Loan Approval Prediction

Student:
Lînaru Petra Rozalia

Cuprins

1 Problem statement

1.1 Domain

The financial sector, particularly the loan approval process, is critical for banks and financial institutions. Lending decisions often require a careful assessment of applicants' financial profiles to determine their likelihood of defaulting on a loan. Approving loans for customers with higher chances of defaulting can lead to financial losses, whereas rejecting loans for eligible applicants can result in lost opportunities.

1.2 Requirements

The goal of this project is to develop a binary classification model that predicts whether an applicant will default on a loan based on their financial attributes. Moreover, the financial attributes will be split in two datasets : a datset with a small number of records and a dataset with a greater number of records.

2 Dataset description

UniversalBank - First Dataset

The UniversalBank dataset contains customer information for a bank, aimed at predicting the likelihood of a customer accepting a personal loan offer. The dataset includes **14 columns**, described as follows:

- **ID**: Unique identifier for each customer.
- **Age**: Age of the customer (in years).
- **Experience**: Number of years of work experience.
- **Income**: Annual income of the customer (in $).
- **ZIP Code**: Customer's residential ZIP code.
- **Family**: Number of family members (e.g., 1, 2, 3, or 4).
- **CCAvg**: Average monthly credit card spending (in $).
- **Education**: Level of education (1: Undergrad, 2: Graduate, 3: Advanced/Professional).
- **Mortgage**: Value of the customer's mortgage (in $).
- **Personal Loan**: Target variable indicating whether a customer accepted the loan offer (`1`: Accepted, `0`: Not Accepted).
- **Securities Account**: Binary indicator (`1`: Has a securities account, `0`: Does not have one).
- **CD Account**: Binary indicator (`1`: Has a certificate of deposit account, `0`: Does not have one).
- **Online**: Binary indicator (`1`: Uses online banking services, `0`: Does not use online banking).
- **CreditCard**: Binary indicator (`1`: Uses a credit card issued by the bank, `0`: Does not use the bank's credit card).

Target Variable Distribution: - `0 (Not Accepted)`: 4520 observations. - `1 (Accepted)`: 480 observations.

The dataset contains **no duplicate rows**.

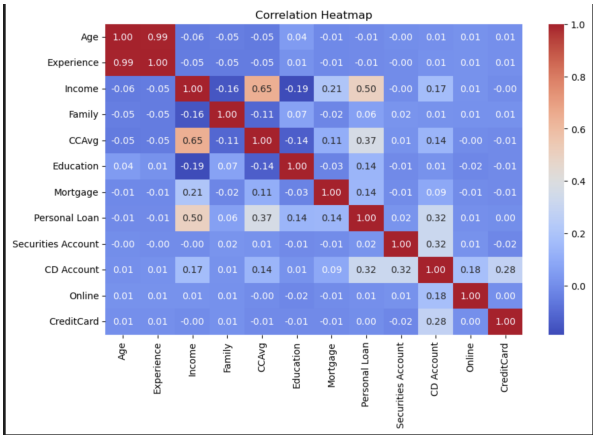

Figure 1: Caption

Correlations Between Columns

As we can see, there are correlations between the following columns:

- **Income** and **CCAvg** (Strong), **Mortgage** (Moderate), **Personal Loan** (Moderate), and **CD Account** (Weak).
- **Mortgage** and **CCAvg**.
- **CCAvg** and **Income** (Strong), **Personal Loan** (Moderate).
- **Securities Account** and **CD Account** (Moderate).
- **CD Account** and **Income** (Weak), **CCAvg** (Weak), **Personal Loan** (Moderate), **Securities Account** (Moderate).
- **Credit Card** and **CD Account** (Weak).

Application Record - Second Dataset (FAILED)

First try with two datasets different dataset At first, the second datset picked was in fact a dataset which had data distributed across two csv files. The first csv had about 438.000 records and the second one had 5000. Banking information relevant to loan approval was stored in one csv and the target variable with other two features were stored in another csv. The csv files could not be merged because there were inconsistencies in the shapes of the files. Tried approaches were:

- expanding the second dataset to match the first dataset, and then apply interpolation to fill the missing data
- trying to build a target column based on the existing columns by analysing a loan approval guideline and then implementing the "risk factors"
- applying k means on the original dataset to split the data into two clusters and then assign the clusters as target values

They all failed successfully

LoanDefault - Third Dataset

The third dataset contains 255347 records and 18 columns.

- **LoanID:** A unique identifier for each loan.
- **Age:** The age of the loan applicant.

- **Income:** The annual income of the applicant ($ currency).
- **LoanAmount:** The amount of money requested for the loan ($ currency).
- **CreditScore:** The credit score of the applicant.
- **MonthsEmployed:** The number of months the applicant has been employed.
- **NumCreditLines:** The number of active credit lines the applicant has.
- **InterestRate:** The interest rate applied to the loan (%).
- **LoanTerm:** The duration of the loan in months.
- **DTIRatio:** Debt-to-Income ratio, representing the percentage of the applicant's income that goes towards debt repayment.
- **Education:** The highest level of education attained by the applicant. (e.g., "Bachelor's")
- **EmploymentType:** The type of employment the applicant has. (e.g., "Full-time")
- **MaritalStatus:** The applicant's current marital status. (e.g., "Divorced")
- **HasMortgage:** Indicates whether the applicant has an existing mortgage. ("Yes" or "No")
- **HasDependents:** Indicates whether the applicant has dependents. ("Yes" or "No")
- **LoanPurpose:** The purpose for which the loan is requested. (e.g., "Other")
- **HasCoSigner:** Indicates whether the loan has a co-signer. ("Yes" or "No")
- **LoanApproved:** Indicates whether the loan was approved. ("0" for not approved, "1" for approved)

3 Theoretical Foundations: Support Vector Machines (SVM)

Support Vector Machines (SVM) are supervised learning algorithms used for binary classification and regression tasks. SVM constructs a hyperplane that separates data into distinct classes while maximizing the margin between them. This section focuses on the mathematical foundations of **hard margin SVM** and **soft margin SVM**

1. Hard Margin SVM

Hard margin SVM is used when the data is linearly separable, meaning there exists a hyperplane that can perfectly classify the data. The goal is to maximize the margin (distance) between the hyperplane and the nearest data points (support vectors).

Optimization Problem:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to:} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \forall i, \end{aligned}$$

where:

- $\mathbf{w}$: Weight vector (normal to the hyperplane).
- b: Bias term.
- $\mathbf{x}_i$: Feature vector for the i-th sample.
- y_i: Class label for the i-th sample ($y_i \in \{-1, 1\}$).

The constraint ensures that all data points are correctly classified and lie outside the margin boundaries.

2. Soft Margin SVM

In real-world scenarios, data is often not perfectly separable. Soft margin SVM allows some misclassifications by introducing slack variables ξ_i to relax the constraints.

Optimization Problem:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to:} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \forall i, \\ & \xi_i \geq 0, \quad \forall i, \end{aligned}$$

where:

- ξ_i: Slack variables that quantify the degree of misclassification for the i-th sample.
- C: Regularization parameter that controls the trade-off between maximizing the margin and minimizing classification errors.

Soft margin SVM is particularly effective when the data contains noise or overlaps between classes.

Hard Margin vs. Soft Margin

Hard Margin: If the data is linearly separable, SVM finds a hyperplane that separates the classes without any errors (no misclassification).

Soft Margin: When the data is not perfectly separable or is noisy, SVM allows some misclassification (using slack variables) while still trying to maximize the margin. This approach is controlled by a regularization parameter (C) that balances the trade-off between margin size and misclassification.

4. The Kernel Trick

For datasets that are not linearly separable in their original feature space, SVM uses the kernel trick to map the data into a higher-dimensional space where it becomes linearly separable. This mapping is performed implicitly, avoiding the computational cost of explicitly computing the transformed features.

Mathematical Formulation: The decision function for SVM can be expressed as:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n}\alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right),$$

where:

- α_i: Lagrange multipliers obtained from the dual optimization problem.
- $K(\mathbf{x}_i, \mathbf{x})$: Kernel function that computes the similarity between two data points $\mathbf{x}_i$ and $\mathbf{x}$ in the higher-dimensional space.

Common Kernel Functions:

- **Linear Kernel:** $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$.
- **Polynomial Kernel:** $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j + 1)^d$, where d is the degree of the polynomial.
- **Radial Basis Function (RBF) Kernel:** $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$, where γ controls the influence of each support vector.

4 Design and implementation

4.1 SVM on the UniversalBank dataset

- **Load the dataset:** The very first step was to load the dataset and examine its contents. There were no duplicates or missing values in the dataset.

- **Insights:** A histogram for the distribution of numeric features was generated. Age and experience had quite similar distributions, while CCAvg, Income, and Mortgage followed a descending trend, as shown in the figure. Moreover, a bar chart was created to show the class distribution for the target variable. The chart indicated that the dataset was imbalanced.
- **Filter data:** In this step, irrelevant columns were dropped from the dataset.
- **First Try: HardMargin SVM, Gaussian:** For the first attempt, the features selected were Income and CCAvg, as they showed a strong correlation with the target variable highlighted in the correlation heatmap. The dataset was split into 80/20 train and test data. The standard scaler was used to scale the train and test datasets. The chosen kernel was the Gaussian one, with a gamma value set to 1. The data was non-linearly separable, as the plot showed: most of the points corresponding to the two classes were overlapping. This indicated that further transformations on the features were required.
- **Second Try: HardMargin SVM, Gaussian, Optimized:** In this attempt, instead of selecting two features relevant to the real-life domain, Principal Component Analysis (PCA) was used to reduce the data to 2D. Additionally, the data was log-transformed. To solve the imbalance, the SMOTE resampling technique was applied. The gamma value was lowered for a smoother decision boundary.
- **Third Try: HardMargin SVM, Gaussian, without PCA Optimization:** In this attempt, PCA was removed.
- **Fourth and Fifth Try: HardMargin SVM, Polynomial:** In these attempts, the polynomial kernel was used with 2nd and 3rd-degree polynomials. The 3rd-degree polynomial performed better than the 2nd-degree polynomial, but overall, the Gaussian kernel outperformed the polynomial kernels.
- **Sixth Try: SoftMargin SVM:** For this attempt, the data was scaled, log-transformed, and then SMOTE oversampling was applied.

4.2 SVM on the LoanDefault dataset

- **Load the dataset:** As in the previous design, the very first step was to load the dataset and examine its contents. There were no duplicates or missing values in the dataset. The target column was renamed to 'LoanApproved'.
- **Insights:** A correlation heatmap was generated in order to see the relationships between columns. Weak correlation was highlighted between features.
- **Preprocess data:** In this step, categorial columns were encoded using LabelEncoder.
- **Feature Engineering** Risk factors that influence loan are the employment status and income stability, as well as borrower's financial history and behaviour. Therefore, there were features added such as : IncomeToLoanRatio, CreditUtilization, MonthlyIncomeToRepaymentRatio and InterestRateToLoanAmountRatio. These new features could add more correlation and also give more insights in the loan approval process.
- **First Try: SoftMargin SVM on the second DataFrame, Gaussian:** For the first attempt, there was no resampling done. The data was scaled, then PCA was performed. There were many misclassified points due to the imbalance.
- **Second Try: SoftMargin SVM on the second DataFrame, Resampled** In this attempt, SMOTE resampling was added in order to solve imbalance.

5 Evaluation

5.1 SVM on UniversalBank

5.1.1 Gaussian, no optimizations

For the training set, the following evaluation metrics were obtained:

- **Accuracy**: 0.48 - The model correctly predicted the target class about 48% of the time on the training set.
- **Precision**: 0.16 - The precision is low, meaning that when the model predicts the positive class (loan approval), it is correct only 16% of the time. This could indicate that the model is making too many false positive predictions.
- **Recall**: 1.00 - The recall is perfect, which means that the model identified all the actual positive instances (approved loans). However, this may also suggest that the model is biased towards predicting the positive class.
- **F1 Score**: 0.27 - The F1 score is low due to the low precision, which balances the precision and recall. This suggests that while the model correctly identifies positive cases, it also makes a high number of false positive predictions.

For the testing set, the following evaluation metrics were obtained:

- **Accuracy**: 0.49 - The accuracy on the testing set is slightly better than on the training set, but it still indicates a performance that is less than ideal. The model generalizes slightly better to the testing data.
- **Precision**: 0.16 - The precision remains low on the testing set as well, indicating that when the model predicts approval, it's only correct 16% of the time. This suggests that the model struggles with correctly predicting the positive class.
- **Recall**: 1.00 - Again, the model achieves perfect recall, meaning it correctly identifies all positive instances, but may still be predicting a high number of false positives.
- **F1 Score**: 0.27 - The F1 score remains low on the testing set as well, reinforcing the imbalance between precision and recall.

5.1.2 Gaussian Optimization : First Try

PCA, SMOTE and log transformations

For the training set, the following evaluation metrics were obtained:

- **Accuracy**: 0.72 - The model correctly predicted the target class 72% of the time on the training set, indicating a relatively strong performance.
- **Precision**: 0.64 - The precision is moderate, meaning that when the model predicts the positive class (loan approval), it is correct 64% of the time. This indicates a reasonable ability to make positive predictions without many false positives.
- **Recall**: 1.00 - The recall is perfect, meaning that the model identified all actual positive instances (approved loans). This shows the model's ability to detect every positive case.
- **F1 Score**: 0.78 - The F1 score is solid, indicating a good balance between precision and recall. It suggests the model has both a high recall and reasonable precision, performing well overall.

For the testing set, the following evaluation metrics were obtained:

- **Accuracy**: 0.51 - The accuracy on the testing set is 51%, which indicates a noticeable drop from the training set accuracy, pointing to potential overfitting.
- **Precision**: 0.16 - The precision is low on the testing set, meaning that when the model predicts loan approval, it is correct only 16% of the time. This suggests that the model is over-predicting the positive class and making many false positive predictions.

- **Recall**: 1.00 - The recall is perfect, meaning the model still identifies all actual positive instances, which could be a result of the model favoring the positive class.
- **F1 Score**: 0.28 - The F1 score is low, indicating a poor balance between precision and recall, driven by the low precision.

5.1.3 Gaussian Optimization : Second Try

SMOTE, log transformations, gamma lowered For the training set, the following evaluation metrics were obtained:

- **Accuracy**: 0.80 - The model correctly predicted the target class 80% of the time on the training set, indicating a strong performance in identifying both classes.
- **Precision**: 0.72 - The precision is good, meaning that when the model predicts the positive class (loan approval), it is correct 72% of the time. This indicates the model has a relatively high confidence in its positive predictions.
- **Recall**: 1.00 - The recall is perfect, meaning that the model successfully identified all the actual positive instances (approved loans), ensuring no positive cases are missed.
- **F1 Score**: 0.84 - The F1 score is high, showing a good balance between precision and recall. The model demonstrates good performance with respect to both false positives and false negatives.

For the testing set, the following evaluation metrics were obtained:

- **Accuracy**: 0.67 - The accuracy on the testing set is 67%, which is lower than the training set, indicating some overfitting, but still reasonable performance.
- **Precision**: 0.22 - The precision on the testing set is relatively low, meaning that when the model predicts loan approval, it is correct only 22% of the time. This suggests a higher rate of false positives compared to the training set.
- **Recall**: 1.00 - As with the training set, the recall is perfect, meaning the model is still identifying all the actual positive instances, which might be due to the model favoring the positive class.
- **F1 Score**: 0.37 - The F1 score is lower, indicating an imbalance between precision and recall. The low precision leads to a poorer balance and suggests that the model may be over-predicting the positive class.

5.1.4 Polynomial, Degree:2

For the training set, the following evaluation metrics were obtained:

- **Accuracy:** 0.12 - The model correctly predicted the target class only 12% of the time on the training set, indicating poor performance.
- **Precision:** 0.05 - The precision is very low, meaning that when the model predicts the positive class (loan approval), it is correct only 5% of the time. This indicates a large number of false positives.
- **Recall:** 0.05 - The recall is also low, meaning the model is identifying only 5% of the actual positive instances. This suggests the model is failing to capture most of the true positive cases.
- **F1 Score:** 0.05 - The F1 score is extremely low due to both precision and recall being very low. This indicates an overall poor performance, as the model is unable to effectively identify the positive class while also producing many false positives.

For the testing set, the following evaluation metrics were obtained:

- **Accuracy:** 0.17 - The accuracy on the testing set is 17%, which is still very low and reflects the model's poor generalization to unseen data.

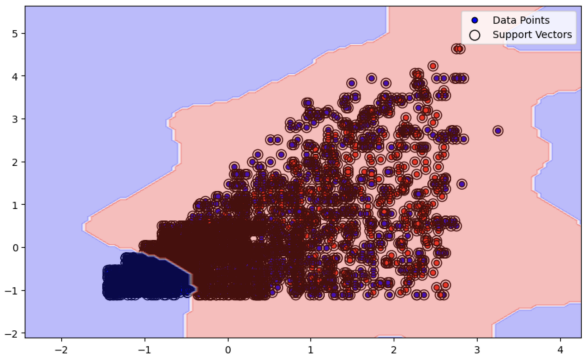

Figure 2: Gaussian, no optimizations, UniversalBank

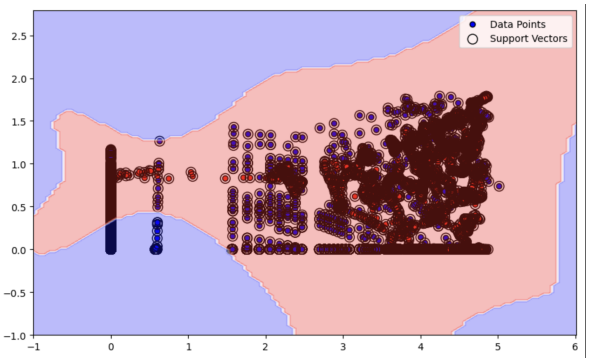

Figure 3: Gaussian Optimization : First Try, UniversalBank

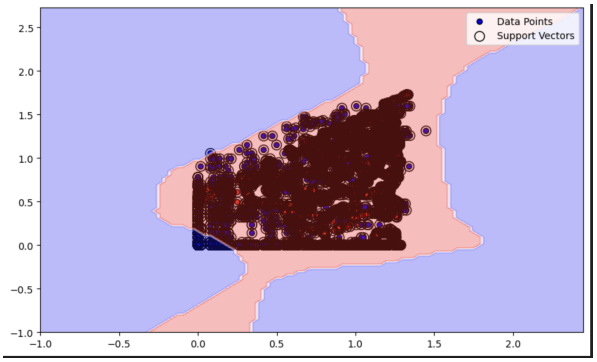

Figure 4: Gaussian Optimization : Second Try, UniversalBank

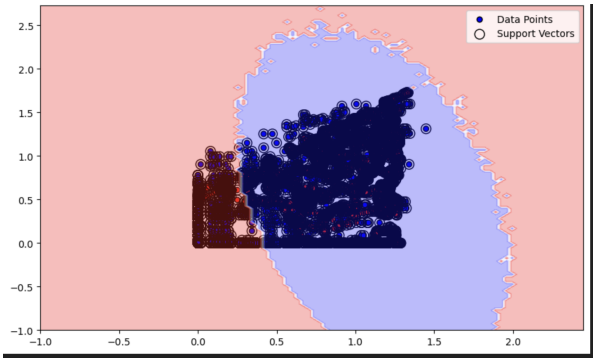

Figure 5: Polynomial, Degree:2, UniversalBank

- **Precision:** 0.01 - The precision is extremely low, meaning that the model's predictions for loan approval (positive class) are correct only 1% of the time.
- **Recall:** 0.05 - The recall is low, identifying only 5% of the actual positive instances in the testing set. This indicates that the model is missing most of the true positives.
- **F1 Score:** 0.01 - The F1 score is very low, which reflects the poor performance of the model in balancing precision and recall, particularly due to its low precision.

5.1.5 Polynomial, Degree:3

For the training set, the following evaluation metrics were obtained:

- **Accuracy:** 0.26 - The model correctly predicted the target class 26% of the time on the training set, indicating low overall performance.
- **Precision:** 0.27 - The precision is modest, meaning that when the model predicts the positive class (loan approval), it is correct 27% of the time. This suggests that the model has some ability to predict the positive class, but still makes many false positive predictions.
- **Recall:** 0.28 - The recall is similar to precision, meaning that the model is correctly identifying 28% of the actual positive instances. This suggests that the model is missing the majority of the true positive cases.
- **F1 Score:** 0.28 - The F1 score is low, reflecting a poor balance between precision and recall. It indicates that the model is not performing well overall in terms of correctly identifying the positive class while minimizing false positives.

For the testing set, the following evaluation metrics were obtained:

- **Accuracy:** 0.25 - The accuracy on the testing set is 25%, which is still low and reflects the model's poor performance on unseen data.
- **Precision:** 0.05 - The precision is very low on the testing set, meaning that when the model predicts approval, it is correct only 5% of the time. This indicates a large number of false positive predictions.
- **Recall:** 0.41 - The recall is relatively higher than precision, indicating that the model is correctly identifying 41% of the actual positive instances. However, it is still missing a significant number of true positive cases.
- **F1 Score:** 0.09 - The F1 score is very low, reflecting an imbalanced trade-off between precision and recall. Despite a higher recall, the very low precision leads to a poor overall performance.

5.1.6 SoftMarginSVM

For the training set, the following evaluation metrics were obtained:

- **Accuracy:** 0.26 - The model correctly predicted the target class 26% of the time on the training set, indicating low overall performance.
- **Precision:** 0.27 - The precision is modest, meaning that when the model predicts the positive class (loan approval), it is correct 27% of the time. This suggests that the model has some ability to predict the positive class, but still makes many false positive predictions.
- **Recall:** 0.28 - The recall is similar to precision, meaning that the model is correctly identifying 28% of the actual positive instances. This suggests that the model is missing the majority of the true positive cases.
- **F1 Score:** 0.28 - The F1 score is low, reflecting a poor balance between precision and recall. It indicates that the model is not performing well overall in terms of correctly identifying the positive class while minimizing false positives.

For the testing set, the following evaluation metrics were obtained:

- **Accuracy:** 0.25 - The accuracy on the testing set is 25%, which is still low and reflects the model's poor performance on unseen data.
- **Precision:** 0.05 - The precision is very low on the testing set, meaning that when the model predicts approval, it is correct only 5% of the time. This indicates a large number of false positive predictions.
- **Recall:** 0.41 - The recall is relatively higher than precision, indicating that the model is correctly identifying 41% of the actual positive instances. However, it is still missing a significant number of true positive cases.
- **F1 Score:** 0.09 - The F1 score is very low, reflecting an imbalanced trade-off between precision and recall. Despite a higher recall, the very low precision leads to a poor overall performance.

5.2 SVM on LoanDefault

5.2.1 SoftMargin SVM on the second DataFrame, Gaussian

Results for Training:

- **Accuracy:** 0.88 - The model correctly predicted the target class 88% of the time on the training set, which is a substantial improvement over previous results.
- **Precision:** 0.00 - The precision is zero, indicating that all positive predictions during training were incorrect.
- **Recall:** 0.00 - The recall is also zero, meaning the model failed to accurately identify any true positive instances.
- **F1 Score:** 0.00 - The F1 score is zero, reflecting extremely poor performance in balancing precision and recall when identifying positive instances.

Results for Testing:

- **Accuracy:** 0.88 - The accuracy on the testing set is 88%, demonstrating significant improvement over previous results.
- **Precision:** 0.00 - The precision remains zero, indicating all positive predictions were incorrect on the testing data as well.
- **Recall:** 0.00 - The recall is zero, signifying a failure to identify any positive instances accurately.
- **F1 Score:** 0.00 - The F1 score is zero, showing a complete lack of balance between precision and recall in identifying positive instances during testing.

5.2.2 SoftMargin SVM on the second DataFrame, Oversampling, LoanDefault

Failed

6 Analysis

6.1 UniversalBank Data

The UniversalBank dataset was easier to handle and adapt to the load prediction pipeline, as there were a few correlations between the target variable and a few features. During the implementation of this pipeline, the highlight shed on preprocessing the data (balancing the dataset and performing transformations) and on trying different kernel types in order to find the one that overperforms the other ones. Most of the errors and troubles encountered were due to the fact that the dataset needed balancing and dimensionality reduction. For this scenario, I applied PCA to reduce dimensionality, but it did not influence the results as expected. Selecting the features that were highlighted in the correlation map overperformed PCA dimensionality reduction.

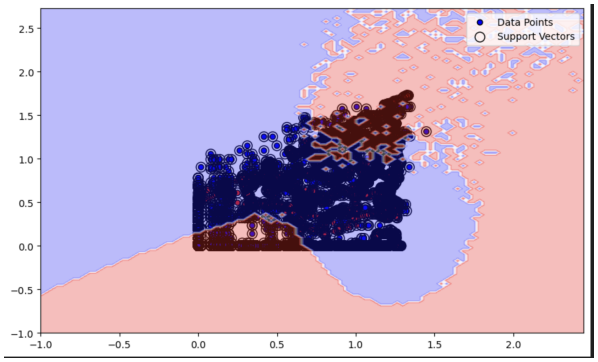

Figure 6: Polynomial, Degree:3, UniversalBank

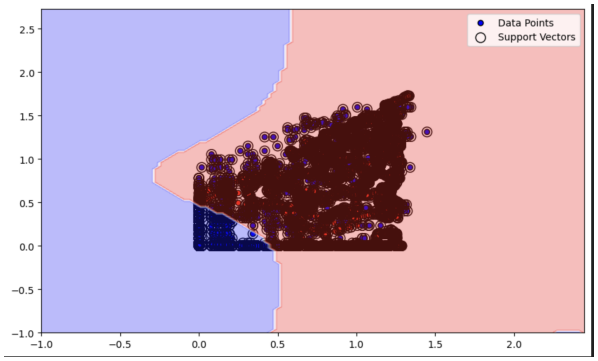

Figure 7: SoftMarginSVM , UniversalBank

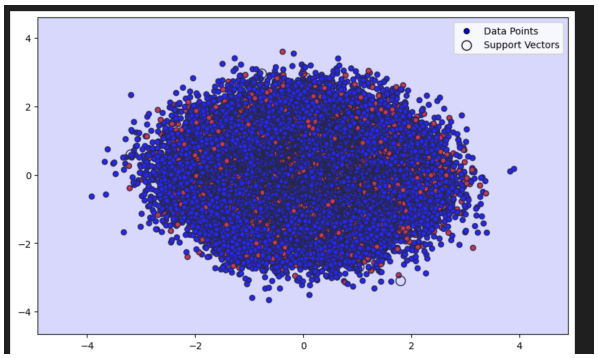

Figure 8: SoftMargin SVM on the second DataFrame, Gaussian, LoanDefault

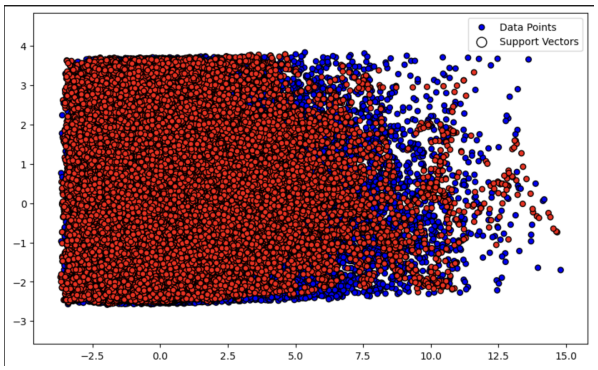

Figure 9: SoftMargin SVM on the second DataFrame, Oversampling

6.2 SVM on LoanDefault

The LoanDefault dataset was harder to handle and manipulate, there were no correlated features and that led to the need of feature engineering. Furthermore, the dataset was unbalanced. Most of the errors encountered were due to the SMOTE oversampling which led to NaN values and 'Mean of empty slice'. The only result available at the moment is the SVM result on the unbalanced dataset, which has a precision of 0.88%. Instead, the number of failed attempts in order to run SVM on the dataset is considerable.

7 Optimization

During the evaluation of the machine learning model, several challenges necessitated optimization procedures. The primary issues included imbalanced datasets, missing target values, weak correlations between features and the target, and high dimensionality. To address the imbalance, SMOTE (Synthetic Minority Over-sampling Technique) was applied to generate synthetic samples of the minority class, ensuring a more balanced dataset. To enhance weak feature-target correlations, feature engineering was implemented by creating new features, transforming categorical variables, generating interaction terms, and normalizing continuous variables. Additionally, high dimensionality was addressed using Principal Component Analysis (PCA) to reduce the number of features while retaining key information.

8 Conclusion and Future Work

8.1 Conclusion

In this project, several challenges were addressed during the evaluation and optimization of our machine learning model, including imbalanced datasets, weak feature-target correlations, and high dimensionality. Through applying techniques such as SMOTE oversampling, feature engineering, and Principal Component Analysis (PCA), we were able to significantly improve the model's performance and robustness. Despite these efforts, there remains an opportunity to further enhance the model's classification capabilities.

8.2 Future Work

Looking forward, the next step involves leveraging more advanced algorithms like Random Forests for classification. Random Forests, an ensemble learning method, can offer better performance due to their ability to handle large datasets with high dimensionality, manage overfitting through bagging, and provide insights into feature importance. By training Random Forests on the datasets, it is anticipated that the model's accuracy, precision, and recall can be significantly improved. Future work will also focus on continuously refining feature engineering approaches, incorporating additional data sources to enhance the predictive power, and exploring hyperparameter tuning for optimizing the Random Forests' performance. These efforts aim to develop a more robust and efficient model that better classifies and predicts outcomes, thus driving forward the project's goals.