CREDIT CARD FRAUD DETECTION

ANOMALY BASED DETECTION

**Introduction :**

In this kernel we will use various predictive models to see how accurate they are in detecting whether a transaction is a normal payment or a fraud. As described 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.

**Our Goals:**

* Understand the little distribution of the "little" data that was provided to us.
* Create a 50/50 sub-dataframe ratio of "Fraud" and "Non-Fraud" transactions. (NearMiss 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 mistaked made with imbalanced datasets.

### **Summary:**

* The transaction amount is relatively **small**. The mean of all the mounts made is approximately USD 88.
* There are no **"Null"** values, so we don't have to work on ways to replace values.
* Most of the transactions were **Non-Fraud** (99.83%) of the time, while **Fraud** transactions occurs (017%) of the time in the dataframe.

### **Feature Technicalities:**

* **PCA Transformation:**The description of the data says that all the features went through a PCA transformation (Dimensionality Reduction technique) (Except for time and amount).
* **Scaling:** Keep in mind that in order to implement a PCA transformation features need to be previously scaled. (In this case, all the V features have been scaled or at least that is what we are assuming the people that develop the dataset did.)

## Scaling and Distributing

In this phase of our kernel, 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 of this notebook 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 this 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)**[**¶**](https://www.kaggle.com/code/janiobachmann/credit-fraud-dealing-with-imbalanced-datasets#Splitting-the-Data-(Original-DataFrame))

Before proceeding with the Random UnderSampling technique we have to separate the orginal dataframe. Why? for testing purposes, remember although we are splitting the data when implementing Random UnderSampling or OverSampling 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 undersample and oversample (in order for our models to detect the patterns), and test it on the original testing set.

## Random Under-Sampling:[¶](https://www.kaggle.com/code/janiobachmann/credit-fraud-dealing-with-imbalanced-datasets#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.

### **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 fradulent and non fradulent transactions.

## Anomaly Detection:[¶](https://www.kaggle.com/code/janiobachmann/credit-fraud-dealing-with-imbalanced-datasets#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 pass 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 substrating 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.

## 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

### **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.

**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.

## A Deeper Look into LogisticRegression:

In this section we will ive a deeper look into the **logistic regression classifier**.

### **Terms:**

* **True Positives:** Correctly Classified Fraud Transactions
* **False Positives:** Incorrectly Classified Fraud Transactions
* **True Negative:** Correctly Classified Non-Fraud Transactions
* **False Negative:** Incorrectly Classified Non-Fraud Transactions
* **Precision:**True Positives/(True Positives + False Positives)
* **Recall:**True Positives/(True Positives + False Negatives)
* Precision as the name says, says how precise (how sure) is our model in detecting fraud transactions while recall is the amount of fraud cases our model is able to detect.
* **Precision/Recall Tradeoff:**The more precise (selective) our model is, the less cases it will detect. Example: Assuming that our model has a precision of 95%, Let's say there are only 5 fraud cases in which the model is 95% precise or more that these are fraud cases. Then let's say there are 5 more cases that our model considers 90% to be a fraud case, if we lower the precision there are more cases that our model will be able to detect.

### **SMOTE Technique (Over-Sampling):**[**¶**](https://www.kaggle.com/code/janiobachmann/credit-fraud-dealing-with-imbalanced-datasets#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:**

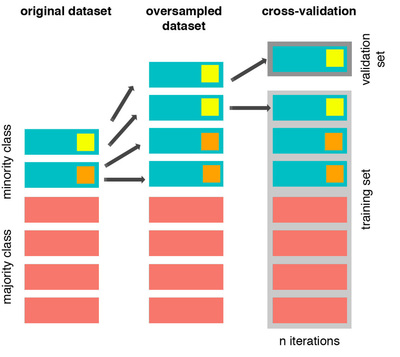
* **Solving the Class Imbalance:**SMOTE creates synthetic points from the minority class in order to reach an equal balance between the minority and majority class.
* **Location of the synthetic points:**SMOTE picks the distance between the closest neighbors of the minority class, in between these distances 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:

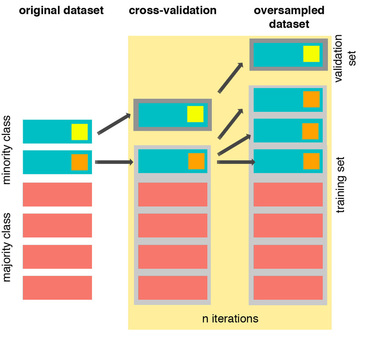
In our undersample analysis I want to show you a common mistake I made that I want to share with all of you. It is simple, if you want to undersample or oversample your data you should not do it before cross validating. Why because you will be directly influencing the validation set before implementing 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:**[**¶**](https://www.kaggle.com/code/janiobachmann/credit-fraud-dealing-with-imbalanced-datasets#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**.

### **Summary:**

* **Random UnderSampling:** We will evaluate the final performance of the classification models in the random undersampling subset. **Keep in mind that this is not the data from the original dataframe.**
* **Classification Models:**The models that performed the best were **logistic regression**and **support vector classifier (SVM)**

## Neural Networks Testing Random UnderSampling Data vs OverSampling (SMOTE):[¶](https://www.kaggle.com/code/janiobachmann/credit-fraud-dealing-with-imbalanced-datasets#Neural-Networks-Testing-Random-UnderSampling-Data-vs-OverSampling-(SMOTE):)

In this section we will implement a simple Neural Network (with one hidden layer) in order to see which of the two logistic regressions models we implemented in the (undersample or oversample(SMOTE)) has a better accuracy for detecting fraud and non-fraud transactions.

### **Our Main Goal:**

Our main goal is to explore how our simple neural network behaves in both the random undersample and oversample dataframes and see whether they can predict accuractely both non-fraud and fraud cases. Why not only focus on fraud? Imagine you were a cardholder and after you purchased an item your card gets blocked because the bank's algorithm thought your purchase was a fraud. That's why we shouldn't emphasize only in detecting fraud cases but we should also emphasize correctly categorizing non-fraud transactions.

### **The Confusion Matrix:**

Here is again, how the confusion matrix works:

* **Upper Left Square:**The amount of **correctly** classified by our model of no fraud transactions.
* **Upper Right Square:** The amount of **incorrectly**classified transactions as fraud cases, but the actual label is **no fraud**.
* **Lower Left Square:** The amount of **incorrectly**classified transactions as no fraud cases, but the actual label is **fraud**.
* **Lower Right Square:** The amount of **correctly** classified by our model of fraud transactions.

### **Summary (Keras || Random UnderSampling):**

* **Dataset:**In this final phase of testing we will fit this model in both the **random undersampled subset** and **oversampled dataset (SMOTE)**in order to predict the final result using the **original dataframe testing data.**
* **Neural Network Structure:**As stated previously, this will be a simple model composed of one input layer (where the number of nodes equals the number of features) plus bias node, one hidden layer with 32 nodes and one output node composed of two possible results 0 or 1 (No fraud or fraud).
* **Other characteristics:** The learning rate will be 0.001, the optimizer we will use is the AdamOptimizer, the activation function that is used in this scenario is "Relu" and for the final outputs we will use sparse categorical cross entropy, which gives the probability whether an instance case is no fraud or fraud (The prediction will pick the highest probability between the two.)

### **Conclusion:**[**¶**](https://www.kaggle.com/code/janiobachmann/credit-fraud-dealing-with-imbalanced-datasets#Conclusion:)

Implementing SMOTE on our imbalanced dataset helped us with the imbalance of our labels (more no fraud than fraud transactions). Nevertheless, I still have to state that sometimes the neural network on the oversampled dataset predicts less correct fraud transactions than our model using the undersample dataset. However, remember that the removal of outliers was implemented only on the random undersample dataset and not on the oversampled one. Also, in our undersample data our model is unable to detect for a large number of cases non fraud transactions correctly and instead, misclassifies those non fraud transactions as fraud cases. Imagine that people that were making regular purchases got their card blocked due to the reason that our model classified that transaction as a fraud transaction, this will be a huge disadvantage for the financial institution. The number of customer complaints and customer disatisfaction will increase. The next step of this analysis will be to do an outlier removal on our oversample dataset and see if our accuracy in the test set improves.