University of Bucharest
Faculty of Mathematics and Computer Science

Loan Risk Prediction

Students:
Dolganiuc Ana - 407
Alexandru Simona-Maria - 410

2024

Introduction

In the modern financial area, understanding the dynamics of loan acceptance is important also for lenders and borrowers. The project uses as dataset the accepted loans in the USA from 2007 to 2018, that can be considered a period with significant economic events, including the aftermath of the 2008 financial crisis and the subsequent recovery. Loans are an essential financial instrument that fuel economic growth by enabling individuals and businesses to invest in various opportunities.

By analyzing this dataset, we want to find patterns and insights that can inform better decision-making in the lending industry.

Dataset Description

The dataset is taken from Kaggle, and the subject it's represented by the accepted loans from 2007 to 2018 in the USA, that contains detailed information and a variety of attributes that provide a view of the lending area during the mentioned period.

Dataset link: https://www.kaggle.com/datasets/wordsforthewise/lending-club/data.

Included key features

Loan information:

Loan amount - the amount borrowed;

Loan term - duration over which the loan is to be repaid;

Borrower information:

Home Ownership - whether the borrower own a home, rents or has a mortgage;

Annual income - the borrower's annual income;

Fico - likelihood that a borrower will repay the loan;

Credit Information:

Credit score - represent the borrower's credit report;

Revolving balance - the amount of revolving(e.g credit card debt) credit the borrower is carrying;

Number of open accounts - the number of open credit lines in the borrower's credit report;

Loan performance and status:

Loan status - whether the loan is current, late or charged off;

Total payment - total amount paid by the borrower to date;

Purpose of loan:

Business investment

Debt consolidation

Home improvement

Geographical information

State - where the borrower resides

Dataset preparation and exploratory data analysis

Data size

The DataFrame has 70052 rows and 100 features.

The target feature is loan_status, and we transformed it into a binary feature by assigning:

1 - Fully Paid

0 - The rest

```
+------------------+
|       loan_status|
+------------------+
|        Fully Paid|
|           Default|
|   In Grace Period|
|       Charged Off|
|Late (31-120 days)|
|           Current|
| Late (16-30 days)|
+------------------+
```

Data cleaning and preprocessing

- Removed duplicates
- Ensured the data types are ok
- Handling missing values:
    - Deleted the columns that have more than 40000 missing values
    - For categorical data we calculated the most frequent items
    - For continuous data:
        - For rows with more values, where the amounts are in dollars, we take the median values(bc_open_to_buy,bc_util)
        - For columns that have less than 3 missing values we deleted the corresponding rows
- One hot encoding for the categorical data, in order to use it further in the analysis:
    - sub_grade(an extension of the credit grade column)
    - term(loan term-36 or 60 months)
    - emp_length(employment length)
    - home_ownership(Rent, Own, Mortgage)

- title(the purpose for the loan)

Descriptive statistics

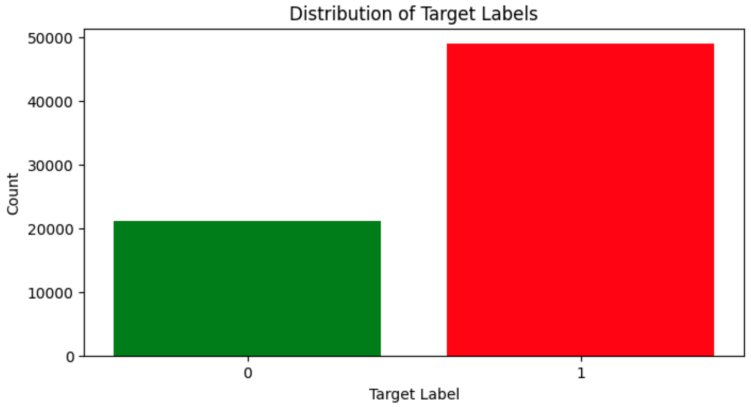

Figure 1: The distribution of the target label across the dataset

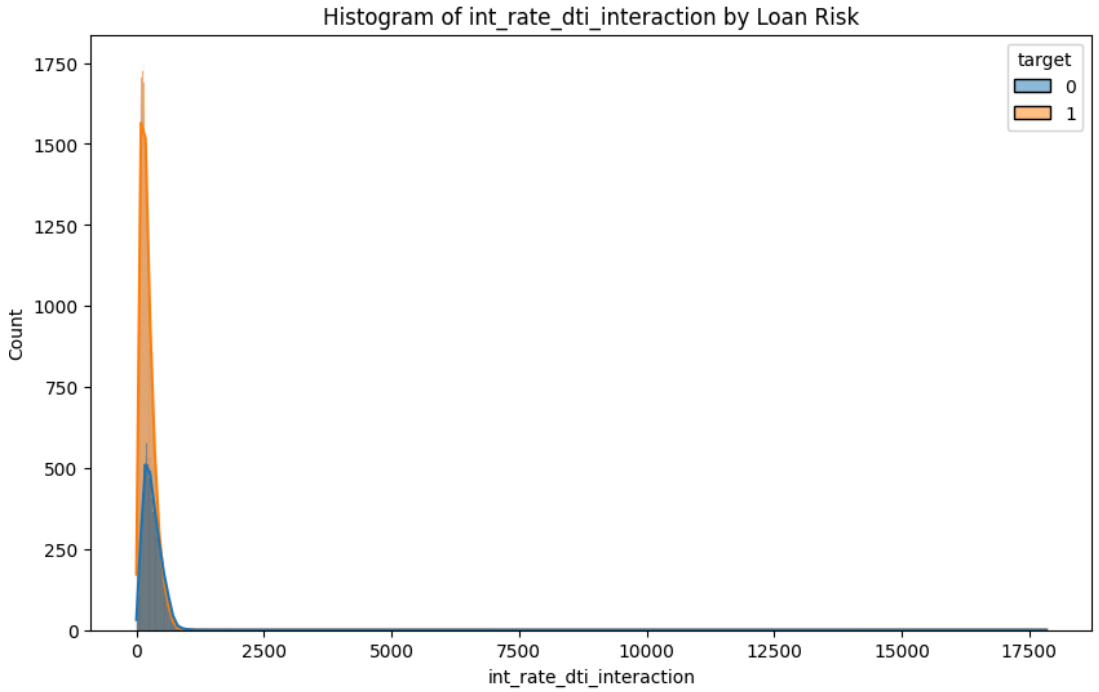

Figure 2: Histogram of interest rate and debt to income by Loan Risk

Categorical data analysis: Chi-square test

Chi-square test is a statistical tool used to measure the strength and direction of the relationship between the categorical variables and the target.

The results from the Chi-square tests for sub_grade, home_ownership, title, and term against your binary target variable indicate the level of association or independence between these categorical features and the target.

Interpretation of Chi-square Test Results:

1. Sub_grade
- pValue: 0.0
- Degrees of Freedom: 34
- Statistic: 7989.3098
- Interpretation: A p-value of 0.0 suggests that there is a statistically significant association between sub_grade and the binary target variable. The high chi-square statistic further confirms a strong relationship, indicating that different sub-grades have different distributions of the binary outcomes.

2. Home Ownership
- pValue: 2.0163e-12
- Degrees of Freedom: 3
- Statistic: 57.4930
- Interpretation: The p-value is significantly small, suggesting that home_ownership statuses are not independent of the binary target variable. The result indicates that the type of home ownership could influence the likelihood of the outcomes encoded in your binary target.

3. Title
- pValue: 0.0
- Degrees of Freedom: 11
- Statistic: 243.3119
- Interpretation: A p-value of 0.0 implies a significant association between the title of the loan and the binary target variable. This suggests that the purpose or title of the loan is a good predictor of the binary outcomes.

4. Term
- pValue: 0.0
- Degrees of Freedom: 1
- Statistic: 15176.5029
- Interpretation: The extremely low p-value and high chi-square statistic indicate a very strong association between the term of the loan and the binary target variable. This means that the loan term (e.g., length of the loan) is highly predictive of the outcome.

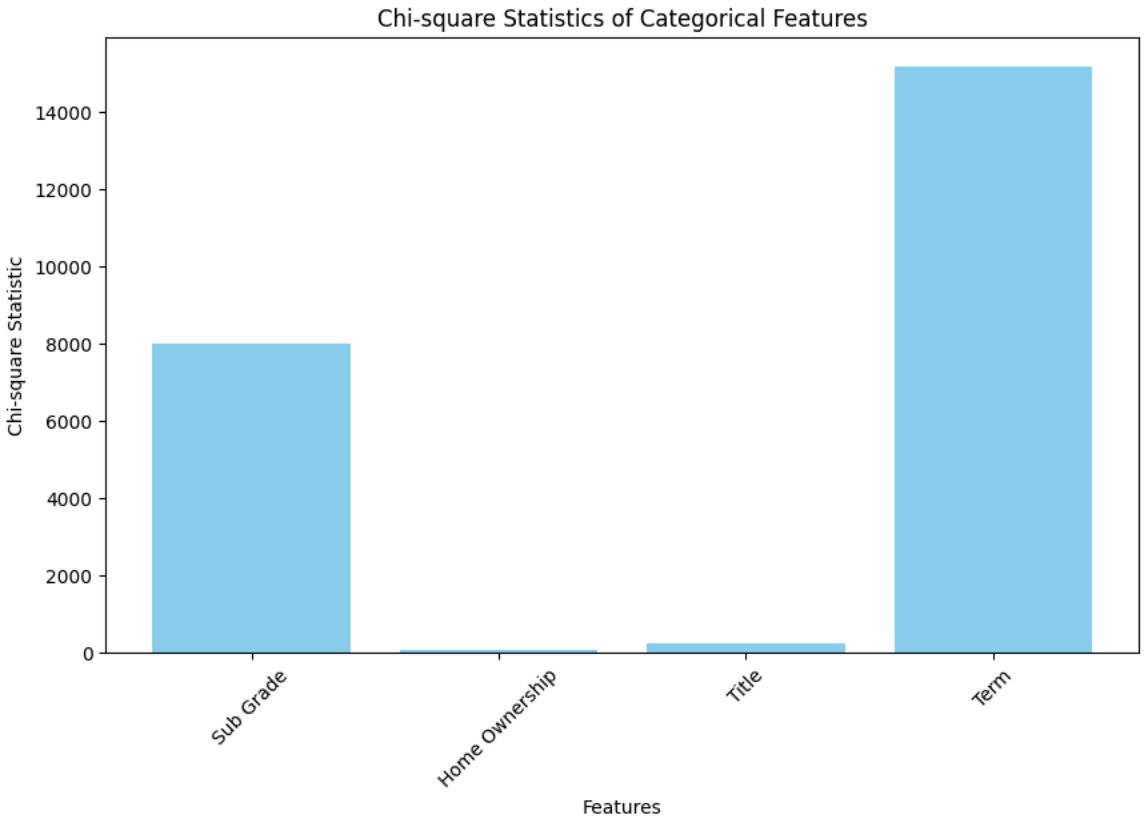

Figure 3: Chi-square statistics of categorical features

Continuous data analysis: Point-biserial correlation

Point-biserial correlation is a statistical tool used to measure the strength and direction of the relationship between the continuous variables and our binary target.

Interpretation of Point-biserial correlation for the most significant features:

- Last_fico_range_high (0.512831) and Last_fico_range_low (0.442641):

Correlation: Strong positive correlation. Interpretation: Higher FICO scores (both high and low ranges) are positively associated with the target being 1. This implies that better credit scores correlate with better loan outcomes.

- Last_pymnt_amnt (0.378579):

Correlation: Moderate positive correlation. Interpretation: Higher amounts in the last payment made are positively associated with the target being 1, indicating that larger payments are linked to more favorable loan statuses.

- Collection_recovery_fee (-0.356965) and Recoveries (-0.359060):

Correlation: Moderate negative correlation. Interpretation: Higher collection recovery fees and recoveries are negatively associated with the target being 1, suggesting that loans requiring recovery efforts are less likely to be in good standing.

- Total_pymnt_inv (0.169610) and Total_pymnt (0.169471):

Correlation: Weak positive correlation. Interpretation: Higher total payments (both invested and general) are slightly associated with better loan outcomes.

- Int_rate_dti_interaction (-0.247408) and Int_rate (-0.382878):

Correlation: Weak to moderate negative correlation. Interpretation: Higher interest rates and their interaction with debt-to-income ratio are negatively associated with the target being 1, indicating higher rates are linked to poorer loan statuses.

General Observations: Credit and Payment History: Features related to outstanding balances, payment amounts, and FICO scores have significant correlations with loan statuses. This is consistent with expectations as these factors are crucial indicators of creditworthiness and financial health.

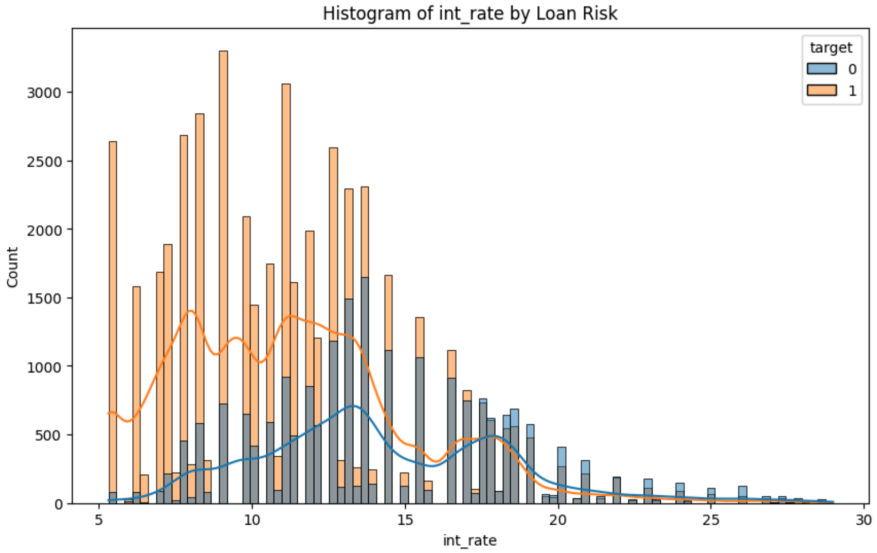

Figure 4: Histogram of interest rate by Loan Risk

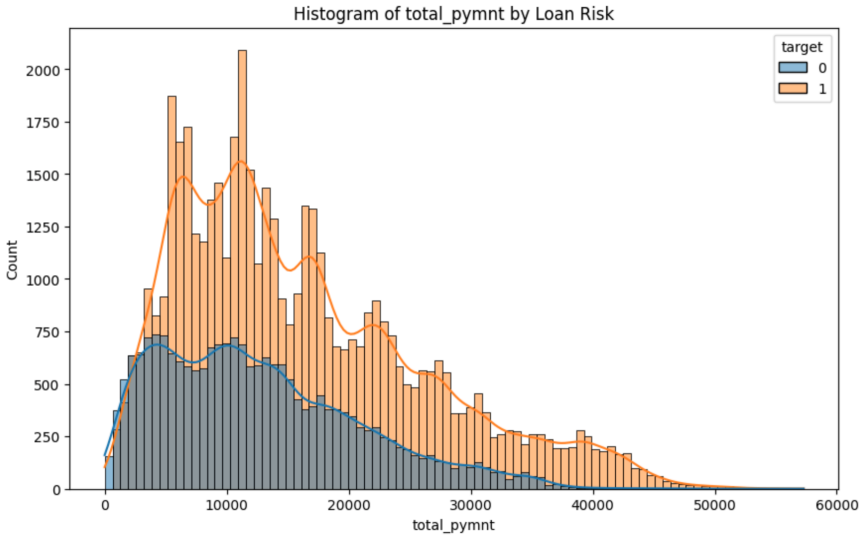

Figure 5: Histogram of total payment by Loan Risk

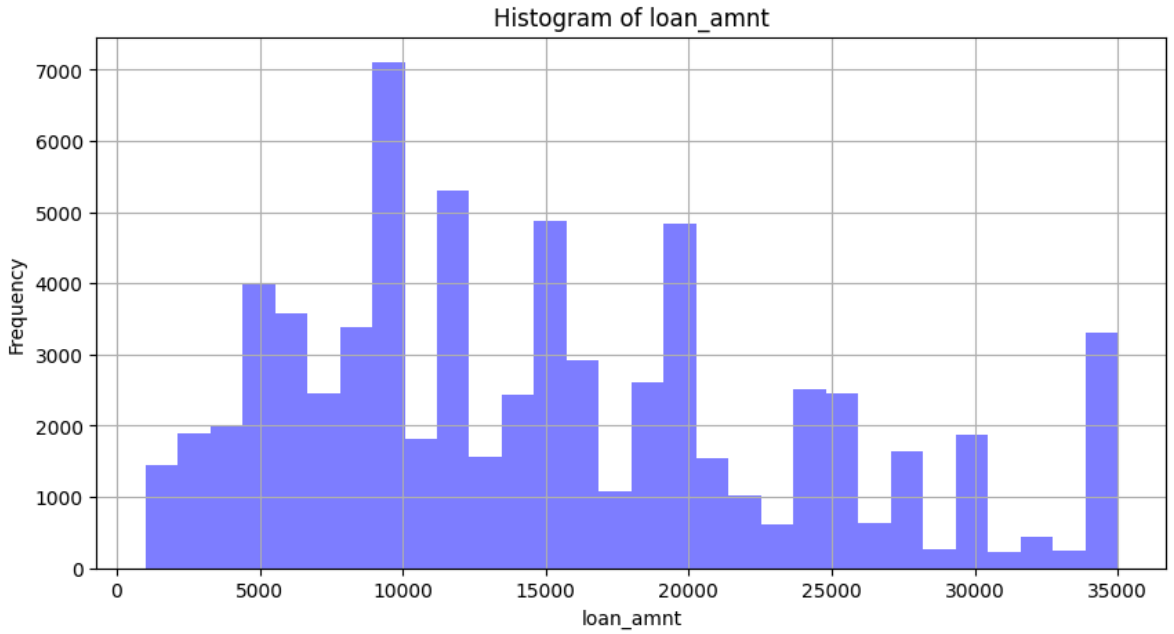

Figure 6: Histogram of Loan Amount

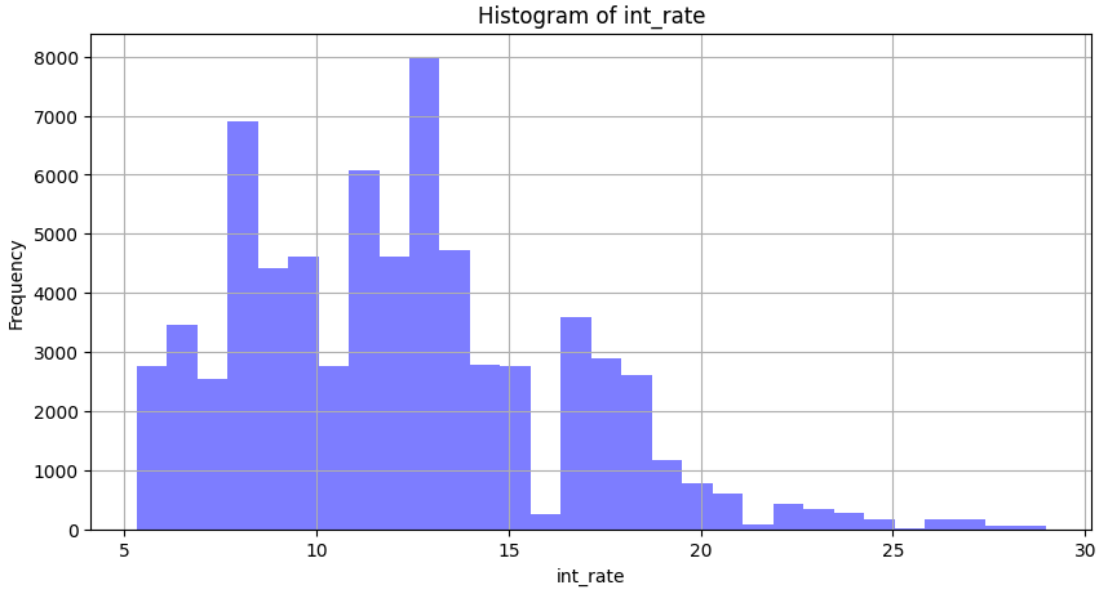

Figure 7: Histogram of Interest Rate

Feature engineering

In the table below we will see the interaction between income (annual_inc) and loan amount (loan_amnt), but also the interaction between interest rate (int_rate) and debt to income (dti).

The income and loan amount interaction aims to capture the correlation between the features and to see if the higher loans are proportionately riskier across different income brackets. As we can see the column income_loan_interaction indicates a stronger interaction between annual income and loan amount, but also

across different income brackets you can assess whether the interaction between load amount and risk varies depending on the income level.

The interaction between interest rate (the percentage of the principal amount charged by a lender to a borrower for the use of assets) and debt to income (evaluate a borrower's ability to manage monthly debt payments and repay the loans) explores whether borrowers with higher dti ratios are more affected by higher interest rate. A stronger interaction between the interest rate and the dti borrower's ratio is indicated.

annual_inc	loan_amnt	income_loan_interaction	int_rate	dti	int_rate_dti_interaction
35000.0	4225.0	1.47875E8	14.85	15.22	226.017
75000.0	20000.0	1.5E9	14.85	18.76	278.586
118000.0	28000.0	3.304E9	15.77	35.91	566.3006999999999
20000.0	6950.0	1.39E8	13.99	24.13	337.57869999999997
57774.0	24000.0	1.386576E9	21.48	24.05	516.594
40000.0	8000.0	3.2E8	11.48	36.6	420.168
108000.0	32000.0	3.456E9	12.88	25.63	330.1144
47000.0	15025.0	7.06175E8	25.09	11.54	289.5386
54500.0	10000.0	5.45E8	9.8	24.1	236.18000000000004
78000.0	20000.0	1.56E9	9.8	16.74	164.052
103000.0	15000.0	1.545E9	5.32	15.54	82.6728
50000.0	14825.0	7.4125E8	21.48	15.89	341.3172
80000.0	18000.0	1.44E9	9.17	6.47	59.329899999999995
190000.0	28000.0	5.32E9	6.99	8.69	60.7431
65000.0	6000.0	3.9E8	7.91	23.06	182.4046
35000.0	11475.0	4.01625E8	21.48	36.98	794.3303999999999
22000.0	6500.0	1.43E8	10.78	33.99	366.4122
90000.0	18000.0	1.62E9	9.17	14.28	130.9476
40000.0	14250.0	5.7E8	9.8	37.62	368.676
72000.0	16000.0	1.152E9	11.99	20.93	250.9507

only showing top 20 rows

Figure 8: Income vs. Loan Amount interaction and Interest Rate vs. Debt to Income

Methods applied and results

PySpark is the main library used, that offers large-scale data processing, but also matplotLib, seaborn, numpy.

- Dimension reduction

Subset selection- The subset selection was needed due to the very large number of features(100) in order to keep just the relevant features. Therefore, we kept the most important features that were identified via the Chi-square test and the Point bi-serial correlation.

Shrinking:

➔ The dataset has been standardized using the *L2 Norm* (Ridge Regression), being a trivial step before applying the PCA analysis in order to preserve

the distribution in case there are features with different units of measurement, for example $ or %. Also it is relevant when reducing overfitting.

➔ The dataset has been normalized using Min-Max scaling , being a trivial step before applying the LDA analysis in order to ensure that all features contribute equally, potentially improving discrimination between classes, especially when features have different units or ranges.

PCA (Principal component analysis) - widely used due to its simplicity and effectiveness in linear scenarios, that relies on computing covariance matrices and eigenvalues.

Below it represents the relevant number of components in PCA analysis.

Since the elbow of the plot is around the fourth component, selecting four components would likely capture most of the significant variance (around 90% of the data) in the data while reducing dimensionality efficiently. This would keep the model simpler and more interpretable, without losing much information.

As we can see, the number of components that are relevant are 20:

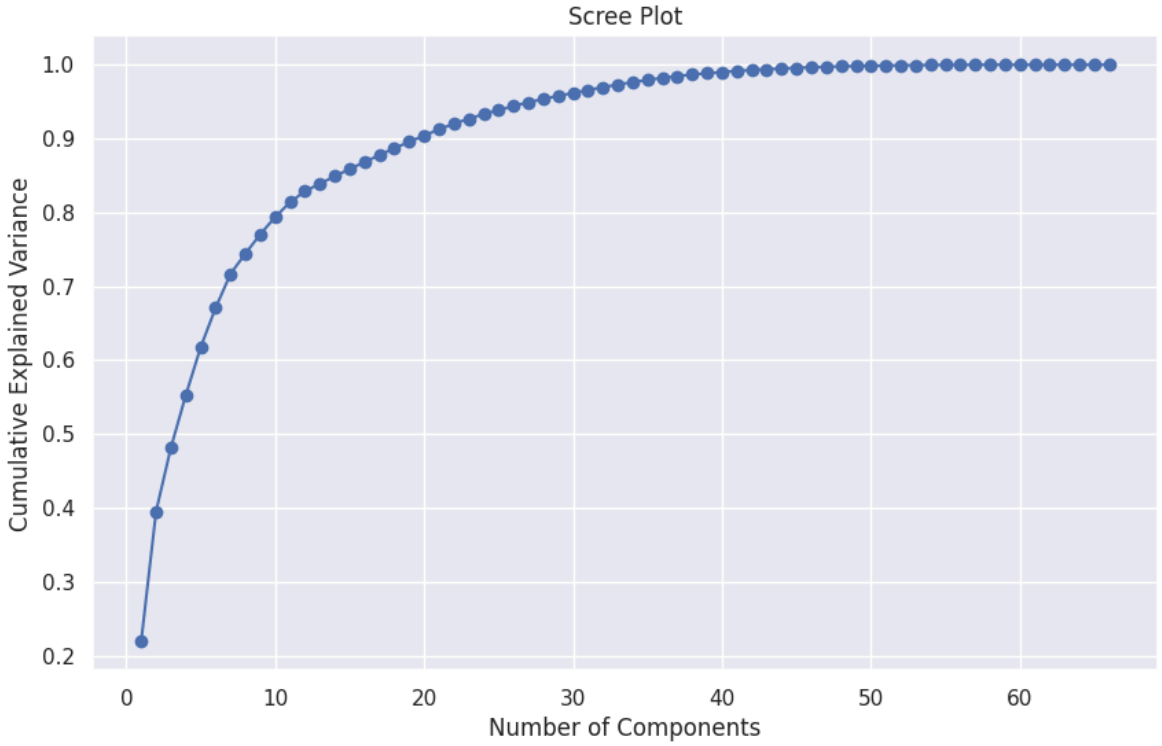

Figure 9: PCA analysis

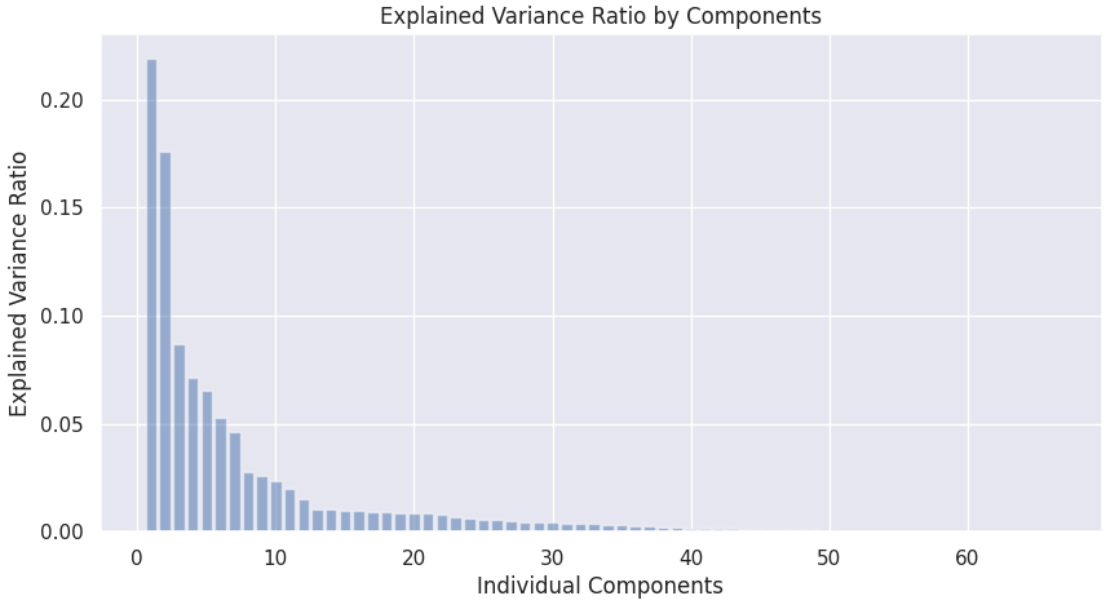

Figure 10: Variance ratio by components

- Supervised algorithms

Logistic regression

The images below showcase several parameters to emphasize the differences between the two models: logistic regression on the original data vs. logistic regression using the dimension reduction algorithm, PCA.

The Accuracy parameter is 98% for the PCA model compared to 97% for the original dataset. Additionally, the computation time is significantly reduced with PCA, taking only 0.21 seconds versus 0.79 seconds for the original dataset.

```
Logistic Regression on PCA data Accuracy: 0.9798628963153385
Logistic Regression computation time: 0.2100 seconds
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.96      0.97      4217
           1       0.98      0.99      0.99      9787

    accuracy                           0.98     14004
   macro avg       0.98      0.97      0.98     14004
weighted avg       0.98      0.98      0.98     14004
```

Figure 11: Logistic regression on PCA

```
Logistic Regression on Original Data Accuracy: 0.975078548986004
Logistic Regression on Original Data time: 0.7984 seconds

                    Logistic Regression:
                    Accuracy: 0.9751
                    Precision: 0.9777
                    Recall: 0.9868
                    F1-score: 0.9823
```

Figure 12: Logistic regression on original data

As we can see from the comparison of the two confusion matrices, the model on original data correctly classified 3997 instances of class 0 and 9658 instances of class 1, while misclassifying 220 instances of class 0 and 129 instances of class 1.

The model with PCA applied shows a slight performance improvement, correctly classifying 4041 instances of class 0 and 9681 instances of class 1, while misclassifying 176 instances of class 0 and 106 instances of class 1.

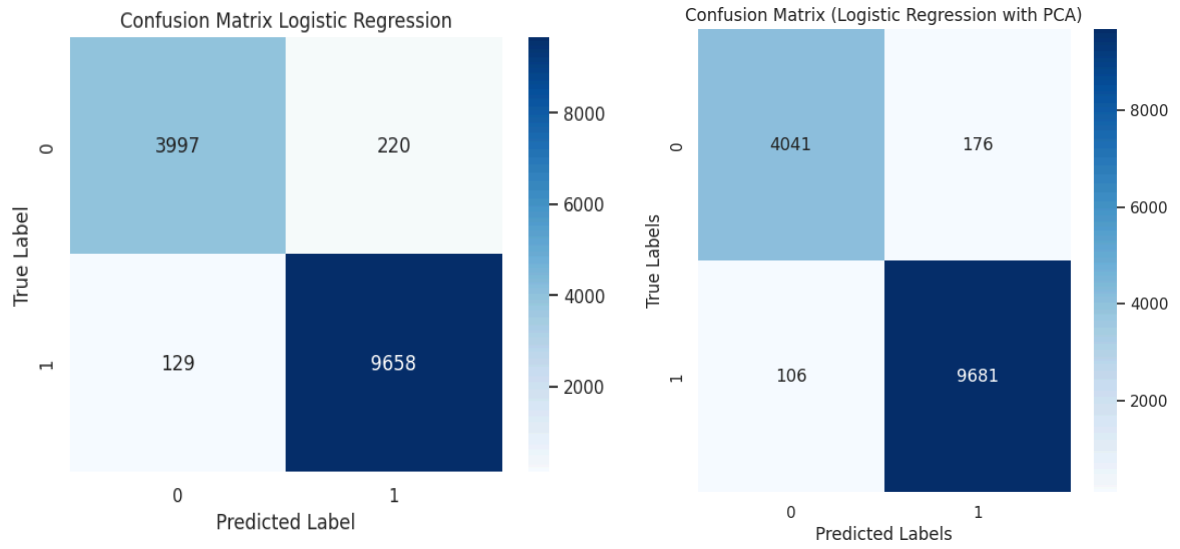

Figure 13: Comparison of logistic regression confusion matrix without and with PCA

Random Forest

Accuracy, precision, recall, and F1-score for the original dataset are slightly higher than those for the PCA model. However, the computation time for the PCA model is significantly lower at 0.45 seconds compared to 9.87 seconds for the original dataset.

```
Random Forest on PCA data Accuracy: 0.9731505284204514
Random Forest computation time: 0.4459 seconds
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.94      0.95      4217
           1       0.97      0.99      0.98      9787

    accuracy                           0.97     14004
   macro avg       0.97      0.96      0.97     14004
weighted avg       0.97      0.97      0.97     14004
```

Figure 14: Random forest with PCA

```
Random Forest on Original Data Accuracy: 0.989502999143102
Random Forest on Original Data time: 9.8674 seconds
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      4217
           1       0.99      1.00      0.99      9787

    accuracy                           0.99     14004
   macro avg       0.99      0.99      0.99     14004
weighted avg       0.99      0.99      0.99     14004
```

Figure 15: Random forest on original data

As we can see from the comparison of the two confusion matrices, the model on original data correctly classified 4111 instances of class 0 and 9746 instances of class 1, while misclassifying 106 instances of class 0 and 41 instances of class 1.

The model with PCA shows a slower performance, correctly classifying 3950 instances of class 0 and 9678 instances of class 1, while misclassifying 267 instances of class 0 and 109 instances of class 1.

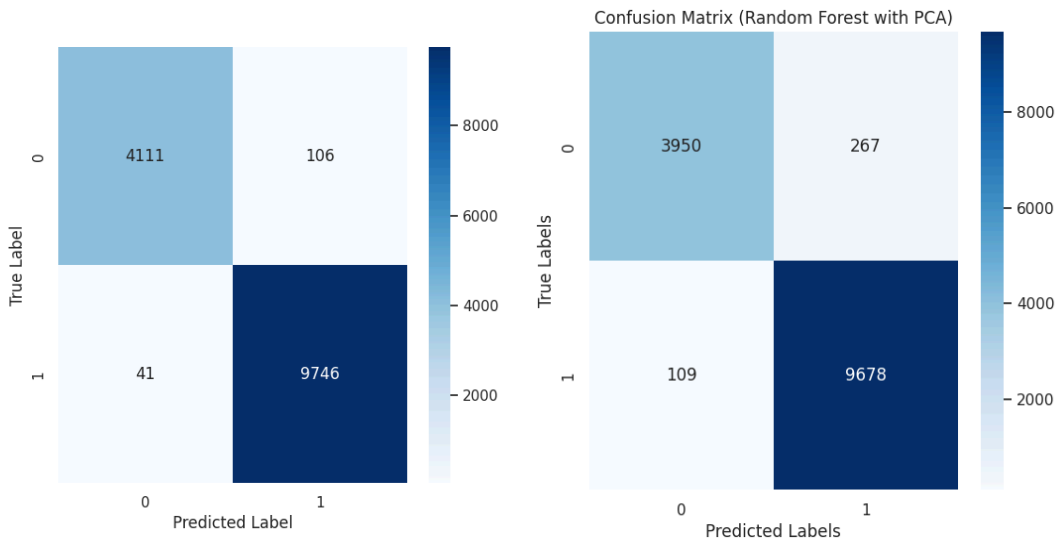

Figure 16: Comparison of Random forest confusion matrix without and with PCA

GBM

A few metrics, including accuracy, recall, precision, and F1-score, for the GBM model with PCA and the GBM model with original data. The second model, without PCA, achieves a higher accuracy of 98.97% compared to 97.42% for the model with PCA. However, it also takes significantly longer to compute, with a runtime of 33.44 seconds versus 11.10 seconds for the model with PCA.

```
GBM on PCA data Accuracy: 0.974150242787775
GBM computation time: 11.0976 seconds
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.95      0.96      4217
           1       0.98      0.99      0.98      9787

    accuracy                           0.97     14004
   macro avg       0.97      0.97      0.97     14004
weighted avg       0.97      0.97      0.97     14004
```

Figure 17: GBM on PCA data

```
GBM:
Accuracy: 0.9897172236503856
Confusion Matrix:
 [[4115  102]
 [  42 9745]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.98      4217
           1       0.99      1.00      0.99      9787

    accuracy                           0.99     14004
   macro avg       0.99      0.99      0.99     14004
weighted avg       0.99      0.99      0.99     14004

GBM data time: 33.4447 seconds
```

Figure 18: GBM on original data

From comparing the two confusion matrices, we can conclude that the initial model accurately classified 4115 instances of class 0 and 9745 instances of class 1, with misclassifications of 102 instances of class 0 and 42 instances of class 1. The second model demonstrates superior performance across all metrics.

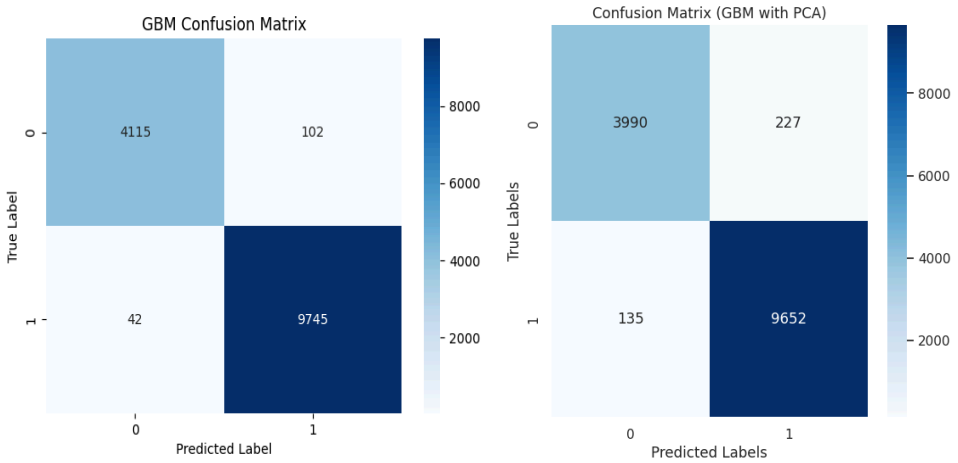

Figure 18: Comparison of GBM confusion matrix without and with PCA

SVM

The model using the original data has slightly better accuracy, precision, recall, and F1-scores, but it requires significantly more computation time 53.01 seconds compared to the PCA model 20.44 seconds.

The SVM model with PCA provides a good balance between performance and computation time, making it a viable option when computational efficiency is a priority.

```
SVM on PCA data Accuracy: 0.980719794344473
SVM computation time: 20.4366 seconds
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.96      0.97      4217
           1       0.98      0.99      0.99      9787

    accuracy                           0.98     14004
   macro avg       0.98      0.98      0.98     14004
weighted avg       0.98      0.98      0.98     14004
```

Figure 19: SVM on PCA

```
SVM
Accuracy: 0.9836475292773493
Confusion Matrix:
 [[4071  146]
 [  83 9704]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.97      0.97      4217
           1       0.99      0.99      0.99      9787

    accuracy                           0.98     14004
   macro avg       0.98      0.98      0.98     14004
weighted avg       0.98      0.98      0.98     14004

SVM data time: 53.0161 seconds
```

Figure 20: SVM on original data

In conclusion, both models perform well, but the choice between them depends on the specific needs for accuracy versus computation time.

The confusion matrix comparison provides a detailed breakdown of model performance, highlighting areas of success and errors in classification.

For instance, in the confusion matrix for the SVM model on original data, 9704 instances are correctly predicted as class 1 and 4071 as class 0, while 146 instances are misclassified as class 1 when they are actually class 0, and 83 instances are misclassified as class 0 when they are actually class 1.

A visual comparison shows that the SVM algorithm with PCA outperforms the SVM model on original data in terms of classification accuracy.

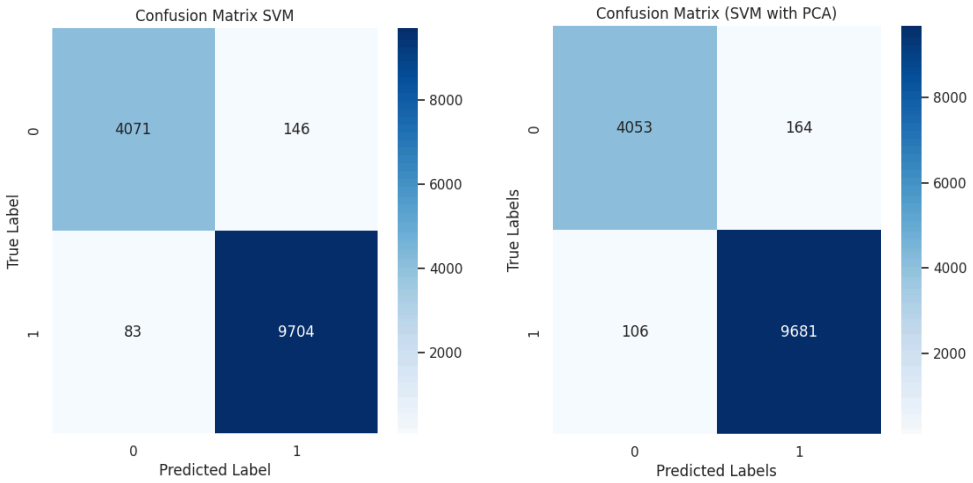

Figure 21: Comparison of SVM confusion matrix without and with PCA

LDA (Linear Discriminant Analysis) - is a dimension reduction technique that transforms the feature space into a lower-dimensional space. After using LDA, the reduced features can be used as input for a classification algorithm.

The performance of the LDA transformed data can be evaluated by using the Confusion Matrix and it helps in order to understand how the classifier performs in the reduced feature space.

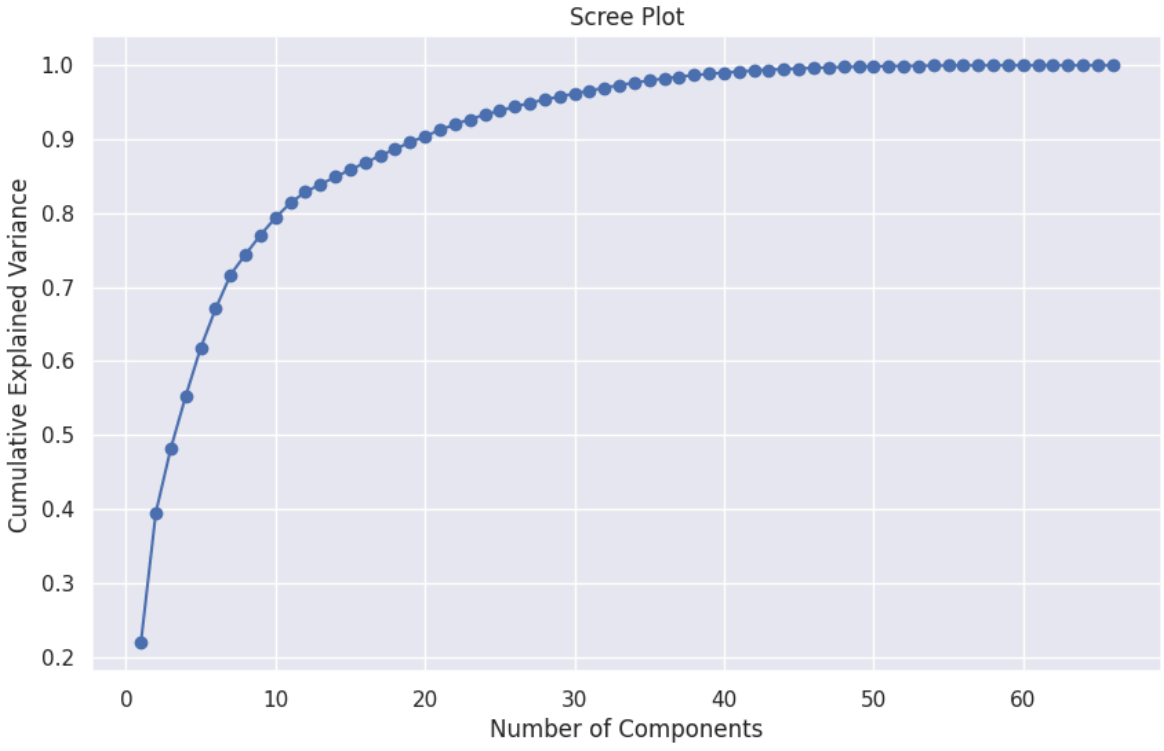

Figure 22: LDA analysis

Confusion matrix is a performance measurement tool for classification models, providing a detailed account of the classifier's predictions vs. the actual labels and showing the number of correct and incorrect predictions for each class.

- Supervised algorithms

Logistic regression

The accuracy of the model on LDA is slightly slower than the value on the original data, 96% compared with 97%, but the computation time is faster from 0.20 seconds to 1.13 seconds.

```
Logistic Regression on LDA data Accuracy: 0.9626535275635533
LDA computation time: 1.5834 seconds
Logistic Regression computation time: 0.2031 seconds
```

Figure 23: Logistic regression with LDA

```
Logistic Regression on Original Data Accuracy: 0.975078548986004
Logistic Regression on Original Data time: 1.1331 seconds

                    Logistic Regression:
                    Accuracy: 0.9751
                    Precision: 0.9777
                    Recall: 0.9868
                    F1-score: 0.9823
```

Figure 24: Logistic regression on original data

From comparing the two confusion matrices, we can conclude that the initial model accurately classified 3997 instances of class 0 and 9658 instances of class 1, with misclassifications of 220 instances of class 0 and 129 instances of class 1.

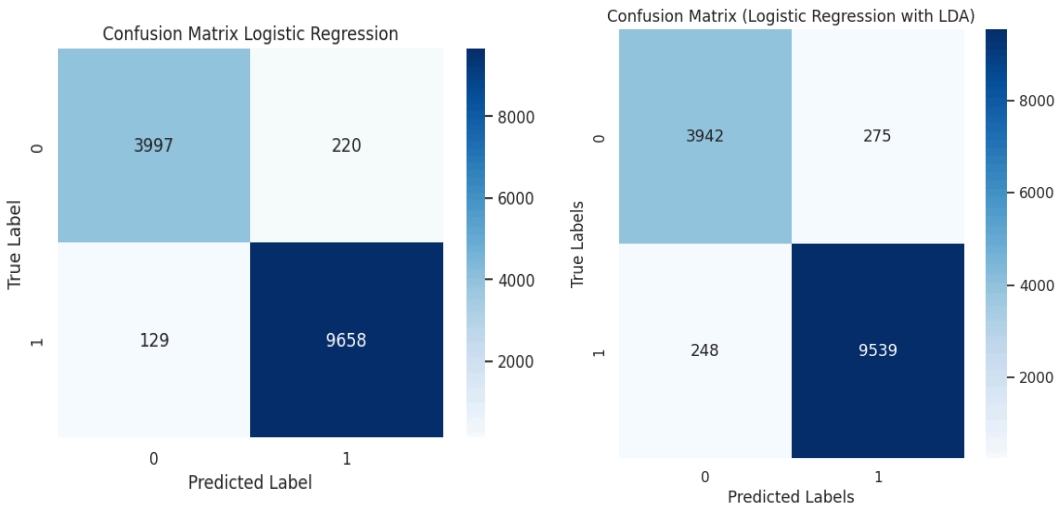

Figure 25: Comparison of Logistic regression confusion matrix without and with LDA

Logistic regression Grid Search

We perform the logistic regression with Grid Search and the results are presented below.

An accuracy of 98% indicates that the logistic regression model predicted correctly a significant number of instances. The Grid search optimization has identified that a low regularization, that means a high 'C' value and the 'liblinear' solver are the optimal hyperparameters, and the process to find these parameters took 51 seconds.

```
Logistic Regression on LDA data (Grid Search) Accuracy: 0.9837903456155385
Best Parameters: {'C': 10000, 'solver': 'liblinear'}
Grid Search for Logistic Regression time: 51.0625 seconds
```

```
Logistic Regression after gridsearch:
Accuracy: 0.9838
Precision: 0.9838
Recall: 0.9932
F1-score: 0.9885
```

Figure 26: Metrics on LR with Grid Search

From the confusion matrix we can conclude that has been correctly classified 9720 instances of class 0 and 4057 instances of class 1, but misclassified 160 of class 0 and 67 of class 1.

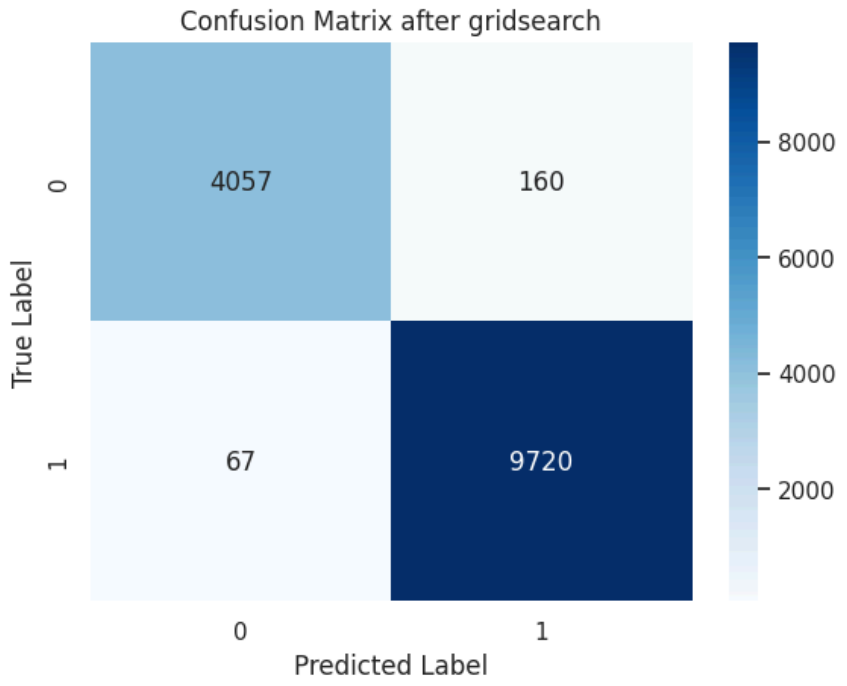

Figure 27: Confusion matrix on LR

Random Forest

The model trained on the original data outperforms the one trained with LDA in terms of accuracy and computation time. Both models have a high accuracy, which means it correctly classified about 97% compared with 98% of the instances in the dataset.

```
Random Forest on LDA data Accuracy: 0.9617252213653242
Random Forest computation time: 25.0022 seconds
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.94      0.94      4217
           1       0.97      0.97      0.97      9787

    accuracy                           0.96     14004
   macro avg       0.95      0.95      0.95     14004
weighted avg       0.96      0.96      0.96     14004
```

Figure 28: RF on LDA

```
Random Forest on Original Data Accuracy: 0.989502999143102
Random Forest on Original Data time: 9.8674 seconds
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      4217
           1       0.99      1.00      0.99      9787

    accuracy                           0.99     14004
   macro avg       0.99      0.99      0.99     14004
weighted avg       0.99      0.99      0.99     14004
```

Figure 29: RF on original data

From comparing the two confusion matrices, we can conclude that the initial model accurately classified 4111 instances of class 0 and 9746 instances of class 1, with misclassifications of 106 instances of class 0 and 41 instances of class 1.

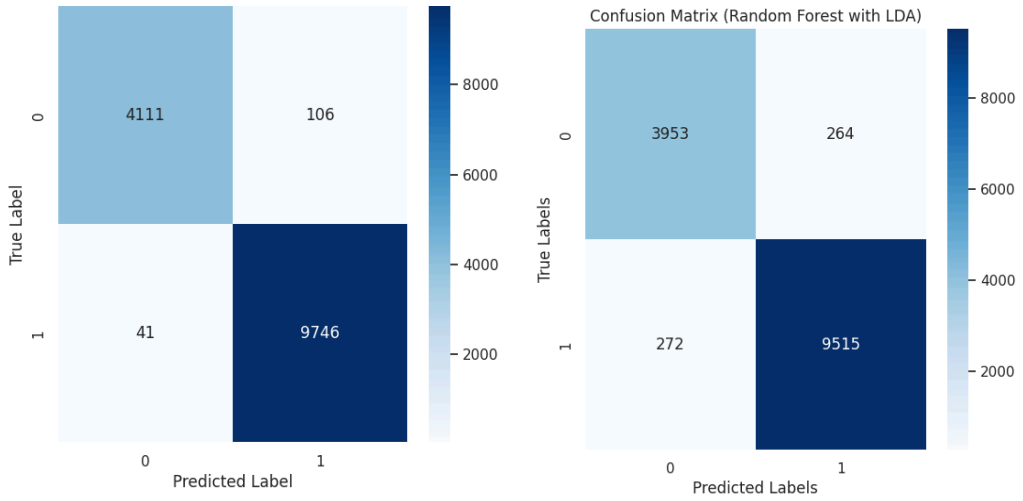

Figure 30: Comparison of Random forest confusion matrix without and with LDA

Grid Search Random Forest

The grid search optimization took 164.96 seconds, that is approximately 27 min, having an accuracy of 98%.

```
Best Random Forest parameters: {'max_depth': 10, 'min_samples_split': 5, 'n_estimators': 300}
Grid Search for Random Forest time: 1635.9553 seconds
Random Forest Grid Search Accuracy: 0.9850756926592402
```

```
Random Forest after grisearch:
Accuracy: 0.9851
Precision: 0.9832
Recall: 0.9957
F1-score: 0.9894
```

Figure 31: Random forest on Grid Search

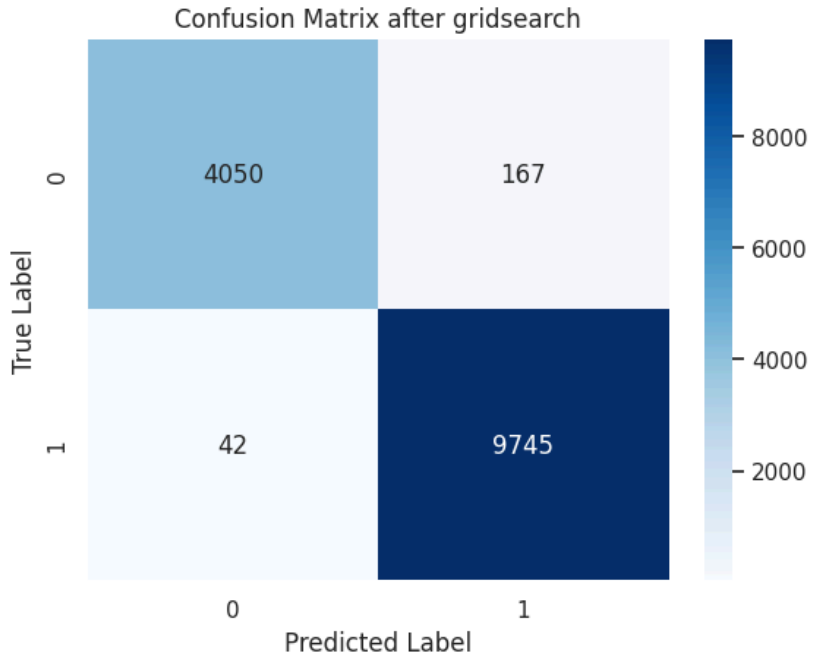

Figure 32: Confusion matrix after Grid search

GBM

The GBM model on the original data performs better with an accuracy of 98% compared with 96% for the GBM model on the LDA. Also the model trained on original data generalizes better and has a better predictive power on all metrics, precision, recall F1-score.

```
GBM on LDA data Accuracy: 0.9607255069980005
GBM computation time: 11.0976 seconds
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.94      0.94      4217
           1       0.97      0.97      0.97      9787

    accuracy                           0.96     14004
   macro avg       0.95      0.95      0.95     14004
weighted avg       0.96      0.96      0.96     14004
```

Figure 33: GBM on LDA

```
GBM:
Accuracy: 0.9897172236503856
Confusion Matrix:
 [[4115  102]
 [  42 9745]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.98      4217
           1       0.99      1.00      0.99      9787

    accuracy                           0.99     14004
   macro avg       0.99      0.99      0.99     14004
weighted avg       0.99      0.99      0.99     14004

GBM data time: 33.4447 seconds
```

Figure 34: GBM on original data

The GBM on LDA has higher misclassifications: 260 instances of class 0 and 270 instances of class 1, compared with GBM on original data, 102 instances of class 0 and 42 on class 1.

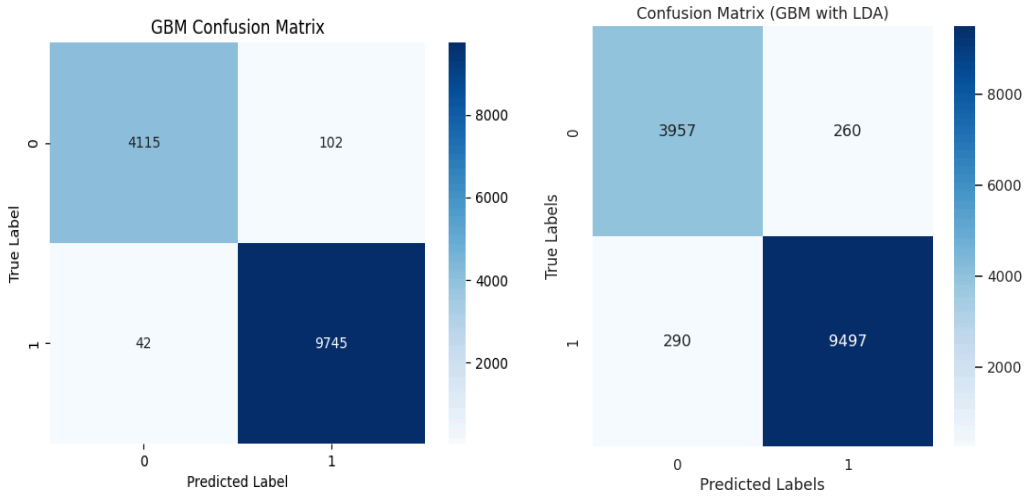

Figure 35: Comparison of GBM confusion matrix without and with LDA

SVM

As we can see the model on original data has a higher accuracy 98% compared with 96%. The SVM classifier achieves a higher precision, recall and F1-scores for both models, but with a slight increase of metrics on the original data.

```
SVM with LDA:
Accuracy: 0.9616538131962297
Confusion Matrix:
 [[3952  265]
 [ 272 9515]]
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.94      0.94      4217
           1       0.97      0.97      0.97      9787

    accuracy                           0.96     14004
   macro avg       0.95      0.95      0.95     14004
weighted avg       0.96      0.96      0.96     14004

SVM data time: 83.5354 seconds
```

Figure 36: SVM on LDA

```
SVM
Accuracy: 0.9836475292773493
Confusion Matrix:
 [[4071  146]
 [  83 9704]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.97      0.97      4217
           1       0.99      0.99      0.99      9787

    accuracy                           0.98     14004
   macro avg       0.98      0.98      0.98     14004
weighted avg       0.98      0.98      0.98     14004

SVM data time: 53.0161 seconds
```

Figure 37: SVM on original data

The SVM on LDA has higher misclassifications: 265 instances of class 0 and 272 instances of class 1, compared with SVM on original data, 146 instances of class 0 and 83 on class 1.

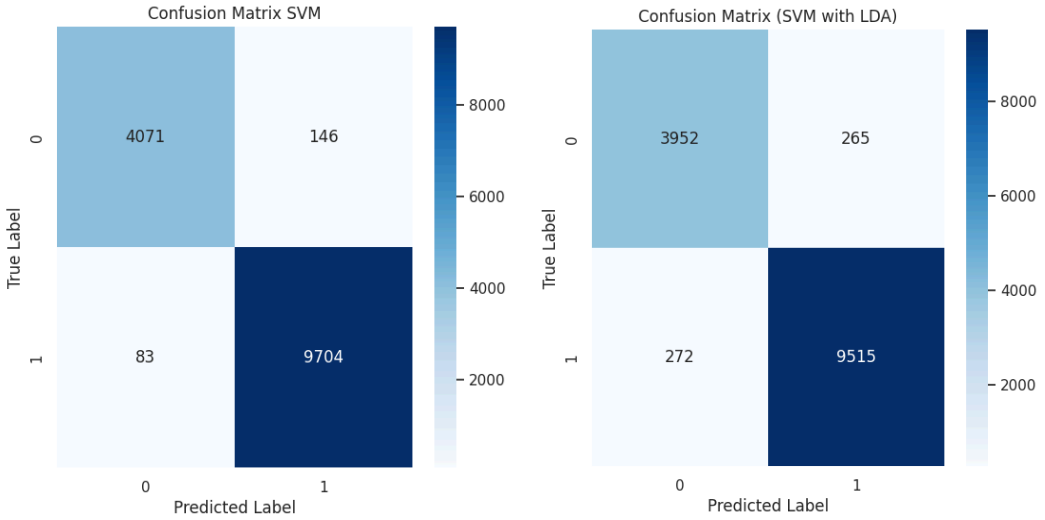

Figure 38: Comparison of SVM confusion matrix without and with LDA

Overall the SVM on original data demonstrates superior performance in terms of accuracy, precision, recall, F1-score and computational efficiency compared to the SVM on LDA.

TruncatedSVD (Truncated Singular Value Decomposition) - is a dimension reduction technique based on matrix factorization preserving as much variance as possible.

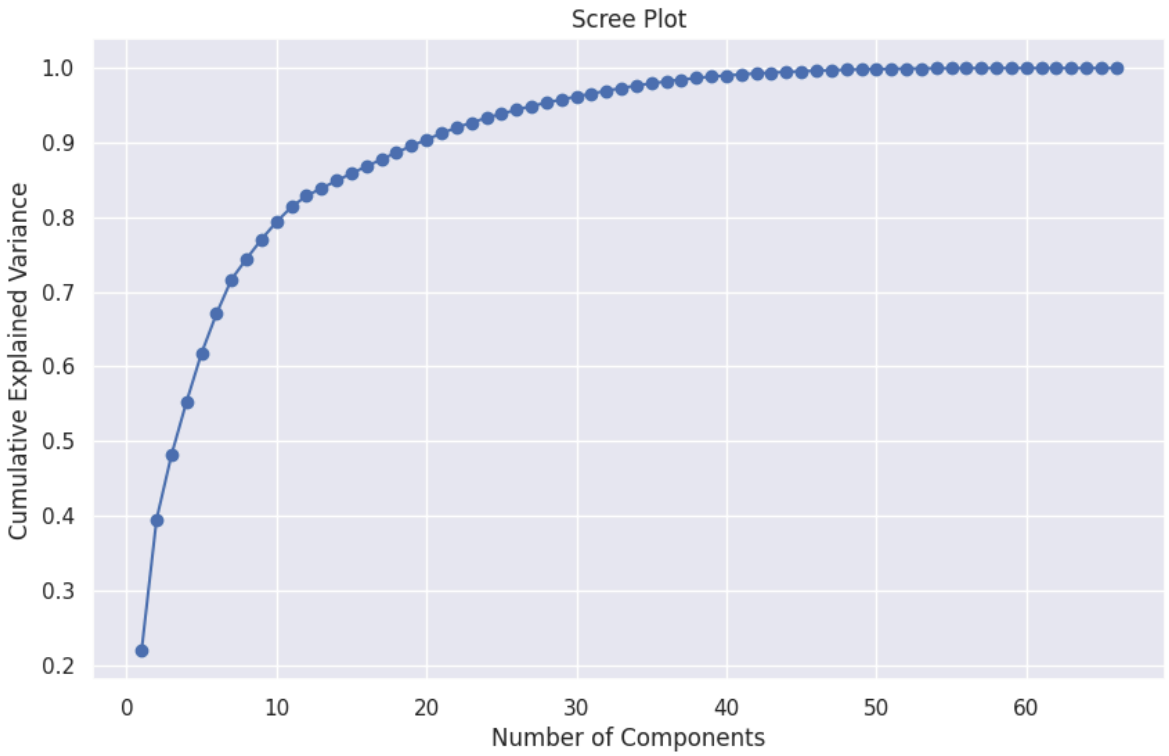

Figure 39: TruncatedSVD

- Supervised algorithms

Logistic regression

The accuracy model on the PCA is slightly higher than the one on the original dataset 98% compared to 0.97%, also the computation time is slower 0.45 seconds compared to 1.13 seconds.

The PCA model leads to a slight improvement in accuracy and a significant reduction in computation time. Precision, recall and F1-scores are generally better or comparable on the PCA model, indicating that the dimension reduction did not compromise the model's ability to correctly classify the instances.

```
Logistic Regression computation time: 0.4459 seconds
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.96      0.97      4217
           1       0.98      0.99      0.99      9787

    accuracy                           0.98     14004
   macro avg       0.98      0.97      0.98     14004
weighted avg       0.98      0.98      0.98     14004
```

Figure 40: Logistic regression on TruncatedSVD

```
Logistic Regression on Original Data Accuracy: 0.975078548986004
Logistic Regression on Original Data time: 1.1331 seconds

                        Logistic Regression:
                        Accuracy: 0.9751
                        Precision: 0.9777
                        Recall: 0.9868
                        F1-score: 0.9823
```

Figure 41: Logistic regression on original data

Random Forest

A better performance can be seen on the model on original data, even if for the rest of models the computation time was better, this time is higher on the model with TruncatedSVD, 72.56 seconds compared with 9.87 seconds.

```
Random Forest on Truncated SVD data Accuracy: 0.9716509568694659
Random Forest computation time: 72.5590 seconds
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.93      0.95      4217
           1       0.97      0.99      0.98      9787

    accuracy                           0.97     14004
   macro avg       0.97      0.96      0.97     14004
weighted avg       0.97      0.97      0.97     14004
```

Figure 42: Random Forest on TruncatedSVD

```
Random Forest on Original Data Accuracy: 0.989502999143102
Random Forest on Original Data time: 9.8674 seconds
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      4217
           1       0.99      1.00      0.99      9787

    accuracy                           0.99     14004
   macro avg       0.99      0.99      0.99     14004
weighted avg       0.99      0.99      0.99     14004
```

Figure 43: Random Forest on original data

The Random forest on TruncatedSVD has higher misclassifications: 291 instances of class 0 and 106 instances of class 1, compared with the model on original data, 106 instances of class 0 and 41 on class 1.

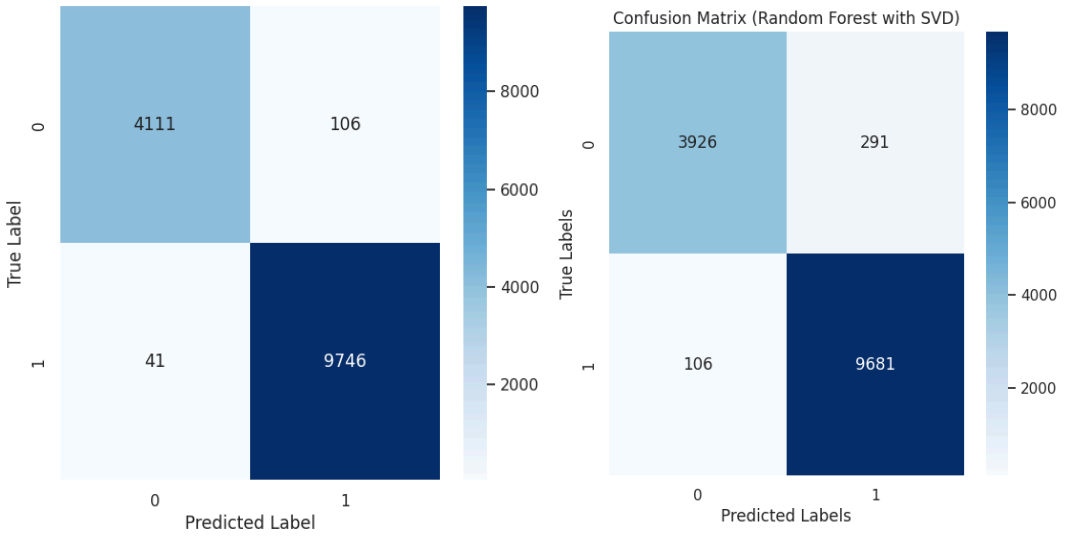

Figure 44: Comparison of Logistic regression confusion matrix without and with TruncatedSVD

GBM

The model on original data has a slightly better performance on metrics and also on computation time, because the GBM on TruncateSVD takes almost twice as long 65.75 seconds compared with 33.44 on original data.

The performance on class 1 is very high on both models, but class 0 shows a decrease in recall 94% with TruncatedSVD compared with original 98%.

```
GBM on Truncated SVD data Accuracy: 0.9727934875749786
GBM computation time: 65.7581 seconds
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.94      0.95      4217
           1       0.98      0.99      0.98      9787

    accuracy                           0.97     14004
   macro avg       0.97      0.96      0.97     14004
weighted avg       0.97      0.97      0.97     14004
```

Figure 45: GBM on TruncatedSVD

```
GBM:
Accuracy: 0.9897172236503856
Confusion Matrix:
 [[4115  102]
 [  42 9745]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.98      4217
           1       0.99      1.00      0.99      9787

    accuracy                           0.99     14004
   macro avg       0.99      0.99      0.99     14004
weighted avg       0.99      0.99      0.99     14004

GBM data time: 33.4447 seconds
```

Figure 46: GBM on original data

The GBM on TruncatedSVDhas higher misclassifications: 245 instances of class 1 and 136 instances of class 0, compared with GBM on original data, 102 instances of class 1 and 42 on class 0.

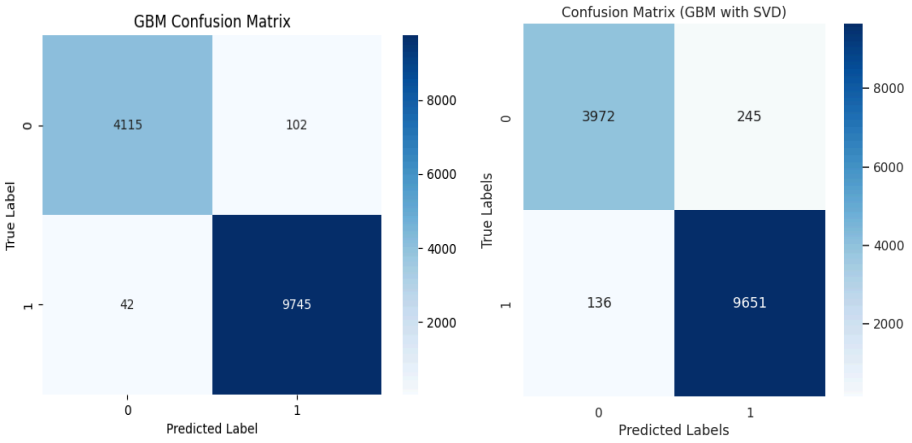

Figure 47: Comparison of GBM confusion matrix without and with TruncatedSVD

SVM

The SVM without TruncatedSVD has a higher accuracy 98.34% compared with 98.06%, but the difference it's not that significant. Also the model without the dimension reduction applied is slower because it takes 53.01 seconds compared to 18.60 seconds.

Conclusion is that the model without TruncatedSVD outperforms the SVM with TruncatedSVDin terms of accuracy, precision, recall and F1-score.

```
SVM on Truncated SVD data Accuracy: 0.9806483861753784
SVM computation time: 18.5957 seconds
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97      4217
           1       0.98      0.99      0.99      9787

    accuracy                           0.98     14004
   macro avg       0.98      0.97      0.98     14004
weighted avg       0.98      0.98      0.98     14004
```

Figure 48: SVM on TruncatedSVD

```
SVM
Accuracy: 0.9836475292773493
Confusion Matrix:
 [[4071  146]
 [  83 9704]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.97      0.97      4217
           1       0.99      0.99      0.99      9787

    accuracy                           0.98     14004
   macro avg       0.98      0.98      0.98     14004
weighted avg       0.98      0.98      0.98     14004

SVM data time: 53.0161 seconds
```

Figure 49: SVM on original data

The SVM without TruncatedSVD has fewer misclassifications: 83 instances of class 0 and 146 instances of class 1, compared with SVM with TruncatedSVD, 170 instances of class 1 and 101 on class 0.

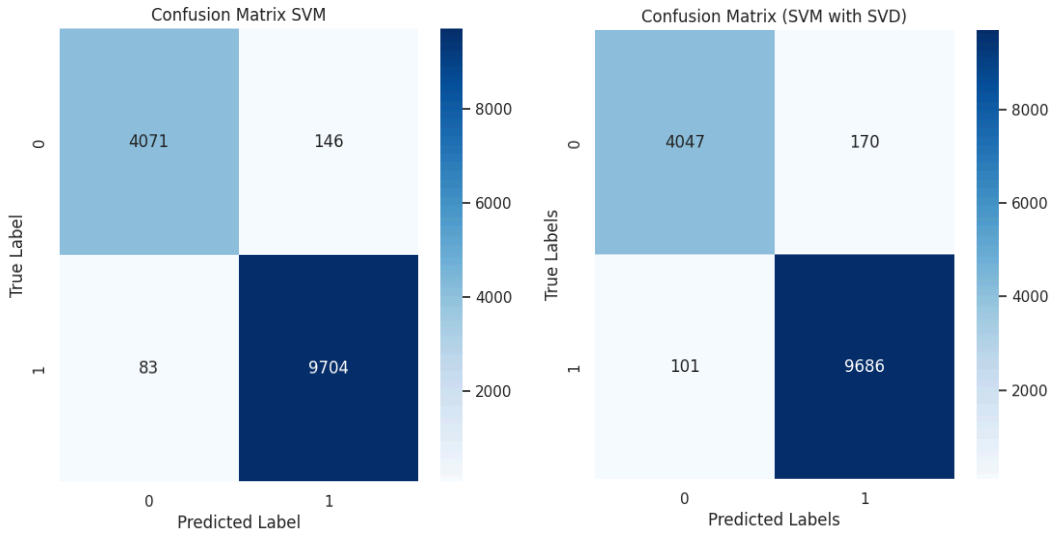

Figure 50: Comparison of SVM confusion matrix without and with TruncatedSVD

What are the best experimental models

As we can see in section **Methods applied and result** there were a few comparisons between models trained with the algorithms of dimension reduction and not.

Starting with the Logistic regression comparison, the model without using the LDA algorithm has a slightly higher accuracy, precision, recall and F1-score, but the model with LDA is significantly faster, and also the GBM algorithm follows the same pattern.

The best model depends on the specific requirements and constraints of what we want to do, and what is a crucial parameter either performance, measured by metrics or computation time.

The Random Forest with LDA follows the same trend as the Logistic regression, in terms of a slightly higher accuracy, precision, recall and F1-score for models on original data, but the computation time is higher than the model trained with dimension reduction for SVM with LDA.

The GBM and SVM models also follow the same pattern, the metrics such as accuracy, precision, recall and F1-score has a better performance for the models without dimension reduction applied, but a training time higher than the ones with LDA and PCA.

GBM and Random forest algorithms have a higher computation time on TruncatedSVD, dimension reduction in comparison with the models on original data.

Optimization utilizing the Grid Search algorithm has been applied to both Random forest and GBM.

Conclusion

Choosing the best model depends on what we want to do. Performance and time computation are the 2 of the important features to take in consideration.

The models trained without dimension reduction perform better than the ones without, because of the information loss after the dimension reduction, model complexity. For the models that have been applied an optimization after the dimension reduction it can be observed a slight performance increase, Random Forest and GBM.

Better performance is obtained for the algorithms without dimension reduction, but using the dimension reduction techniques we can significantly improve the computation time.

The table below provides a summary of all discussed algorithms highlighting their accuracy and computation time characteristics. We can see that all models have an increased accuracy and the computation time depends on each model by the chosen algorithm.

Random forest and SVM on LDA have a computation time slower than the model on original data, but also the random forest and GBM on TruncatedSVD are the other 2 algorithms that have a slower computation time compared with the model on original data.

Dimension reduction algorithm	Optimal number of components
PCA	20
LDA	1
Truncated SVD	20

Table 1: Optimal number of components for each dimension reduction method

Algorithm	Logistic Regression	Random Forest	GBM	SVM
Accuracy on original data	98%	99%	99%	98%
Computation time on original data	1.13 seconds	9.88 seconds	33.44 seconds	53.01 seconds

Accuracy with PCA	98%	97%	97%	98%
Computation time with PCA	0.21 seconds	0.45 seconds	11.09 seconds	20.43 seconds
Accuracy with LDA	96%	96%	96%	96%
Computation time with LDA	0.20 seconds	25 seconds	11.09 seconds	83.54 seconds
Accuracy with TruncatedSVD	98%	97%	97%	98%
Computation time with TruncatedSVD	0.45 seconds	72.56 seconds	65.76 seconds	18.60 seconds

Table 2: Comparison table between algorithms accuracy and computation time metrics