|  |
| --- |
| AI ALGORITHMS I  Credit Card Fraud Detection (V2)  D |
|  |
| December 2,2020   |  |  | | --- | --- | | Submitted To: | Submitted By: | | Marcos Bittencourt | Aman Verma | | AI ALGORITHMS I | 100799391 | |

Contents

1. Executive Summary……………………………………………………………………………………….….3
2. Problem Statement……………………………………………………………………………….………….3
3. Rational Statement………………………………………………………………………………….……….4
4. Data Requirement …………………………………………………………………………………………...4
5. Data Analysis Approach………………...………………………………………………………………….4
6. Outline to follow and Test Process…………………………………………………………………….5
7. Data Scaling and Distribution…………………………………………………………………………….5
8. Exploratory Data Analysis………………………………………………………………………………….7
9. Feature Engineering Benefits……………………………………………………………………………12

## Executive Summary:

What is Credit Card Fraud?

Credit card fraud is when someone uses another person's credit card or account information to make unauthorized purchases or access funds through cash advances. Credit card fraud doesn’t just happen online; it happens in brick-and-mortar stores, too. As a business owner, you can avoid serious headaches – and unwanted publicity – by recognizing potentially fraudulent use of credit cards in your payment environment.



## Problem Statement

The Credit Card Fraud Detection Problem includes modeling past credit card transactions with the knowledge of the ones that turned out to be a fraud. This model is then used to identify whether a new transaction is fraudulent or not. Our aim here is to detect 100% of the fraudulent transactions while minimizing the incorrect fraud classifications.

## Rational Statement

Build a machine learning models, in order to make real-time fraud detection decisions tailored to the individual customer. Applying supervised learning and classification algorithms to build models which provide better insights, accurate decision and find interesting pattern in customer data. Improve decision making process by correctly predicting probability of default for detecting fraud.

## Data Requirements

We will use a dataset from the KAGLE to identify the fraud on Credit Card. Application and previous history are required for model building. The labels are included in the training data and train a model to predict the labels from the features. Details of the data required are as follow:

In the dataset, the features are scaled and the names of the features are not shown due to privacy reasons. Nevertheless, we can still analyze some important aspects of the dataset.

Assumption, Limitations and Constraints with the data:

* Understand the little distribution of the "little" data that was provided to us.
* Create a 50/50 sub-data frame ratio of "Fraud" and "Non-Fraud" transactions. (Near miss Algorithm)
* Determine the Classifiers we are going to use and decide which one has a higher accuracy.
* Create a Neural Network and compare the accuracy to our best classifier.
* Understand common mistake made with imbalanced datasets.

This dataset contains total of 284707 records which further composed of 31 columns and 284707 rows.

## Data Analysis Approach

This is a Classification problem and it is a scenario of Supervised Learning. This problem requires classification algorithms like Logistic Regression, Clustering, smote for sampling, Classifier etc., and try to understand the accuracy. This would be determined in the modeling phase of the project.

The software tools used will be: -

* Python – for EDA (Exploratory Data Analysis), Data Cleaning, Model Building, and Testing
* Jupyter Notebook – IDE for development
* Tableau/Power BI – for detailed visualizations

## Outline to Follow and Test Process:

I. Understanding our data

a) Gather Sense of our data

II. Preprocessing

a) Scaling and Distributing

b) Splitting the Data

III. Random Under Sampling and Oversampling

a) Distributing and Correlating

b) Anomaly Detection

c) Dimensionality Reduction and Clustering (t-SNE)

d) Classifiers

e) A Deeper Look into Logistic Regression

f) Oversampling with SMOTE

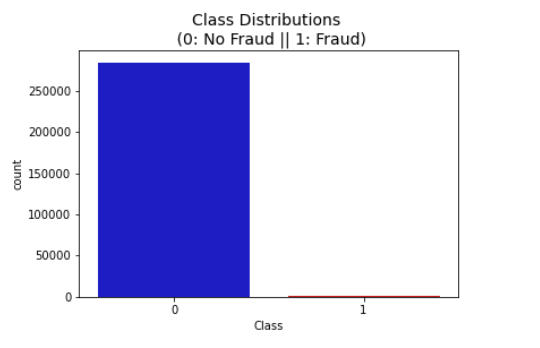
IV. Test Process

a) Testing with Logistic Regression

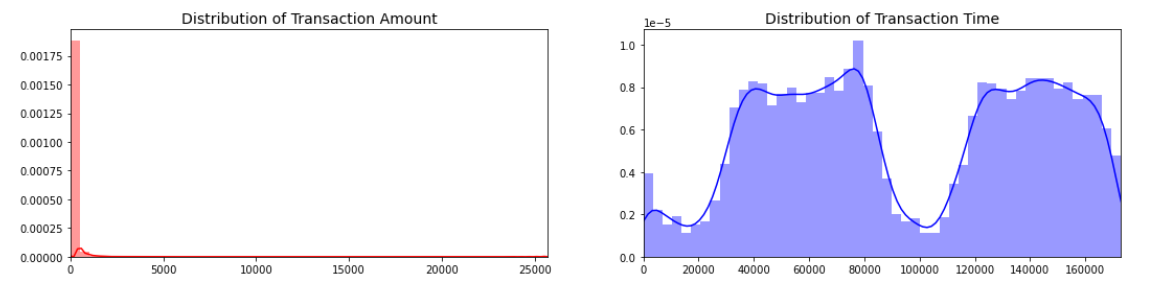
b) Neural Networks Testing (Under sampling vs Oversampling)

## Data Scaling and Distribution:

Class Distribution using Countplot



Distributions: By seeing the distributions we can have an idea how skewed are these features, we can also see further distributions of the other features. There are techniques that can help the distributions be less skewed which will be implemented.



Scaling and Distributing In this phase, we will first scale the columns comprise of Time and Amount. Time and amount should be scaled as the other columns. On the other hand, we need to also create a sub sample of the dataframe in order to have an equal amount of Fraud and Non-Fraud cases, helping our algorithms better understand patterns that determines whether a transaction is a fraud or not.

What is a sub-Sample? In this scenario, our subsample will be a dataframe 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.

Why do we create a sub-Sample? In the beginning, we saw that the original dataframe was heavily imbalanced! Using the original dataframe will cause the following issues:

Overfitting: Our classification models will assume that in most cases there are no frauds! What we want for our model is to be certain when a fraud occurs.

Wrong Correlations: Although we don't know what the "V" features stand for, it will be useful to understand how each of these features influence the result (Fraud or No Fraud) by having an imbalance dataframe we are not able to see the true correlations between the class and features.

Summary:

* Scaled amount and scaled time are the columns with scaled values.
* There are 492 cases of fraud in our dataset so we can randomly get 492 cases of non-fraud to create our new sub dataframe.
* We concat the 492 cases of fraud and non fraud, creating a new sub-sample.

### Splitting the Data (Original DataFrame)

* Before proceeding with the **Random Under Sampling technique** we have to separate the original dataframe. **Why? for testing purposes, remember although we are splitting the data when implementing Random Under Sampling or Over Sampling techniques, we want to test our models on the original testing set not on the testing set created by either of these techniques.** The main goal is to fit the model either with the dataframes that were under sample and oversample (in order for our models to detect the patterns), and test it on the original testing set.

**Random Under-Sampling:**

In this phase of the project we will implement "Random Under Sampling" which basically consists of removing data in order to have a more **balanced dataset** and thus avoiding our models to overfitting.

**Steps:**

* The first thing we have to do is determine how imbalanced is our class (use "value\_counts()" on the class column to determine the amount for each label)
* Once we determine how many instances are considered fraud transactions (Fraud = "1") , we should bring the non-fraud transactions to the same amount as fraud transactions (assuming we want a 50/50 ratio), this will be equivalent to 492 cases of fraud and 492 cases of non-fraud transactions.
* After implementing this technique, we have a sub-sample of our dataframe with a 50/50 ratio with regards to our classes. Then the next step we will implement is to shuffle the data to see if our models can maintain a certain accuracy everytime we run this script.

**Note:** The main issue with "Random Under-Sampling" is that we run the risk that our classification models will not perform as accurate as we would like to since there is a great deal of information loss (bringing 492 non-fraud transaction from 284,315 non-fraud transaction)

## Exploratory Data Analysis:

**Equally Distributing and Correlating**:

Now that we have our dataframe correctly balanced, we can go further with our analysis and data preprocessing.



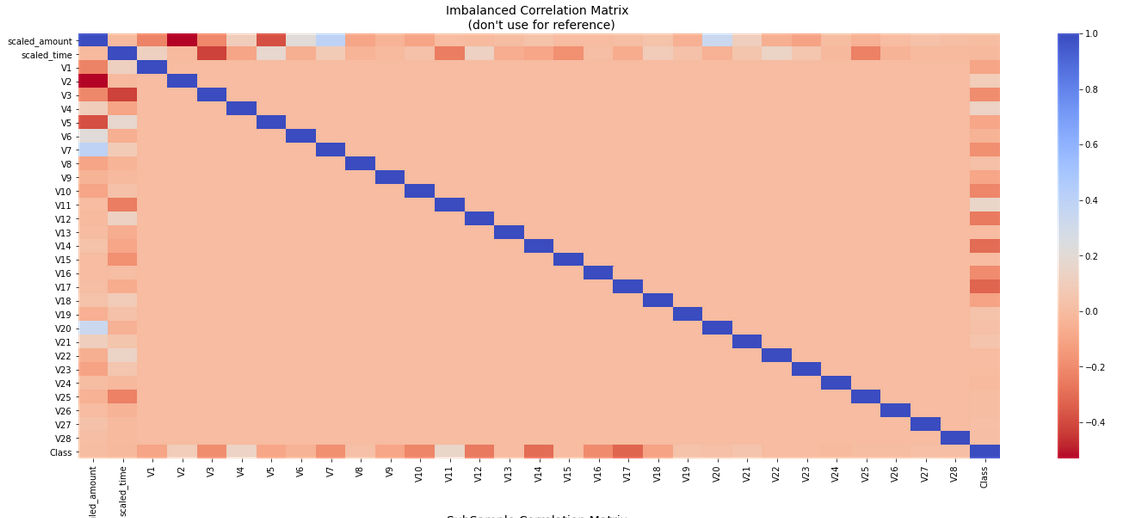
**Correlation Matrices**

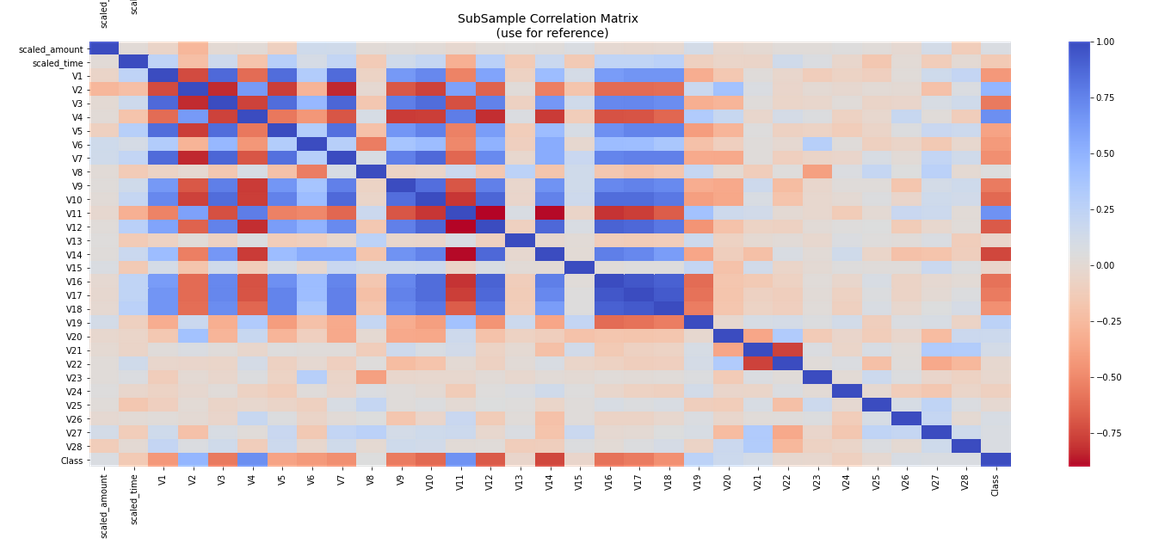
Correlation matrices are the essence of understanding our data. We want to know if there are features that influence heavily in whether a specific transaction is a fraud. However, it is important that we use the correct dataframe (subsample) in order for us to see which features have a high positive or negative correlation with regards to fraud transactions.

Summary and Explanation:

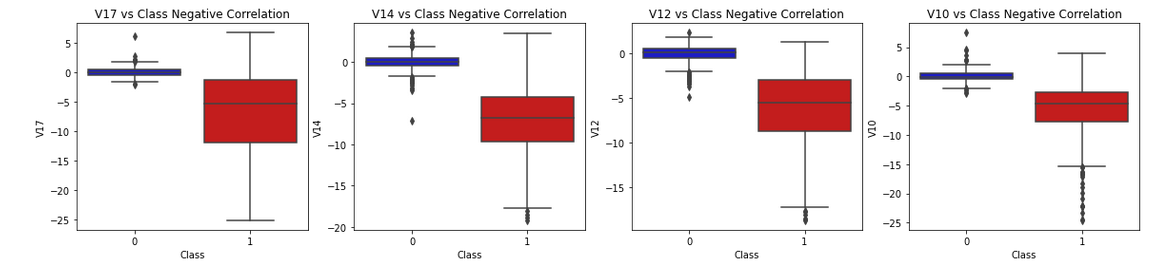
* **Negative Correlations**: V17, V14, V12 and V10 are negatively correlated. Notice how the lower these values are, the more likely the end result will be a fraud transaction.
* **Positive Correlations**: V2, V4, V11, and V19 are positively correlated. Notice how the higher these values are, the more likely the end result will be a fraud transaction.
* **BoxPlots**: We will use boxplots to have a better understanding of the distribution of these features in fraudulent and non fraudulent transactions.

**Note**: We have to make sure we use the subsample in our correlation matrix or else our correlation matrix will be affected by the high imbalance between our classes. This occurs due to the high class imbalance in the original dataframe.

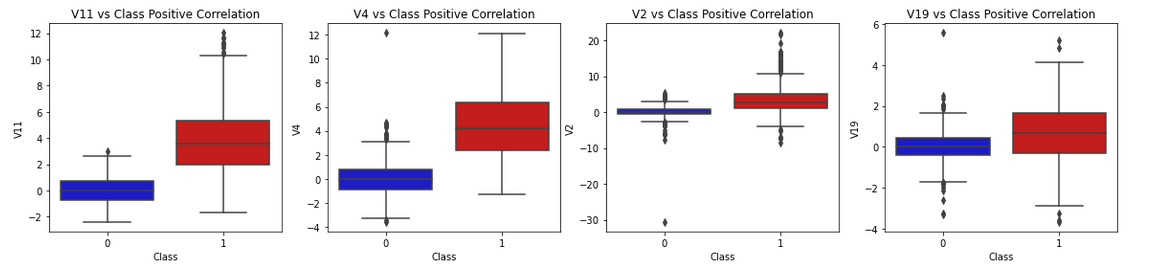




Negative Correlations with our Class (The lower our feature value the more likely it will be a fraud transaction)



Positive correlations (The higher the feature the probability increases that it will be a fraud transaction)



**Anomaly Detection:**

Our main aim in this section is to remove "extreme outliers" from features that have a high correlation with our classes. This will have a positive impact on the accuracy of our models.

**Interquartile Range Method:**

* **Interquartile Range (IQR):** We calculate this by the difference between the 75th percentile and 25th percentile. Our aim is to create a threshold beyond the 75th and 25th percentile that in case some instance passes this threshold the instance will be deleted.
* **Boxplots:** Besides easily seeing the 25th and 75th percentiles (both end of the squares) it is also easy to see extreme outliers (points beyond the lower and higher extreme).

**Outlier Removal Tradeoff:**

We have to be careful as to how far do we want the threshold for removing outliers. We determine the threshold by multiplying a number (ex: 1.5) by the (Interquartile Range). The higher this threshold is, the less outliers will detect (multiplying by a higher number ex: 3), and the lower this threshold is the more outliers it will detect.

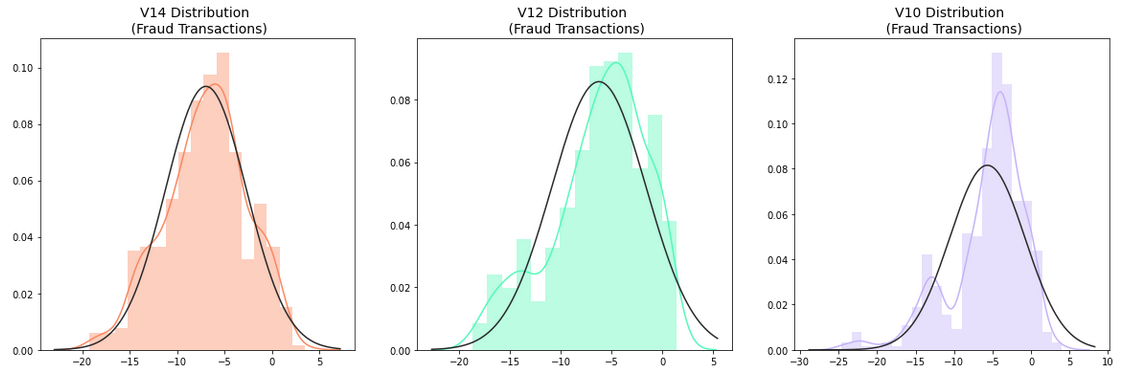
**The Tradeoff:** The lower the threshold the more outliers it will remove however, we want to focus more on "extreme outliers" rather than just outliers. Why? because we might run the risk of information loss which will cause our models to have a lower accuracy. You can play with this threshold and see how it affects the accuracy of our classification models.

**Summary:**

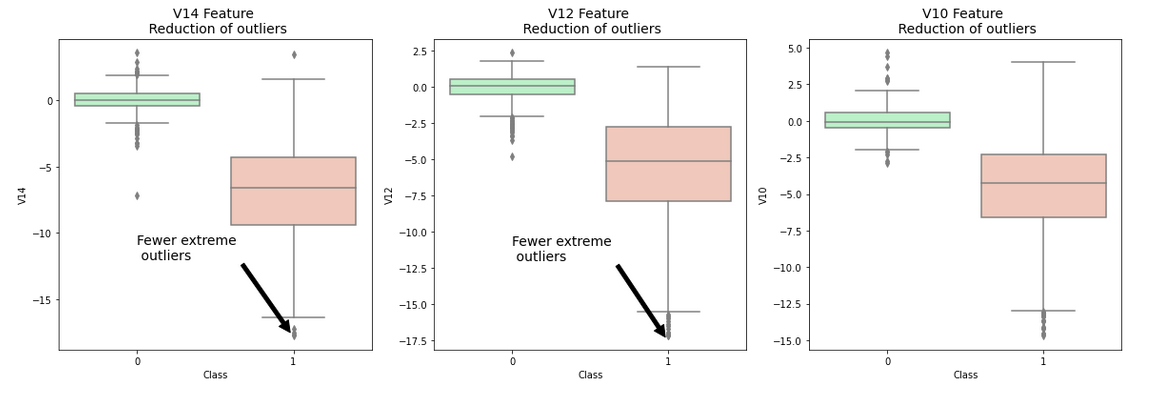
* Visualize Distributions: We first start by visualizing the distribution of the feature we are going to use to eliminate some of the outliers. V14 is the only feature that has a Gaussian distribution compared to features V12 and V10.
* Determining the threshold: After we decide which number, we will use to multiply with the iqr (the lower more outliers removed), we will proceed in determining the upper and lower thresholds by subtracting q25 - threshold (lower extreme threshold) and adding q75 + threshold (upper extreme threshold).
* Conditional Dropping: Lastly, we create a conditional dropping stating that if the "threshold" is exceeded in both extremes, the instances will be removed.
* Boxplot Representation: Visualize through the boxplot that the number of "extreme outliers" have been reduced to a considerable amount.

**Note:** After implementing outlier reduction our accuracy has been improved by over 3%! Some outliers can distort the accuracy of our models but remember, we have to avoid an extreme amount of information loss or else our model runs the risk of underfitting.

**Reference**: More information on Interquartile Range Method: How to Use Statistics to Identify Outliers in Data by Jason Brownless (Machine Learning Mastery blog).



**Boxplots with outliers removed**



**Dimensionality Reduction and Clustering:**

**Understanding t-SNE:**

In order to understand this algorithm, you have to understand the following terms:

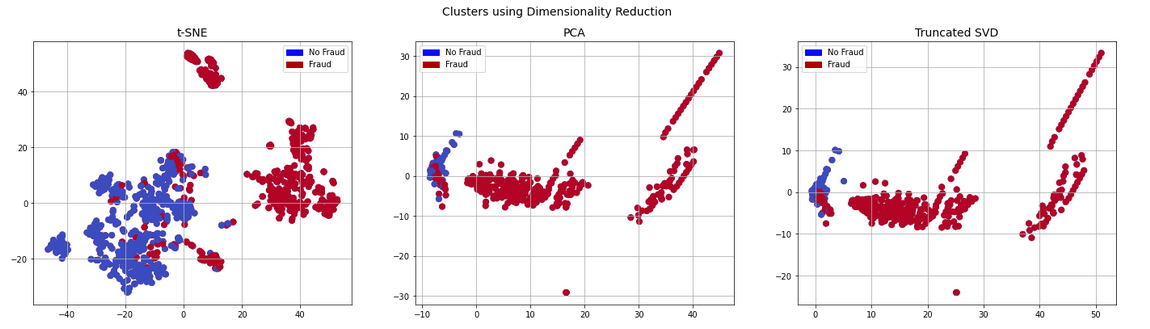
* Euclidean Distance
* Conditional Probability
* Normal and T-Distribution Plots

**Note**: If you want a simple instructive video look at StatQuest: t-SNE, Clearly Explained by Joshua Starmer

**Summary**:

* t-SNE algorithm can pretty accurately cluster the cases that were fraud and non-fraud in our dataset.
* Although the subsample is pretty small, the t-SNE algorithm is able to detect clusters pretty accurately in every scenario (I shuffle the dataset before running t-SNE)
* This gives us an indication that further predictive models will perform pretty well in separating fraud cases from non-fraud cases.

PCA scatter plot and Truncated SVD scatter plot



**Classifiers (UnderSampling):**

In this section we will train four types of classifiers and decide which classifier will be more effective in detecting **fraud transactions**. Before we have to split our data into training and testing sets and separate the features from the labels.

**Summary:**

* **Logistic Regression** classifier is more accurate than the other three classifiers in most cases. (We will further analyze Logistic Regression)
* **GridSearchCV** is used to determine the paremeters that gives the best predictive score for the classifiers.
* Logistic Regression has the best Receiving Operating Characteristic score (ROC), meaning that LogisticRegression pretty accurately separates **fraud** and **non-fraud** transactions.

**Learning Curves:**

* The **wider the gap** between the training score and the cross-validation score, the more likely your model is **overfitting (high variance)**.
* If the score is low in both training and cross-validation sets this is an indication that our model is **underfitting (high bias)**
* **Logistic Regression Classifier** shows the best score in both training and cross-validating sets.

## Feature Engineering Benefits:

Three benefits of performing feature selection before modeling your data are:

* Reduces Overfitting: Less redundant data means less opportunity to make decisions based on noise.
* Improves Accuracy: Less misleading data means modeling accuracy improves.
* Reduces Training Time: Less data means that algorithms train faster.

**SMOTE Technique (Over-Sampling):**

**SMOTE** stands for Synthetic Minority Over-sampling Technique. Unlike Random UnderSampling, SMOTE creates new synthetic points in order to have an equal balance of the classes. This is another alternative for solving the "class imbalance problems".

**Understanding SMOTE:**

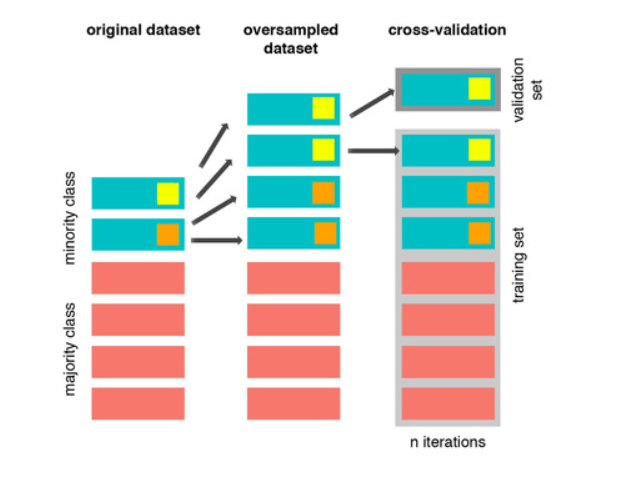
* **Solving the Class Imbalance:** SMOTE creates synthetic points from the minority class in order to reach an equal balance between the minority and majority class.
* **Location of the synthetic points:** SMOTE picks the distance between the closest neighbors of the minority class, in between these distances it creates synthetic points.
* **Final Effect:** More information is retained since we didn't have to delete any rows unlike in random undersampling.
* **Accuracy || Time Tradeoff:** Although it is likely that SMOTE will be more accurate than random under-sampling, it will take more time to train since no rows are eliminated as previously stated.

**Cross** Validation **Overfitting Mistake:**

**Overfitting during Cross Validation:**

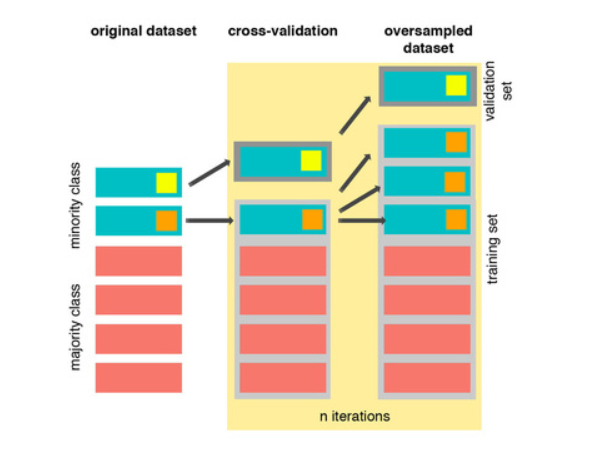
In our undersample analysis I want to show you a common mistake I made that I want to share with all of you. It is simple, if you want to undersample or oversample your data you should not do it before cross validating. Why because you will be directly influencing the validation set before implementing cross-validation causing a "data leakage" problem. **In the following section you will see amazing precision and recall scores but in reality, our data is overfitting!**

**The Wrong Way:**



As mentioned previously, if we get the minority class ("Fraud) in our case, and create the synthetic points before cross validating, we have a certain influence on the "validation set" of the cross-validation process. Remember how cross validation works, let's assume we are splitting the data into 5 batches, 4/5 of the dataset will be the training set while 1/5 will be the validation set. The test set should not be touched! For that reason, we have to do the creation of synthetic datapoints "during" cross-validation and not before, just like below:

**The Right Way:**



As you see above, SMOTE occurs "during" cross validation and not "prior" to the cross-validation process. Synthetic data are created only for the training set without affecting the validation set.

**Test Data with Logistic Regression:**

**Confusion Matrix:**

**Positive/Negative:** Type of Class (label) ["No", "Yes"] **True/False:** Correctly or Incorrectly classified by the model.

**True Negatives (Top-Left Square):** This is the number of **correctly** classifications of the "No" (No Fraud Detected) class.

**False Negatives (Top-Right Square):** This is the number of **incorrectly** classifications of the "No"(No Fraud Detected) class.

**False Positives (Bottom-Left Square):** This is the number of **incorrectly** classifications of the "Yes" (Fraud Detected) class

**True Positives (Bottom-Right Square):** This is the number of **correctly** classifications of the "Yes" (Fraud Detected) class.

