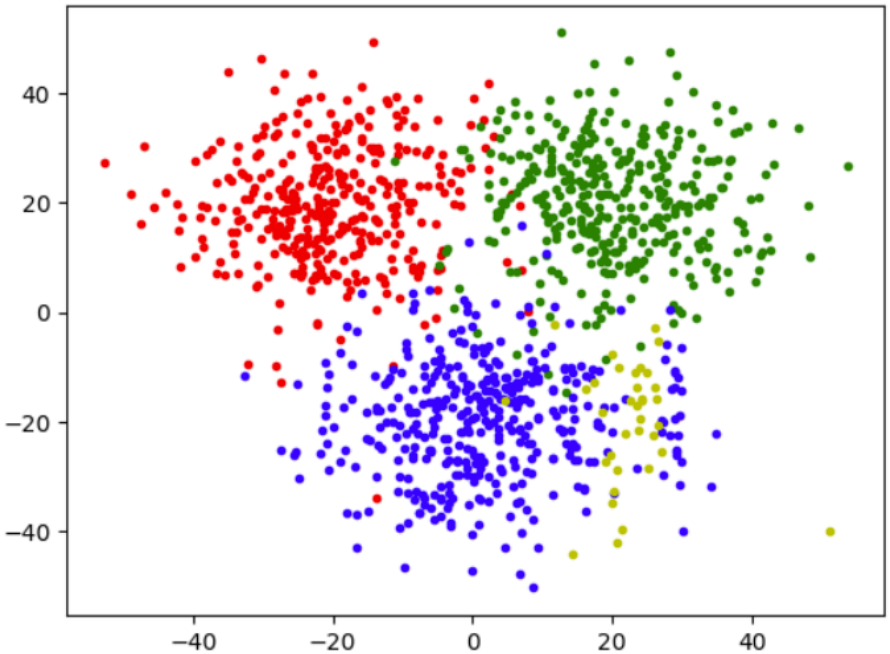
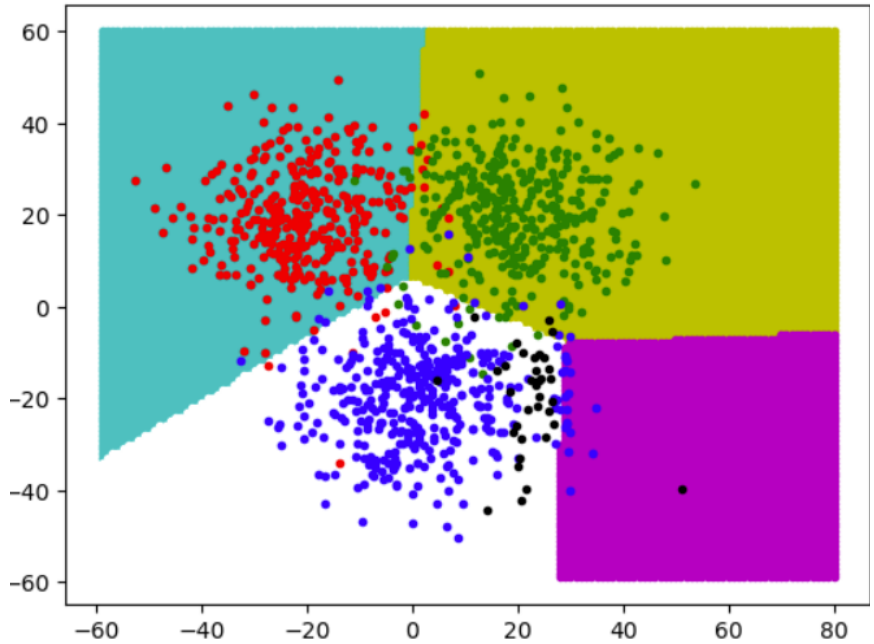
# Data Imbalance




*One of the classes (class 3) has inadequate representation*




*Most instances are wrongly classified*

```
Classification Report:
              precision    recall  f1-score   support

           1       0.96      0.96      0.96       360
           2       0.95      0.95      0.95       360
           3       0.88      0.92      0.90       383
           4       0.06      0.03      0.04        35

Confusion Matrix:
[[345   9   6   0]
 [  7 343  10   0]
 [  8   7 351  17]
 [  0   2  32   1]]
```

# Dealing with Data Imbalances

Data imbalance is a common problem in machine learning, where one class has a significantly higher number of observations than the other. This can lead to biased models and poor performance on the minority class. There are several techniques for dealing with data imbalance, including:

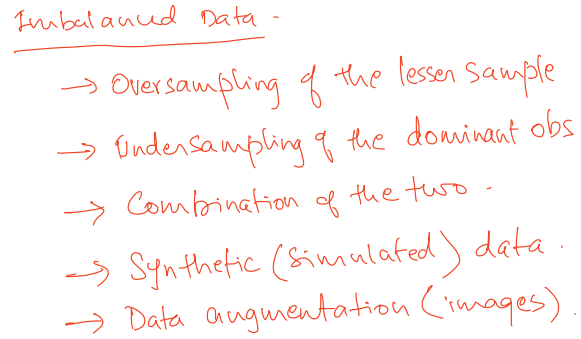
1. Use the right evaluation metrics: Accuracy is not always the best metric to evaluate model performance on imbalanced data. Instead, metrics such as precision, recall, F1-score, and AUC-ROC can provide a better understanding of model performance.

2. Resample the training set: Resampling techniques involve modifying the training set to balance the class distribution. This can be done by either **undersampling** the majority class or **oversampling** the minority class, **SMOTE** (Synthetic Minority Oversampling Technique), or a mix of these strategies.

3. Change the algorithm: Some algorithms are better suited for imbalanced data than others. For example, decision trees and random forests can handle imbalanced data well.

4. Use ensemble methods: Ensemble methods such as bagging, boosting, and stacking can be used to combine multiple models and improve the performance on imbalanced data.

5. Collect more data: Collecting more data can help to balance the class distribution and improve model performance.




The Python package **imblearn** (**imbalanced-learn)** implements these strategies

https://imbalanced-learn.org/stable/

# Data Imbalance

| Metric | Formula | Description |
|---|---|---|
| Accuracy | (TP + TN) / (TP + TN + FP + FN) | Overall correctness |
| Precision | TP / (TP + FP) | Proportion of positive predictions that were correct |
| Recall | TP / (TP + FN) | Proportion of actual positive instances that were correctly identified |
| Sensitivity | TP / (TP + FN) | True Positive Rate |
| Specificity | TN / (TN + FP) | True Negative Rate |
| F1-score | 2 * (Precision * Recall) / (Precision + Recall) | Harmonic mean of precision and recall |

Review the following confusion matrix:

| | 0 | 1 | Support |
|---|---|---|---|
| Actual class 0 | 253 | 5 | 258 |
| Actual class 1 | 5 | 12 | 17 |

CONFUSION MATRIX.

In this case, clearly there is data imbalance since 258 observations belong to Class-0, while only 17 observations belong to Class-1

Observe that even though only 12 out of 17 Class-1 observations were correctly classified (corresponding to = 70.58% accuracy figure for the class, the **overall 'accuracy'** of the classifier is:

(12 + 253) / (12 + 253 + 5 + 5) = 265 / 275 = **96.36%**

The metrics of the classifier are as follows:

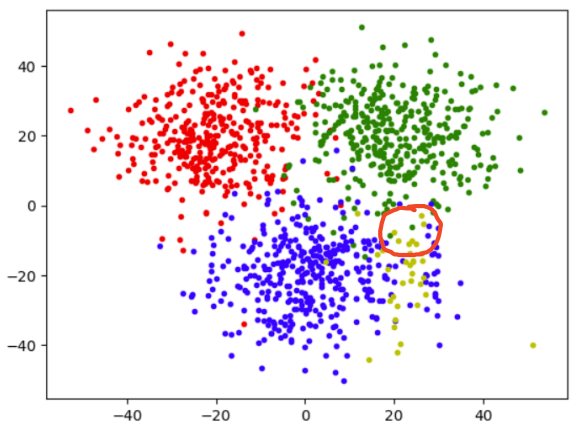
- **Accuracy: 96.36%**
- Sensitivity (Recall): 70.59%
- Specificity: 98.06%
- Precision: 70.59%
- **F1-Score: 70.59%**

IMBALANCED DATA:
"The dominant class will hijack the model"
→ Accuracy values will be (falsely) high.

**Handling Class Imbalances**

- Over-sampling of minority class
- Under-sampling of majority class
- Combination

The scikit-learn module **imblearn** implements functions to tackle data-imbalance problems.




**ENN (Noise Reduction)**

ENN is often used as a preprocessing step **in combination with other techniques** such as over-sampling (e.g., SMOTE) or under-sampling to address the imbalance issue while simultaneously improving the quality of the data.

Cleaning using ENN refers to the use of Edited Nearest Neighbors (ENN) as a method for cleaning or refining imbalanced datasets. ENN is a technique commonly employed in combination with over-sampling or under-sampling methods to improve the quality of the dataset, particularly in the context of classification tasks. ENN works as follows:

1. **Selecting a data point**: Iterate through each instance in the dataset.
2. **Identifying nearest neighbors**: Determine the k-nearest neighbors of the selected instance using a distance metric (e.g., Euclidean distance).
3. **Majority voting**: Check if the majority class label among the nearest neighbors differs from the label of the selected instance. If it does, consider the instance for removal.
4. **Removing inconsistent instances**: Remove the instances for which the majority class label among its nearest neighbors is different from its own label.

By removing instances that are misclassified by their nearest neighbors, ENN aims to improve the quality of the dataset by eliminating noisy or borderline instances. This can lead to a more robust and reliable training dataset, which in turn can improve the performance of classifiers, especially in scenarios where class imbalance is a concern.

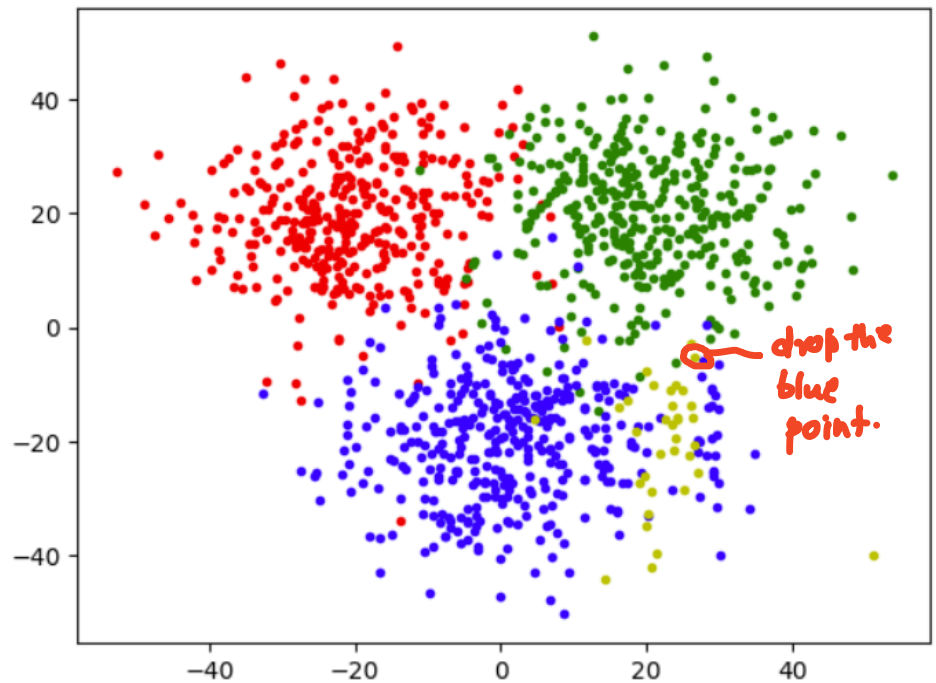
**Tomek Links (Undersampling)**

Tomek links are pairs of instances in a dataset that belong to different classes and are close to each other, meaning they are nearest neighbors.

Specifically, a Tomek link exists between two instances if there are no other instances closer to each other than these two instances are to each other.

Tomek links are often used as a technique for cleaning or refining the dataset. The idea is to remove the majority class instance of each Tomek link, which can help improve the decision boundary of a classifier, especially in scenarios where there is significant overlap between classes or where the majority class instances dominate the decision-making process.

Removing the majority class instances that form Tomek links can help in addressing the issue of imbalanced classes by potentially reducing the dominance of the majority class and improving the overall performance of the classifier, particularly in terms of metrics like precision, recall, and F1-score.



**SMOTE**

SMOTE stands for Synthetic Minority Over-sampling Technique. It's a popular method used in the context of handling imbalanced datasets, particularly in machine learning classification tasks.

The SMOTE method works by generating synthetic samples from the minority class to alleviate the class imbalance problem. It does this by interpolating between existing minority class instances. Here's how it typically works:

1. **Selecting a minority class instance**: Randomly choose an instance from the minority class.
2. **Finding its nearest neighbors**: Find the k-nearest neighbors, **belonging to the same minority class**, for this instance (typically using Euclidean distance or other distance metrics).
3. **Creating synthetic samples**: Randomly select one of the nearest neighbors and use it to create a new synthetic instance. This synthetic instance is created by choosing a point along the line segment joining the selected minority class instance and its nearest neighbor.
4. **Repeat**: Repeat steps 1-3 until the desired balance between classes is achieved.

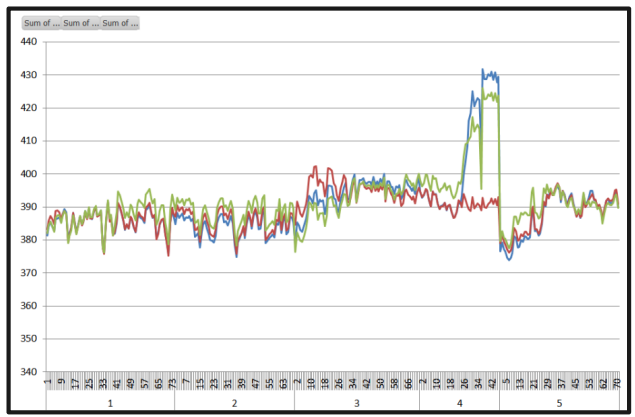
By generating synthetic samples, SMOTE effectively increases the representation of the minority class in the dataset without simply duplicating existing instances, thus helping to address the class imbalance problem. This can lead to better generalization and performance of classifiers trained on imbalanced datasets, especially when combined with other techniques like Tomek links or under-sampling of the majority class.

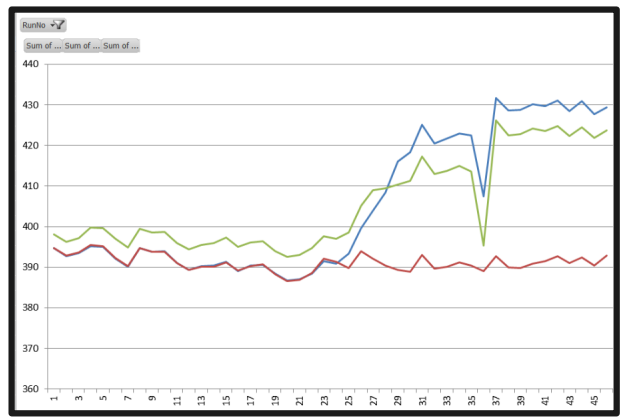
SMOTE & other methods in 'imblearn'

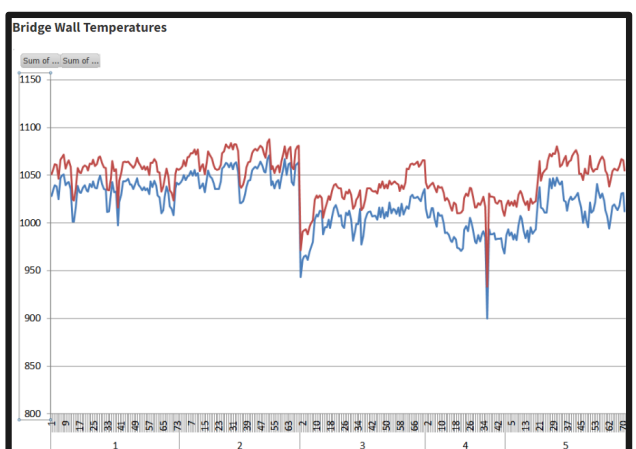
# EDA / Data Preparation : Features (ie. Columns)

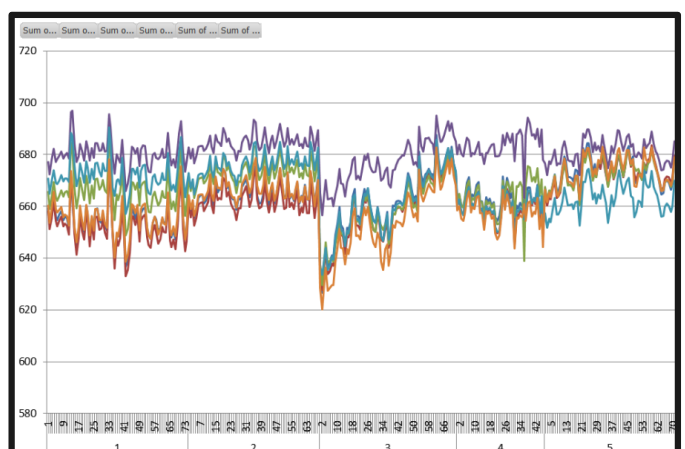
- Number of features and their impact on models: Reduce the number of features to improve model quality
- Identification of interdependent features
    - Feature Correlation : Heat Maps
    - Multicollinearity detection : VIF - Variance Inflation
    - Feature encoding : Categorical Features
    - Feature Engineering
    - Feature Reduction
        - PCA : Principal Component Analysis
        - t-SNE : t-distributed Stochastic Neighbour Embedding

# Pair-wise Correlation of Features



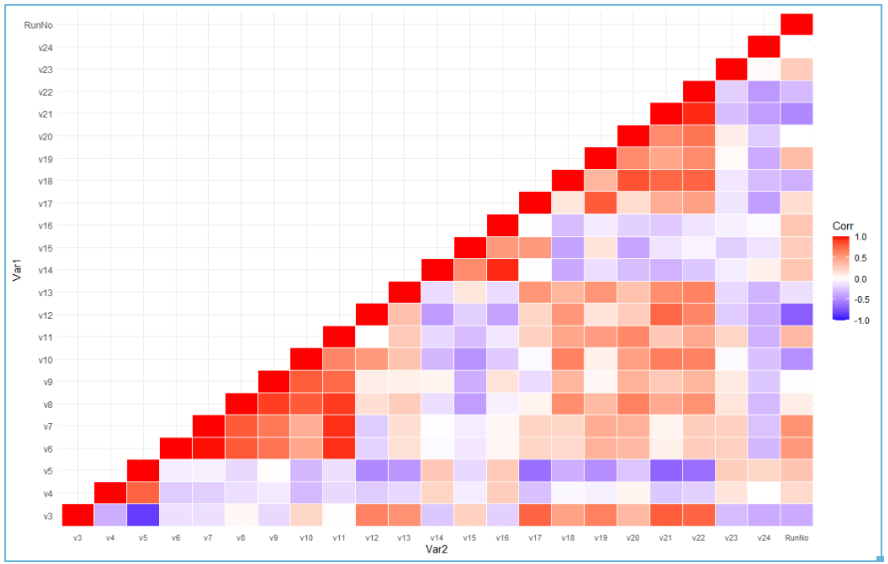



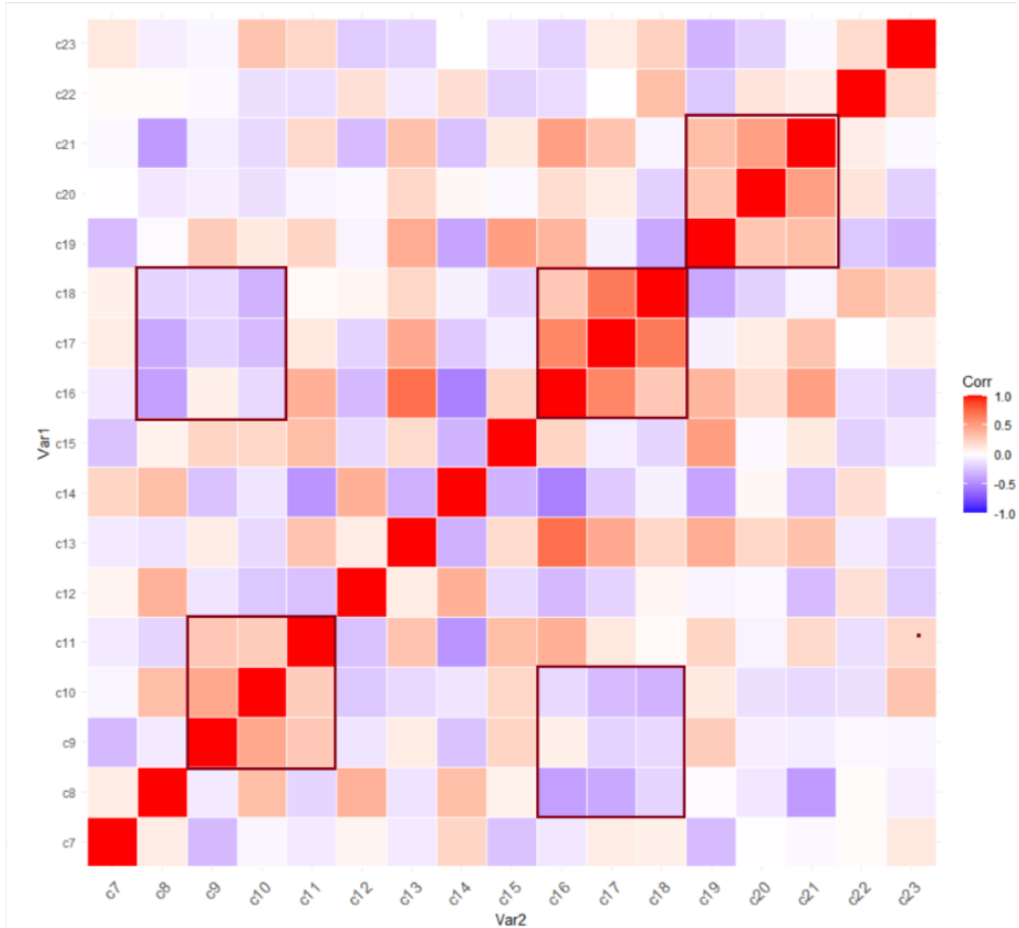




This method of pair-wise or group-wise plotting of features are convenient to detect feature dependencies if there are only a handful of features.

For a very large feature set, correlation **heat maps** are used. They can be automatically, and exhaustively created, and they provide a consolidated view of **feature correlations**.

# Correlation Heat Maps



# Multicollinearity Detection
## Variance Inflation Factor (**VIF**)

The Variance Inflation Factor (VIF) quantifies how much the variance of a regression coefficient is inflated due to multicollinearity in the model. Mathematically, the VIF for a particular feature is calculated by performing a linear regression of that feature on all the other features.

For a given feature $X_i$, the VIF is defined as:

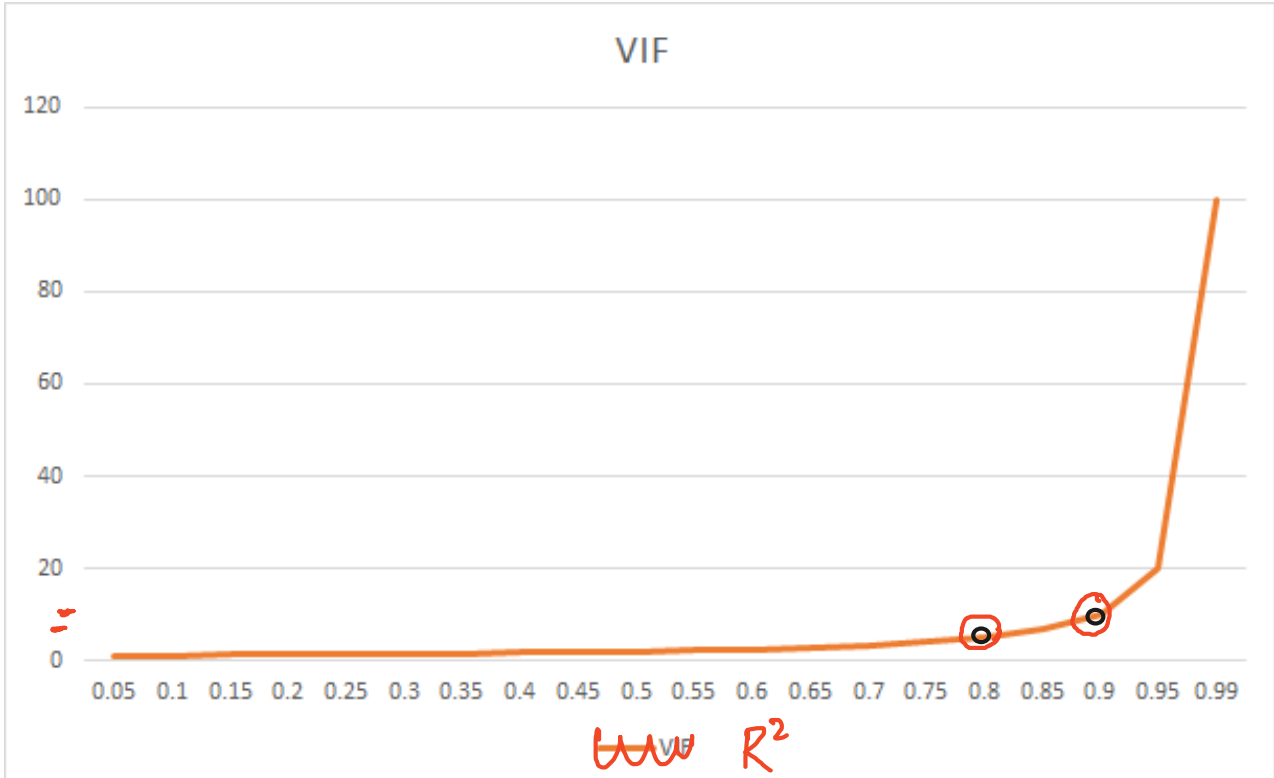
$$\text{VIF}(X_i) = \frac{1}{1-R_i^2}$$

Where:

- $R_i^2$ is the $R^2$ value obtained by regressing the feature $X_i$ against all other features.

1. For each feature $X_i$ in your dataset, run a linear regression using $X_i$ as the dependent variable and all other features as independent variables.
2. Obtain the $R^2$ value from this regression, denoted as $R_i^2$.
3. Calculate the VIF for $X_i$ using the formula above.

A VIF of 1 indicates that there's no multicollinearity, whereas a VIF above 1 suggests that the feature is correlated with other features. A common rule of thumb is that a VIF above 10 indicates high multicollinearity, but this threshold can vary depending on the context.




VIF v/s R-square

| R-square | VIF |
|---|---|
| 0.05 | 1.052631579 |
| 0.1 | 1.111111111 |
| 0.15 | 1.176470588 |
| 0.2 | 1.25 |
| 0.25 | 1.333333333 |
| 0.3 | 1.428571429 |
| 0.35 | 1.538461538 |
| 0.4 | 1.666666667 |
| 0.45 | 1.818181818 |
| 0.5 | 2 |
| 0.55 | 2.222222222 |
| 0.6 | 2.5 |
| 0.65 | 2.857142857 |
| 0.7 | 3.333333333 |
| 0.75 | 4 |
| 0.8 | 5 |
| 0.85 | 6.666666667 |
| 0.9 | 10 |
| 0.95 | 20 |
| 0.99 | 100 |

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | c1 | c2 | c26 | c27 | c28 | c29 | c30 | c31 | c32 |
| 2 | 43344 | 2 | 493.7968 | 104.5539 | 41.1876 | 290.9653 | 14.37955 | 71.73199 | 48.67901 |
| 3 | 43345 | 2 | 493.6619 | 104.5132 | 41.58075 | 290.6212 | 14.31532 | 78.59982 | 48.05742 |
| 4 | 43346 | 2 | 495.6449 | 104.5025 | 40.74457 | 292.1524 | 14.56618 | 78.83246 | 47.32059 |
| 5 | 43347 | 2 | 494.354 | 104.4529 | 40.28818 | 292.6762 | 14.60518 | 72.73663 | 47.98046 |
| 6 | 43348 | 2 | 492.0514 | 104.4886 | 41.26669 | 289.0175 | 14.54893 | 76.62107 | 48.2173 |
| 7 | 43349 | 2 | 491.2304 | 104.5097 | 42.63592 | 286.8439 | 14.31609 | 76.83934 | 48.43282 |
| 8 | 43350 | 2 | 493.0255 | 104.4522 | 41.96914 | 289.3589 | 14.47512 | 77.22903 | 48.33145 |
| 9 | 43351 | 2 | 492.9835 | 104.479 | 42.39755 | 290.8424 | 14.18896 | 73.14102 | 47.37012 |
| 10 | 43352 | 2 | 492.5621 | 104.531 | 43.21701 | 290.1102 | 14.11881 | 72.42022 | 47.36809 |
| 11 | 43353 | 2 | 493.8725 | 104.5807 | 43.00113 | 291.9806 | 14.21165 | 73.24146 | 47.38106 |
| 12 | 43354 | 2 | 490.2188 | 104.604 | 42.60375 | 287.1388 | 14.25636 | 73.24146 | 48.72817 |
| 13 | 43355 | 2 | 489.2321 | 104.472 | 42.89505 | 284.6288 | 14.26377 | 74.96662 | 49.60527 |
| 14 | 43356 | 2 | 491.5731 | 104.4412 | 44.05164 | 286.618 | 14.26902 | 76.23442 | 50.06172 |
| 15 | 43357 | 2 | 492.4099 | 104.4934 | 44.50718 | 287.3763 | 14.27284 | 77.44611 | 50.30747 |
| 16 | 43358 | 2 | 490.4405 | 104.4737 | 43.78121 | 285.1875 | 14.26373 | 80.78736 | 50.59053 |
| 17 | 43359 | 2 | 490.9521 | 104.5132 | 43.82019 | 285.7735 | 14.26143 | 79.81733 | 50.24661 |
| 18 | 43360 | 2 | 491.9521 | 104.5001 | 43.59271 | 286.4335 | 14.27795 | 81.64099 | 50.50682 |
| 19 | 43361 | 2 | 492.0228 | 104.5208 | 43.05299 | 287.2706 | 14.36223 | 81.6224 | 50.40502 |
| 20 | 43362 | 2 | 491.425 | 104.5444 | 43.69969 | 285.4068 | 14.16046 | 81.08435 | 50.34545 |
| 21 | 43363 | 2 | 491.4424 | 104.4848 | 43.57138 | 285.7367 | 14.1837 | 80.95044 | 50.32148 |
| 22 | 43364 | 2 | 492.6671 | 104.4642 | 42.89322 | 287.0949 | 14.25549 | 82.38933 | 50.27493 |

**How to 'drop' features based on VIF analysis?**

| Features | VIF |
|---|---|
| const | 1.031338e+07 |
| c26 | 1.328880e+01 |
| c28 | 3.225659e+01 |
| c29 | 1.157359e+01 |
| c31 | 1.785831e+01 |
| c32 | 8.938850e+01 |
| c33 | 9.569349e+01 |
| c139 | 1.186148e+02 |
| c142 | 3.273434e+01 |
| c143 | 2.579679e+01 |
| c155 | 1.036910e+01 |
| c16 | 1.055594e+01 |
| c19 | 1.122551e+02 |

| Features | VIF |
|---|---|
| c27 | 1.060532 |
| c30 | 4.330190 |
| c39 | 4.803262 |
| c157 | 6.593328 |
| c158 | 4.129364 |
| c160 | 1.843026 |
| c161 | 3.467590 |
| c162 | 3.160080 |
| c163 | 1.878506 |
| c7 | 2.774363 |
| c8 | 5.818705 |
| c9 | 6.480519 |
| c10 | 6.367653 |
| c11 | 2.923712 |
| c12 | 3.019417 |
| c13 | 6.077563 |
| c15 | 4.695586 |
| c17 | 3.781340 |
| c20 | 4.543217 |
| c21 | 3.737963 |
| c22 | 2.498634 |
| c23 | 7.632798 |
| c34 | 2.127750 |
| c35 | 3.572582 |
| c36 | 1.075612 |

## How to 'drop' features based on VIF analysis?

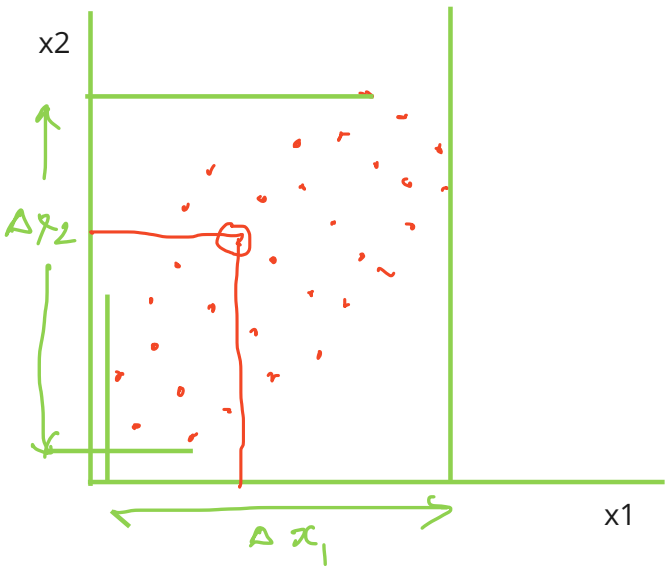
The following strategy can be used to remove features:

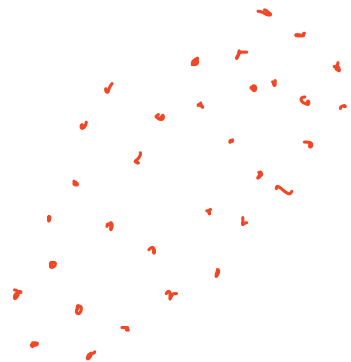
1. Identify the features with high VIF values: The first step is to identify the features with high VIF values, typically above 5 or 10, which indicate that the feature is highly correlated with other features in the model.
2. Evaluate the importance of the features: The next step is to evaluate the importance of the features based on their relevance to the research question, their interpretability, and their contribution to the model's performance.
3. Remove the least important features: Based on the evaluation, the least important features can be removed from the model to reduce multicollinearity and improve the model's performance.
4. Re-evaluate the model: After removing the features, it is important to re-evaluate the model's performance using appropriate metrics and cross-validation techniques to ensure that the model is still accurate and reliable.

### CAVEAT

It is important to note that **removing features can result in loss of information and may affect the interpretability of the model**. Therefore, it is important to carefully consider the trade-offs between multicollinearity and model performance and to choose the appropriate strategy based on the research question, the data, and the assumptions of the model

# PCA - Principal Component Analysis

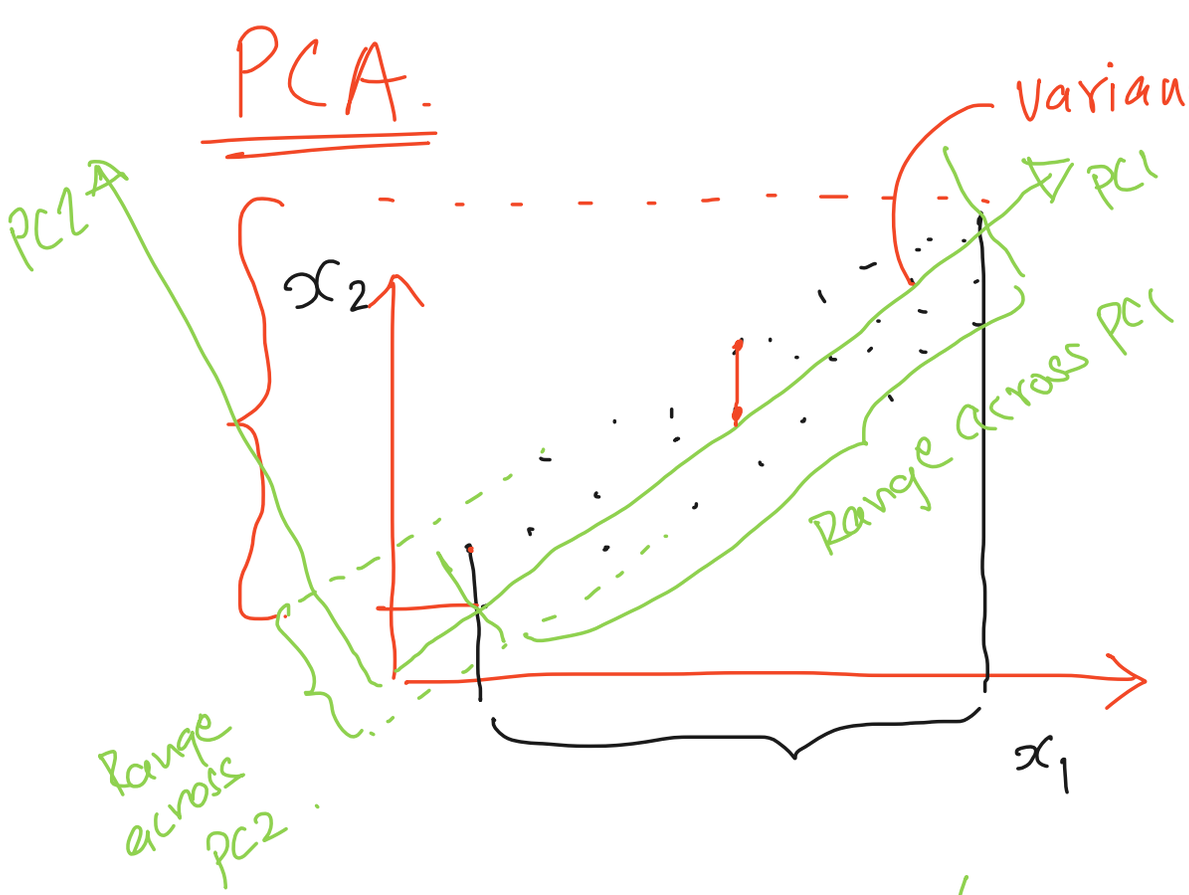


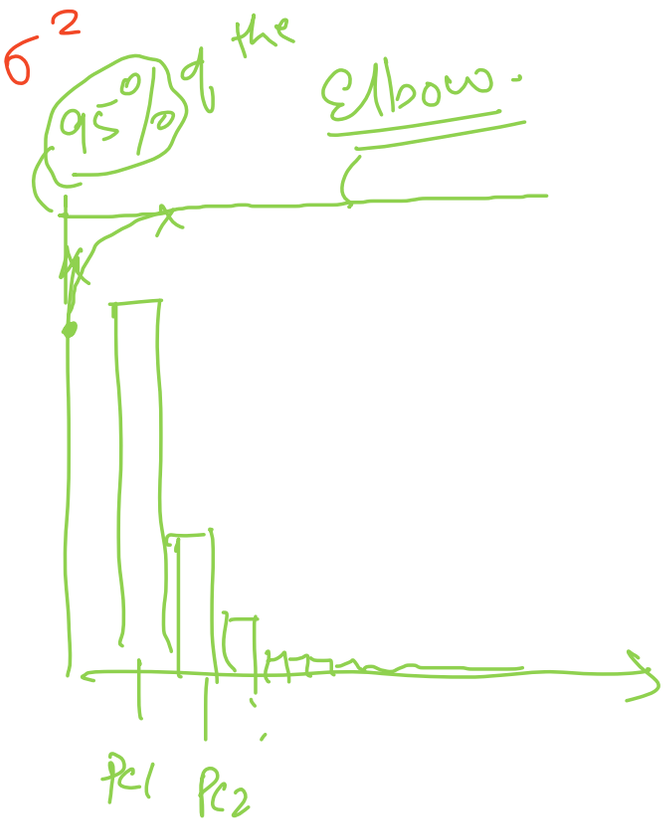



Principal Components are components along re-aligned axes such that PC1 'captures' or 'explains' or 'accounts for' the maximum variance, followed by PC2, PC3, and so on.

If the first few PCs cumulatively explain more than, say, 90% of the data variance, they may be sufficient for creating effective ML models. PCA thus serves the purpose of 'feature reduction'.

# PCA.




$$Y = f'(PC1, PC2, PC3, \cdots PC_K)$$

Downside → "NO EXPLAINABILITY"

PC = linear Combination of $X_i$

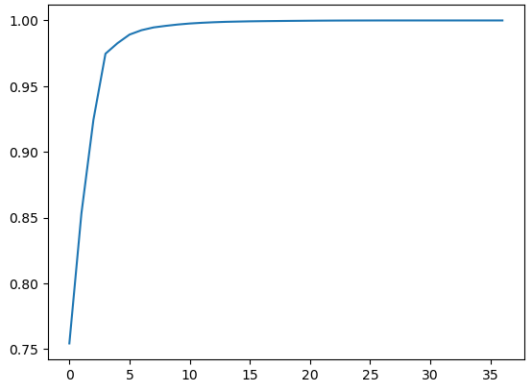
$$PC_1 = a_1 x_1 + b_1 x_2 + c_1 x_3 + c_1 x_4 + \cdots$$

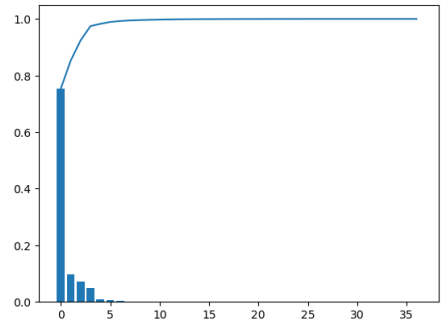
This data has more than 40 columns (features)

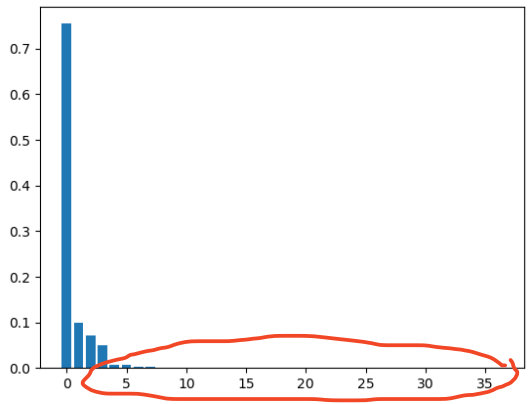
| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | c1 | c2 | c26 | c27 | c28 | c29 | c30 | c31 | c32 |
| 2 | 43344 | 2 | 493.7968 | 104.5539 | 41.1876 | 290.9653 | 14.37955 | 71.73199 | 48.67901 |
| 3 | 43345 | 2 | 493.6619 | 104.5132 | 41.58075 | 290.6212 | 14.31532 | 78.59982 | 48.05742 |
| 4 | 43346 | 2 | 495.6449 | 104.5025 | 40.74457 | 292.1524 | 14.56618 | 78.83246 | 47.32059 |
| 5 | 43347 | 2 | 494.354 | 104.4529 | 40.28818 | 292.6762 | 14.60518 | 72.73663 | 47.98046 |
| 6 | 43348 | 2 | 492.0514 | 104.4886 | 41.26669 | 289.0175 | 14.54893 | 76.62107 | 48.2173 |
| 7 | 43349 | 2 | 491.2304 | 104.5097 | 42.63592 | 286.8439 | 14.31609 | 76.83934 | 48.43282 |
| 8 | 43350 | 2 | 493.0255 | 104.4522 | 41.96914 | 289.3589 | 14.47512 | 77.22903 | 48.33145 |
| 9 | 43351 | 2 | 492.9835 | 104.479 | 42.39755 | 290.8424 | 14.18896 | 73.14102 | 47.37012 |
| 10 | 43352 | 2 | 492.5621 | 104.531 | 43.21701 | 290.1102 | 14.11881 | 72.42022 | 47.36809 |
| 11 | 43353 | 2 | 493.8725 | 104.5807 | 43.00113 | 291.9806 | 14.21165 | 73.24146 | 47.38106 |
| 12 | 43354 | 2 | 490.2188 | 104.604 | 42.60375 | 287.1388 | 14.25636 | 73.24146 | 48.72817 |
| 13 | 43355 | 2 | 489.2321 | 104.472 | 42.89505 | 284.6288 | 14.26377 | 74.96662 | 49.60527 |
| 14 | 43356 | 2 | 491.5731 | 104.4412 | 44.05164 | 286.618 | 14.26902 | 76.23442 | 50.06172 |
| 15 | 43357 | 2 | 492.4099 | 104.4934 | 44.50718 | 287.3763 | 14.27284 | 77.44611 | 50.30747 |
| 16 | 43358 | 2 | 490.4405 | 104.4737 | 43.78121 | 285.1875 | 14.26373 | 80.78736 | 50.59053 |
| 17 | 43359 | 2 | 490.9521 | 104.5132 | 43.82019 | 285.7735 | 14.26143 | 79.81733 | 50.24661 |
| 18 | 43360 | 2 | 491.9521 | 104.5001 | 43.59271 | 286.4335 | 14.27795 | 81.64099 | 50.50682 |
| 19 | 43361 | 2 | 492.0228 | 104.5208 | 43.05299 | 287.2706 | 14.36223 | 81.6224 | 50.40502 |
| 20 | 43362 | 2 | 491.425 | 104.5444 | 43.69969 | 285.4068 | 14.16046 | 81.08435 | 50.34545 |
| 21 | 43363 | 2 | 491.4424 | 104.4848 | 43.57138 | 285.7367 | 14.1837 | 80.95044 | 50.32148 |
| 22 | 43364 | 2 | 492.6671 | 104.4642 | 42.89322 | 287.0949 | 14.25549 | 82.38933 | 50.27493 |

```
[7.54261365e-01 9.87869904e-02 7.12964455e-02 5.03510215e-02
 7.98749378e-03 6.55441064e-03 3.36504308e-03 2.09332441e-03
 1.18752903e-03 1.01662107e-03 7.74969179e-04 5.43124838e-04
 4.13982528e-04 3.22268906e-04 1.99904846e-04 1.82358764e-04
 1.22357384e-04 9.84701179e-05 8.87915581e-05 8.02456161e-05
 7.55522105e-05 6.24074825e-05 5.04485632e-05 3.08481492e-05
 1.86827712e-05 1.49416419e-05 1.33783774e-05 2.66135642e-06
 2.02403308e-06 9.51393151e-07 7.49623526e-07 2.39654410e-07
 1.99355029e-07 8.26718788e-08 7.06173620e-08 4.41103905e-08
 2.33592435e-11]
```

PCA reveals that only about 6 PCs explain more than 90% variance in the data, and only the first 6 PCs need be used to create ML models



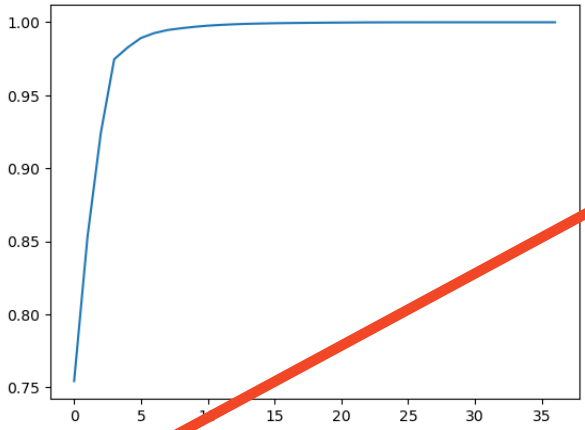



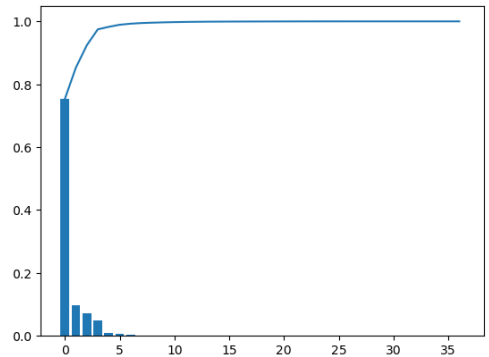

However, there is a problem with these results … see next slide.

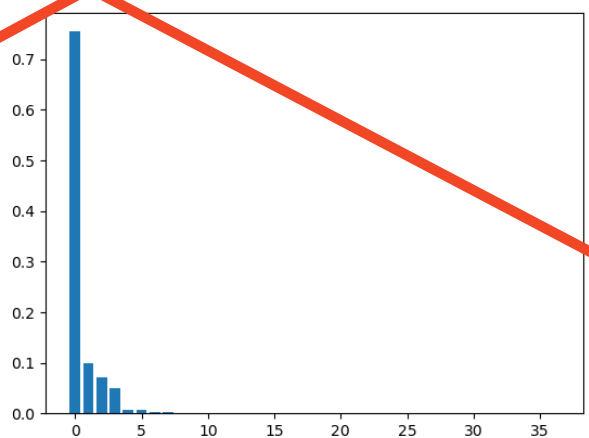
| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | c1 | c2 | c26 | c27 | c28 | c29 | c30 | c31 | c32 |
| 2 | 43344 | 2 | 493.7968 | 104.5539 | 41.1876 | 290.9653 | 14.37955 | 71.73199 | 48.67901 |
| 3 | 43345 | 2 | 493.6619 | 104.5132 | 41.58075 | 290.6212 | 14.31532 | 78.59982 | 48.05742 |
| 4 | 43346 | 2 | [illegible] | 104.5025 | 40.74457 | 292.1524 | 14.56618 | 78.83246 | 47.32059 |
| 5 | 43347 | 2 | [illegible] | 104.4529 | 40.28818 | 292.6762 | 14.60518 | 72.73663 | 47.98046 |
| 6 | 43348 | 2 | 492.0514 | [illegible] | 41.26669 | 289.0175 | 14.54893 | 76.62107 | 48.2173 |
| 7 | 43349 | 2 | 491.2304 | [illegible] | 42.63592 | 286.8439 | 14.31609 | 76.83934 | 48.43282 |
| 8 | 43350 | 2 | 493.0255 | 104.4522 | [illegible] | 289.3589 | 14.47512 | 77.22903 | 48.33145 |
| 9 | 43351 | 2 | 492.9835 | 104.479 | 42.39755 | [illegible] | 14.18896 | 73.14102 | 47.37012 |
| 10 | 43352 | 2 | 492.5621 | 104.531 | 43.21701 | [illegible] | 14.11881 | 72.42022 | 47.36809 |
| 11 | 43353 | 2 | 493.8725 | 104.5807 | 43.00113 | 291.9806 | [illegible] | 73.24146 | 47.38106 |
| 12 | 43354 | 2 | 490.2188 | 104.604 | 42.60375 | 287.1388 | [illegible] | 73.24146 | 48.72817 |
| 13 | 43355 | 2 | 489.2321 | 104.472 | 42.89505 | 284.6288 | 14.26377 | [illegible] | 49.60527 |
| 14 | 43356 | 2 | 491.5731 | 104.4412 | 44.05164 | 286.618 | 14.26902 | 76.23442 | 50.06172 |
| 15 | 43357 | 2 | 492.4099 | 104.4934 | 44.50718 | 287.3763 | 14.27284 | 77.44611 | [illegible] |
| 16 | 43358 | 2 | 490.4405 | 104.4737 | 43.78121 | 285.1875 | 14.26373 | 80.78736 | 50.59053 |
| 17 | 43359 | 2 | 490.9521 | 104.5132 | 43.82019 | 285.7735 | 14.26143 | 79.81733 | 50.24661 |
| 18 | 43360 | 2 | 491.9521 | 104.5001 | 43.59271 | 286.4335 | 14.27795 | 81.64099 | 50.50682 |
| 19 | 43361 | 2 | 492.0228 | 104.5208 | 43.05299 | 287.2706 | 14.36223 | 81.6224 | 50.40502 |
| 20 | 43362 | 2 | 491.425 | 104.5444 | 43.69969 | 285.4068 | 14.16046 | 81.08435 | 50.34545 |
| 21 | 43363 | 2 | 491.4424 | 104.4848 | 43.57138 | 285.7367 | 14.1837 | 80.95044 | 50.32148 |
| 22 | 43364 | 2 | 492.6671 | 104.4642 | 42.89322 | 287.0949 | 14.25549 | 82.38933 | 50.27493 |

```
[7.54261365e-01 9.87869904e-02 7.12964455e-02 5.03510215e-02
 7.98749378e-03 6.55441064e-03 3.36504308e-03 2.09332441e-03
 1.18752903e-03 1.01662107e-03 7.74969179e-04 5.43124838e-04
 4.13982528e-04 3.22268906e-04 1.99904846e-04 1.82358764e-04
 1.22357384e-04 9.84701179e-05 8.87915581e-05 8.02456161e-05
 7.55522105e-05 6.24074825e-05 5.04485632e-05 3.08481492e-05
 1.86827712e-05 1.49416419e-05 1.33783774e-05 2.66135642e-06
 2.02403308e-06 9.51393151e-07 7.49623526e-07 2.39654410e-07
 1.99356029e-07 8.26718788e-08 7.06173620e-08 4.41103905e-08
 2.33592435e-11]
```

Problem: One feature seems to be very 'dominant'. This is happening because of incompatible scales of the features



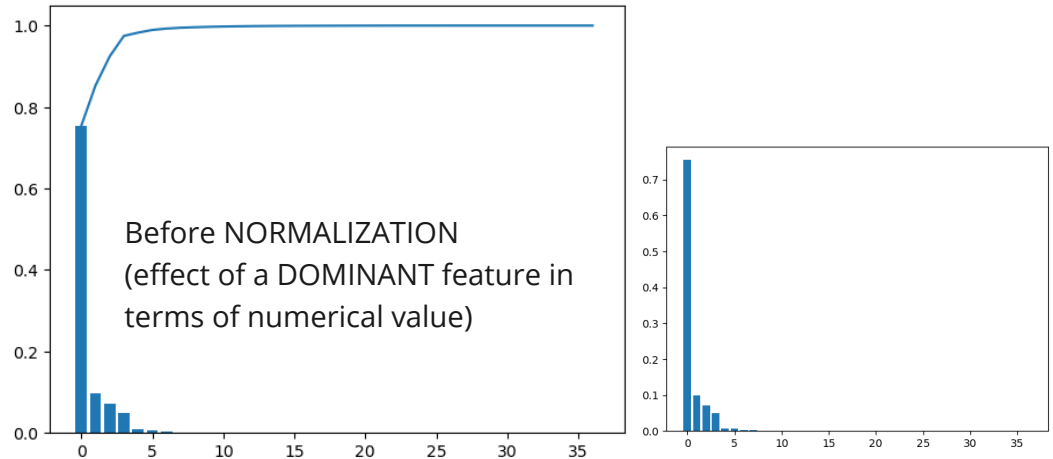



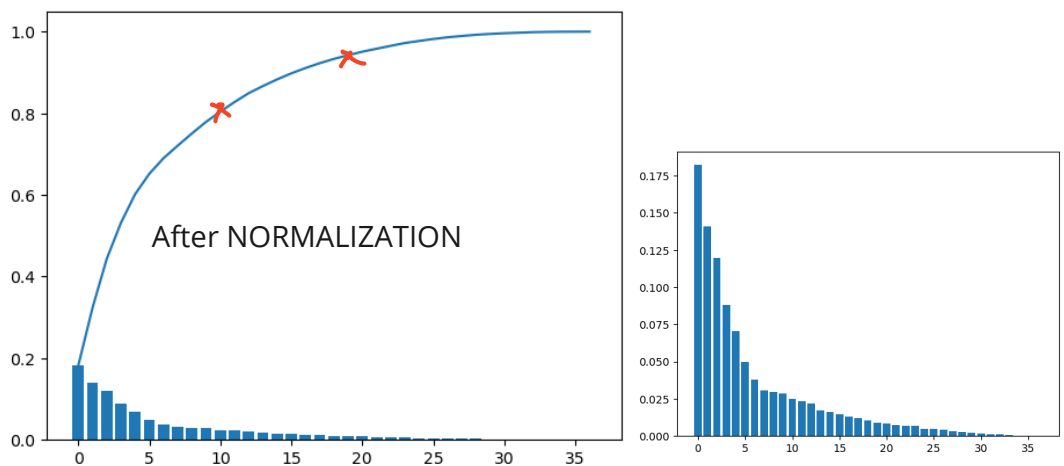
# Importance of NORMALIZATION

Note : Data needs to be normalized before performing PCA

```
[7.54261365e-01 9.87869904e-02 7.12964455e-02 5.03510215e-02
 7.98749378e-03 6.55441064e-03 3.36504308e-03 2.09332441e-03
 1.18752903e-03 1.01662107e-03 7.74969179e-04 5.43124838e-04
 4.13982528e-04 3.22268906e-04 1.99904846e-04 1.82358764e-04
 1.22357384e-04 9.84701179e-05 8.87915581e-05 8.02456161e-05
 7.55522105e-05 6.24074825e-05 5.04485632e-05 3.08481492e-05
 1.86827712e-05 1.49416419e-05 1.33783774e-05 2.66135642e-06
 2.02403308e-06 9.51393151e-07 7.49623526e-07 2.39654410e-07
 1.99355029e-07 8.26718788e-08 7.06173620e-08 4.41103905e-08
 2.33592435e-11]
```




```
[1.82187813e-01 1.41144034e-01 1.19882002e-01 8.83869333e-02
 7.02808938e-02 4.99639610e-02 3.80999668e-02 3.08267651e-02
 2.98342213e-02 2.84815612e-02 2.49268628e-02 2.35483667e-02
 2.17418607e-02 1.73831972e-02 1.63501067e-02 1.47978112e-02
 1.30720181e-02 1.18307523e-02 1.05092250e-02 9.01490576e-03
 8.63737539e-03 7.21624942e-03 7.11038058e-03 6.89243449e-03
 5.03197986e-03 4.82978737e-03 4.14962998e-03 3.18816726e-03
 2.93711660e-03 2.14386597e-03 1.64215733e-03 1.35758835e-03
 1.28773381e-03 6.51744979e-04 3.97614261e-04 1.45522755e-04
 1.17392623e-04]
```



# Uses of PCA

**Data understanding:**

- PCA can be used to identify the most important variables in a dataset and to understand the relationships between them.
- PCA can be used to detect outliers and anomalies in the data by identifying data points that are far from the rest of the data.

**Data visualization:**

- PCA can be used to visualize high-dimensional data in a low-dimensional space, typically 2D or 3D, by reducing the dimensionality of the data while preserving the most important information.
- PCA can be used to identify clusters and patterns in the data that are not visible in the high-dimensional space.

**Data simplification:**

- PCA can be used to simplify the data by reducing the number of variables or features in the dataset while preserving the most important information.
- PCA can be used to remove noise and redundancy from the data by identifying the most important components and ignoring the rest.

**Dimensionality reduction:**

- PCA can be used to reduce the dimensionality of the data by finding a new set of variables that are smaller than the original set but still contain most of the information in the original set.
- PCA can be used to transform a large set of variables into a smaller one that still contains most of the information in the large set.

**Machine learning:**

- PCA can be used as a preprocessing step in machine learning to reduce the dimensionality of the data and to improve the performance of the model.
- PCA can be used to remove multicollinearity and to improve the interpretability of the model by identifying the most important variables.