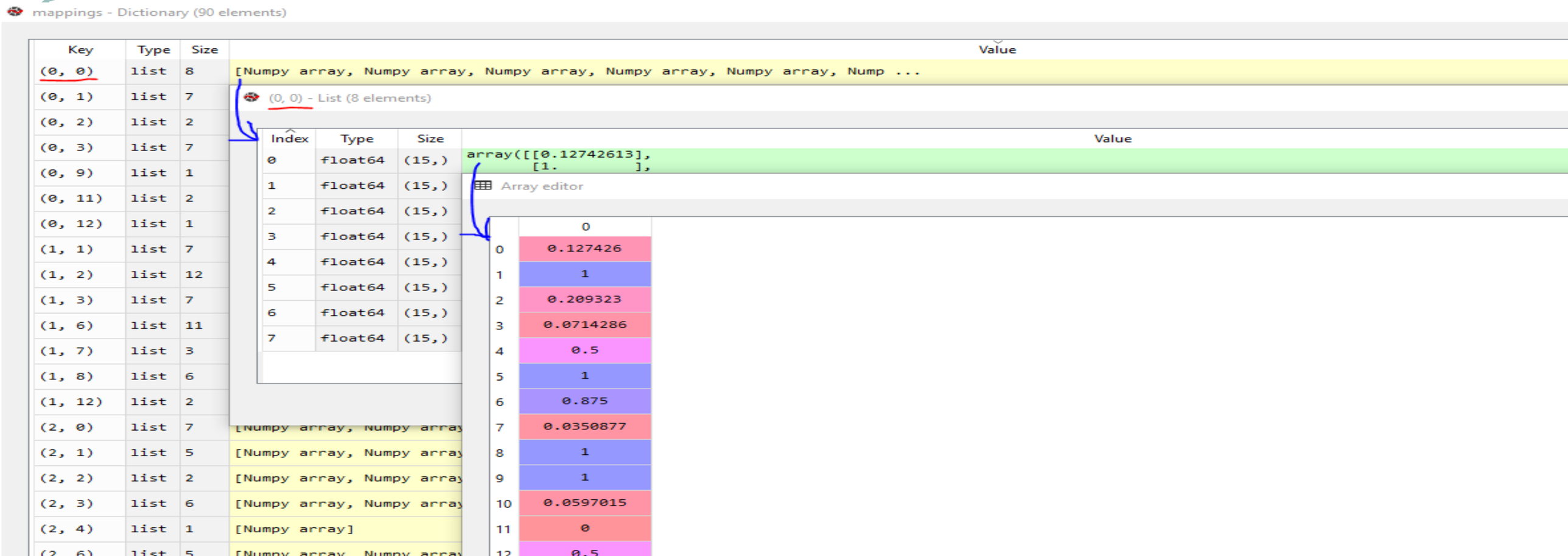
```
frauds = np.concatenate((mappings[(5,10)], mappings[(6,4)]), axis = 0)
frauds = sc.inverse_transform(frauds)
```



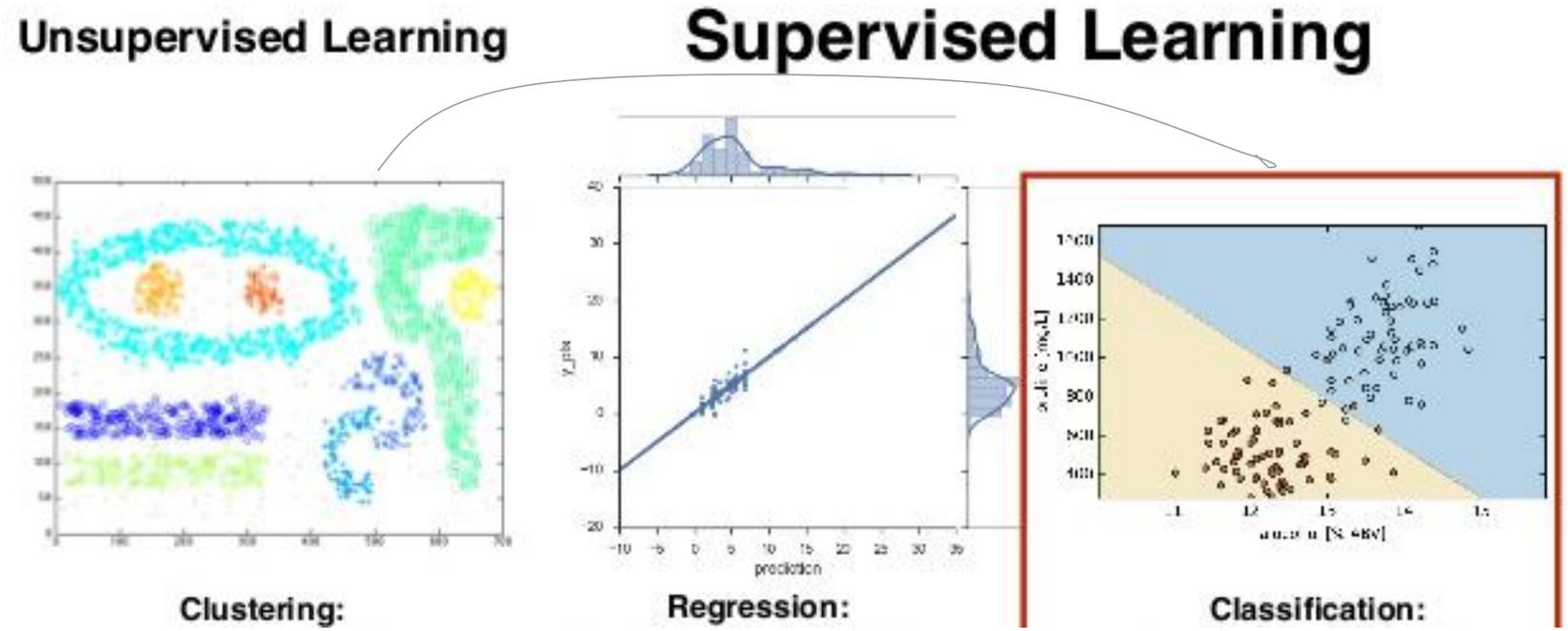
UMDERSTANDING MAPPINGS OF USERID(S):

```
# Finding the frauds
mappings = som.win_map(X)
```

```
frauds = np.concatenate((mappings[(5,10)], mappings[(6,4)]), axis = 0)
frauds = sc.inverse_transform(frauds)
```



UNSUPERVISED TO SUPERVISED DEEP LEARNING



For detecting the Frauds.

customer (X) + is_fraud (y)

Unsupervised to Supervised Deep Learning

In [385]:
```
# Creating the matrix of features
customers = dataset.iloc[:, 1:].values
```

In [386]:
```
# Creating the dependent variable
is_fraud = np.zeros(len(dataset))
for i in range(len(dataset)):
    if dataset.iloc[i,0] in frauds:
        is_fraud[i] = 1
```

In [387]:
```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
customers = sc.fit_transform(customers)
```

customers - NumPy array

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	22.08	11.46	2	4	4	1.585	0	0	0	1	2	100	1213	0
1	0	22.67	7	2	8	4	0.165	0	0	0	0	2	160	1	0
2	0	29.58	1.75	1	4	4	1.25	0	0	0	1	2	280	1	0
3	0	21.67	11.5	1	5	3	0	1	1	11	1	2	0	1	1
4	1	20.17	8.17	2	6	4	1.96	1	1	14	0	2	60	159	1

is_fraud - NumPy array

	0
0	0
1	0
2	0
3	0
4	0

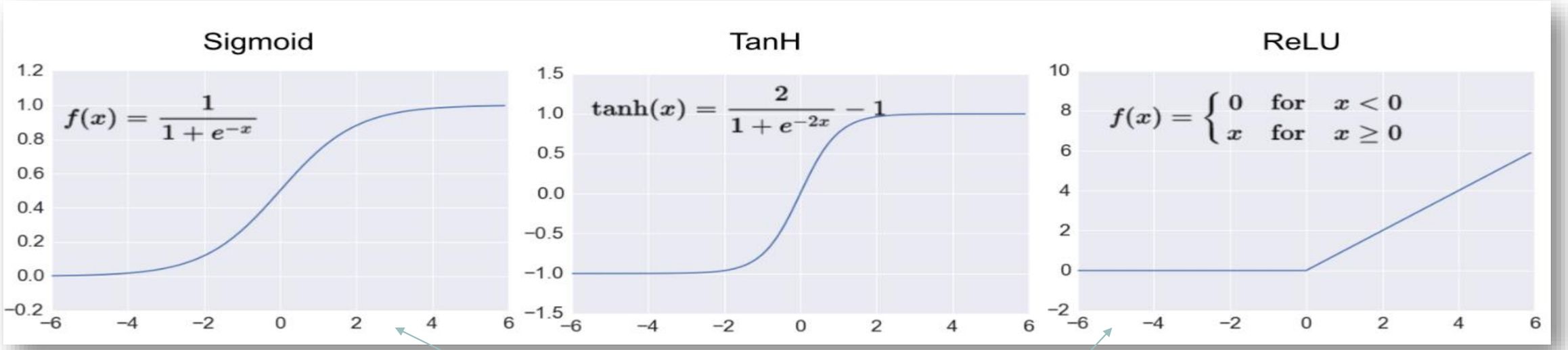
customers - NumPy array

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0.688737	-0.801052	1.34711	0.54295	-0.916282	-0.347965	-0.190906	-1.0475	-0.864196	-0.493887	1.08791	0.237828	-0.488358	0.0373804	-0.89530
1	-1.45193	-0.75124	0.450548	0.54295	0.170499	-0.347965	-0.615536	-1.0475	-0.864196	-0.493887	-0.919195	0.237828	-0.139591	-0.195413	-0.89530
2	-1.45193	-0.167856	-0.604823	-1.78398	-0.916282	-0.347965	-0.291083	-1.0475	-0.864196	-0.493887	1.08791	0.237828	0.557943	-0.195413	-0.89530
3	-1.45193	-0.835667	1.35515	-1.78398	-0.644587	-0.850257	-0.664877	0.95465	1.15714	1.76976	1.08791	0.237828	-1.06964	-0.195413	1.11694
4	0.688737	-0.962306	0.685745	0.54295	-0.372892	-0.347965	-0.0787676	0.95465	1.15714	2.38712	-0.919195	0.237828	-0.72087	-0.165066	1.11694

TRAIN & TEST SPLIT

In [388]:
```
# Part 2 - Now let's make the ANN!

# Importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Dense
```

In [389]:
```
from sklearn.model_selection import train_test_split
customers_train, customers_test, is_fraud_train, is_fraud_test = train_test_split(customers, is_fraud, test_size =.30,random_state
```



ANN

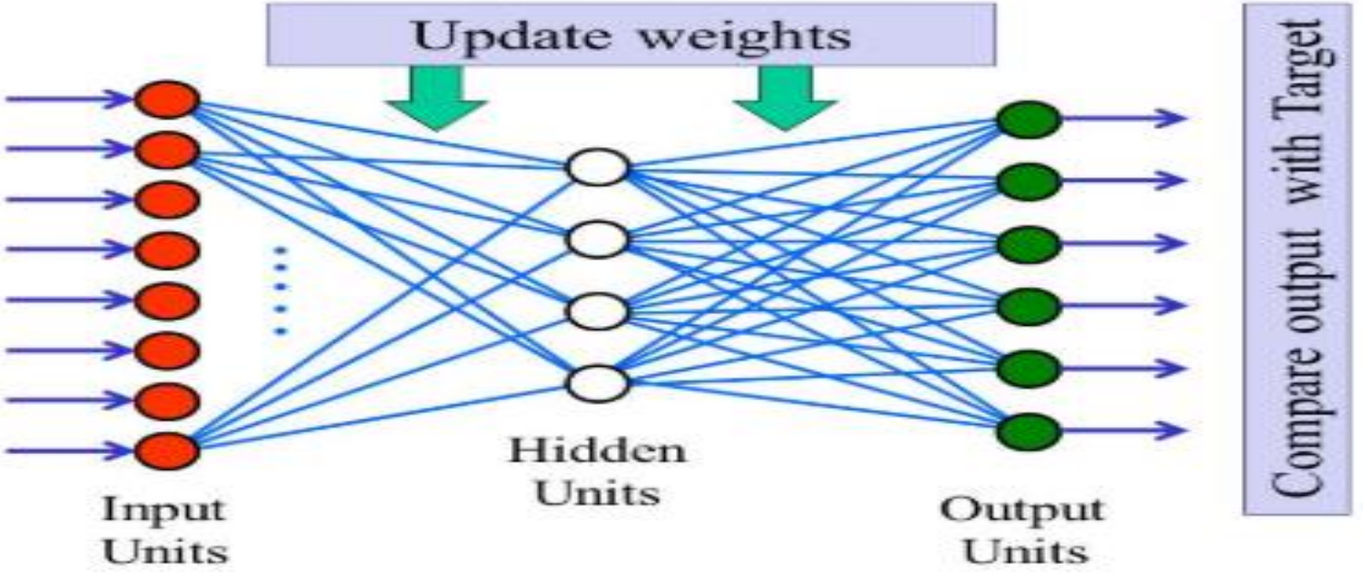
In [390]:
```
# Initialising the ANN
classifier = Sequential()

# Adding the input layer and the first hidden layer
classifier.add(Dense(units = 8, kernel_initializer = 'uniform', activation = 'relu', input_dim = 15))


# Adding the output layer
classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))

# Compiling the ANN
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

In [391]:
```
# Fitting the ANN to the Training set
classifier.fit(customers_train, is_fraud_train, batch_size = 2, epochs = 2)
```

```
Epoch 1/2
483/483 [==============================] - 1s 2ms/step - loss: 0.5691 - acc: 0.9752
Epoch 2/2
483/483 [==============================] - 0s 812us/step - loss: 0.1908 - acc: 0.9876
```

Out[391]:
```
<keras.callbacks.History at 0x14809a25a90>
```

Artificial Neural Network



In [392]:
```
# Predicting the probabilities of frauds
y_pred = classifier.predict(customers_test)
```

In [393]:
```
y_pred = (y_pred > 0.5)
```

In [394]:
```
from sklearn.metrics import classification_report, accuracy_score,confusion_matrix
cm = confusion_matrix(is_fraud_test, y_pred)
```

In [395]:
```
cm
```

Out[395]:
```
array([[205,   0],
       [  2,   0]], dtype=int64)
```

In [396]:
```
#print accuracy
print(accuracy_score(is_fraud_test,y_pred))
```

```
0.9903381642512077
```

AFTER TRYING:

- ANN
- LOGISTIC REGRESSION
- KNN ALGORITHM
- RANDOM FOREST ETC.

WINNER FOR US WITH MINIMAL EFFORT IS

- **Random forest**

ACCURACY SCORE OF 99.27%

RandomForestClassifier

In [410]:
```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(random_state = 42)
# Fit the regressor object into the training set
classifier.fit(customers_train, is_fraud_train)
# regressor is the machine that learns the corelation of the training set to make some future predictions

y_pred = classifier.predict(customers_test)
print(accuracy_score(is_fraud_test,y_pred))
```

```
0.9927536231884058
```

THANK YOU

REESHU ROY
JAYANTA ROY