ITMO UNIVERSITY

Information Technologies and Programming Faculty

# Financial Fraud Detection

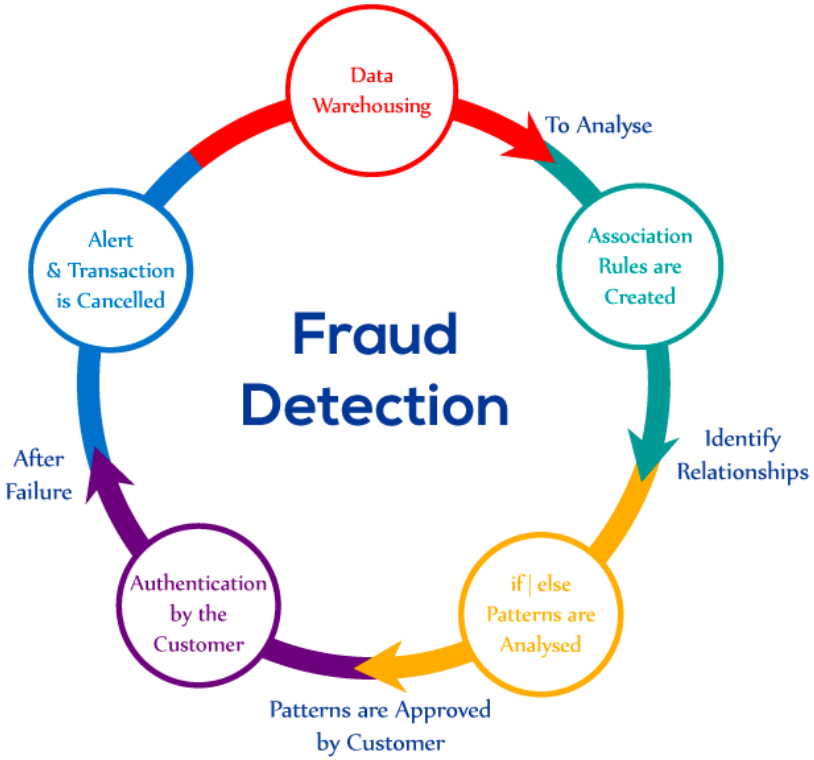
Supervised: Prof. Andrey Filchenkov

Presented: M Saber – M41332

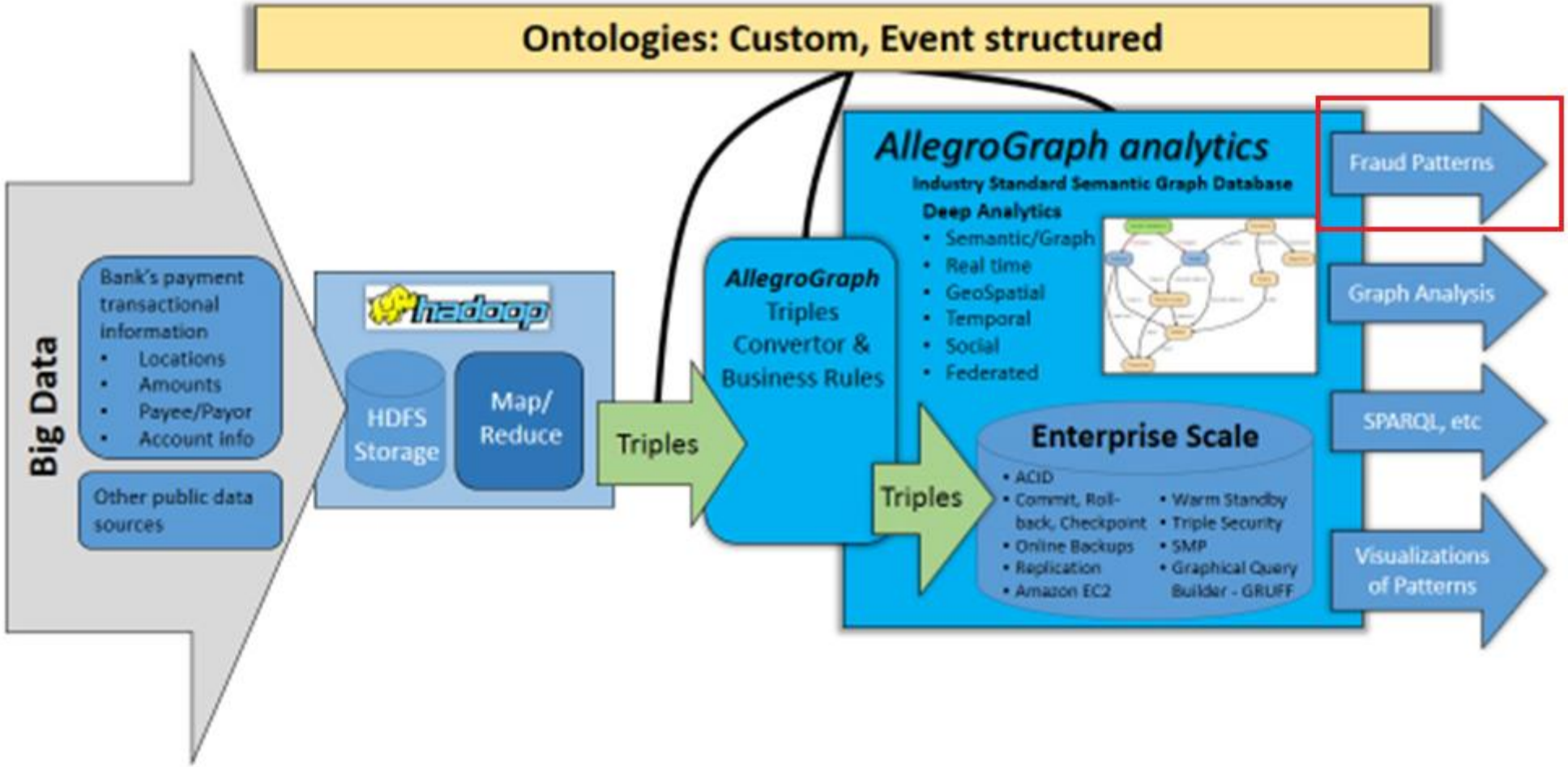
# Objective

The finance and banking is very important sector in our present day generation, where almost every human has to deal with bank either physically or online. The productivity and profitability of both public and private sector has tremendously increased because of banking information system. Nowadays most of E-commerce application system transactions are done through credit card and online net banking. These systems are vulnerable with new attacks and techniques at alarming rate.

Fraud detection in banking is one of the vital aspects nowadays as finance is major sector in our life



# Online Bank Fraud Detection



# Steps Processing FFD

| | |
|---|---|
| 1- Exploration Data Analysis (EDA) | |
| 2- Data Preprocessing | Features Scaling \| Standardization |
| 3- Handling Imbalanced data | Manual Undersample - Auto processing |
| 4- Visualization | Correlation - Boxplot |
| 5- Handing Outlier | IQR |
| 6- Dim. Reduction | t-SNE |
| 7- Classification Algorithms | …. |

# 1- Exploration Data Analysis (EDA)

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 | V11 | V12 | V13 | V14 | Amount | Class |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.0 | -1.359807 | -0.072781 | 2.536347 | 1.378155 | -0.338321 | 0.462388 | 0.239599 | 0.098698 | 0.363787 | 0.090794 | -0.551600 | -0.617801 | -0.991390 | -0.311169 | 149.62 | 0 |
| 1 | 0.0 | 1.191857 | 0.266151 | 0.166480 | 0.448154 | 0.060018 | -0.082361 | -0.078803 | 0.085102 | -0.255425 | -0.166974 | 1.612727 | 1.065235 | 0.489095 | -0.143772 | 2.69 | 0 |
| 2 | 1.0 | -1.358354 | -1.340163 | 1.773209 | 0.379780 | -0.503198 | 1.800499 | 0.791461 | 0.247676 | -1.514654 | 0.207643 | 0.624501 | 0.066084 | 0.717293 | -0.165946 | 378.66 | 0 |
| 3 | 1.0 | -0.966272 | -0.185226 | 1.792993 | -0.863291 | -0.010309 | 1.247203 | 0.237609 | 0.377436 | -1.387024 | -0.054952 | -0.226487 | 0.178228 | 0.507757 | -0.287924 | 123.50 | 0 |
| 4 | 2.0 | -1.158233 | 0.877737 | 1.548718 | 0.403034 | -0.407193 | 0.095921 | 0.592941 | -0.270533 | 0.817739 | 0.753074 | -0.822843 | 0.538196 | 1.345852 | -1.119670 | 69.99 | 0 |

Max , Min [Time]

172792.0 , 0.0

Need Features Scaling

Max , Min [Amount]

25691.16 , 0.0

```
# Check is null
print ( "Count of Null = " , df.isnull().sum().max())  # >> Count of Null =  0
```

```
# Percentage of each part of fraud and non-fraud from dataset
print('No Frauds = ', round(df['Class'].value_counts()[0]/len(df) * 100,2), '%')  # >> No Frauds =  99.83 %
print('Frauds = ', round(df['Class'].value_counts()[1]/len(df) * 100,2), '%') # >> Frauds =  0.17 %
```


Need Handing Imbalance data

No Frauds = 99.83 %
Frauds = 0.17 %

(284315, 492)

# 2- Handling Imbalanced data

## Manual Processing

**Create Sub-sampling**

subsample will be a data frame with a 50/50 ratio of fraud and non-fraud transactions. Meaning our sub-sample will have the same amount of fraud and non fraud transactions

**Handling Imbalance data**

**Handling Outlier in:**

- - Correlation (0,1)
- + Correlation (0,1)

## Resample Processing

**OVER-sampling**

Which is adding copies of the under-represented class
Better when you have little data

||

**UNDER-sampling**

Which deletes instances from the over-represented class
Better when he has lots of data

||

**SMOTE**

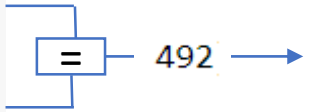
Synthetic Minority Over-Sampling Technique
It is combination of oversampling and undersampling

# 2- Features Scaling / Standardization

| | scaled_amount | scaled_time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 | V11 | V12 | V13 | V28 | Class |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.783274 | -0.994983 | -1.359807 | -0.072781 | 2.536347 | 1.378155 | -0.338321 | 0.462388 | 0.239599 | 0.098698 | 0.363787 | 0.090794 | -0.551600 | -0.617801 | -0.991390 | -0.021053 | 0 |
| 1 | -0.269825 | -0.994983 | 1.191857 | 0.266151 | 0.166480 | 0.448154 | 0.060018 | -0.082361 | -0.078803 | 0.085102 | -0.255425 | -0.166974 | 1.612727 | 1.065235 | 0.489095 | 0.014724 | 0 |
| 2 | 4.983721 | -0.994972 | -1.358354 | -1.340163 | 1.773209 | 0.379780 | -0.503198 | 1.800499 | 0.791461 | 0.247676 | -1.514654 | 0.207643 | 0.624501 | 0.066084 | 0.717293 | -0.059752 | 0 |
| 3 | 1.418291 | -0.994972 | -0.966272 | -0.185226 | 1.792993 | -0.863291 | -0.010309 | 1.247203 | 0.237609 | 0.377436 | -1.387024 | -0.054952 | -0.226487 | 0.178228 | 0.507757 | 0.061458 | 0 |
| 4 | 0.670579 | -0.994960 | -1.158233 | 0.877737 | 1.548718 | 0.403034 | -0.407193 | 0.095921 | 0.592941 | -0.270533 | 0.817739 | 0.753074 | -0.822843 | 0.538196 | 1.345852 | 0.215153 | 0 |

# 3- Handling Imbalanced data > Create Sub-sample

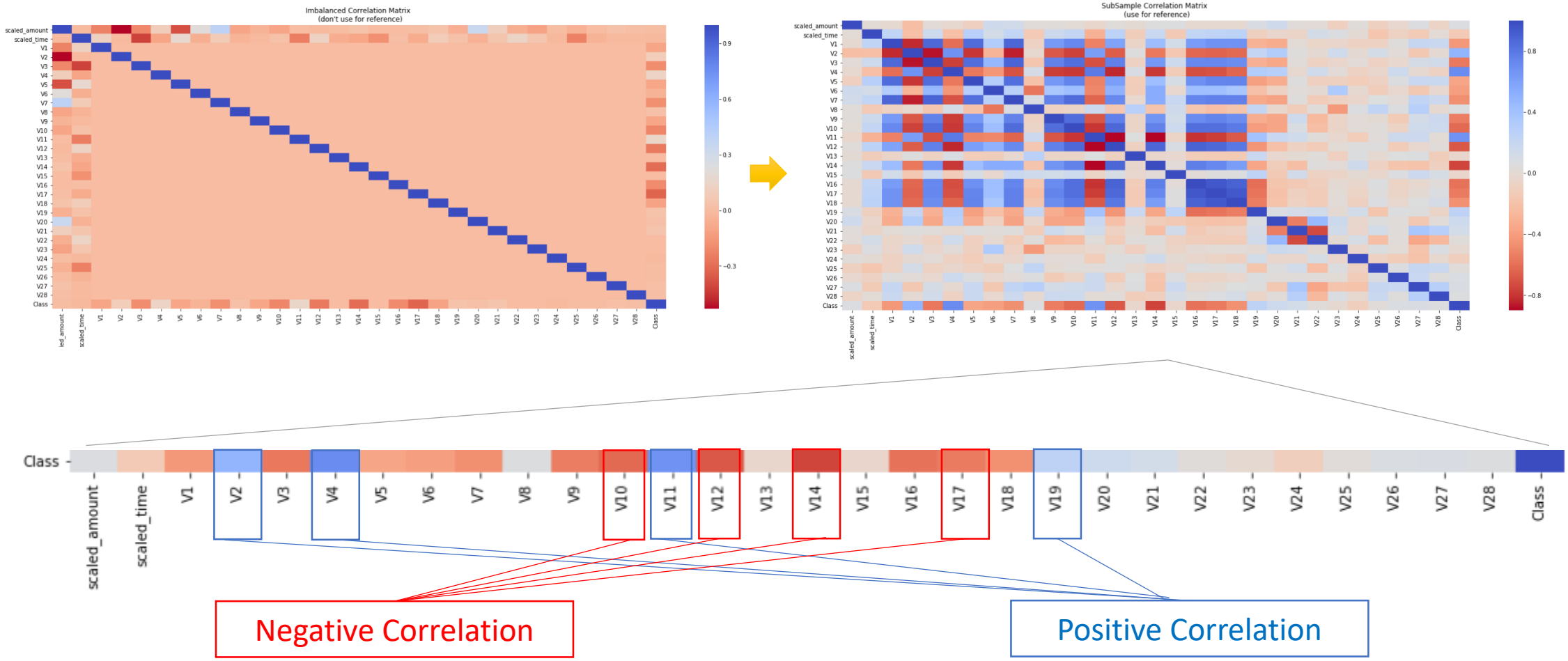
```
# amount of fraud classes 492 rows.
fraud_df = df.loc[df['Class'] == 1] # aready count = 492
non_fraud_df = df.loc[df['Class'] == 0][:492] # select only 492 rows
```




# 4- Visualization > Correlation

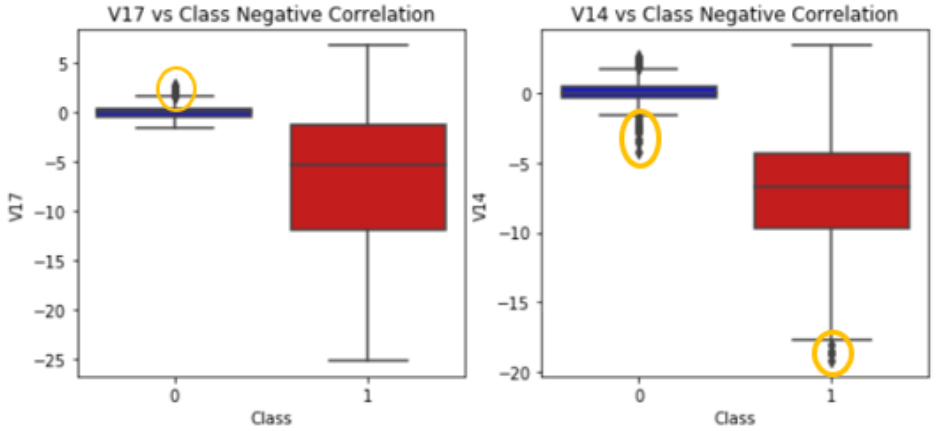
Imbalanced Correlation Matrix

Sub-sample Correlation Matrix



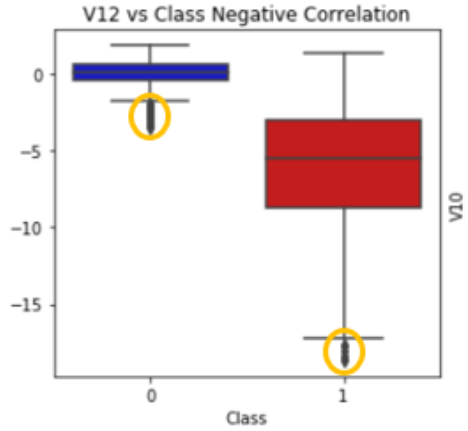
# 4- Visualization > Boxplot

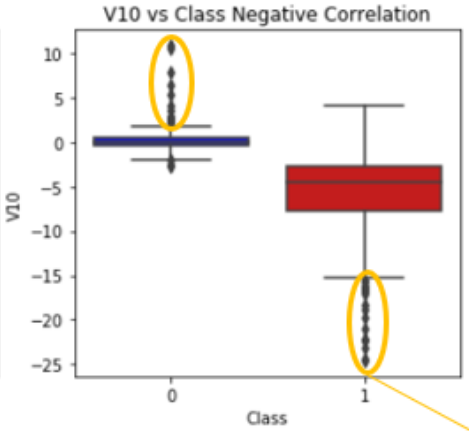
Negative Correlation



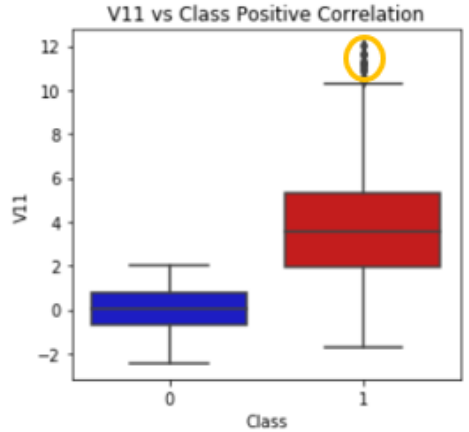


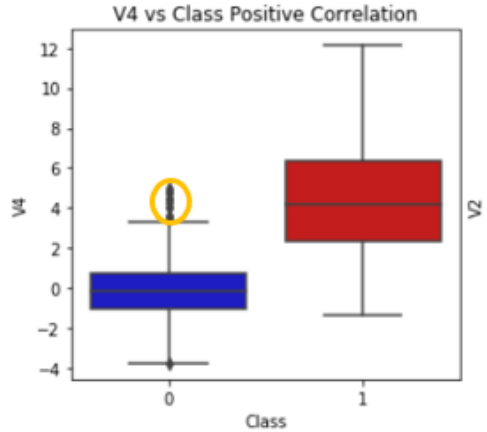


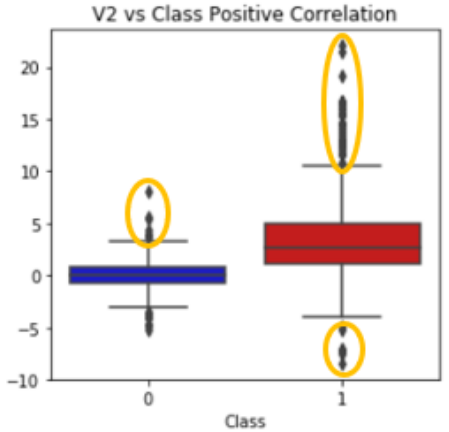


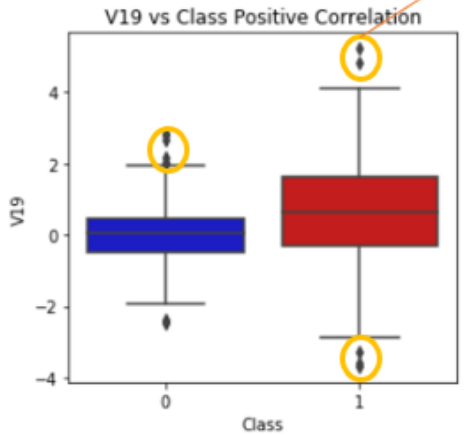



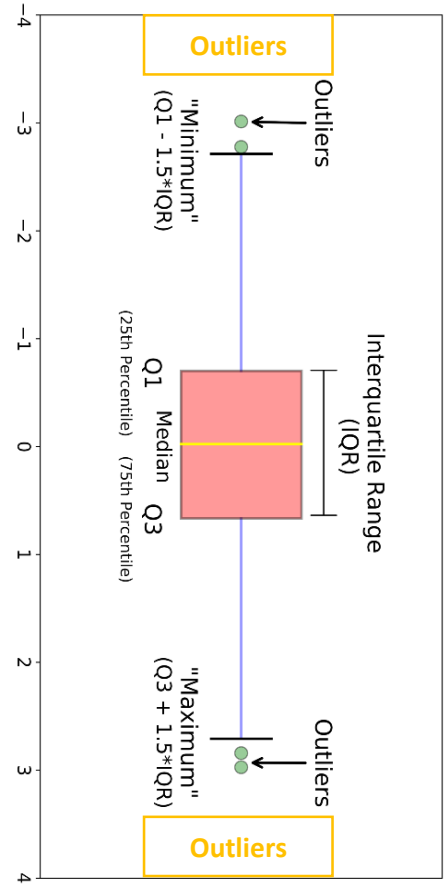

Positive Correlation



# 5- Handling Outliers

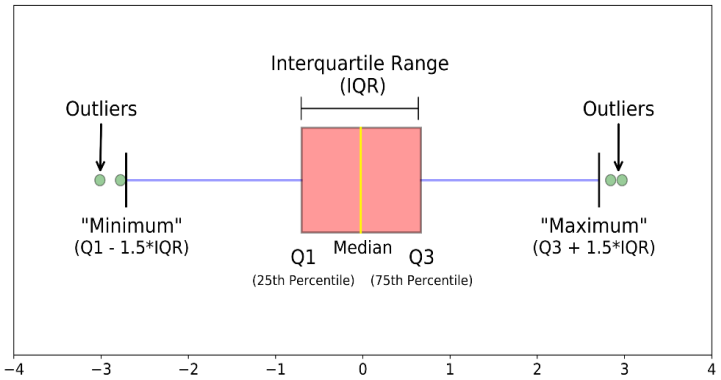
## Math Expression

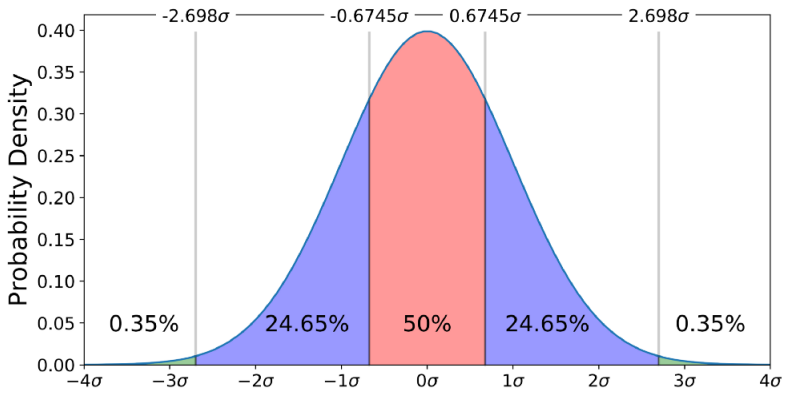
$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$$

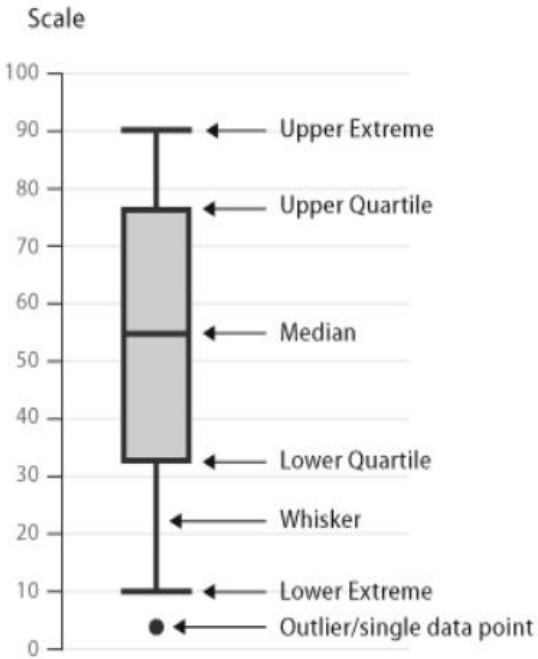
## Code to Integrate

```
# Make PDF for the normal distribution a function
def normalProbabilityDensity(x):
    constant = 1.0 / np.sqrt(2*np.pi)
    return(constant * np.exp((-x**2) / 2.0) )

# Integrate from -inf to +inf
result, _ = quad(normalProbabilityDensity,
                 -np.inf,
                 np.inf,
                 limit = 1000)

print(result)
```

```
1.0
```



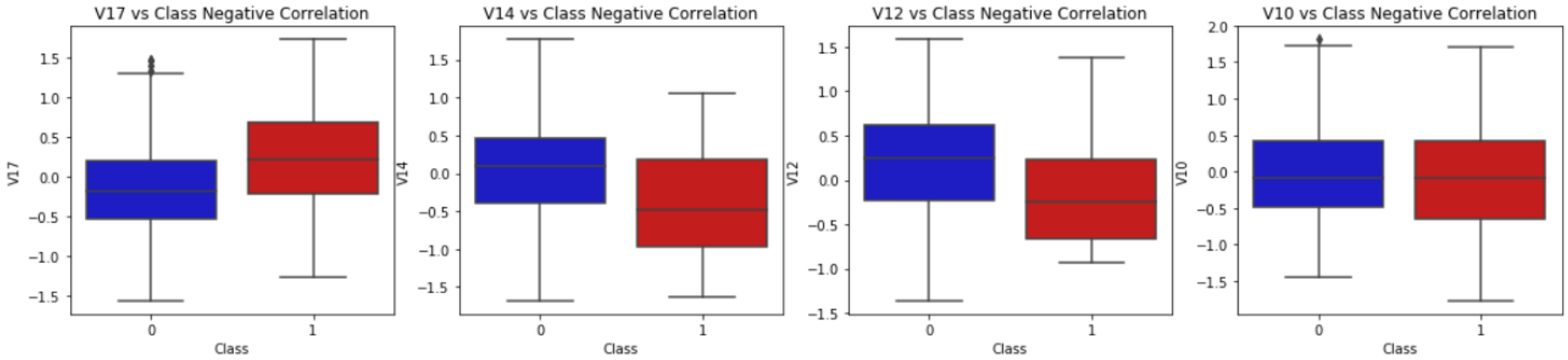



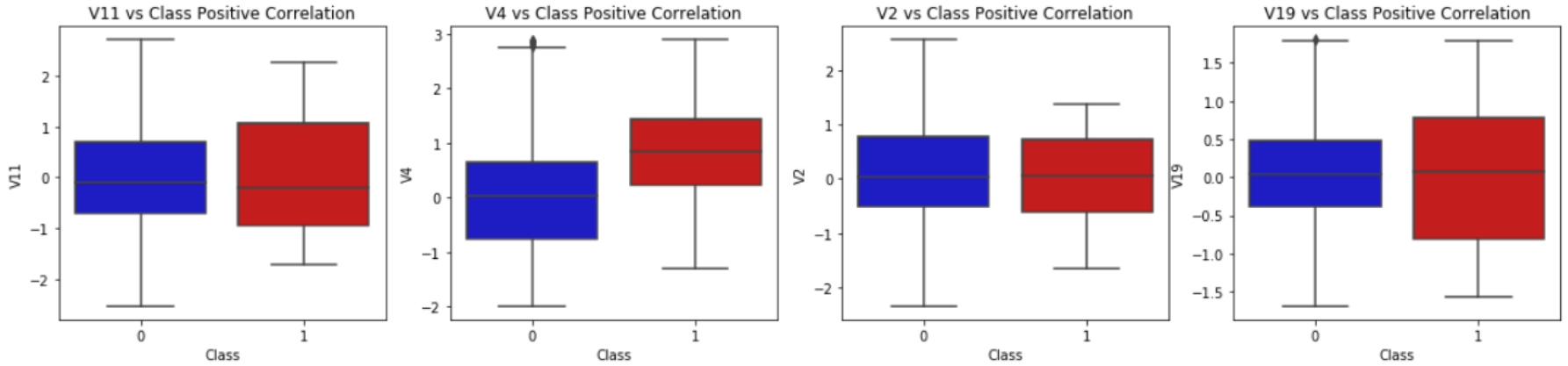

**Interquartile Range (IQR)**

# 5- Handling Outliers > Boxplot

Negative Correlation

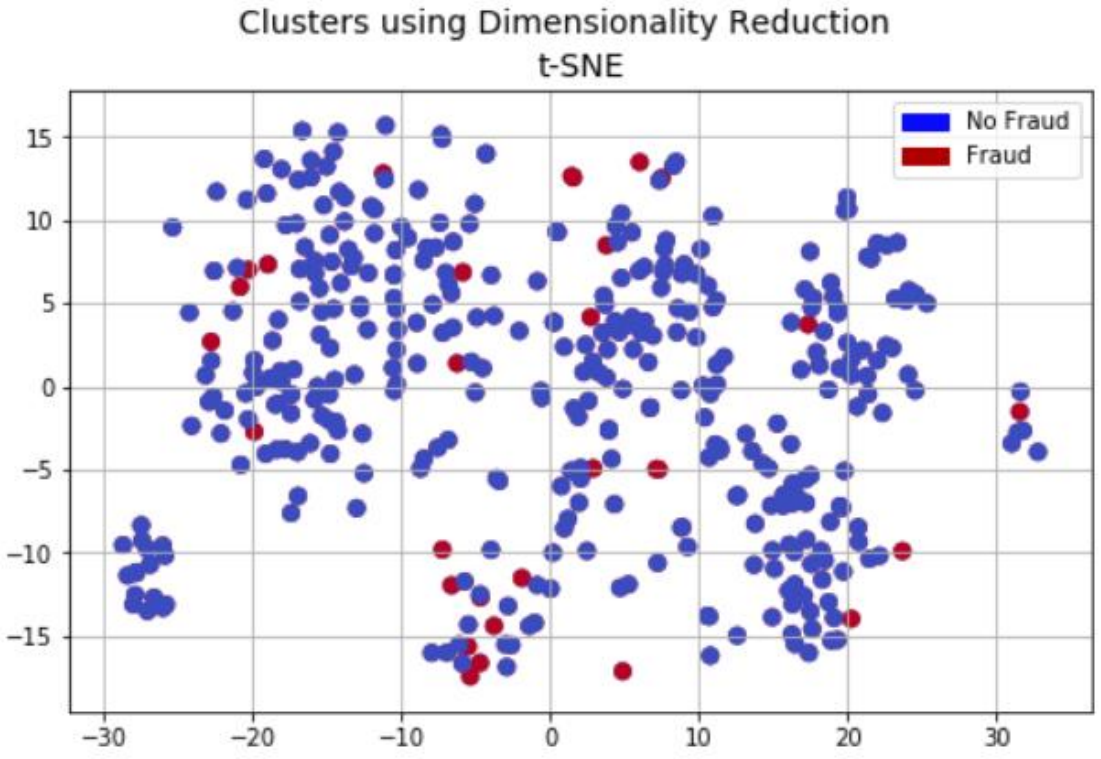



Positive Correlation



# 6- Dim. Reduction and Clustering (t-SNE)

t-SNE: non-linear technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets



# 6- Classification Algorithms

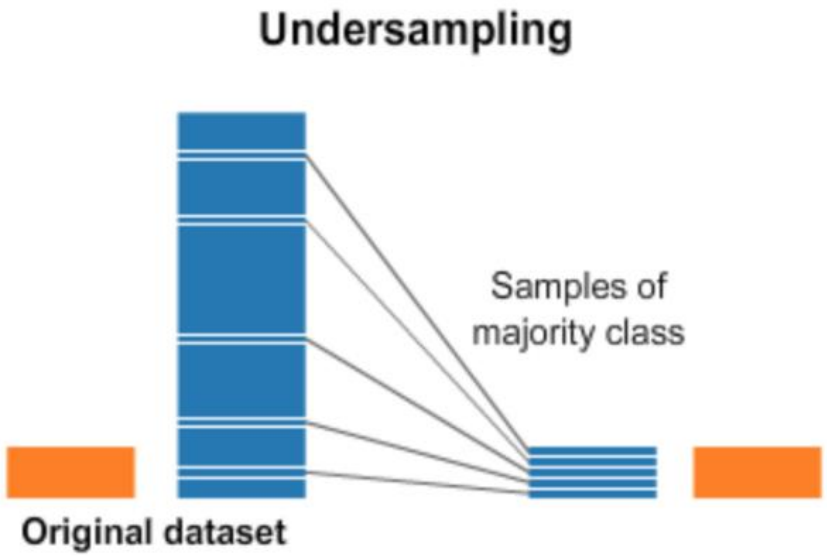
Don't use **accuracy score** as a metric with imbalanced datasets (will be usually high and misleading), instead use **f1-score**, **precision/recall score** or **confusion matrix**

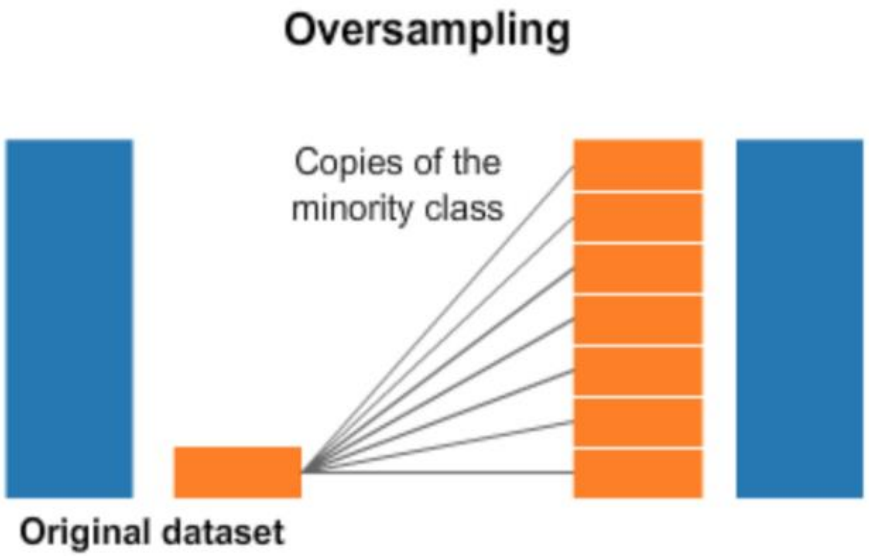
| ALGORITHMS | BEFOTR GRIDSEARCHCV | AFTER GRIDSEARCHCV | |
|---|---|---|---|
| Logistic Regression | 92.0 % | 93.01% | ↑ |
| K-neighbors Classifier | 93.0 % | 92.72% | ↓ |
| Support Vector Classifier | 93.0% | 93.3% | ↑ |
| Decision Tree Classifier | 86.0 % | 90.1% | ↑ |

| after to found the best parameters

# 7- SMOTE with Imbalance Data

- **Sample Data:**



# 7- SMOTE with Imbalance Data

SMOTE It is oversampling

SMOTEEN It is combination of oversampling and undersampling

- Build Model STME : to handling imbalance data between fraud and non-fraud
- Implement GridSearchCV
- Applied best parameter in SMOTE

| ALGORITHMS | BEFOTR GRIDSEARCHCV | AFTER GRIDSEARCHCV | AG-SMOTE |
|---|---|---|---|
| Logistic Regression | 92.0 % | 93.01% | |
| K-neighbors Classifier | 93.0 % | 92.72% | |
| Support Vector Classifier | 93.0% | 93.3% | 98.38 % |
| Decision Tree Classifier | 86.0 % | 90.1% | |

New: This example was implemented based on combination of traditional method of data processing so that selected best algorithm and pass it to SMOTE to reach high accuracy

Thank you