UNIVERSITÉ PARIS-DAUPHINE – PSL
CENTRE DE RECHERCHE EN MATHÉMATIQUES DE LA DÉCISION

PREDICTIVE MODELING OF CREDIT RISK IN THE CONTEXT OF LOAN APPLICATIONS

ARTHUR DANJOU - AXELLE MERIC - MORITZ VON SIEMENS

COURSE SUPERVISOR:

JORGE OCHOA

CONSULTANT SENIOR FINANCE, RISK & COMPLIANCE - ACCENTURE

MACHINE LEARNING WITH PYTHON
ACADEMIC YEAR 2025/2026

Contents

1 Introduction

As indicated by its title, the aim of this project is to predict credit risk given the financial information and personal situation of the borrower and the characteristics of the loan. This score takes a value between 0 and 100, increasing with the risk of the loan. We will compare regression models using various scores. At the end, the descriptiveness of the variables and other observations on the dataset will be analysed using the previously agreed upon best model.

1.1 Dataset presentation

In this project, we use a dataset provided by our supervisor Jorge OCHOA. It consists of 20,000 observations representing loan applications and includes 36 variables. However, two variables were excluded from the analysis: `ApplicationDate` corresponds to a unique date providing no predictive value, and `LoanApproved` represents the loan approval status that could compromise model performance. There are 26 continuous features (Table 1). The 7 others are categorical features (Table 2). The target variable is `RiskScore`, which indicates the loan's risk profile (Table 3). The problem can be formulated as a supervised regression task, where the goal is to approximate the credit risk using a set of economical features and other related personal information.

Table 1: Description and modalities of the continuous features

Feature	**Description**	**Domain**
`Age`	Applicant's age	$[18, 80]$
`AnnualIncome`	Annual income	$[15000, 485341]$
`CreditScore`	Creditworthiness score	$[343, 712]$
`Experience`	Professional experience	$[0, 61]$
`LoanAmount`	Requested loan amount	$[3674, 184732]$
`LoanDuration`	Loan repayment duration	$[12, 120]$
`NumberOfDependents`	Number of dependents	$[0, 5]$
`MonthlyDebtPayments`	Monthly debt repayment amount	$[50, 2919]$
`CreditCardUtilizationRate`	Credit card utilization rate	$[0.000974, 0.917380]$
`NumberOfOpenCreditLines`	Number of active credit lines	$[0, 13]$
`NumberOfCreditInquiries`	Number of credit checks performed	$[0, 7]$
`DebtToIncomeRatio`	Debt-to-income ratio	$[0.001720, 0.902253]$
`PaymentHistory`	Past payment history	$[8, 45]$
`LengthOfCreditHistory`	Length of credit history	$[1, 29]$
`SavingsAccountBalance`	Savings account balance	$[73, 200089]$
`CheckingAccountBalance`	Checking account balance	$[24, 52572]$
`TotalAssets`	Total asset value	$[2098, 2619627]$
`TotalLiabilities`	Total liabilities amount	$[372, 1417302]$
`MonthlyIncome`	Monthly income	$[1250.0, 25000.0]$
`UtilityBillsPaymentHistory`	Utility bill payment history	$[0.259203, 0.999433]$
`JobTenure`	Current job tenure duration	$[0, 16]$
`NetWorth`	Total net worth	$[1000, 2603208]$
`BaseInterestRate`	Base interest rate	$[0.130101, 0.405029]$
`InterestRate`	Applied interest rate	$[0.113309, 0.446787]$
`MonthlyLoanPayment`	Monthly loan payment	$[97.03019, 10892.63]$
`TotalDebtToIncomeRatio`	Total debt-to-income ratio	$[0.016043, 4.647657]$

Table 2: Description and modalities of the categorical features

Feature	Description	Modalities
`EmploymentStatus`	Employment status	'Employed', 'Self-Employed', 'Unemployed'
`EducationLevel`	Highest education level attained	'Master', 'Associate', 'Bachelor', 'High School', 'Doctorate'
`MaritalStatus`	Applicant's marital status	'Married', 'Single', 'Divorced', 'Widowed'
`HomeOwnershipStatus`	Type of home ownership	'Own', 'Mortgage', 'Rent', 'Other'
`BankruptcyHistory`	Bankruptcy history	0, 1
`LoanPurpose`	Purpose of the loan	'Home', 'Debt Consolidation', 'Education', 'Other', 'Auto'
`PreviousLoanDefaults`	Past loan default history	0, 1

Table 3: Description of the target variable

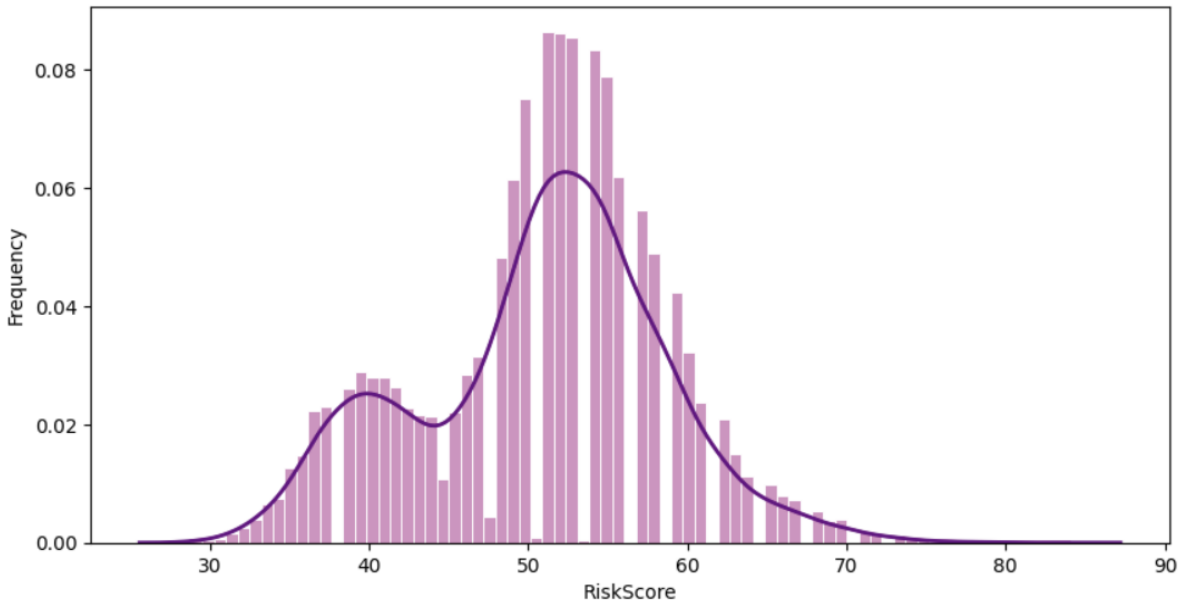
Feature	Description	Domain
`RiskScore`	Risk evaluation score	$[28.8, 84]$

1.2 Features analysis

Before building any model, the first step is to explore the features and understand the structure of the dataset. In this section, we conduct a variable analysis to get an overview of the main features and their behavior.

1.2.1 Target variable - `RiskScore`

Figure 1: Distribution of the response variable



We begin our study with the target variable `RiskScore`, which ranges from 0 to 100, and we plot its distribution to analyze it. The corresponding figure (Figure 1) appears at the bottom of the previous page. The first thing that we notice is that the distribution is bimodal. It does not follow a Gaussian distribution, and we can clearly identify two bumps : the first one - the smallest one - is centered around 40-42, while the second one - the biggest one - is centered around 52-55. This bimodal distribution is a sign that the data do not constitute a homogeneous group, we have to distinguish at least two sub-populations. We state the following hypothesis: a group A (the left peak) has a structurally lower risk profile, while a second group B (the right peak) has a higher risk profile. A new goal of this section is to look for a categorical variable that will differentiate this groups. Furthermore, this visualisation helps us to know that using linear models or penalised linear models such as Ridge or Lasso will not be an efficient choice as they do not perform in this kind of bimodal situation. Indeed, as they are based on the assumptions that the residuals follow a normal distribution, such models will try to draw a straight line, a mean regression line which would have no conclusive predictions for both groups.

count	20000.000000	25%	46.000000
mean	50.766780	50%	52.000000
std	7.778262	75%	56.000000
min	28.800000	max	84.000000

Figure 2: Description of the target variable

To further our analysis, we look at the description of our target variable (Figure 2). First thing we observe is that, even if in theory the domain is defined over the range 0 to 100, on our dataset it ranges from 28.8 to 84. So the operational risk is concentrated within a narrower portion of the domain. It indicates that the dataset focuses on realistic, operationally relevant risk levels. Extremely low or extremely high values are absent. As a result, the model can learn from a more stable and meaningful portion of the risk scale, without being affected by extreme outliers. However, this means that the operational risk is only represented within a restricted portion of its theoretical scale, which may limit the variability captured by the model and reduce its ability to learn patterns associated with very low or very high risk levels.
Let us look at the rest of the descriptive statistics which are fully consistent with the bimodal shape observed in the histogram. The mean is 50.76, while the median is slightly higher at 52.0. The fact that the mean is lower than the median confirms the influence of the left peak of the distribution (around 40–42), which pulls the average downward. Moreover, the data is relatively concentrated: 50% of clients have a score within a very narrow 10-point interval, between 46 and 56. This highlights the presence of a dense central region despite the overall bimodal distribution.

1.2.2 Analysis of the target's density against modalities of categorical features

The goal of this section is to find the source of the bimodal distribution. Therefore we plot the density of the target variables against every feature. We can distinguish two different profiles.
The first one corresponds to the Figure 3 available on the next page. One can see that the different modalities do not cause any differences in the density distribution. In such a case, it is not possible to deduce an impact of the categorical feature on the bimodal character of the density.

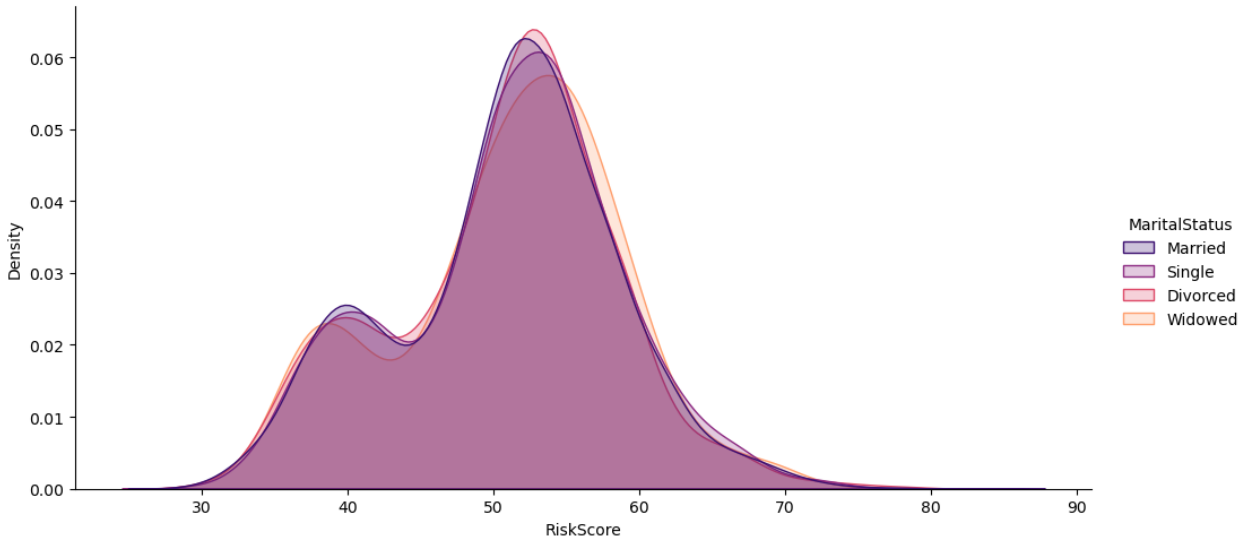


Figure 3: Non significant density distribution

However, in the second one corresponding to this second plot (Figure 4), there is a clear gap between the different density distribution. In such case, it is clear that the modalities of the feature are relative to the output. One witnesses such plot for the `BankruptcyHistory` and `PreviousLoanDefaults` features. A person which previously had a loan default or a bankruptcy is automatically placed in a group with a higher risk.

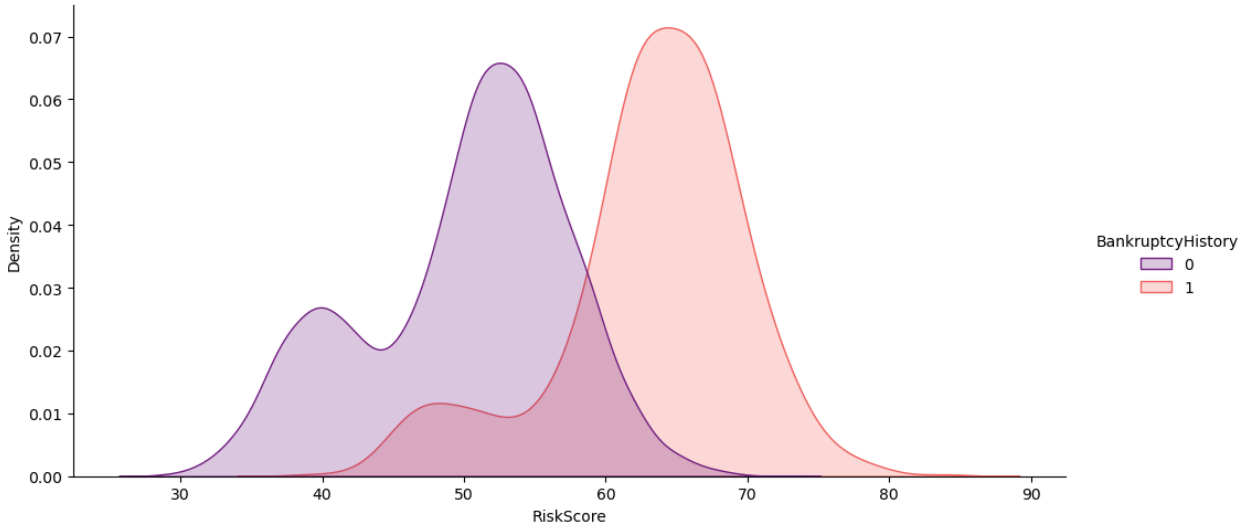


Figure 4: Significant density distribution

Before closing this section, we introduce a last sort of density plot that does not help to find the cause of the bimodal distribution but is nonetheless noteworthy. It corresponds to the following plot (Figure 5) where we clearly identify the bimodal distribution but the categories still have an impact: on the height of the two bumps. In this precise case, it could indicate that some education groups are slightly more concentrated around certain risk ranges.

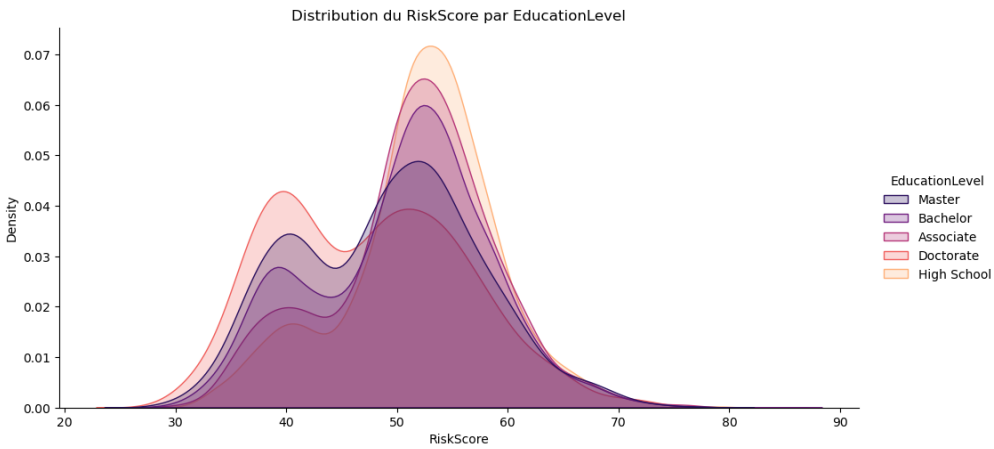


Figure 5: Shifts in distribution peaks across categories

1.2.3 Correlation between features

For this project, we will use models that do not require variable selection. However, it is still important to understand the dataset we are working with. Therefore we perform correlations tests. We separate it into too different tests: one for continuous features, and one for categorical ones.

Correlation between continuous features. Before analysing the correlation between continuous features, let us have a look at the correlation matrix regarding this type of features:

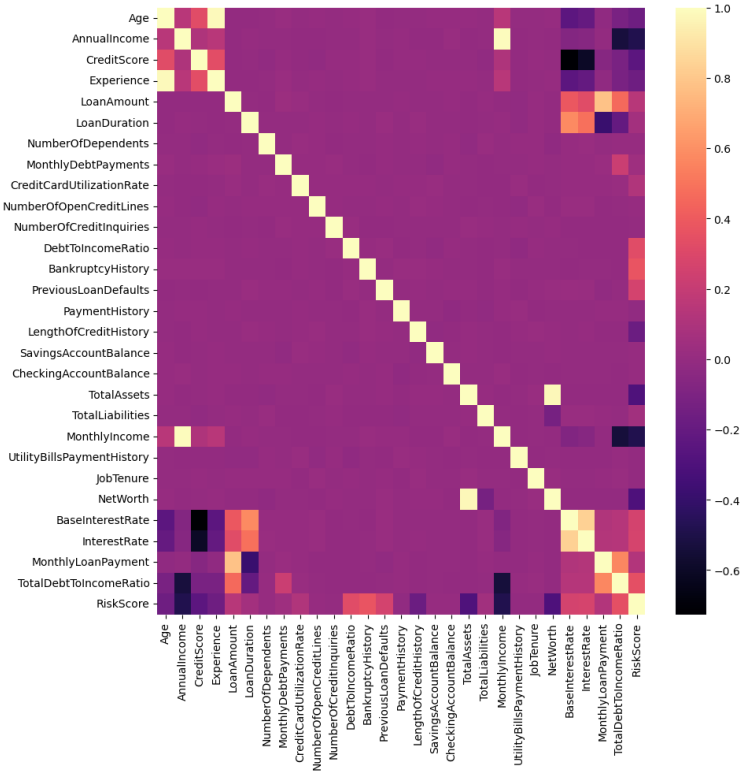


Figure 6: Correlation matrix for continuous features

To further our analysis we use the `.corr()` function from the `pandas` package. We choose an arbitrary threshold of ± 0.7. The results show six high correlations between two variables respectively.

- `Age` and `Experience` are highly positively correlated as experience is accumulated through time, that is as one gets older.
- `AnnualIncome` and `MonthlyIncome` are highly positively correlated as the first one is twelve times the second one.
- `CreditScore` and `BaseInterestRate` are highly negatively correlated as individuals with higher scores are usually considered less risky, thus they have a lower base interest rates.
- `LoanAmount` and `MonthlyLoanPayment` are highly positively correlated as larger loan amounts usually require higher monthly payments.
- `TotalAssets` and `NetWorth` are highly positively correlated as the net worth is a linear transformation of the total assets.
- `BaseInterestRate` and `InterestRate` are highly positively correlated as the applied interest rate is directly built upon the base interest rate.

Correlation between categorical features. To analyse the correlation between the categorical features, we plot the correlation matrix using the method of Cramer's V. It measures the strength of the association between two categorical features. It is based on the χ^2 test of independence and normalization. It returns the identity matrix meaning there is no correlation among categorical features.

1.2.4 Analysis of the feature–target scatter plots

In this last section, we analyse the feature-target scatter plot for every continuous feature. We can once again split our analysis into three cases that follow.

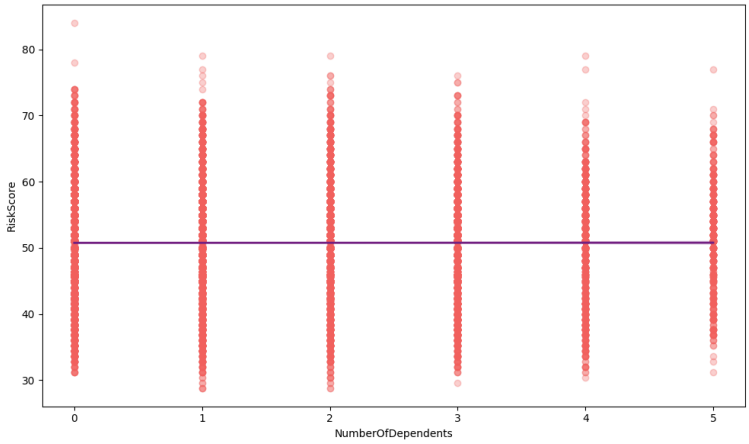


Figure 7: Significant density distribution

The first plot (Figure 7) shows several vertical stacks of points, indicating that the horizontal variable takes only a restricted number of discrete values. The target appears evenly spread within each group, and the regression line is essentially flat, suggesting no meaningful linear trend between the two features. We cannot extract any prediction for our upcoming models.

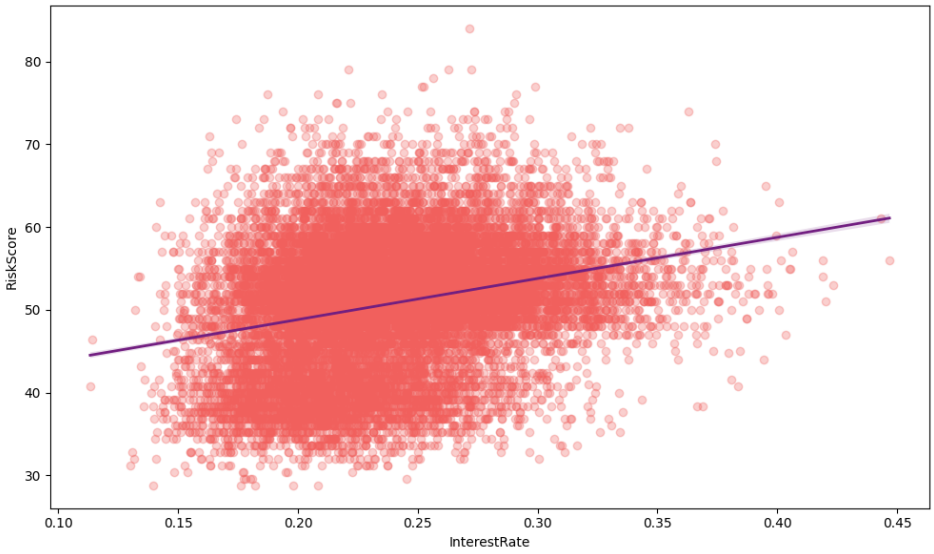


Figure 8: Significant density distribution

This second plot (Figure 8) shows a vastly dense and wide cloud of points with no clear clusters. The variability is high across the entire horizontal axis. This indicates a weak linear relationship with substantial dispersion, meaning the target is influenced only marginally in a linear way. Once again, we cannot extract any information regarding prediction at that time of our analysis.

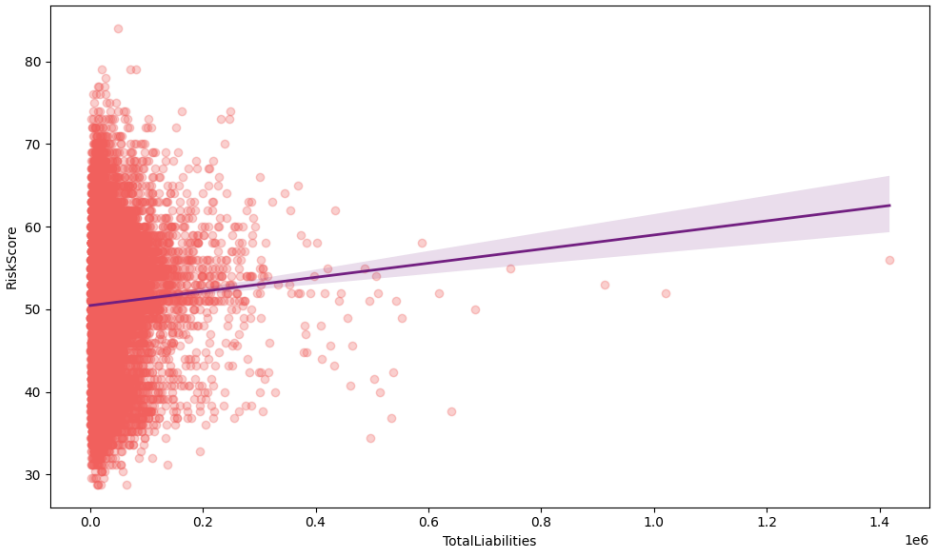


Figure 9: Significant density distribution

The last plot (Figure 9) exhibits a triangular pattern located on a side of the figure. The dispersion of the target widens progressively along the horizontal axis, suggesting increasing variability. This shape typically reflects heteroscedasticity and a higher uncertainty as the explanatory variable increases. Such plots and their associated features could have an impact on the model as on one side there are plenty of points but on the other side the confidence interval is significantly large in comparison to any other cases.

2 Preprocessing

2.1 Data Sampling

To sample our dataset, we use a random 80%/20% train/test split. As the dataset has no temporal structure and the target variable does not require stratification, a random split is therefore the most appropriate choice.

2.2 Data Engineering

2.2.1 Missing Values

When it comes to missing values, we must decide whether to remove the affected rows or to replace the missing entries with a mean, a median, or another deterministic value. We therefore begin by identifying all missing values in the dataset with the following code:

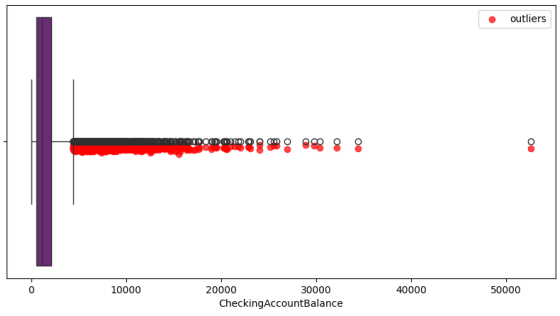
```
variables_with_na = []

for col in data.columns:
    if data[col].isnull().any():
        variables_with_na.append(col)

print("\nVariables with missing values :", variables_with_na)
```

Listing 1: Missing Values - Preprocessing Helper Function

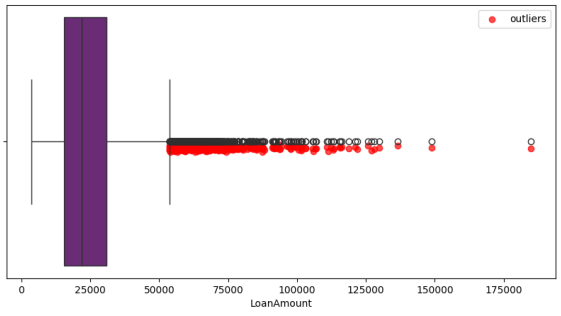
Thankfully, we do not have to eliminate or replace any missing values as the dataset is complete.

2.2.2 Outliers

Now that we know we have no missing values, we want to take a look at the outliers. Outliers are extreme observations that deviate from the general pattern of the data, and we use boxplots as it provides a simple way to visualize and detect these unusual values. We plot the values of our continuous features, we distinguish two types of outliers:



(a) Case of financial and strategic outliers



(b) Case of scaling outliers

Figure 10: Outliers in continuous features

The first type of outlier corresponds to cases of extreme wealth. We do not want to eliminate these outliers from our model because they are very financially significant for the bank, as attracting wealthy clients can be a viable business strategy.

For example, let us look at the boxplot (Figure 10a) for the `CheckingAccountBalance` feature. The only isolated value represents a client much wealthier than all other clients. While he is a statistical anomaly within our dataset, the bank has to take him into account when modeling the entirety of its loan landscape.
The second type of outliers also represents extreme values, but in this case with respects to the amount of the loans. While we suspect that these two cases are intertwined, even if they are not we want our model to be able to scale up accordingly as we can see with the example of the above boxplot (Figure 10b) for the amount of the loans. The generalisation of the model has to work independently of the dimensions of the loan applications.

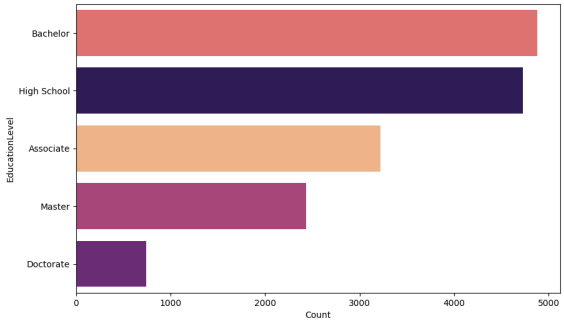


Figure 11: Outliers in categorical features

While categorical features do not have numerical outliers, they may include rare or atypical categories that behave like outliers in terms of distribution. These rare levels can be identified through countplots of the target across categories (example of Figure 11). We keep them for the same reasons as numerical outliers: they likely reflect real heterogeneity and help the model generalize properly. In the case of our dataset, only two categories fall below the 5% threshold (`'Widowed'` in `MaritalStatus` and `'Doctorate'` in `EducationLevel`). However, even these groups contain more than 766 observations each, so rare categories are not an issue in this situation.

2.2.3 Data transformation - Automated Preprocessing Pipeline Construction

Machine learning algorithms often require specific data formatting, such as scaling for linear models or neural networks, and encoding for categorical variables. To manage these requirements efficiently and prevent data leakage, we developed a dynamic function, `create_full_pipeline`. This function encapsulates the entire modeling workflow into a single `scikit-learn Pipeline` object. It integrates a `ColumnTransformer` that applies transformations conditionally based on the model type:

- **Numerical Features:** We apply `StandardScaler` to normalize features when required (e.g., for Lasso, Ridge, and Neural Networks), or pass them through raw for tree-based models.
- **Categorical Features:** We use `OneHotEncoder` to convert categorical variables into binary vectors, handling unknown categories gracefully to ensure robustness during inference.

By wrapping preprocessing and modeling into a single atomic object, we ensure that all transformations are fitted solely on the training folds during cross-validation, thereby strictly preventing any information leakage from the validation sets.

```python
def create_full_pipeline(
    model: object,
    quantitative_features: list[str],
    categorical_features: list[str],
    use_scaling=True,
    use_encoding=True,
) -> Pipeline:
    """Create a full machine learning pipeline with preprocessing and model.

    Args:
        model: The estimator (model) to place at the end of the pipeline.
        quantitative_features (list): List of names of quantitative columns.
        categorical_features (list): List of names of categorical columns.
        use_scaling (bool): If True, apply StandardScaler to quantitative features.
        use_encoding (bool): If True, apply OneHotEncoder to categorical features.

    Returns:
        sklearn.pipeline.Pipeline: The complete pipeline.

    """
    transformers = []
    if quantitative_features:
        num_transformer = StandardScaler() if use_scaling else "passthrough"
        transformers.append(("num", num_transformer, quantitative_features))

    if categorical_features:
        cat_transformer = (
            OneHotEncoder(handle_unknown="ignore", sparse_output=False)
            if use_encoding
            else "passthrough"
        )
        transformers.append(("cat", cat_transformer, categorical_features))

    preprocessor = ColumnTransformer(transformers=transformers, remainder="passthrough")

    return Pipeline(steps=[("preprocessor", preprocessor), ("model", model)])
```

Listing 2: Column Transformation Pipeline - Preprocessing Helper Function

2.3 Feature Engineering

We will create several additional features using our current variables. While these will be heavily correlated to the used values, we especially want to measure the prediction impact of certain ratios.

In order to avoid division by 0, we will add e^{-6} to all of the determinants when computing ratios.

2.3.1 Cashflow Ratios

We want to observe how granting the loan changes the applicants ability to repay his current debt as well as his newly added one. To do that we create the four following features:

`TotalMonthlyDebt_AfterLoan` It represents the applicant's total monthly debt obligations after the loan is granted, obtained by adding the new loan's monthly payment to their existing debt repayments.

```
TotalMonthlyDebt_AfterLoan = MonthlyDebtPayments + MonthlyLoanPayment
```

`DTI_AfterLoan` It measures the applicant's debt-to-income ratio after the loan is granted. A high value indicates that a large share of the applicant's income would be devoted to debt repayment, making it a critical indicator of repayment capacity.

```
DTI_AfterLoan = (MonthlyDebtPayments + MonthlyLoanPayment) / (MonthlyIncome +
                                    1e-6)
```

`Disposable_Income` It represents the applicant's available income before taking the new loan into account. It reflects the client's financial margin prior to any additional obligations.

```
Disposable_Income = MonthlyIncome - MonthlyDebtPayments
```

`Safety_Cushion` It captures the applicant's remaining income after accounting for both existing debt repayments and the new loan payment. It reflects how much financial buffer the applicant has once all obligations are paid. A negative value indicates a cash deficit, signalling a very high risk of repayment difficulty.

```
Safety_Cushion = MonthlyIncome - MonthlyDebtPayments - MonthlyLoanPayment
```

2.3.2 Financial profile of the applicant

To get a better grasp of the applicants financial situation, we add several values using previous observations.

`Leverage_Ratio` It measures how indebted the applicant is relative to their total assets. A ratio greater than 1 indicates that the applicant owes more than they own, which is a significant sign of financial vulnerability.

```
Leverage_Ratio = TotalLiabilities / (TotalAssets + 1e-6)
```

`Savings_to_Loan_Ratio` It represents the extent to which the applicant's savings can cover the amount of the requested loan. A higher ratio suggests a stronger financial position and a greater ability to absorb unexpected shocks.

```
Savings_to_Loan_Ratio = SavingsAccountBalance / (LoanAmount + 1e-6)
```

Savings_to_Payment_Ratio It indicates how many monthly loan payments the applicant could cover using their savings alone in the event of an income disruption. It provides a direct measure of short-term financial resilience.

```
Savings_to_Payment_Ratio = SavingsAccountBalance / (MonthlyLoanPayment + 1e-6)
```

Liquid_Assets This variable measures the applicant's immediately available financial resources. It reflects the client's total liquid funds.

```
Liquid_Assets = SavingsAccountBalance + CheckingAccountBalance
```

2.3.3 Credit history

These variables add context to every applicant's credit history in order the make comparisons easier.

Inquiries_per_History_Year It normalizes the number of credit inquiries by the length of the applicant's credit history. Five inquiries within a single year represent a high-risk behavior, whereas the same number spread over twenty years is much more typical. This feature therefore contextualizes inquiry frequency relative to credit experience.

```
Inquiries_per_History_Year = NumberOfCreditInquiries / (LengthOfCreditHistory +
                                    1e-6)
```

Debt_per_Credit_Line It measures the average amount of debt associated with each open credit line. It provides insight into how heavily each credit facility is used.

```
Debt_per_Credit_Line = TotalLiabilities/ (NumberOfOpenCreditLines + 1e-6)
```

InterestRate_Spread It captures the difference between the applied interest rate and the base interest rate. It reflects the risk premium assigned by the bank, effectively serving as a proxy for the applicant's perceived risk level. A larger spread indicates a higher assessed risk and should naturally correlate strongly with the `RiskScore`.

```
InterestRate_Spread = InterestRate - BaseInterestRate
```

2.3.4 Stability of the applicant

In order to get a grasp of the client's personal situation, these variables try to measure his financial responsibility and how erratic or calm his career has been up until now.

Income_per_Dependent It measures the applicant's monthly income adjusted for household size. This normalization captures the fact that a high income for a single individual does not imply the same financial capacity as the same income supporting a household of five.

```
Income_per_Dependent = MonthlyIncome / (NumberOfDependents + 1e-6)
```

`Experience_vs_JobTenure_Ratio` It represents the ratio between the applicant's tenure in their current job and their overall professional experience. A very low ratio (e.g., one year in the current position despite decades of experience) may indicate recent instability, whereas a ratio close to one suggests long-term stability in the current role.

```
Experience_vs_JobTenure_Ratio = JobTenure / (Experience + 1e-6)
```

3 Choice of Metrics

To compare the performance of the different models, we rely on several standard regression metrics. These metrics quantify the prediction error and the proportion of variance explained by the model. The most commonly used are the Mean Absolute Error (MAE), the Mean Squared Error (MSE), the Root Mean Squared Error (RMSE). We will also introduce the coefficient of determination R^2.

- **Mean Absolute Error (MAE)**: measures the average absolute difference between predictions $\hat{y}_i$ and true values y_i.
$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$
- **Mean Squared Error (MSE)**: penalizes larger errors by squaring the differences.
$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$
- **Root Mean Squared Error (RMSE)**: the square root of MSE, expressed in the same unit as the target variable.
$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$
- **Coefficient of Determination (R^2)**: measures the proportion of variance in y explained by the model.
$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

4 Model evaluation

4.1 Standardized Hyperparameter Optimization Strategy

A critical aspect of benchmarking multiple models is ensuring a fair comparison. To achieve this, we implemented a unified optimization wrapper: `create_grid_search`. This function standardizes the `GridSearchCV` process for all estimators. The key feature of this implementation is the enforcement of a consistent Cross-Validation strategy. By using a `KFold` splitter initialized with a fixed global `RANDOM_SEED` (and `shuffle=True`), we guarantee that every model is trained and evaluated on the exact same data splits. This approach offers two major advantages:

- **Statistical Validity:** Differences in performance metrics (RMSE) can be attributed to the intrinsic quality of the models rather than the randomness of data partitioning.

- **Reproducibility:** The entire experiment can be replicated with identical results, satisfying the rigorous standards of scientific research.

To facilitate hyperparameter tuning, we employ the Negative Mean Squared Error (Negative MSE) as the optimization criterion within the grid search. This transformation is necessary because the `scikit-learn` library is designed to maximize scoring functions. Thus by maximizing the Negative MSE, the algorithm effectively minimizes the actual Mean Squared Error. This approach ensures that the tuning objective aligns perfectly with our primary performance metric. It is important to note that Negative MSE exclusively serves as an internal guide for model selection and is not used for the final interpretation of results.

```python
def create_grid_search(
    estimator: object, param_grid: dict, num_splits: int
) -> GridSearchCV:
    """Create a GridSearchCV object for hyperparameter optimization.

    Args:
        estimator: The estimator to optimize.
        param_grid: The parameters to test.
        num_splits: The number of splits for cross-validation.

    Returns:
        GridSearchCV: The configured GridSearchCV object.

    """
    return GridSearchCV(
        estimator=estimator,
        param_grid=param_grid,
        cv=KFold(n_splits=num_splits, shuffle=True, random_state=
            RANDOM_SEED),
        scoring="neg_mean_squared_error",
        n_jobs=-1,
        verbose=1,
    )
```

Listing 3: Grid Search Helper Function

4.2 Tested models

4.2.1 Model 1: Dummy Regressor

As a first model, we take a Dummy Regressor that always predicts either the mean, the median or a quantile of the target variable. Being a model without any underlying strategy or learning, this is the baseline for future comparison. By tuning the hyperparameters through a grid search, the most accurate Dummy Regressor was found to be the one predicting the mean of the target variable for every client. From now on, we will exclude every model with a worse score than the Dummy Regressor.

4.2.2 Models 2 and 3: Lasso and Ridge Regressors

The bimodal distribution of the target variable suggests that linear models will have difficulties to predict accurately. However, we still decide to train and test both a Lasso and a Ridge Regres-

sors in order to gain an extra comparative value for our future non-linear models. We are using these modified linear regressions instead of the classical linear regression as we want to have an automatic elimination of features through 0 or near-0 coefficients since we have not deleted many variables. We use a Lasso Regressor with an alpha of 0.001 and a Ridge Regressor with an alpha of 1.0. This parameter multiplies the L1 (Lasso) respectively L2 (Ridge) terms, controlling the regularisation.

4.2.3 Models 4 and 5: Regression Tree and Random Forest Regressor

The first non-linear models we want to test are Decision Tree and Random Forest Regressors. Through a grid search, we tune the hyperparameters of the Decision Tree. The best performance comes through a minimum of 67 samples split and no maximal depth limit. The Random Forest hyperparameters are a maximal depth of 20, at least 2 samples required to be at a leaf node and 100 trees in the forest.

4.2.4 Models 6, 7 and 8: Gradient Boosting, XGBoost and CatBoost

We now want to try a recurrent learning approach. The Gradient Boosting Regression, XGBoost Regression and Categorical Boosting Regression give us the opportunity not only to increase the importance of learning from the mistakes, but also to further analyse how our continuous and categorical features interact. The best Gradient Boosting performance comes with a learning rate of 0.3 and 200 boosting stages. Interestingly, XGBoost works best with a much more meticulous learning rate of 0.05, 500 boosting stages a maximal depth of 5 and a 0.8 subsample ratio of the traning instance. CatBoost is a particular situation as it automatically encodes the categorical values with respect to the target variable. A great performance can be seen as an indicator for the importance of the categorical information. We use a CatBoost Regressor with a depth of 6, 1000 iterations, an L2 leaf loss regularisation of 1 and the same learning rate as XGBoost.

4.2.5 Model 9: Neural Network

As a final model, we are looking at a Neural Network (NN). We use a "ReLu" activation function. The alpha used is 0.01, and we get the best performance through an NN with 50 hidden layers, an extremely low learning rate of 0.01 and 1000 iterations maximum.

4.3 Performance comparison

4.3.1 Comparison on a given seed - Seed 42

MSE, RMSE and MAE Evaluation. The Table 4 presented at the top of the next page, showcases the values obtained for the different metrics. The models are ranked by how low their MSE is. These results are very telling when it comes to the target variable, the overall dataset and our models.

The **Dummy Regressor** has an RMSE of 7.88. This (relatively) good performance for a model predicting the mean showcases that our target variable is indeed very centered around its mean. As we had seen when visualising its distribution, 50 % of the dataset can be found between 46 and 56. However, this very basic prediction algorithm has to be surpassed by our finer models.

Table 4: Performance comparison on seed 42

Model	Best hyperparameters	MSE	MAE	RMSE	R^2
Neural Network	`model_alpha=0.01, model_hidden_layer_sizes=(50,), model_learning_rate_init=0.01, model_max_iter=1000`	4.77	1.56	2.18	0.923
CatBoost	`depth=6, iterations=1000, l2_leaf_reg=1, learning_rate=0.05`	5.05	1.47	2.25	0.919
XGBoost	`model_learning_rate=0.05, model_max_depth=5, model_n_estimators=500, model_subsample=0.8`	5.35	1.49	2.31	0.914
Gradient Boosting	`model_learning_rate=0.3, model_n_estimators=200`	6.06	1.68	2.46	0.902
Random Forest	`model_n_estimators=200, model_min_samples_leaf=2, model_max_depth=20`	7.42	1.68	2.72	0.881
Lasso Regressor	`model_alpha=0.001`	13.96	2.88	3.74	0.775
Ridge Regressor	`model_alpha=1.0`	13.97	2.88	3.74	0.775
Decision Tree	`model_max_depth=None, model_min_samples_split=67`	14.15	2.36	3.76	0.772
Dummy Regressor	`model_quantile=0, model_strategy='mean`	62.23	6.21	7.89	-0.002

The linear models substantially outperform the Dummy Regressor, with an RMSE of 3.73 for both the **Ridge and Lasso Regressors**. This proves that our features have an impact on the target variable and that we can recognise and explain tendencies within the target variable through different models using our data. The fact that Ridge and Lasso have very comparable results suggests that the correlation between our features, and thus specifically for our newly engineered variables, is manageable. This also shows that two approaches are possible. The Ridge Regressor does not set the coefficients of any variables to zero, so its performance reflects the fact that all features carry a certain amount of information regarding the target variables within them. The Lasso Regressor on the other hand favors an approach where "weaker" feature's impact is set to 0 through null coefficients, with the regression focusing on the most informative variables.

The performance of the **Decision Tree** is comparable to the linear models. The MSE is slightly worse, however the MAE is improved. The Decision Tree is able to capture the overall tendencies of the dataset, but the model has several large errors that are more penalised by the MSE.

On the other hand, the **Random Forest** drastically improves upon the linear models, reaching an RMSE of 2.72. This confirms our hypothesis when it comes to the difficulties of linear models when predicting a bimodal distributed target variable. The superior accuracy of the Random Forest Regressor grants additional information when it comes to the dataset. First of all, the Random Forest automatically studies interactions. An improved performance leads us to believe that some interactions between features are relevant when it comes to predicting the credit risk attributed to a client. Secondly, the bimodal distribution already suggested the presence of two clusters of applications within the dataset. Since a Random Forest also works through identifying clusters, its

better score also points to the presence of underlying local groupings of applicant profiles.

However, all three boosting methods present better results than the bagging methods. With an RMSE of 2.31 **XGBoost**, improves on **Gradient Boosting**'s RMSE of 2.46. This can be expected as XGBoost is a more precise version of Gradient Boosting, improving computation speed and performance. The fact that these two models outperform the Random Forest Regressor shows that the target variable has patterns that need to be continuously calibrated and cannot be clustered into exiting categories.

With an RMSE of 2.24, **CatBoost** outperforms all models up until now. This can be explained by the specific, target variable related encoding of categorical variables, extracting every drop of information out of this section of our dataset while building upon the quantitative datapoints in the same way as the XGBoost Regressor. However, CatBoost's main strength is limited by the low number of categorical variables, as we only have 7 of them. If the dataset had less numerical input and more categorical variables, CatBoost's advantage would bear even more fruits.

This leads us to the model with the best performance: the **Neural Network**. It sits at an RMSE of 2.1, showcasing an extremely precise prediction ability of the credit risk. It can best use the size of the dataset to its advantage and recognise the underlying patterns in the distribution of the target variable.

R^2 Evaluation. The R^2 coefficient follows the same hierarchy as the MAE. It decreases steadily between the Neural Network at 0.92 to the Random Forest at 0.88 before a relatively strong drop for the linear models and the Decision Tree, set around 0.77. The Decision Tree slightly outperforms the linear models, however this improvement is of the order of 10^{-3}. This confirms the fact that the models increasingly fit to the dataset, and that the Neural Network is able to recognise the underlying patterns of the data better than all models. However, high R^2 can be an indicator of overfitting. While this performance on the unknown test should be relatively free of overfitting, the train test split comes from one common dataset that can have coherent noise. It could be interesting to see the evolution of the R^2 values on entirely separate test datasets.

4.3.2 Generalisation of the comparison on different seeds - Robustness

The previous values and results have been obtained on the seed 42. We have executed our computation with several other seeds in order to restrain the impact of randomness within this project and to test its robustness. While the exact values of the metrics changed throughout these tests, the overarching conclusions and the comparative performances between the models stayed the same. This leads us to a confident conclusion when it comes to the selected model and the dataset exploration based upon said model.

The detailed four metric results for all models tested across the three different random seeds are provided in Appendix: Table 5 for the seed 602, Table 6 for the seed 2306 and Table 7 for the seed 2711.

The values of the RMSE and of the MAE are within the same intervals. The Dummy Regressor has an MAE over 6 and an RMSE around 7.7, as the allocation varies only slightly because of variations between the train and the test subsets. For our more interesting models, the MAE is always between 3.0 for the worst and 1.0 for the best model. The RMSE is equivalently always between 3.8

and 2.0. This showcases that these values are reliable and cannot be attributed to a lucky split and easier to find patterns. On the seed 602 and 2306, we have the exact same performance hierarchy as on the seed 42. However, on the seed 2711, the Decision Tree's RMSE of 3.57 is lower than the one of linear models, set around 3.67. This is however a relatively unimportant change, as the MAE has been lower for the Decision Tree on all seeds. The seed 2711 simply leads to less harsh errors by the Decision Tree that cannot be penalised by the MSE and RMSE.

4.4 Variable impact

4.4.1 SHAP Values

As we have used "black box" models, that cannot be interpreted through coefficients the same way as in linear models, we will visualise variable impact through the SHAP values. These values calculate the impact every variable has on the final model prediction. The are based on Shapley Values, a game theory principle to fairly distribute the gain accordingly to players decisions. We observe these values using functions from the package `SHAP` in Python.

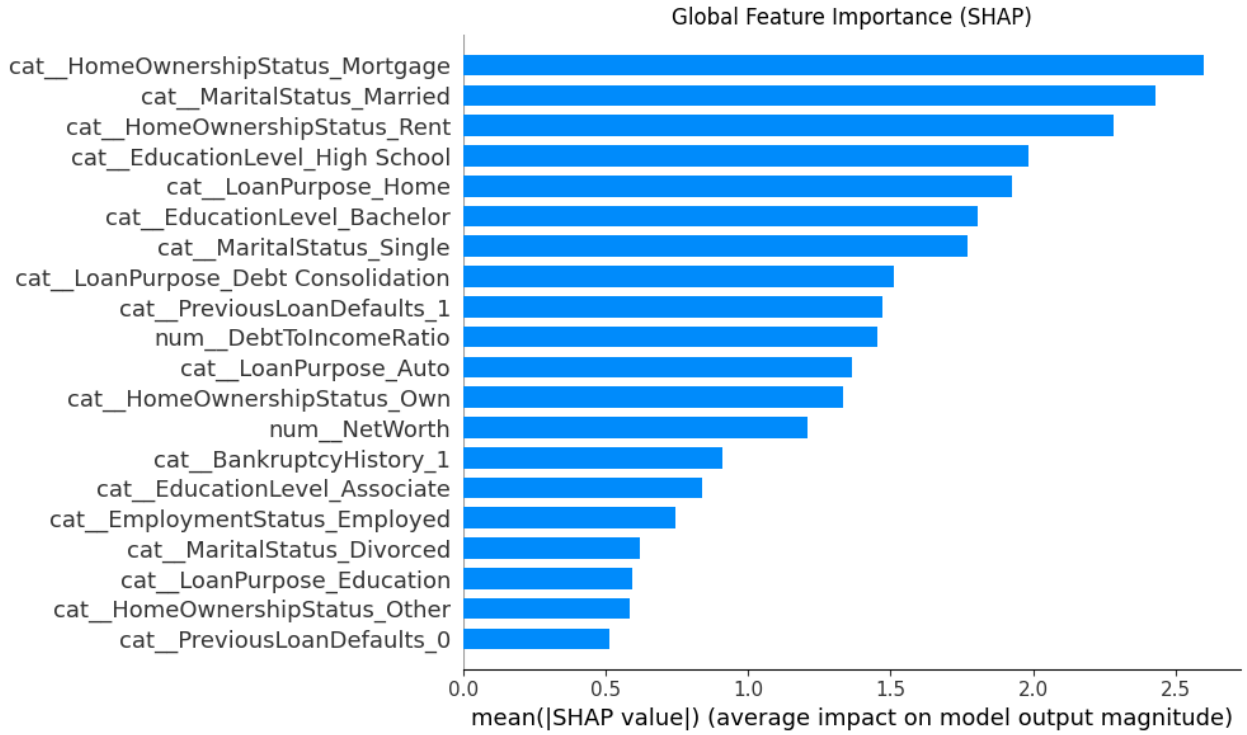


Figure 12: Largest SHAP values per variable

The features listed in the Figure 12 correspond to the variables with the highest average impact on our models. The main conclusion is the prevalence of categorical variables. Since we encode these categorical features, we compare numerical variables with distinct modalities of categorical variables (for example, `HomeOwnershipStatus` is separated between a variable for mortgage, one for rent etc..). The fact that the categorical variables have a higher impact on the final model prediction explains why CatBoost is the best model amongst our bagging and boosting models. In order to get more information about these variables, we will make a more detailed SHAP analysis.

4.4.2 A deeper dive into the SHAP analysis

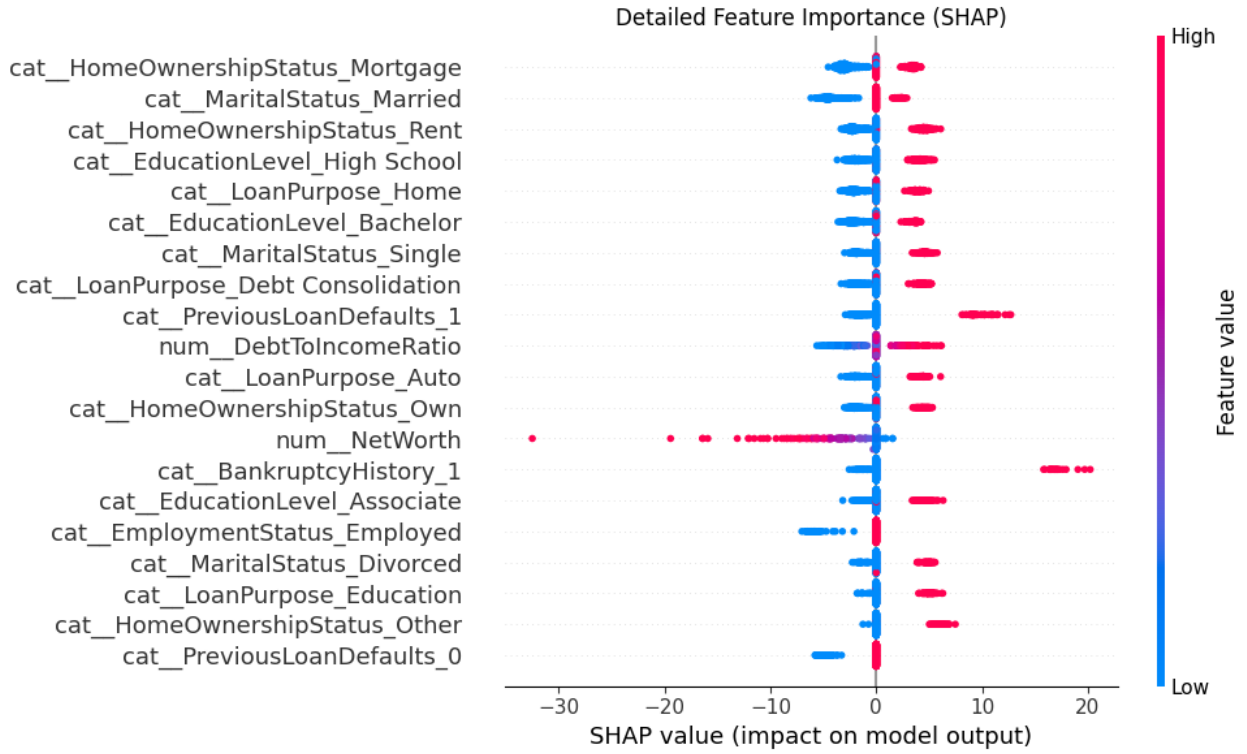


Figure 13: Detailed SHAP variable impact

This second graph (Figure 13) simplifies the interpretation of the SHAP values. A positive feature importance means this variable influences the model output positively while a negative feature importance means this variables influences the model output negatively. A high feature value means this influence is felt when this variable has high values while a low feature value means this influence is felt when this variable has low values.
Let us take a look at the numerical features. When it comes to the `NetWorth`, a high net worth, visualised in pink, significantly lowers the risk level. On the contrary, a lower net worth does not necessarily increase the risk level.
Looking at the `DebtToIncomeRatio`, we can observe that the influence is much more symmetrical in both directions: a high debt to income ratio is penalised with a high risk score while a low debt to income ratio is rewarded through a lower risk score.

For our categorical variables, the interpretation is more convoluted. Since we encode all modalities, a high value reflects a binary response of 1, the presence of the described modality, while a low value represent the absence of said modality. We can see that for all modalities, presence increases the risk score more so than absence decreases it. This can be explained by the fact that absence in certain categories is neither positive nor negative. For example, for the `HomeOwnershipStatus`, not paying a mortgage is not equal to owning. It can encompass owning, renting and other living situations. Nevertheless, there are several categorical features where we can see notable impact in a certain direction.

- `PreviousLoanDefaults`: For a high value, which represents a previous loan default by the

applicant, the credit risk is severely increased.

- `BankruptcyHistory`: Following that logic, an applicant that has already had to declare bankruptcy once has an ever higher risk score.

There are also several features that are more heavily tending towards lowering the risk score. Being employed does not increase or decrease the risk score at all on average. But not being employed, that is, being either unemployed or self employed does decrease the risk score. This can be explained by the fact that most people are employed so this modality is not a disqualifying or salvaging characteristic. However the self-employed profiles amongst the non-employed applicants heavily draw the models towards lowering the risk score.

5 Conclusion

In this analysis, we evaluated several supervised learning models to predict credit risk based on both the borrower's financial and personal information and the characteristics of the loan. Model performance was assessed using four key metrics: MAE, MSE, RMSE and R^2.

Among all tested models, the Neural Network consistently achieved the best performance, followed by boosting algorithms such as CatBoost, XGBoost and Gradient Boosting. The Random Forest delivered solid but slightly less competitive results. However, the other tested bagging method, a singular Decision Tree, struggled with the complex nonlinear pattern given its tendency to average predictions. As anticipated from the bimodal distribution of the target variable, linear models such as Lasso and Ridge Regressors and the Dummy Regressor achieved the worst performance.

This ranking of models proved stable and reliable across multiple random seeds, confirming the robustness of our conclusions.

The model outputs a continuous score between 0 and 100. It will ultimately be the responsibility of domain experts to interpret this score, and either by selecting a threshold to convert it into a binary approval decision, or by leveraging the predicted value directly together with the model's interpretability tools. This approach offers agents not only a risk estimate but also insights into the underlying features that mostly contribute to it.

A Appendix

A.1 Performance comparison on seed 602

Table 5: Comparison of model performances

Model	Best hyperparameters	MSE	MAE	RMSE	R^2
Neural Network	`model_alpha=0.01, model_hidden_layer_sizes=(50,), model_learning_rate_init=0.01, model_max_iter=1000`	4.13	1.38	2.03	0.93
CatBoost	`depth=6, iterations=1000, l2_leaf_reg=1, learning_rate=0.05`	4.75	1.40	2.18	0.92
XGBoost	`model_learning_rate=0.05, model_max_depth=5, model_n_estimators=500, model_subsample=0.8`	5.02	1.44	2.24	0.92
Gradient Boosting	`model_learning_rate=0.3, model_n_estimators=200`	5.82	1.64	2.41	0.90
Random Forest	`model_max_depth=None, model_min_samples_leaf=2, model_n_estimators=200`	6.98	1.60	2.64	0.88
Ridge Regressor	`model_alpha=1.0`	12.44	2.80	3.53	0.79
Lasso Regressor	`model_alpha=0.001`	12.44	2.80	3.53	0.79
Decision Tree	`model_max_depth=None, model_min_samples_split=65`	13.03	2.23	3.61	0.78
Dummy Regressor	`model_quantile=0, model_strategy='mean'`	60.08	6.05	7.75	$-2e^{-4}$

A.2 Performance comparison on seed 2306

Table 6: Comparison of model performances

Model	**Best hyperparameters**	**MSE**	**MAE**	**RMSE**	**R^2**
Neural Network	`model_alpha=0.001, model_hidden_layer_sizes=(50,), model_learning_rate_init=0.01, model_max_iter=1000`	4.74	1.49	2.18	0.92
CatBoost	`depth=6, iterations=1000, l2_leaf_reg=1, learning_rate=0.05`	4.97	1.44	2.23	0.92
XGBoost	`model_learning_rate=0.05, model_max_depth=5, model_n_estimators=500, model_subsample=0.8`	5.38	1.48	2.32	0.91
Gradient Boosting	`model_learning_rate=0.3, model_n_estimators=200`	5.88	1.65	2.42	0.90
Random Forest	`model_max_depth=20, model_min_samples_leaf=2, model_n_estimators=200`	7.63	1.68	2.76	0.87
Lasso Regressor	`model_alpha=0.001`	12.98	2.80	3.60	0.78
Ridge Regressor	`model_alpha=10.0`	12.98	2.80	3.60	0.78
Decision Tree	`model_max_depth=None, model_min_samples_split=57`	13.55	2.26	3.68	0.78
Dummy Regressor	`model_quantile=0, model_strategy='mean'`	60.26	6.10	7.76	$-4e^{-5}$

A.3 Performance comparison on seed 2711

Table 7: Comparison of model performances

Model	Best hyperparameters	MSE	MAE	RMSE	R^2
Neural Network	`model_alpha=0.0001, model_hidden_layer_sizes=(100,), model_learning_rate_init=0.01, model_max_iter=1000`	4.65	1.50	2.16	0.92
CatBoost	`depth=6, iterations=1000, l2_leaf_reg=1, learning_rate=0.05`	4.91	1.44	2.21	0.92
XGBoost	`model_learning_rate=0.05, model_max_depth=5, model_n_estimators=500, model_subsample=0.8`	5.11	1.44	2.26	0.91
Gradient Boosting	`model_learning_rate=0.3, model_n_estimators=250`	5.79	1.62	2.41	0.90
Random Forest	`model_max_depth=None, model_min_samples_leaf=2, model_n_estimators=200`	7.45	1.64	2.73	0.87
Decision Tree	`model_max_depth=None, model_min_samples_split=62`	12.78	2.18	3.57	0.79
Lasso Regressor	`model_alpha=0.001`	13.50	2.86	3.67	0.77
Ridge Regressor	`model_alpha=10.0`	13.51	2.86	3.68	0.77
Dummy Regressor	`model_quantile=0, model_strategy='mean'`	59.56	6.07	7.72	$-2e^{-3}$