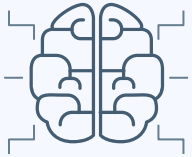
Machine Learning Project



Done by:

- Fady Mounir 49-0716
- Ingy Fam 49-2499
- Mira Emad 49-4627
- George Reda 49-0301
- Tony Amir 49-10132

Machine learning has become an essential part of our daily lives.

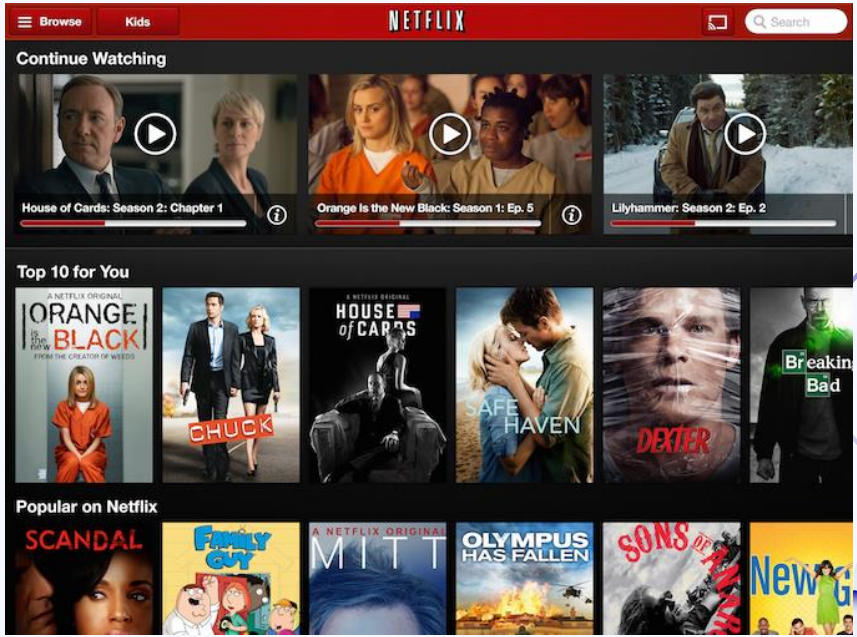


Table of contents

01

Background

Context and motivation for the project, Description of the problem

Background

"Imagine building a machine learning project like crafting a digital superhero. Each bit of code becomes a superpower, helping us teach computers to learn from data, predict outcomes, and solve problems. It's like giving our technology the ability to understand, adapt, and make smart choices. The motivation? To use this tech superhero to make things easier, solve puzzles, and make our digital world a bit more awesome each day!"

The challenge is to assess the eligibility of individuals or organizations for loans, considering factors like credit score, income, employment status, loan term, amount, assets, and past loan performance. **Solving this problem matters because** it streamlines the lending process, enhances decision-making, and promotes fair, inclusive, and efficient lending practices, ultimately benefiting both applicants and the lending institutions.

“Without big data analytics, companies are blind and deaf, wandering out on the web like deer on a freeway”

— Geoffery Moore

02

Dataset

Description of the data being used

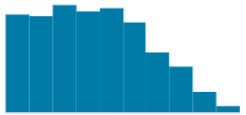
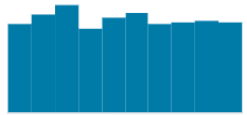
Dataset

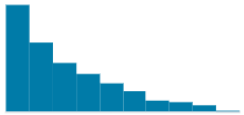
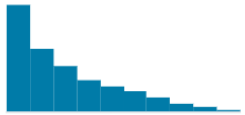
The loan approval dataset

The loan approval dataset is a robust financial resource for evaluating loan eligibility, featuring **13 columns and 4269 rows**. The columns encompass a mix **of continuous numbers, discrete variables, and object/string data types**. Essential details, such as credit score, income, employment status, loan term, amount, and asset values, are captured **within columns like** loan_id, no_of_dependents, education, self_employed, income_annum, loan_amount, loan_term, cibil_score, residential_assets_value, commercial_assets_value, luxury_assets_value, bank_asset_value, and loan_status. This dataset is instrumental in developing predictive models and algorithms to determine the likelihood of loan approval based on provided features.

Dataset

The loan approval dataset

loan_id	no_of_dependents	education	self_employed	income_annum	loan_amount	loan_term
	Number of Dependents of the Applicant	Education of the Applicant	Employment Status of the Applicant	Annual Income of the Applicant	Loan Amount	Loan Term in Years
 1 4269	 0 5	Graduate 50% Not Graduate 50%	true 0 0% false 0 0%	 200k 9.90m	 300k 39.5m	 2 20

cibil_score	residential_asset...	commercial_asse...
Credit Score		
 300 900	 -100000 29.1m	 0 19.4m

true and false are 0% as the attributes are yes and no.

Objective/Goal & Method

Objective/Goal

The primary objective of applying machine learning models to the loan approval dataset is to **predict the outcome of the "loan_status"** column. This entails developing loan predictive algorithms that leverage various features within the dataset to accurately forecast **whether a loan will be approved or denied.** By focusing on this key column, the goal is to **create models** that contribute to more informed **decision-making in the lending process, enhancing the efficiency and effectiveness of loan approvals.**

Method

In the upcoming slides, we will delve into the methodologies employed to achieve our objective. The use of machine learning techniques and data analysis to harness the information embedded in the dataset. We'll explore the process of model development, feature engineering, and the evaluation of model performance

03 Objective/Goal & Method

what the project aims to achieve and how we did it.

Methodologies

Pre-processing

1-Data Cleaning
2-Mapping(encoding)
3-Normalization

Classification

1- For normalized dataset
2- For PCA dataset(without target column)

Clustering

1- PCA dataset with target column
2-PCA dataset without target column

1- Preprocessing

Data Cleaning:

- Checking against duplicates and null values in the dataset however, the loan approval data set was clean

Mapping:

- Mapping discrete columns to 0s and 1s:
  1- education -> Graduated: 1 , Not Graducated: 0
  2- self_employed: Yes:1 , No: 0
  3- loan_status: Approved:1 , Rejected: 0

Then, the new columns for the mapped values are education_numeric, self_eployed_numeric, and loan_status_numeric, all added to the dataset that will be used later. (df)

Dataset before mapping:

education	self_employed	income_annum	loan_amount	loan_term	cibil_score	loan_status
Graduate	No	9600000	29900000	12	778	Approved
Not Graduate	Yes	4100000	12200000	8	417	Rejected
Graduate	No	9100000	29700000	20	506	Rejected
Graduate	No	8200000	30700000	8	467	Rejected
Not Graduate	Yes	9800000	24200000	20	382	Rejected

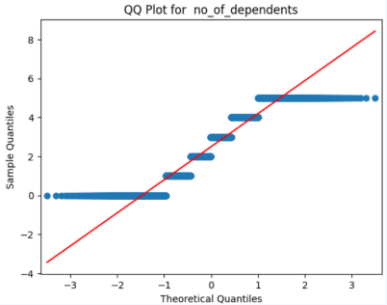
Dataset after mapping:

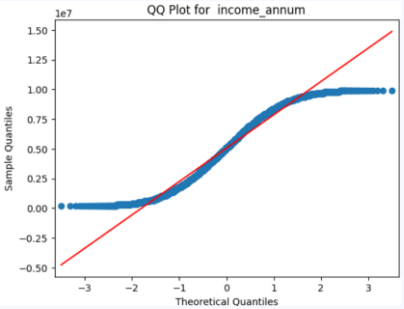
residential_assets_value	commercial_assets_value	luxury_assets_value	bank_asset_value	loan_status_numeric	education_numeric	self_employed_numeric
2400000	17600000	22700000	8000000	1	1	0
2700000	2200000	8800000	3300000	0	0	1
7100000	4500000	33300000	12800000	0	1	0
18200000	3300000	23300000	7900000	0	1	0
12400000	8200000	29400000	5000000	0	0	1
...	...	...	...	...	...	...
2800000	500000	3300000	800000	0	1	1
4200000	2900000	11000000	1900000	1	0	1
1200000	12400000	18100000	7300000	0	0	0
8200000	700000	14100000	5800000	1	0	0
17800000	11800000	35700000	12000000	1	1	0

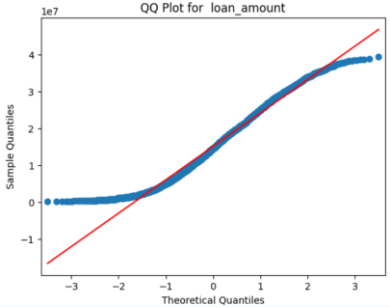
1- Preprocessing

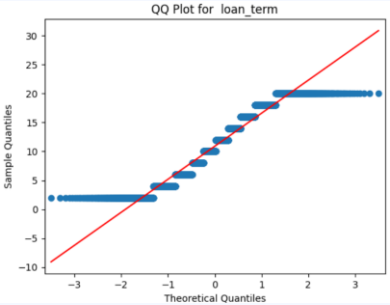
Normalization:

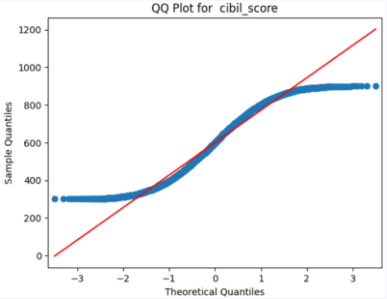
- QQ-plot for dataset (df) :

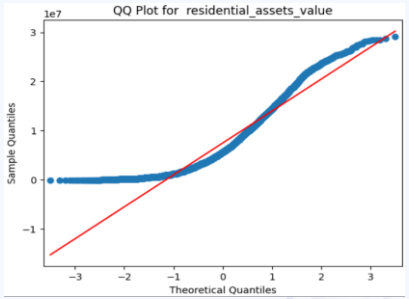


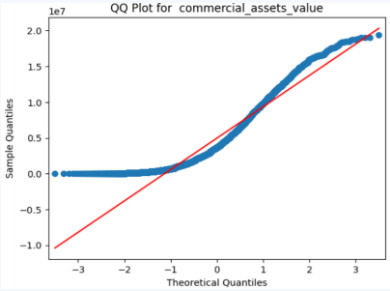


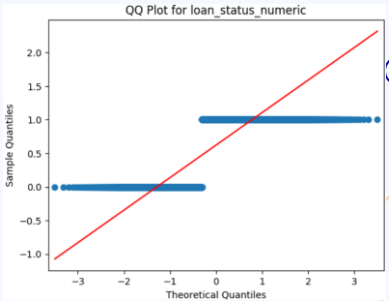












1- Preprocessing

Normalization:

- As the graphs illustrate, all columns need to be normalised because of the strong deviation from the line.

- A snippet of the dataset after normalisation (all values lie from 0 to 1)

no_of_dependents	income_annum	loan_amount	loan_term	cibil_score	residential_assets_value	commercial_assets_value	luxury_assets_value
0.4	0.969072	0.755102	0.555556	0.796667	0.085616	0.907216	0.575835
0.0	0.402062	0.303571	0.333333	0.195000	0.095890	0.113402	0.218509
0.6	0.917526	0.750000	1.000000	0.343333	0.246575	0.231959	0.848329
0.6	0.824742	0.775510	0.333333	0.278333	0.626712	0.170103	0.591260
1.0	0.989691	0.609694	1.000000	0.136667	0.428082	0.422680	0.748072

Dimensionality Reduction

Principle Component Analysis:

- PCA reduces the dimensionality of data while preserving essential information
- PCA was done on the dataset without target column(loan_status_numeric) and with the target column (pca_raw and X_pca).
- Number of components was determined using the cumulative variance.
- Number of components for PCA with target is 8 and 7 for PCA without target.

PCA with target column

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
0	-0.406195	0.912950	-0.501107	0.081242	-0.114405	0.086839	0.571287	-0.088507
1	0.646920	-0.517540	0.589134	-0.023853	-0.468830	-0.290960	0.015560	0.035982
2	0.730059	0.851903	-0.531598	0.061790	0.012486	0.481266	-0.010255	0.218800
3	0.688293	0.677468	-0.578635	0.039307	0.116355	-0.180314	-0.325494	-0.033461
4	0.839949	0.370549	0.803954	0.067739	0.412887	0.496466	-0.000614	0.067858

PCA without target column

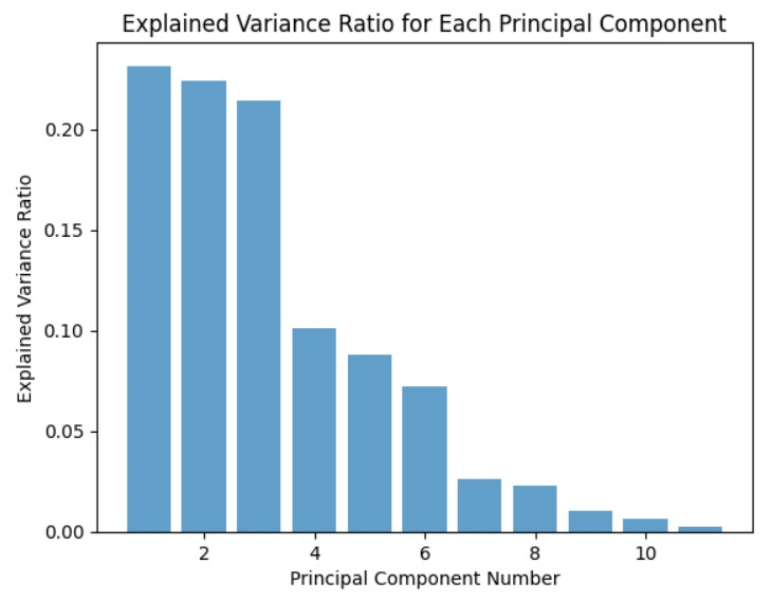
	PC1	PC2	PC3	PC4	PC5	PC6	PC7
0	0.857227	-0.535376	0.074296	-0.125412	0.046968	0.312984	0.571999
1	-0.438440	0.625253	-0.015584	-0.457591	-0.245201	-0.316892	0.015652
2	0.915544	-0.517109	0.069296	0.025357	0.512357	-0.161563	-0.009920
3	0.736636	-0.562692	0.046869	0.125553	-0.148705	-0.206805	-0.324940
4	0.469545	0.831614	0.075441	0.431293	0.547185	-0.358283	-0.001324

Dimensionality Reduction

Principle Component Analysis:

- Variance bar graph: (with target)
The graph illustrates how much variance each PC contributes to the overall dataset.

It helps in deciding how many principal components to retain based on the amount of variance explained



Feature extraction and selection:

Correlated features:

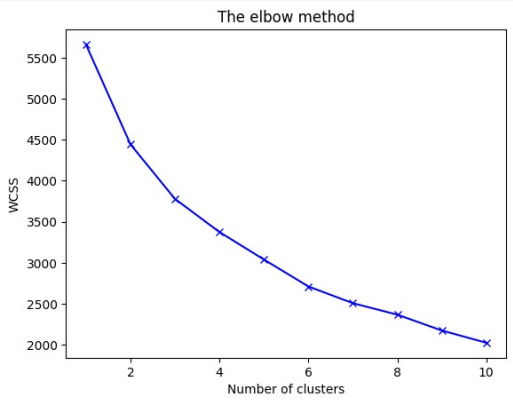
- Table illustrates the correlation between features.

	no_of_dependents	income_annum	loan_amount	loan_term	cibil_score	residential_assets_value	commercial_assets_value	luxury_assets_value	bank_asset_value	education_numeric	self_employed_numeric
no_of_dependents	1.000000	0.000256	-0.007731	-0.012531	-0.008442	-0.006673	-0.004101	-0.006367	0.005316	-0.010196	0.007422
income_annum	0.000256	1.000000	0.928383	0.017354	-0.036682	0.634352	0.633718	0.928056	0.849639	0.013002	0.002045
loan_amount	-0.007731	0.928383	1.000000	0.018343	-0.026847	0.588828	0.590508	0.858396	0.782245	0.011665	-0.000249
loan_term	-0.012531	0.017354	0.018343	1.000000	0.008453	0.009412	0.000632	0.017262	0.020203	-0.008671	-0.003858
cibil_score	-0.008442	-0.036682	-0.026847	0.008453	1.000000	-0.043259	-0.006271	-0.037326	-0.018143	-0.003754	-0.004163
residential_assets_value	-0.006673	0.634352	0.588828	0.009412	-0.043259	1.000000	0.402003	0.580027	0.533244	0.015495	0.006839
commercial_assets_value	-0.004101	0.633718	0.590508	0.000632	-0.006271	0.402003	1.000000	0.577120	0.550251	0.005898	-0.015673
luxury_assets_value	-0.006367	0.928056	0.858396	0.017262	-0.037326	0.580027	0.577120	1.000000	0.788922	0.015413	0.002447
bank_asset_value	0.005316	0.849639	0.782245	0.020203	-0.018143	0.533244	0.550251	0.788922	1.000000	0.010710	-0.005404
education_numeric	-0.010196	0.013002	0.011665	-0.008671	-0.003754	0.015495	0.005898	0.015413	0.010710	1.000000	-0.015505
self_employed_numeric	0.007422	0.002045	-0.000249	-0.003858	-0.004163	0.006839	-0.015673	0.002447	-0.005404	-0.015505	1.000000

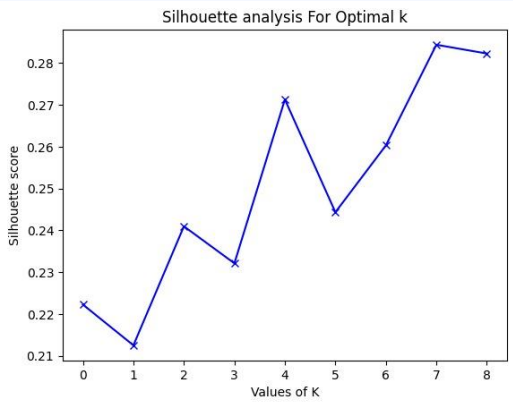
2- Clustering

1- PCA dataset with the target column

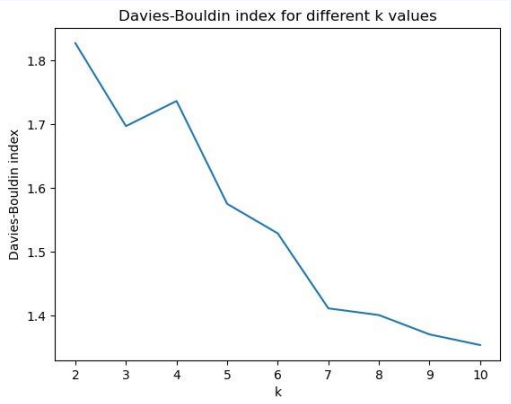
a) K-means:



Elbow method



Silhouette method

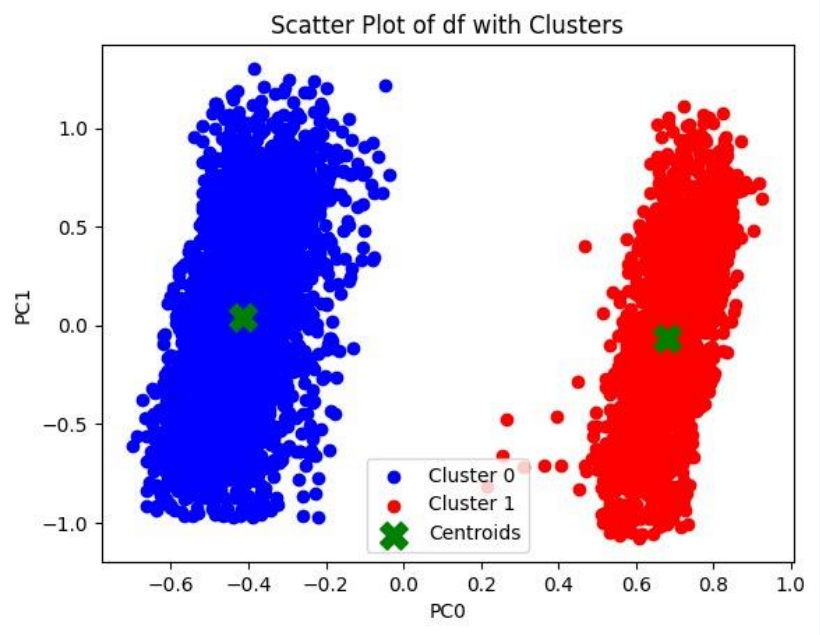


Davis-Bouldin

2- Clustering

1- PCA dataset with the target column

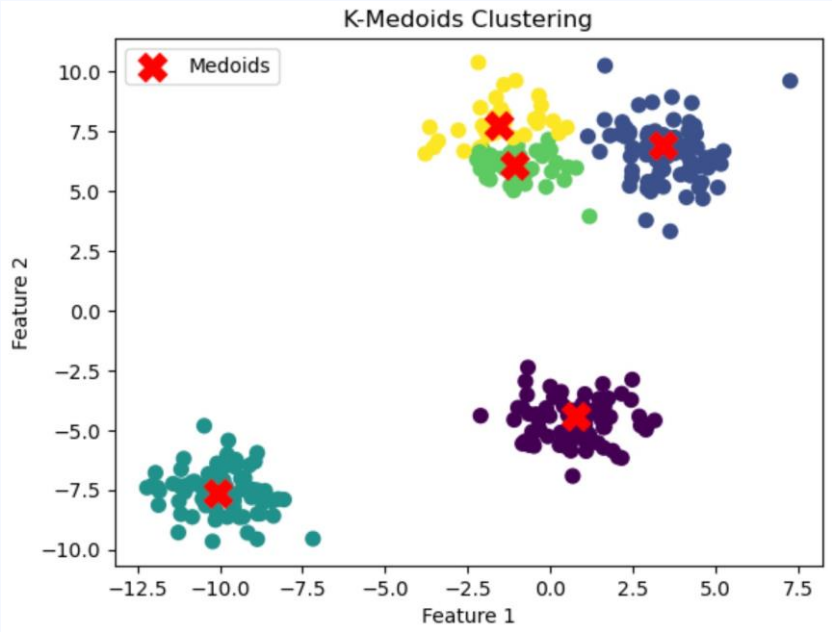
a) **K-means:**



2- Clustering

1- PCA dataset with the target column

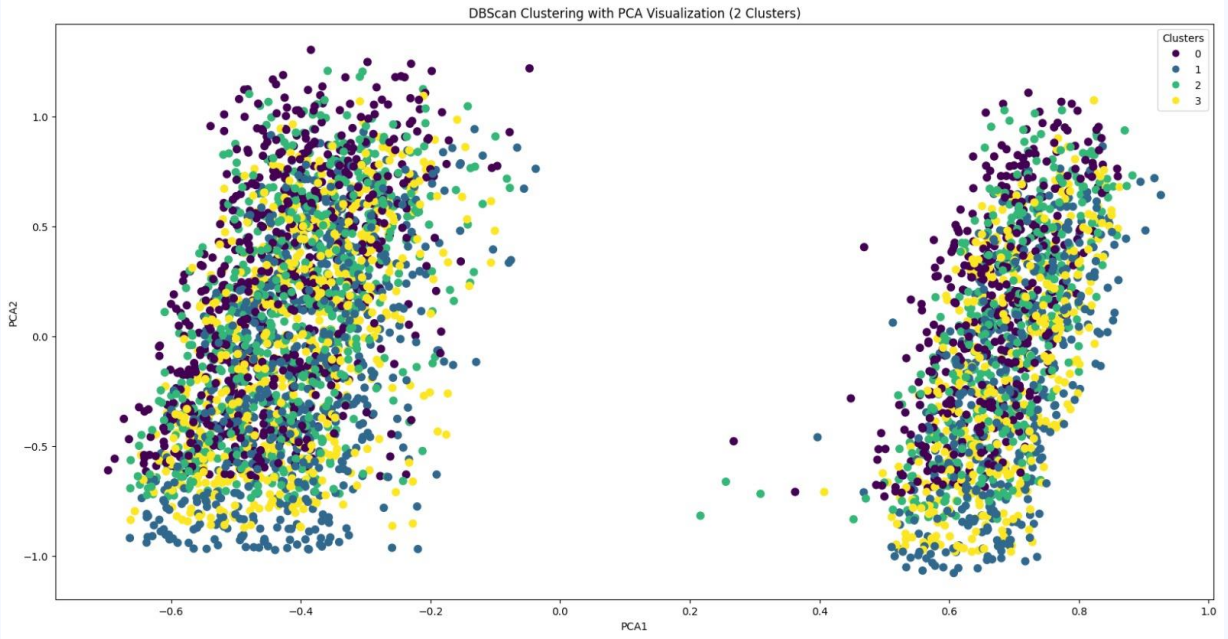
b) K-medoids:



2- Clustering

1- PCA dataset with the target column

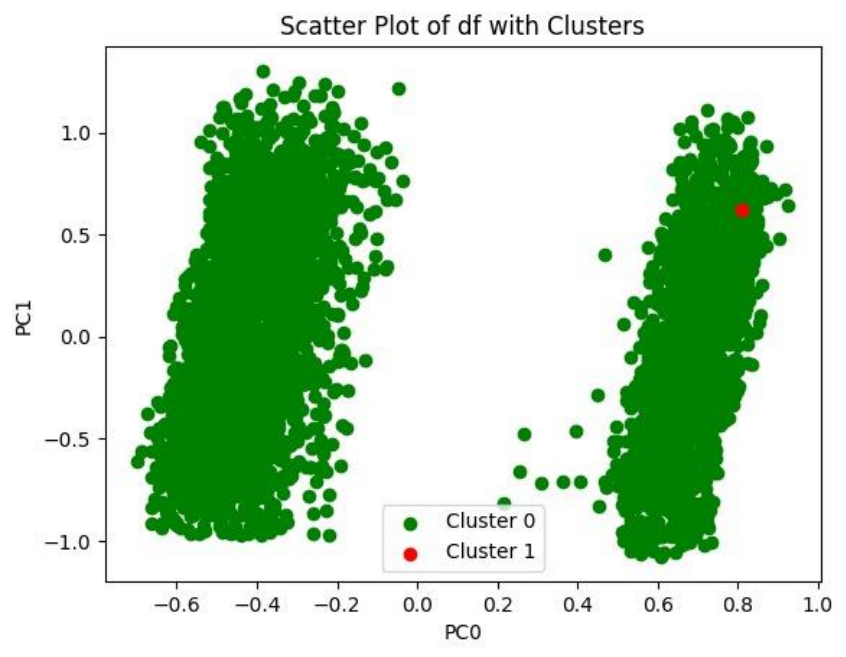
c) DBSCAN



2- Clustering

1- PCA dataset with the target column

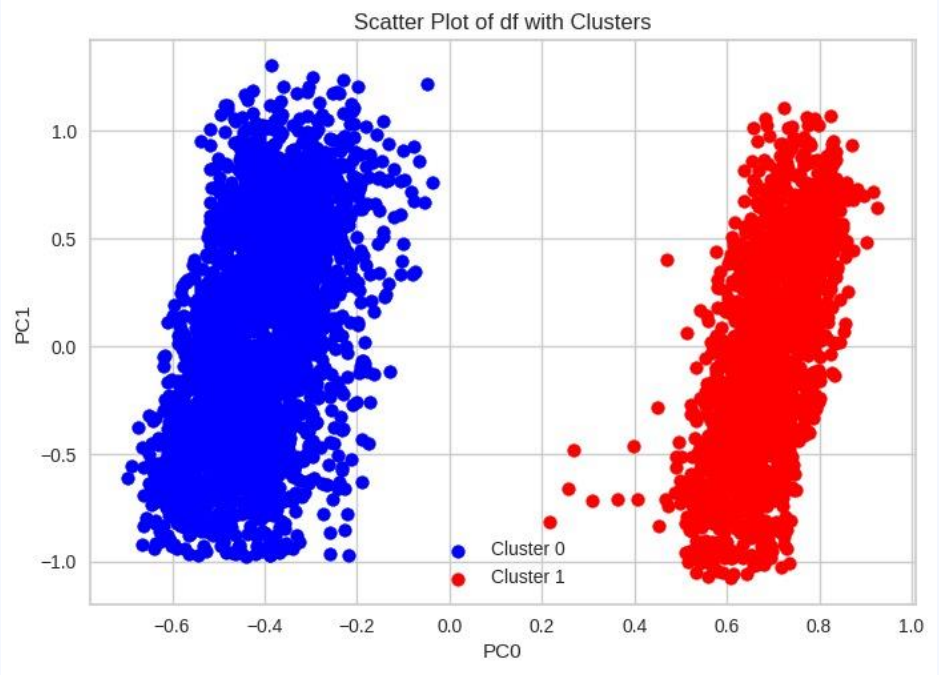
d) K-modes



2- Clustering

1- PCA dataset with the target column

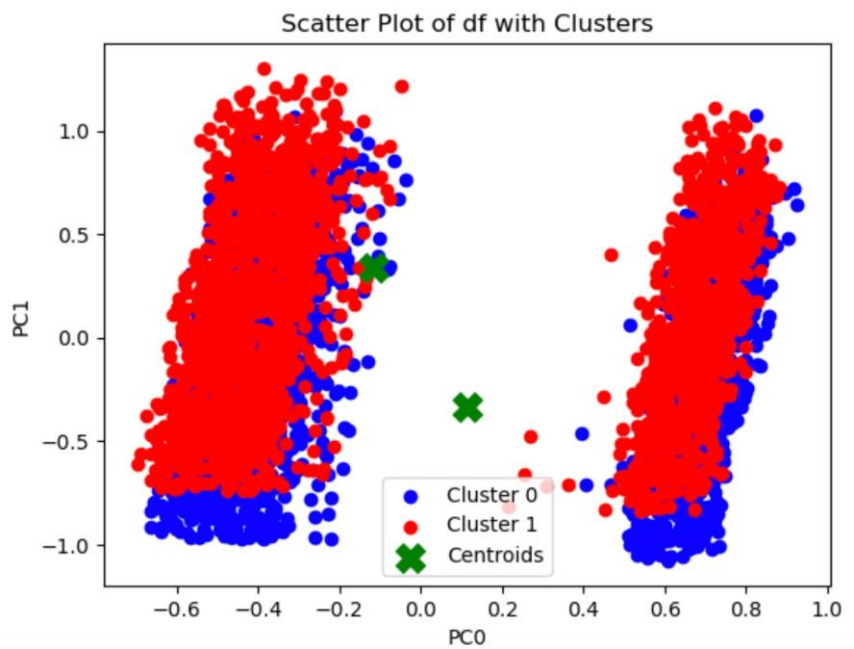
e) Hierarchical:



2- Clustering

2- PCA dataset withOUT the target column

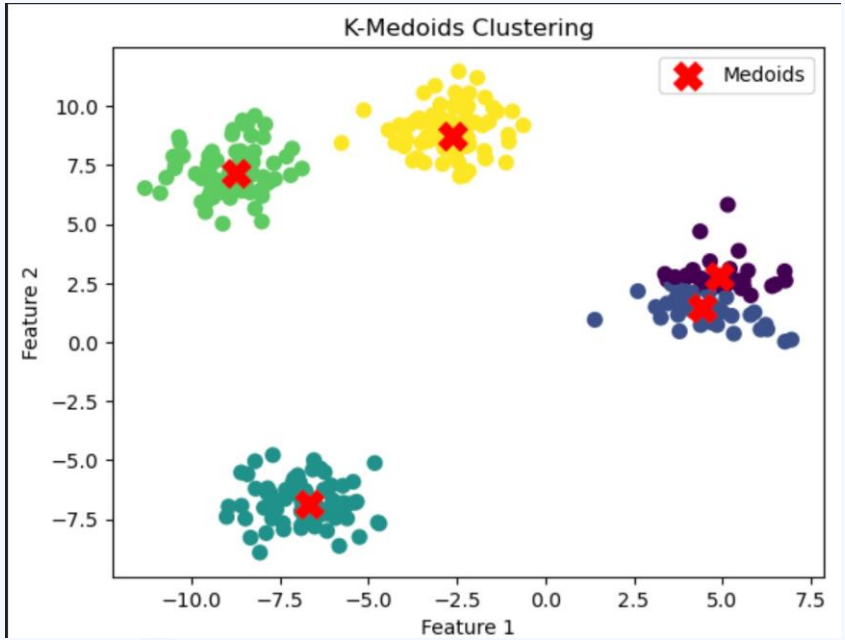
a) K-means:



2- Clustering

2- PCA dataset withOUT the target column

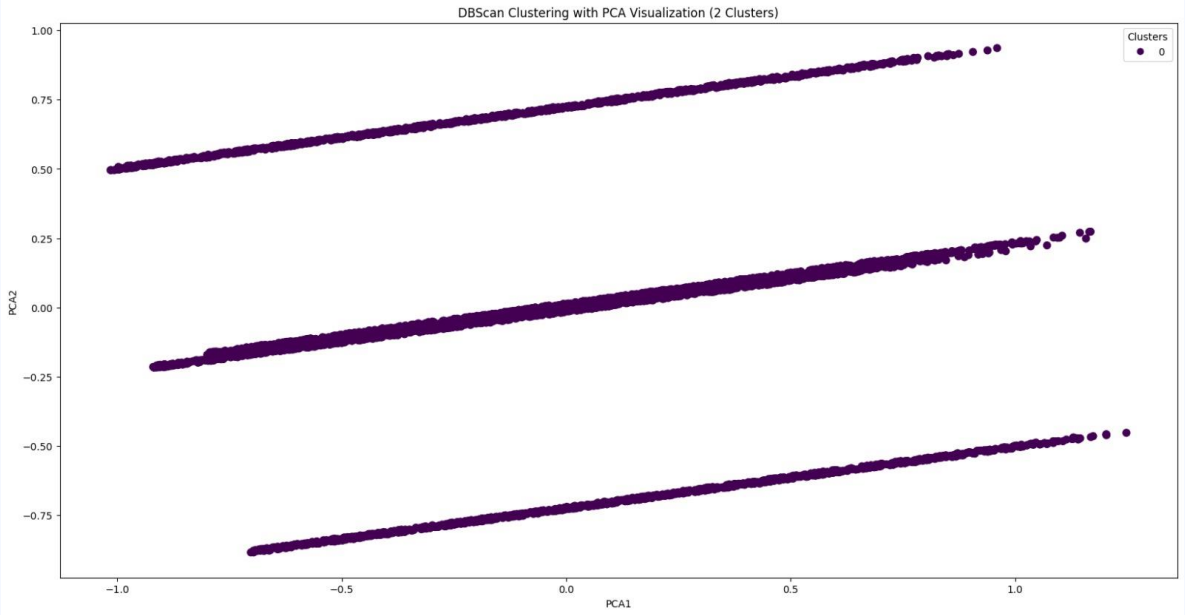
b) K-medoids



2- Clustering

2- PCA dataset withOUT the target column

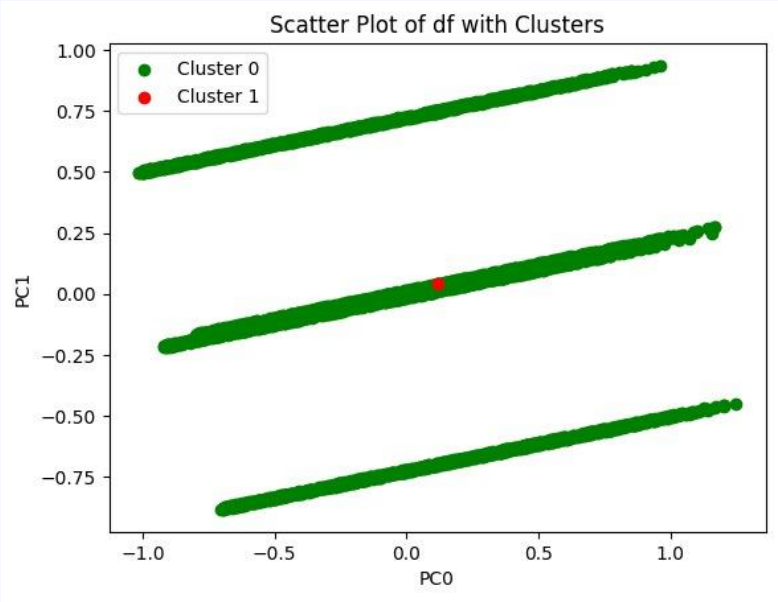
c) DBSCAN



2- Clustering

2- PCA dataset withOUT the target column

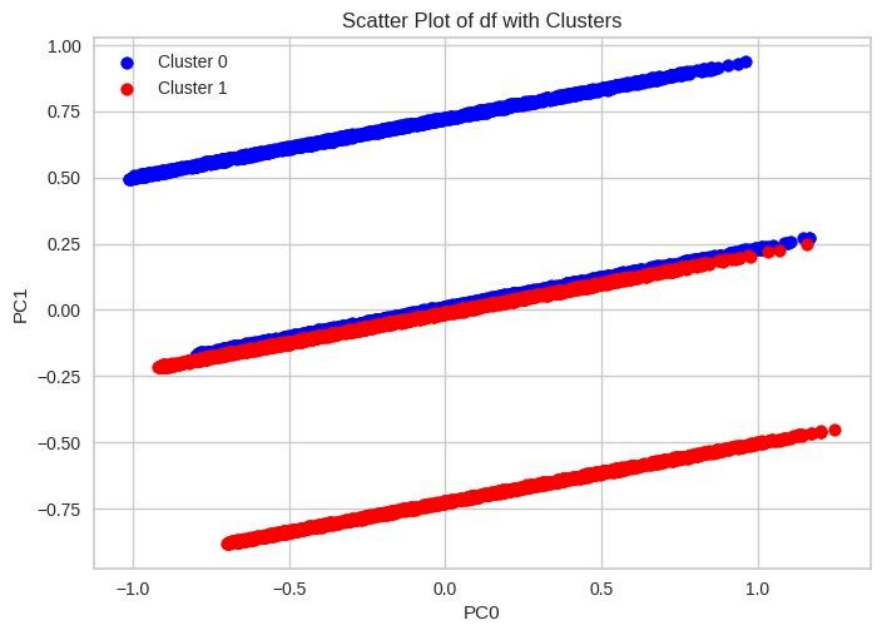
d) K-modes



2- Clustering

2- PCA dataset withOUT the target column

e) Hierarchical :



2- Clustering

Comparison between kmeans, kmedoids, DBSCAN:

1- K-means:
Method: Divides data into K clusters based on mean values.
Centroids: Utilizes cluster centroids (means) to represent each cluster.
Outlier Sensitivity: Sensitive to outliers as they heavily impact the mean calculation.
Applicability: Works well for globular-shaped clusters but struggles with outliers and non-linear shapes.

2- Clustering

Comparison between kmeans, kmedoids, DBSCAN:

2- K-medoids:

Method: Divides data into K clusters based on medoid points (data points with the least average dissimilarity to all other points in the cluster).
Centroids: Uses actual data points as cluster representatives, providing robustness against outliers.
Outlier Sensitivity: Less sensitive to outliers compared to K-means due to the use of medoids.
Applicability: Suitable for non-Euclidean distances and arbitrary-shaped clusters

2- Clustering

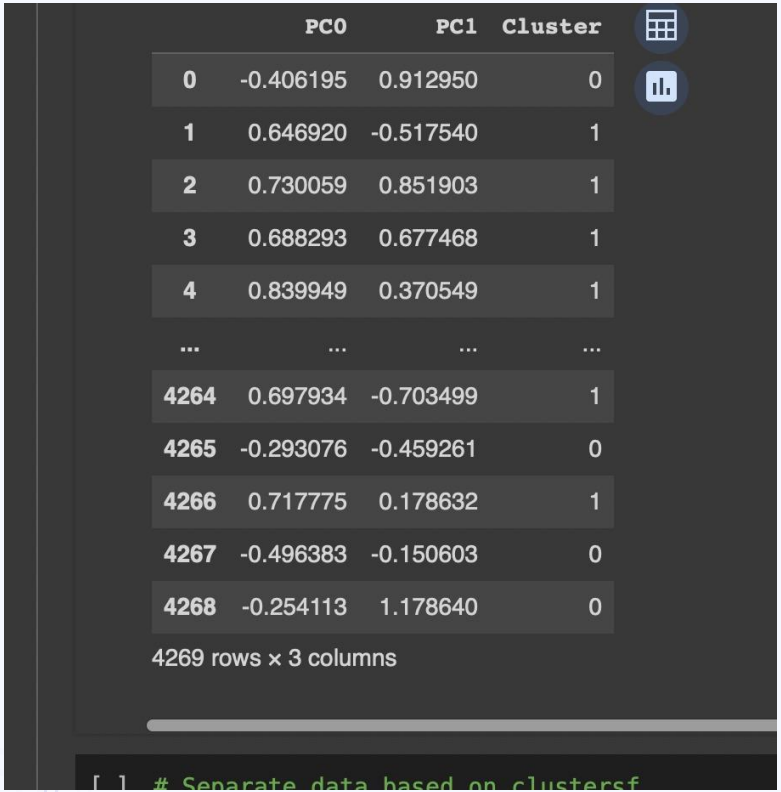
Comparison between kmeans, kmedoids, DBSCAN:

3- DBSCAN (Density-Based Spatial Clustering of Applications with Noise):

Method: Identifies clusters based on the density of data points.
Cluster Shape: Capable of identifying clusters of arbitrary shapes and sizes, and it detects noise as well.
Outlier Sensitivity: Less sensitive to outliers as it defines clusters based on density.
Applicability: Effective for datasets with varied cluster densities and shapes, resistant to noise, and capable of discovering clusters of irregular shapes.

Visualization of clustering results:

Use PCA to reduce dimensions of original data to 2



	PC0	PC1	Cluster
0	-0.406195	0.912950	0
1	0.646920	-0.517540	1
2	0.730059	0.851903	1
3	0.688293	0.677468	1
4	0.839949	0.370549	1
...	...	...	...
4264	0.697934	-0.703499	1
4265	-0.293076	-0.459261	0
4266	0.717775	0.178632	1
4267	-0.496383	-0.150603	0
4268	-0.254113	1.178640	0

4269 rows × 3 columns

3- Classification

Classification on normalised dataset:

Splitting data: → 70% training (2988 rows)

→ 30% testing (1281 rows)

We used 3 Classification techniques:

- Logistic Regression
- Naïve Bayes
- Random Forest

3- Classification

Why did we use these types of Classification

Logistic Regression:

Used for binary classification, predicting the probability of an observation belonging to a particular class. It models the relationship between a dependent binary variable and one or more independent variables by estimating probabilities using a logistic function.
Best choice for problems where the outcome is binary or categorical.

3- Classification

Why did we use these types of Classification

Naïve Bayes:

probabilistic classification algorithm based on Bayes' theorem with an assumption of independence between predictors. It calculates the probability that a data point belongs to a certain class given the presence of certain features.

Pros:

It's fast, efficient, and performs surprisingly well in many real-world applications

3- Classification

Why did we use these types of Classification

Random Forest:

learning method that constructs a multitude of decision trees during training and outputs the mode of the classes for classification tasks.

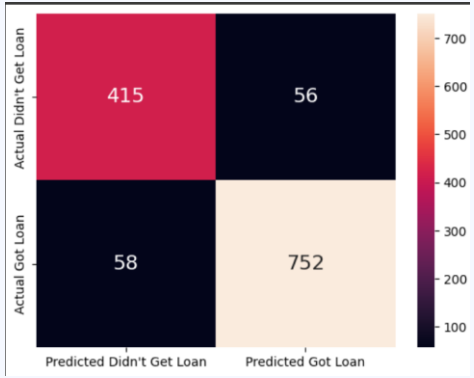
Pros:

Robust, less prone to overfitting, and capable of handling complex relationships in the data. It excels in capturing diverse patterns, handling large datasets, and providing high accuracy, making it a versatile and powerful tool for various machine learning tasks.

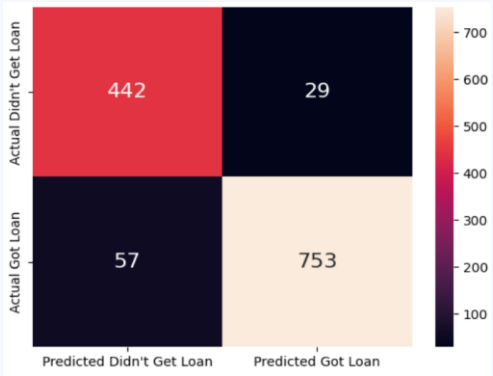
3- Classification

Classification on normalised dataset:

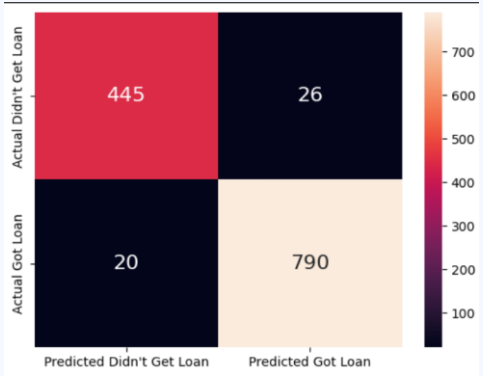
Confusion Matrix:



Logistic Regression



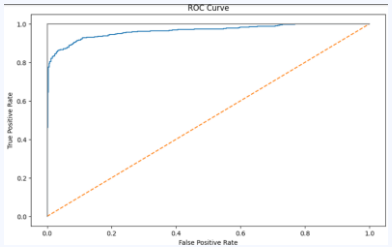
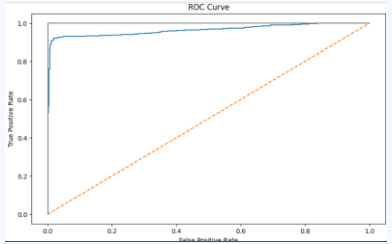
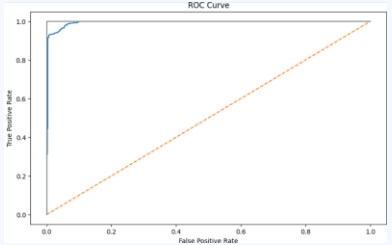
Naïve Bayes



Random Forest

3- Classification

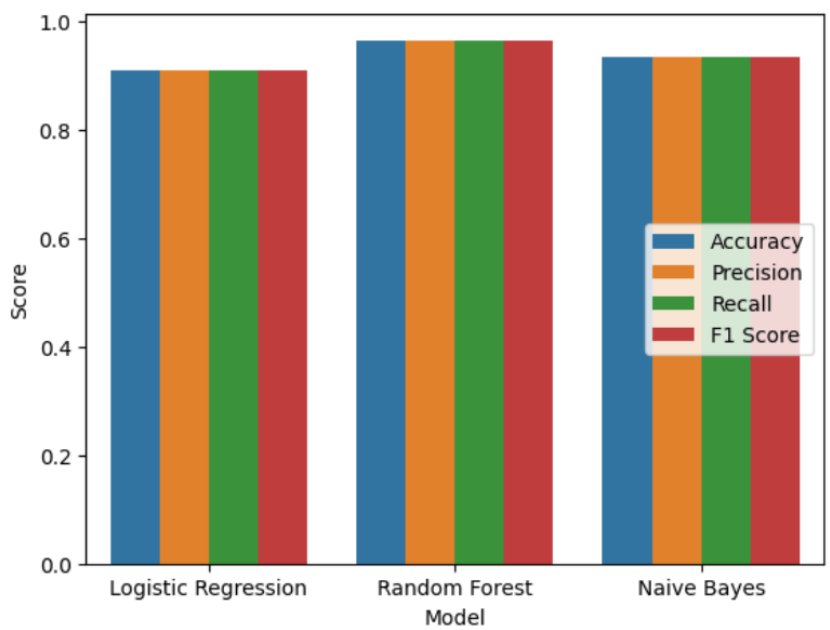
Classification on normalised dataset:

Logistic Regression	Naïve Bayes	Random Forest
• Accuracy: 91.10 % • Precision: 91.11 % • Recall: 91.10% • F1 score: 91.10 • ROC AUC: 96.3%	• Accuracy: 93.29 % • Precision: 93.46% • Recall: 93.29% • F1 score: 93.34% • ROC AUC: 96.35%	• Accuracy: 96.41 % • Precision: 96.40% • Recall: 96.41% • F1 score: 96.41% • ROC AUC: 99.40
		

3- Classification

Classification on normalised dataset:

Comparison between 3 algorithms:



3- Classification

Classification on normalized dataset:

Visualization of the decision boundary along with the original data points :SVM

```
from sklearn.metrics import classification_report

svm_report = classification_report(y_test,y_pred)

print(svm_report)
```

```
              precision    recall  f1-score   support

         0.0       0.88      0.91      0.90       471
         1.0       0.95      0.93      0.94       810

    accuracy                           0.92      1281
   macro avg       0.92      0.92      0.92      1281
weighted avg       0.93      0.92      0.92      1281
```

3- Classification

Classification on PCA dataset (without target column) (raw_pca dataset) :

Splitting data: → 70% training

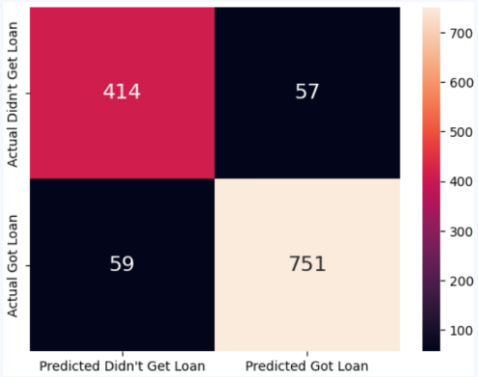
→ 30% testing

- Test data: The y-train and y-test were taken from normalized dataset as the PCA dataset used does not include target column (y value).
- Therefore, the PCA dataset only contains x values, which were divided into 30% for testing and 70% for training.

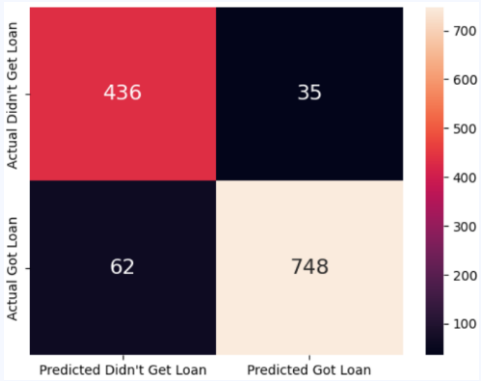
3- Classification

Classification on PCA dataset:

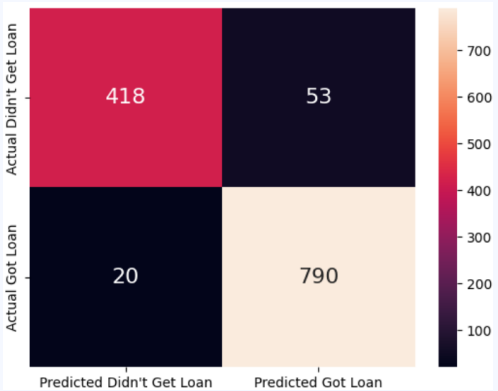
Confusion Matrix:



Logistic Regression



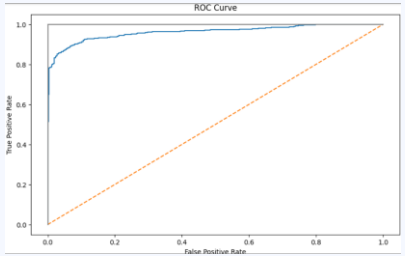
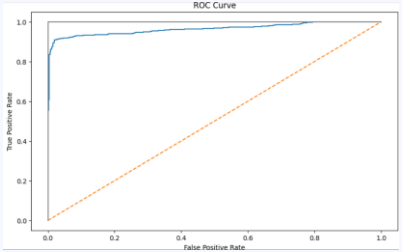
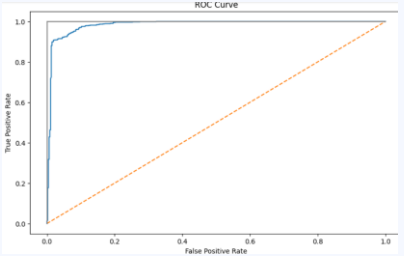
Naïve Bayes



Random Forest

3- Classification

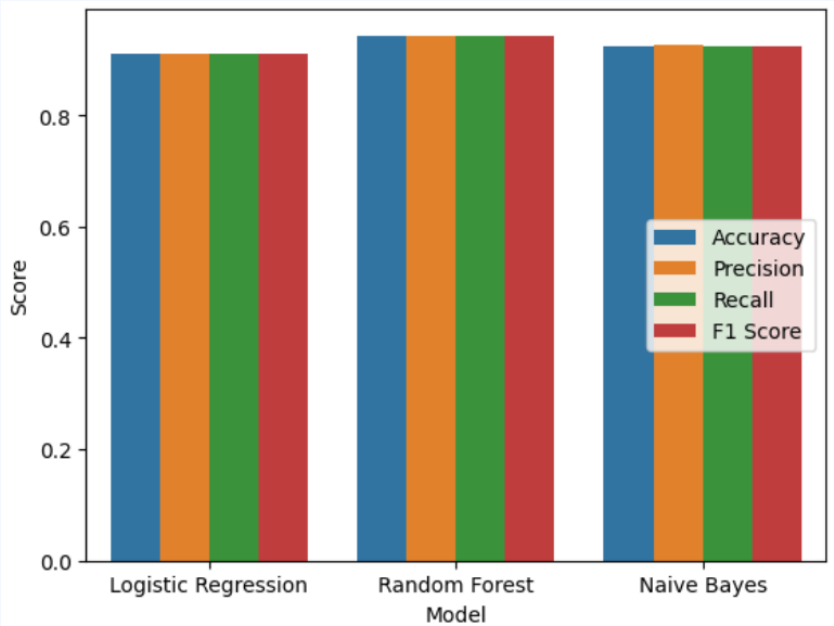
Classification on PCA dataset:

Logistic Regression	Naïve Bayes	Random Forest
• Accuracy: 90.94 % • Precision: 90.95 % • Recall: 90.94% • F1 score: 90.95% • ROC AUC: 96.13%	• Accuracy: 92.43 % • Precision: 92.60% • Recall: 92.43% • F1 score: 92.47% • ROC AUC: 96.39%	• Accuracy: 94.30 % • Precision: 94.35% • Recall: 94.30% • F1 score: 94.25% • ROC AUC: 98.38%
		

3- Classification

Classification on PCA dataset:

Comparison between 3 algorithms:



3- Classification

Classification on PCA dataset:

Visualization of the decision boundary along with the original data points :SVM

```
from sklearn.metrics import classification_report

svm_report = classification_report(y_test,y_pred_pca)

print(svm_report)
```

```
              precision    recall  f1-score   support

         0.0       0.87      0.92      0.89       471
         1.0       0.95      0.92      0.93       810

    accuracy                           0.92      1281
   macro avg       0.91      0.92      0.91      1281
weighted avg       0.92      0.92      0.92      1281
```

04
Discussion and Future Insight

Summarizing the findings, discuss how the results can be improved

4- Discussion

Classification on Normalized dataset:

Results showcase the performance of three classification algorithms—Logistic Regression, Naïve Bayes, and Random Forest—applied to a dataset that has been normalized. **Random Forest demonstrated the best performance** among the three methods, with higher accuracy, precision, recall, and F1 score around 96%. **Indicating its exceptional predictive Prowers** compared to both Logistic Regression and Naïve Bayes in this specific scenario.

ROC AUC score was high for all classifications algorithms:

- Logistic regression: 96.3 %
- Naïve Bayes: 96.35 %
- Random Forest: 99.4%

4- Discussion

Classification on PCA dataset:

Results indicate the performance of three different classification algorithms—Logistic Regression, Naïve Bayes, and Random Forest—on a dataset that was pre-processed using PCA (Principal Component Analysis) for dimensionality reduction. **Random Forest outperformed both** Logistic Regression and Naïve Bayes with higher accuracy, precision, recall, and F1 score around 94-95%.This suggests that the **Random Forest algorithm** yielded the best performance among the three methods on this **PCA-transformed dataset,** offering **higher accuracy and predictive power** for the classification task at hand.

ROC AUC score was high for all classifications algorithms:

- Logistic regression: 96.13 %
- Naïve Bayes: 96.39 %
- Random Forest: 98.38%

4- Discussion

Comparing Results between Normalized and PCA datasets

- **Performance Improvement:** Across all algorithms, there's a consistent improvement in performance metrics when using the **normalized dataset compared to the PCA dataset.**

- **Random Forest:** Random Forest exhibits **significantly higher performance on both datasets**, with a notable boost in accuracy, precision, recall, and F1 score on the normalized dataset compared to the PCA dataset, 94% compared to 96%.

4- Discussion

Conclusion of Classification results

- **Data Scaling:** Normalization scales the data to a common range, facilitating better convergence for various algorithms. It helps different algorithms by making all features equally important in the decision-making process.

- **PCA's Dimensionality Reduction:** PCA reduces features, possibly discarding some information crucial for classification, impacting the performance of algorithms that rely on the full feature set.

- **Impact of Preprocessing:** Normalization standardizes the range of features, aiding algorithms to make more informed decisions. PCA, while reducing dimensions, might discard some variance which could impact performance.


Thank
you