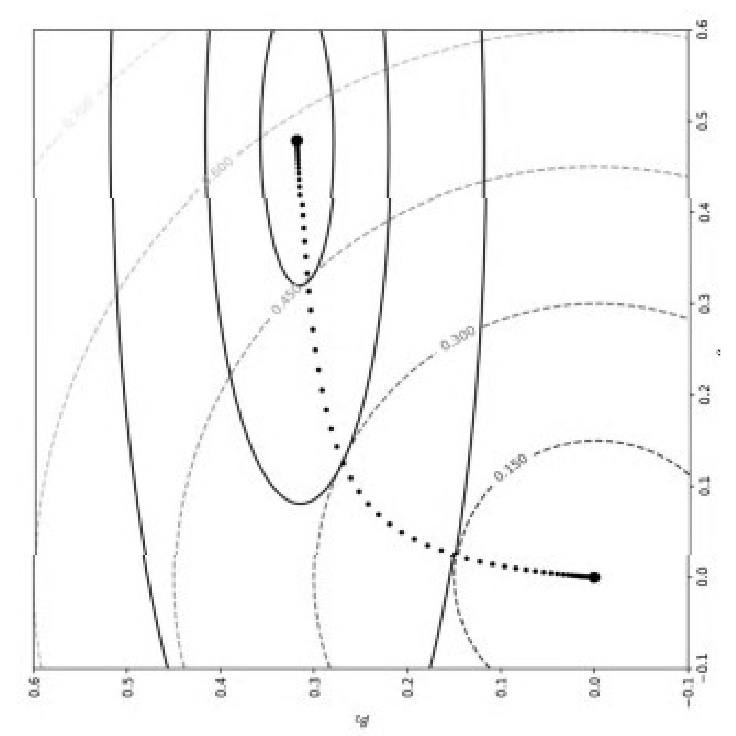
**2.1.3 Regularisation methods**

Regularization techniques are used in econometric modelling to minimize the variance of estimated parameters and hence the variability of model predictions in an ill-posed problem or overfitting the model. The first scenario of an ill-posed problem pertains to econometric models that display estimation instability. A slight change in the training set might cause huge changes in the model’s predictions; this is undesirable, particularly in credit scoring, can lead to an unsuitable model. It may be argued that model stability is a prerequisite for technological robustness in ethical and trustworthy machine learning. The multicollinearity of explanatory variables may cause the ill-posed issue; for example, variables representing credit arrears in multiple tenors, distinct tenors of DPD, and dummy behavioural variables are instances of highly correlated characteristics. The second situation of overfitting the model occurs when a model performs well on the training set. However, the prediction error increases considerably on different data sets. The model is considered to be overfitted to the initial data set (training set) and hence unable to infer on other data sets (Kaszyński et al., 2020). The inclusion of too many independent variables on a training set may lead to a factitiously high prediction power on training sets but not on test sets (Johnston & DiNardo, 1997). The three types of regularisation methods, ridge regression, LASSO regression, and elastic net regularisation, are discussed below.

**2.1.3.1 Ridge regression**

Ridge regression, a regularization technique used to estimate parameters when the independent variables display near multicollinearity, is a method of biased estimation with biased estimated parameters ( i.e. ). However, this allows for the reduction of the variance of estimated parameters. Hence, model streamlining may reduce the model predictions and parameter values variance while incurring low estimator bias. That trade-off (reduced variance and low bias) is desirable from the commercial point of view unless it significantly impairs the model’s interpretability. Let  = ( 1,…, *p*)*T* . The values of independent variables are generally standardized, i.e. Σi *xi,j**/ N* = 0 and Σi *x2i,j /N* = 1. The ridge regression parameters are defined as an optimal solution for the following inequality-constrained quadratic programming problem (Tikhonov & Arsenin, 1977):

(2.13)

where t ≥ 0 is the penalty term or the tuning parameter.

Based on the minimisation of the prediction error on the validation set, a model with a specific tuning parameter (target model) is chosen for a given set of *t* parameters.

# Figure 2.1: Contour curve of ridge regression loss function

[Source: Reproduced with permission from Kaszyński et al. (2020, p. 99)]

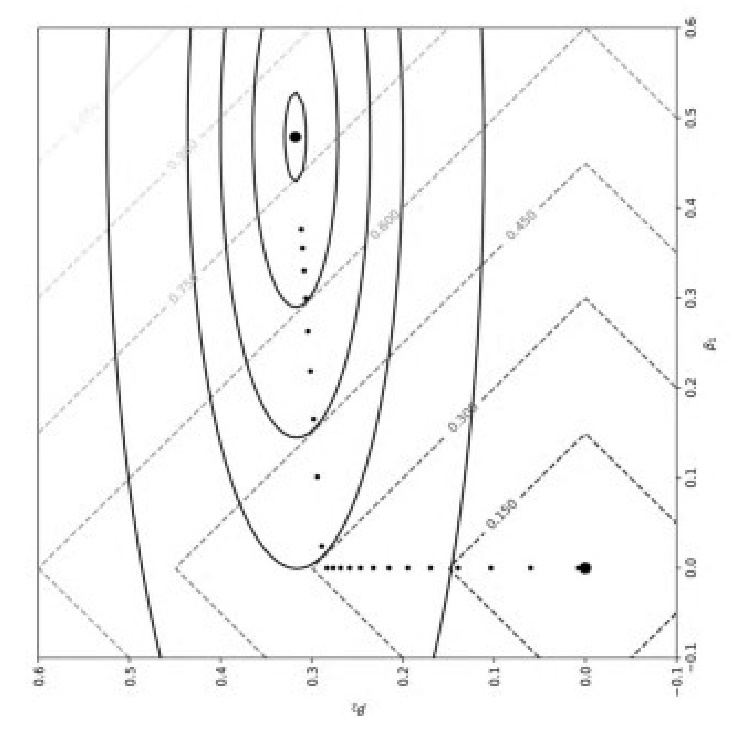
**2.1.3.2 Lasso Regression**

In the case of Lasso ( *Least Absolute Shrinkage and Selection Operator* ) regression, the regression parameters are defined as an optimal solution for the following constrained problem (Tibshirani, 1996):

(2.14)

where t ≥ 0 is the penalty term or the tuning parameter.

Akin to ridge regression, lasso regression sets a limit on the norm of model parameters; in the case of lasso regression, it is the *l*1 norm or urban norm. Even though these two regularisation methods have similar model structures (i.e. an additional penalty component imposed on the loss function of the logistic regression), they may be differentiated as follows:

* Lasso regression allows the exclusion of individual parameters (i.e. *βi*= 0). In contrast, ridge regression allows the values of the model parameters to be reduced to minimal values that are not equal to zero.
* The lasso regression makes parameter interpretation simpler than ridge regression.
* In the case of ridge regression, the change in the sign for a specific parameter may be observed, whereas, in the case of lasso regression, the model parameters change more linearly with the tuning parameter.
* Ridge regression penalises the model’s coefficients with larger values than lasso regression.
* In the case of two similar or highly correlated characteristics, the ridge regression will yield coefficients with equal weights. In contrast, the lasso regression will produce coefficients with the same sign but not identical values.

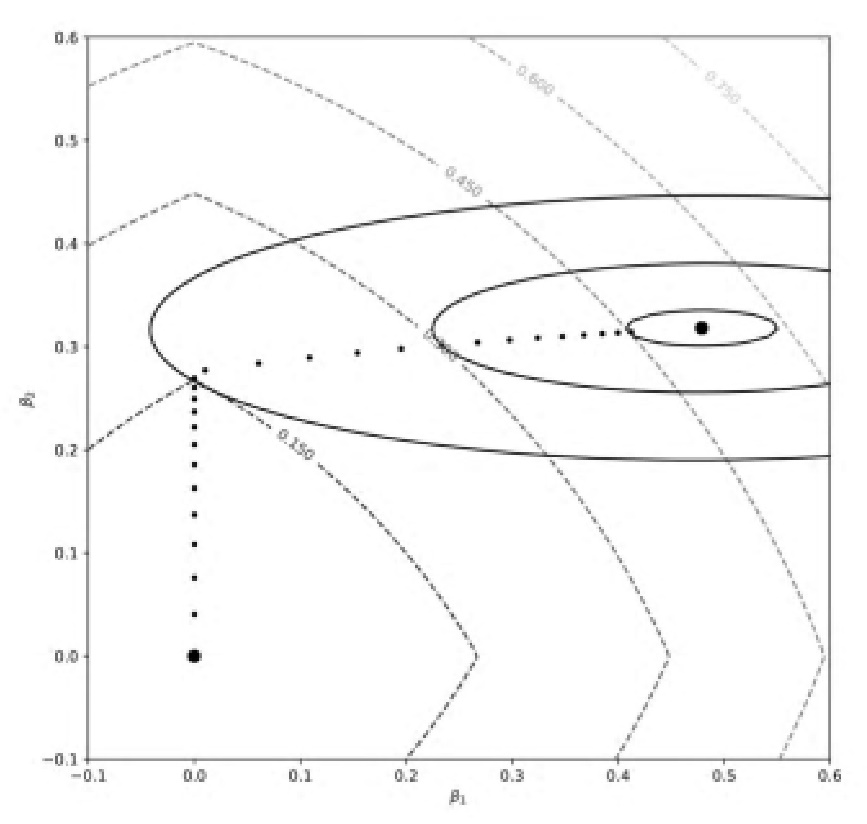
# Figure 2.2: Contour curve of lasso regression loss function

[Source: Reproduced with permission from Kaszyński et al. (2020, p. 100)]

**2.1.3.3 Elastic net regression**

The *elastic net* regularization combines both the penalties terms used in ridge and lasso regressions. Zhou et al. (2015) show that the elastic net may be reduced to the linear Support Vector Machine (SVM); for each setup of the elastic net (concerning the parametrization of penalties) in the case of binary classification problem, the solution of the elastic net is similar to the hyper-plane solution of linear SVM, thus enabling the use of highly optimized SVM solvers and Graphics Processing Unit (GPU) acceleration for elastic net problems. The logistic regression loss function with elastic net regularization assumes the following form:

(2.15)



# Figure 2.3: Contour curve of the elastic net regression loss function

[Source: Reprinted with permission from Kaszyński et al. (2020, p. 102)]

**2.2 Non-parametric models used in credit scoring**

The non-parametric models that require few or no assumptions concerning the relationship between dependent variable and explanatory variables, distribution of the variables or the errors include decision trees, random forest, extreme gradient boosting (XGBoosting), and neural network discussed below.

**2.2.1 Decision trees**

**2.2.1.1 Concept**

A decision tree is an algorithm of supervised learning (Kaszyński et al., 2020) in the form of a graphical tool, with a branch- or root-like structure of boxes and lines used to show possible turns of events that may or may not be controllable (Anderson, 2007). When the dependent variable is nominal and accepts multiple values indicating object classes, trees are often utilized in classification issues. For example, in the context of credit, A classification tree is a machine learning technique that classifies a collection of observations into subgroups based on attribute value tests. The division’s objective is to create subsets. The most often used techniques for building classification trees minimize the measure of diversity in the resulting subgroups (Hastie et al., 2009). Initially, efforts to derive decision trees were made by trial and error. According to Thomas et al. (2002), Breiman and Friedman separately proposed utilizing analytical techniques to derive the rule set in 1973. However, it was not until 1984 that they collaborated with Olshen and Stone to create Trees (CART) algorithms and C4.5 algorithms (Breiman et al., 1984; Quinlan, 2014). Makowski and Coffman applied their approach to credit scoring for the first time in 1985 and 1986, respectively. Another subset of algorithms is those that split the set using statistical testing. This category includes methods such as the CHAID algorithm, which uses the *χ*2 statistics (Kass, 1980) and permutation tests-based algorithms (Hastie et al., 2009).

**2.2.1.2 Structure of decision trees**

The tree structure represents the recursive divisions. The nodes and leaves signify the obtained subsets, while the tree branches illustrate the division rules. When the dependent variable is nominal and accepts multiple values indicating object classes, trees are often utilized in classification issues. The explanatory variables establish the divisions of observation sets. The tree nodes indicate the results of tests on the variable values. The root node is the first node that encompasses all observations. The branches correlate to the findings of many tests. The leaves display the category labels for the observations. Each tree represents a collection of classification rules, and each leaf represents a distinct classification rule. A classification rule is a statement of the form:

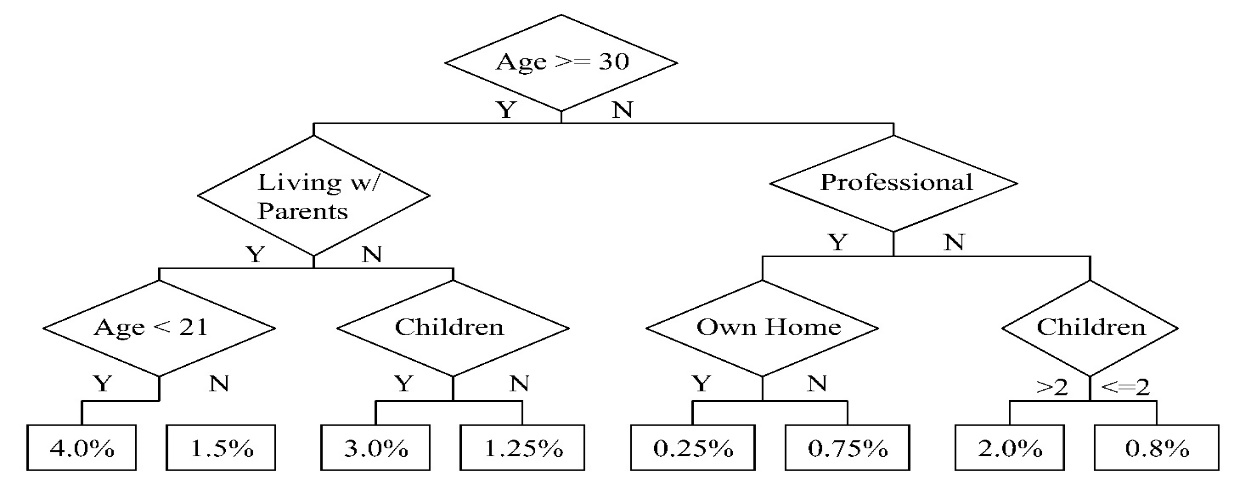
IF conditions are met, THEN a decision is taken.

Classification rules are a helpful technique to describe information in an easily interpretable manner. They suggest the appropriate options to make when the criteria are satisfied. In the instance of classification, the rule is referred to as classification and is as follows:

IF conditions are met, THEN a category is created.

The rule’s left side is a collection of criteria put on the attribute values. At the same time, the right side identifies the object’s category. The classification tree’s leaves correspond to the classification rules for a specific cut-off point. Changing the cut-off point may alter some of the rules, causing items to be allocated to a different class than before (Kaszyński et al., 2020).

The splits in the example depicted in Figure 2.4 are calculated from the top down. The *root node* is at the top of the tree, each succeeding level is referred to as a *child node*, and the *terminal nodes* are at the bottom. There will be two or more splits each time, and the number of levels will vary based on the branch. When completed, the values of the terminal nodes may be utilized as estimations (scores) or as a grouping tool. For a binary result, the value is a probability, such as P(Bad), and any branches with probabilities more significant than a preset cut-off, such as greater than the average P(Bad), are classified as Bad (Anderson, 2007).

**Figure 2.4: Decision tree**

[Source: Reprinted with permission from Anderson (2007, p. 173)]

The fundamental approach used is called a Recursive Partitioning Algorithm (RPA) which outlines the methods of finding the branches through repeated attempts to locate the optimal split. The following criteria define the RPA process:

* **binning**, ascertaining how predictors should be binned;
* **splitting**, choosing which characteristic to utilise;
* **stopping**, when to stop the creation of new sub-nodes;
* **pruning**, how to remove nodes to prevent overfitting;
* **assignment**, how to categorise each node as good or bad (Anderson, 2007).

**(A) Splitting Criteria**

Selecting the optimum test for dividing the set entails deciding on the characteristic to test and the available test outcomes. Each node of the same tree may have a unique test set accessible. Additionally, tests conducted at distinct nodes may be based on the same variable yet provide distinct sets of findings. Typically, the functions presented below are employed as criteria for dividing a set in algorithms that minimize the diversity measure. In practice, the disparity between defaults and non-defaults is significant in credit assessment (Kaszyński et al., 2020). The various approaches to the splitting criteria, such as entropy and information gain, information gain ratio, Gini index, Twoing criteria and CHAID, are discussed below.

**(I) Entropy and Information Gain**

One of the factors for selecting the test is to increase the expansion of information (Quinlan, 2014). The amount of information or entropy included in a collection of labelled instances is proportional to the number of categories in the set. The set’s entropy is proportional to its impurity and is denoted by the formula

where

C is a set of categories described by the dependent variable;

*X* is the entire set;

*X d* is a set of observations.

This value shows the amount of information required to classify a randomly picked sample correctly. The set’s minimal entropy value may be zero, but this is only possible if the set is entirely homogenous. The maximum entropy value varies according to the number of object categories. However, it is always obtained when the set’s category distribution is uniform. The goal of test selection and set division is to decrease the set’s entropy.

The Entropy of the set of observations with the result *r* in test *t* is given by:

where

is the set of observations with the result *r* in test *t*;

is the set of observations with the result *r* in test *t* and with category *d*;

The entropy of the entire set *X* after applying test *t* is a weighted average of the entropies for all available results for a given test, *r ∈ Rt* . and is given by:

The set’s increase of information after applying test *t* is expressed as:

*gt*  ( *X* ) = *I* ( *X* ) − ( *X* ) (2.19)

The test that results in the maximum increase in information in the set after its division, that is, *gt* ( *X* ), is chosen as the optimal. This is comparable to selecting a test that minimises the set’s entropy after division (Kaszyński et al., 2020).

**(II) Information Gain Ratio**

The information gain ratio, another criterion for the division of the set (Quinlan, 2014), uses the information gain mentioned above. The value of the following ratio determines the test’s choice:

*grt*  ( *X* ) = (2.20)

where *IVt* , test *t*’s informational value,is given by:

The optimal test is chosen as the one that maximizes the information gain ratio, *grt* ( *X* ).

The information value is independent of the distribution of classes within the subsets produced. It solely quantifies the homogeneity of the set’s subset distribution. This indicator has a high value when the set is subdivided into similar-sized subsets. On the other hand, this number drops when one of the subsets is much bigger than the others. Dividing the gain in information by the information value helps to normalise the *gt* . The downside of the information gain criterion is that it favours tests with a large number of results even when this is not warranted, i.e. when the chosen test is not truly more effective than alternatives with fewer results (Quinlan, 2014). Due to the imposed normalization, the information gain ratio criterion does not have this drawback. Additionally, the information gain associated with a bias for tests with many results and the most homogenous sets may result in overfitting models. Using the information gain ratio rather than information gain helps avoid this issue (Kaszyński et al., 2020).

**(III) Gini index**

The Gini index, another criterion indicating a test for minimising set impurity after division (Breiman et al., 1984), is given for a set of labelled samples by:

where the terms have the same meaning as in equation (2.6).

The Gini index for the entire set *X* after applying the test *t* is the weighted average of the Gini indexes for all available test results and is given by:

The optimal test is selected as the one that minimizes the Gini index, *GIt* ( *X* ) (Kaszyński et al., 2020).

**(IV) Twoing Criterion**

Twoing Criterion is a function applied to binary trees, i.e. trees in which two subsets are generated each division (Breiman et al., 1984), one of them being expressed by the index L (left) and the other by R (right). The function determining the subset’s purity is:

where

*PL* or *PR* is the share of observations assigned to the left or right subset respectively in the entire set *X*;

or is the share of observations with category *d* in the left or right subset, respectively.

The optimal test is selected as the one that maximises (Kaszyński et al., 2020).

**(V) The *χ* 2 statistics (CHAID)**

Chi-square Automatic Interaction Detection (CHAID) is an example of an algorithm based on *χ* 2 statistics (Kass, 1980). This statistic is used to assess the independence of two variables having discrete distributions. The null hypothesis is that the variables are independent. It is used in decision trees to determine the probability of independence of a dependent variable with nominal or categorical explanatory variables. A contingency table with a dependent variable having k values is constructed for each explanatory variable having m values. The table summarises the frequency of occurrences of the *i*-th explanatory variable’s value and the *j*-th dependent variable’s value for *i* = 1,…, m and *j* = 1,…, k. The *χ* 2 statistic comparing the observed frequencies of cumulative occurrences in the table with theoretical values calculated under the assumption of the independence of variables is given by:

where

is the frequency of common occurrences of the explanatory variable’s *i*-th value and the depen-dent variable’s *j*-th value;

, the expected value of the frequency of common occurrences assuming the independence of the variables, is expressed as:

= (2.26)

where

is the number of occurrences of the *i*-th value of the explanatory variable;

is the number of occurrences of the *j*-th value of the explanatory variable;

is the total number of observations.

The statistics constructed in this manner follow a *χ* 2 distribution with degrees of freedom (m - 1)(k - 1). It computes the difference between the observed and theoretical distributions, assuming that the variables are independent. The bigger the value of this statistic, the more different the distributions are. Exceeding the critical value contradicts the null hypothesis at a certain confidence level. The test with the greatest statistic value will be chosen using this criterion (Kaszyński et al., 2020).

**(VI) Comments on the criteria:**

It is tough to choose one of the above criteria that consistently produces the greatest outcome. Using various criteria might result in trees with a variety of architectures. Additionally, the form of a tree is determined by the number and structure of classes in a collection. When there are several classes of items in the set, i.e. when the explanatory variable has a large number of values, the comparison of the test selection criteria for binary splits reveals considerable disparities in the produced tree topologies. In the instance of credit scoring, when the dependent variable has just two classifications and two values, all the criteria provide comparable results (Breiman, 1996; Breiman et al., 1984). When non-binary partitions are used, the Twoing criterion is not applicable. However, the Gini index and Entropy still provide comparable findings. Both criteria produce homogenous nodes, but entropy produces more balanced partitions regarding the number of observations in the subsets, which may be undesirable in an imbalanced sample (Kaszyński et al., 2020).

**(B) Binning and variable report**

Binning is a data pre-processing method involving discretizing a continuous variable by allocating its values to a series of bins. This method contributes to the variable’s manageability, for example, by bucketing outliers in the lowest or highest intervals of the range with less extreme values. In this method, outliers become identical to other values in the distribution’s tail. Binning may also resolve issues caused by the variable’s high degree of skewness. Binning may be unsupervised (bins are created solely based on the variable’s distribution) or supervised (bins are produced using external information, e.g. target variable). Fitting a decision tree to predict target variables, such as default rates, using just the variable the analyst wishes to discretize is one of the supervised ways of binning. It boils down to finding a split that minimises the Gini coefficient or entropy. The resultant leaves (or bins) naturally exhibit decreasing entropy (Kamiski & Zawisza, 2012). In reality, variable binning should be performed for logistic regression; for decision trees, feature binning should be performed inside the algorithm (generation of leaves) (Kaszyński et al., 2020).

**(C) Stopping criteria**

Stopping criteria determine whether or not the tree-building process should be halted. A tree that has been over-expanded is overfitted to the training set. As a result, its quality is often substantially worse on the test set. Preventing a tree from overfitting is accomplished via the use of halting criteria and tree trimming. The halting condition may prevent overfitting by prohibiting further division of the node. The criteria for halting may also be technological in character. These technical requirements include the following:

* set homogeneity, i.e. when a node contains only data from a single class;
* absence of accessible tests, i.e. when all observations in a node have the same values for the explanatory variables.

The following criteria are used to prevent a tree from becoming overfit:

* maximum depth of the tree – when a tree reaches the maximum depth specified in the process parameters, the tree’s construction is complete;
* minimum node size – a node will not be split if its child nodes are less than the specified minimum node size;
* minimum leaf size – a node will not be split if the obtained leaves are less than the specified minimum leaf size; the node will become a leaf;
* Minimum leaf purity - if node’s purity does not fall below a certain threshold, the node becomes a leaf;
* minimal increase in purity – if the increase in purity is less than the provided amount, the node will not be split:

< *β*  (2.27)

where *X*  is the node to be split;

*T* is the set of available tests;

is the increase in the purity of X after applying test *t*;

*β*  is the minimum required increase in purity.

If node *X* purity is measured by entropy, then = *gt*(*X*) (Kaszyński et al., 2020).

**2.2.1.3 Advantages of classification trees**

* A classification tree’s structure enables easy comprehension of the resulting relationships and categorization criteria, even those without sophisticated statistical expertise. The tree’s shape illustrates precisely which variables, how they interact, and in what sequence they impact the assignment of data to certain categories. The relevance of specific variables (which indicates how much it helps reduce impurity in the derived subsets) within the tree structure and hence within the classification process may be determined.
* Classification trees are an excellent way to visualise the non-linear connections in the data. They may be constructed using any variable type - real, ordinal, or nominal. Moreover, they do not need any specific variable processing prior to model construction, but they may be treated in the same way as other approaches; for example, binary variables or weighted averages can substitute nominal variables. Furthermore, these methods perform well when dealing with outliers, missing numbers, and errors in data. There is no need for the pre-selection of variables.
* A tree-building approach based on recursive divisions enables the selection of the most relevant variables for the classification process. They are invariant to numerical variable monotone transformations. However, monotonicity restrictions may be implemented, which increases the transparency and compliance of derived models with accessible information (Kaszyński et al., 2020).

**2.2.1.4 Disadvantages of classification trees**

* A minimal change in the value of a single variable may often totally alter the categorization result. On the other hand, a minor modification to the data set may change the tree’s overall structure.
* The interpretability of decision trees rapidly decreases as the number of nodes increases (Kaszyński et al., 2020).

**2.2.2 Random Forest**

**2.2.2.1 Concept**

A random forest consists of a collection of several categorization trees. It is generated using a bagging algorithm, in which individual trees are formed using randomly chosen bootstrap samples from the training set. The usage of random forests enables a reduction in forecast variance compared to using a single classification tree with a high variation (Hastie et al., 2009). The variance reduction occurs due to averaging the findings from separate trees. The generalisation error in a random forest is dependent on the number of trees, the predictive capability of individual trees, and the correlation between the trees (Breiman et al., 1984). In the case of that technique, a monotonicity constraint on the specific characteristics that are beneficial and crucial in credit scoring from the business and regulatory point of view may be imposed both in the case of a single tree or entire forest granularity (Kaszyński et al., 2020).

**2.2.2.2 Construction of trees in a random forest**

The random forest is constructed using independent bootstrap samples drawn from the training set. This reduces the degree of correlation between the trees. All trees are constructed without being pruned. A subset of variables is drawn in each node during creating a single tree, which will be used to determine the definitive test that separates this node. Each node has its own set of variables. Each of the acquired classifiers is used to classify the observations, and the ultimate choice is made by voting. A significantly predicted variance defines a single decision tree. The forecast variance is reduced by averaging the findings from several trees in a random forest. The mean-variance of *B* identical random variables with variance *σ 2* and a correlation coefficient between two variables *ρ* is given by

(Hastie et al., 2009):

Hence, the decrease of the average result’s fluctuation is dependent on the number of trees B and the correlation coefficient. The correlation between individual trees may be reduced by using separate bootstrap samples and independent sampling of a subset of variables prior to each division. A subset of *m* ≤ *p* variables is chosen from among p variables, assuming that *m* = √*p*. To ascertain the classification error of a random forest, it is advised to judge for each observation using only trees that were not constructed from the sample containing that observation (Kaszyński et al., 2020).

**2.2.2.3 Significance of variables in a random forest**

Like individual classification trees, random forests allow one to assess the importance of specific variables in model creation. Several strategies for identifying the relevance of variables based on defined heterogeneity indicators are discussed, for example, in (Breiman, 2001; Louppe et al., 2013). Generally, the Gini index is often used to assess the relevance of variables, and this metric is known as Mean Decrease Gini (Louppe et al., 2013). This metric depicts the average drop in the Gini index throughout the set due to the application of a particular variable. The factors that contribute the most to lowering the Gini index in a set are the most significant. Shapley values (Molnar, 2019) and Variable Importance Measure (VIM) computations are two other methods for evaluating the relevance of variables in a random forest (Breiman, 2001).

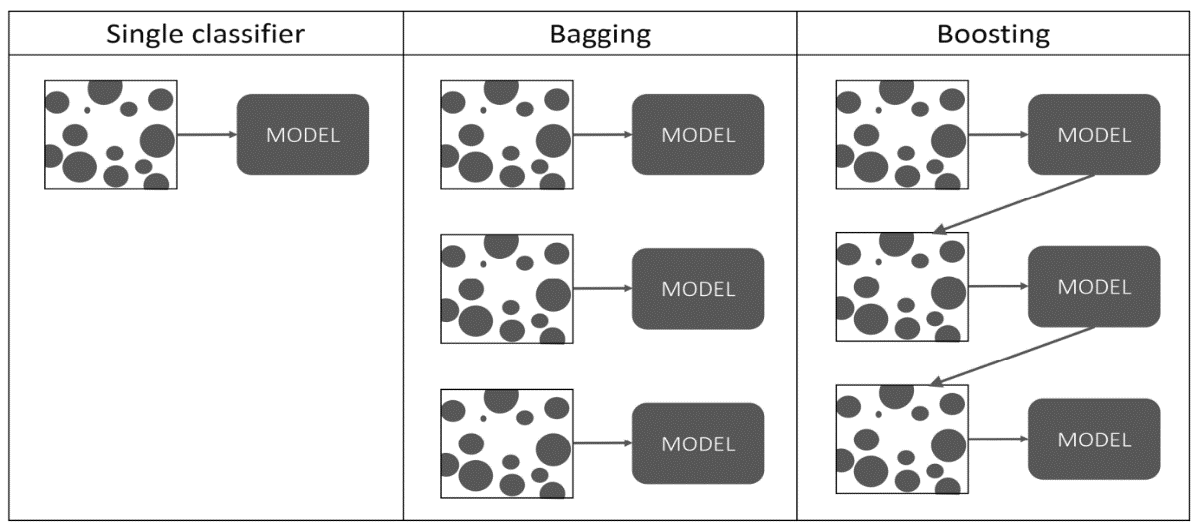
**2.2.3 Extreme Gradient Boosting (XGBoosting)**

**2.2.3.1 Introduction**

XGBoost is another popular tree ensemble model. Since its first release in 2014, it has shown to be a quick and effective machine learning algorithm and a winning approach in various data science contests (Nielsen, 2016). XGBoost, like the random forest, is an ensemble learning approach, which means that the outcome is derived from an aggregated output of numerous models. Decision trees have a significant variance, and ensemble learning may assist in lessening this tendency. It is based on the idea that averaging a group of data lowers variance (Hastie et al., 2009). In such a technique, the monotonicity constraint may also be imposed on the specific characteristics in both a single tree or the entire forest granularity (Kaszyński et al., 2020).

**2.2.3.2 Ensemble Learners**

There are two types of ensemble learners: bagging and boosting. In the case of bagging, it makes no difference which model was developed initially as the models are trained in a parallel manner. In contrast, boosting models are trained in a sequential manner with each model learning from the preceding model’s error. Boosting is built on the same ideas as bagging. However, it addresses one of the bagging’s limitations, i.e., all models may be incorrect in the same region and make errors regarding the same data. Boosting overcomes this problem by chaining the models - generating one tree leads to creating another. In addition, each model focuses on areas where its predecessor underperformed (Freund et al., 1999). The following figures illustrate these concepts.

****

**Figure 2.5: Comparison of decision trees ensemble methods**

[Source: Reprinted with permission from Kaszyński et al. (2020, p. 119)]

**2.2.3.3 Adaptive Boosting and Gradient Boosting**

In the case of Adaptive Boosting (abbreviated as AdaBoost), successive trees learn from the errors of the preceding model by altering weights appropriately. The greater the inaccuracy, the greater the weight for this observation. Therefore, the penalty of a faulty forecast is also more significant. However, they are not the only weights in the algorithm. Additional weights of the tree, referred to as Amount of Say, are linked and originate from the tree’s prediction quality. In this model, the better the tree, the more critical the boost provided to incorrectly predicted observations or significant residual. Let us look at an example of how such a system may operate. The following measures are computed after beginning with equal weights and fitting the first tree:

Amount of Say:

where Total Error is a selected weighted error measure or weighted error rate. It is to be noted that a classifier with 50% accuracy with the Amount of Say equal to zero will not contribute to the final prediction.

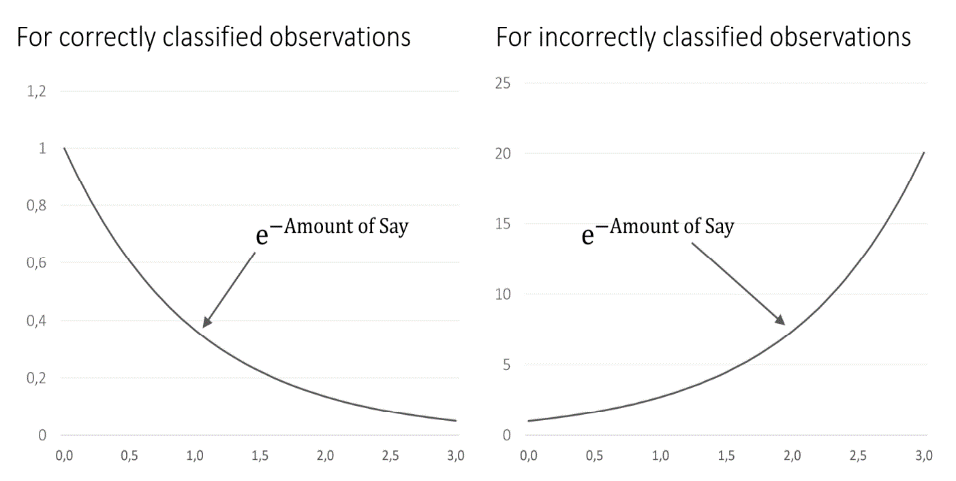
New Sample Weight for correctly classified observations:

(2.30)

where Sample Weight refers to the sample weight used in the algorithm’s last iteration.

New Sample Weight for wrongly classified observations:

(2.31)   
Following that, weights are adjusted to equal one. A graphic representation of the Amount of Say and the updated sample weights are depicted in Figure 2.6 below (Kaszyński et al., 2020).

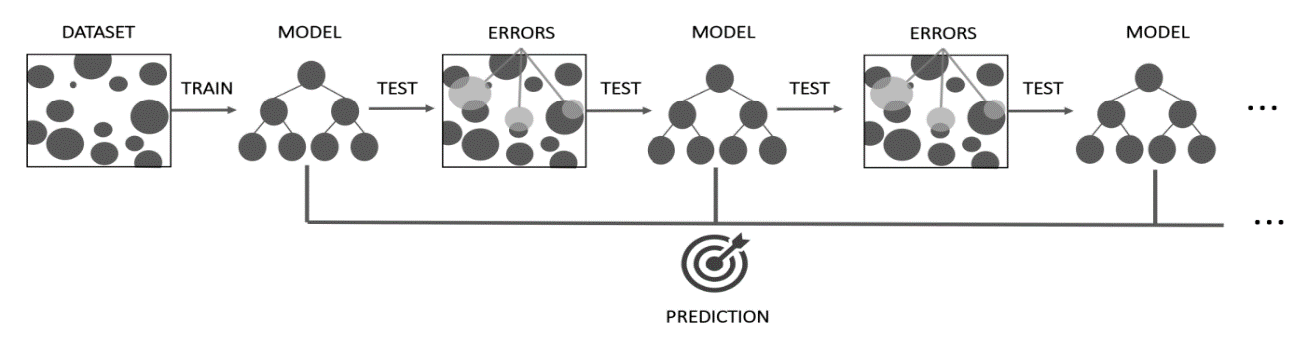


**Figure 2.6: Visualization of the calculation of new sample weights**

[Source: Reprinted with permission from Kaszyński et al. (2020, p. 121)]

When the Amount of Say is large, the sample weight is scaled by a factor close to zero in the case of correctly classified samples and close to 1 for the misclassified samples. The more complex the tree, the more weight will be assigned to misclassified observations. When the Amount of Say is low, multipliers will have comparable values; for example, for the poorest feasible tree with the Amount of Say equal to zero, both multipliers will be equal to one, with no change in weight. In contrast to a random forest, where trees are developed to their greatest depth, AdaBoost trees are often produced with few splits. Often, they just have one split and are referred to as stumps. Combining numerous weak learners may create a robust and robust model with a high predictive value. While AdaBoost is severe in this regard, ordinary tree depth in Gradient Boosting is somewhat greater, generally between 2 and 8. It is instrumental in credit scoring, where ABT variables are often generated at a depth of 2 to 3. The approach enables us to determine how to generate certain variables in the ABT by examining a subset of the most significant trees. Gradient Boosting with a logistic objective function may be seen as a linear model constructed from ABT variables created automatically. Another significant distinction between Adaptive and Gradient Boosting is that the latter does not rely on weights to fix prior models’ faults; instead, models learn directly from their predecessors’ errors (Kaszyński et al., 2020).

The following diagram illustrates the Gradient Boosting algorithm’s high-level architecture.

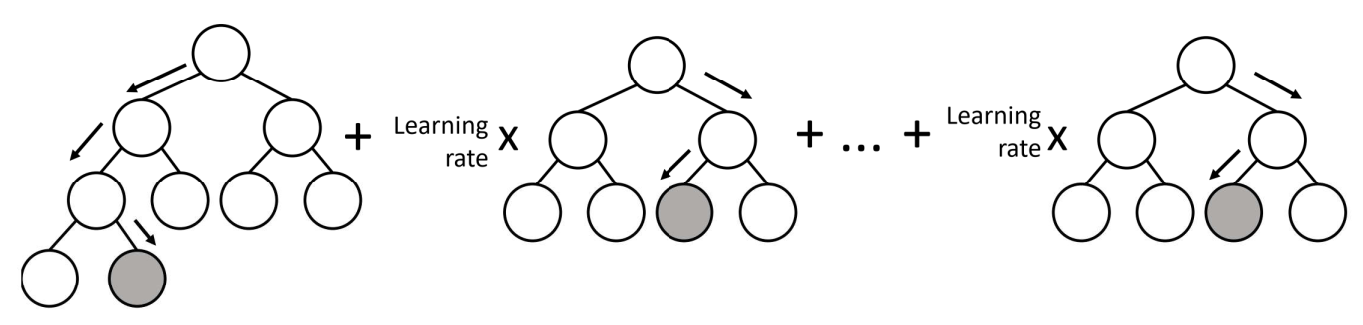


**Figure 2.7: High-level architecture of the gradient boosting algorithm**

[Source: Reprinted with permission from Kaszyński et al. (2020, p. 121)]

**2.2.3.4 Structure of Gradient Boosting**

GradienBoosting’s fundamental structure may be summarised as follows in the context of a regres-sion issue. Beginning with a fundamental learner and depending on the algorithm’s implementation, the first estimate may be a stump, a tiny decision tree, or even a linear model. A primary learner in its simplest form would be a single leaf with a single value given to all observations, i.e., the response variable’s average. Once a reasonable estimate is obtained, the not-too-advance model’s residuals may be calculated. The next iteration will simulate these residuals rather than the actual response variable. The prior estimate is modified with modelled residuals with each iteration by adding the prediction. Notably, while computing the outcome, a hyperparameter called learning rate is employed to avoid overfitting and make only incremental moves in the proper direction. For instance, 10% of it may be added instead of adding the projected residual in its entirety. As a result of this addition, more decision trees are used to arrive at the ultimate solution. When predictions are made after developing the subsequent trees, the initial forecast from the base learner is added to all of the anticipated residuals, each of which is multiplied by the defined learning rate. Figure 2.8 below presents the form of the final prediction (Kaszyński et al., 2020).



**Figure 2.8: Visualization of prediction making in Gradient Boosting**

[Source: Reprinted with permission from Kaszyński et al. (2020, p. 121)]

The successive models are trained directly on the residuals in the preceding description. Gradient Boosting is used in this example to solve a Mean Squared Error loss function. However, Gradient Boosting is indifferent to the loss function working with all differentiable loss functions. Gradient Descent is a well-known iterative optimization technique for optimising function parameters (θ) by minimising a loss function (L) that assumes the predicted values of the response variable Furthermore, *y* itself and produces a loss value. The hyperparameter *η* is the learning rate at which the gradient of the loss function L is scaled down.

) (2.32)

Comparing this to the example given above:

) (2.33)

) (2.34)

The function architecture F is optimised instead of the parameters of a function. The gradient of the loss function is referred to as pseudo residuals, which become true residuals in the case of MSE loss, as shown below, with a constant added to simplify calculations (Kaszyński et al., 2020):

(2.35)

**2.2.3.4 Extreme Gradient Boosting (XGBoosting)**

The most prevalent Gradient Boosting implementation is XGBoost, which significantly enhanced the algorithm’s capabilities. The X in its name stands for extreme. Upgrades and enhancements are many methods, procedures, and settings that a user may configure (Chen & Guestrin, 2016). As a result, we shall limit our discussion to the method’s fundamental capabilities. The primary benefit of XGBoost is its sophisticated method for avoiding overfitting. In addition, basic Gradient Boosting already provides a learning rate for lowering weights (eta in XGBoost).

On the other hand, this implementation went far further in this regard. *Gain* is a critical measure introduced in XGBoost. It is a measure of the feature’s relative contribution to the model. The greater the value, the more significant a divide is. Apart from terminating fitting when a node reaches a preset maximum depth, it also enables automated tree trimming through *Gain*. XGBoost evaluates if the *Gain* from a particular split falls below gamma, a threshold of *Gain* enhancement necessary to maintain a division, starting at the leaves level. If *Gain* is less than this value, the node is pruned, and the process proceeds to the root branch by branch.

On the contrary, where the node’s *Gain* value is more significant than gamma, not only is the node retained but the parent nodes are not pruned. This is because, in the end, it makes little difference if a good split occurs after a very bad one; it is still a split favourable to the model. A regularisation parameter λ adds an additional degree of complexity to this operation. It is a constant used in *Gain* and forecast calculations. It affects *Gain* by being added to the denominator of *Gain*. As a result, by lowering *Gain*, each tree becomes more pruneable and less prone to overfitting. The greater the parameter value, the more conservative the model is. It is worth mentioning that lambda has an asymmetric effect on *Gain*. The fewer the observations in a node, the greater the influence of lambda, which is intended to combat overfitting. As indicated by Chen, the creator of the approach, xgboost uses a more regularised model formalisation to control overfitting, which offers it more excellent performance (Hastie et al., 2009). XGBoost enables parallel processing and distributed computing, and bespoke optimization targets. It provides an in-built technique of managing missing values and executes cross-validation at each iteration. It enables optimal memory consumption and is highly scalable. The basis of XGBoost is Gradient Boosting, which has expanded systems and algorithmic optimization (Kaszyński et al., 2020).

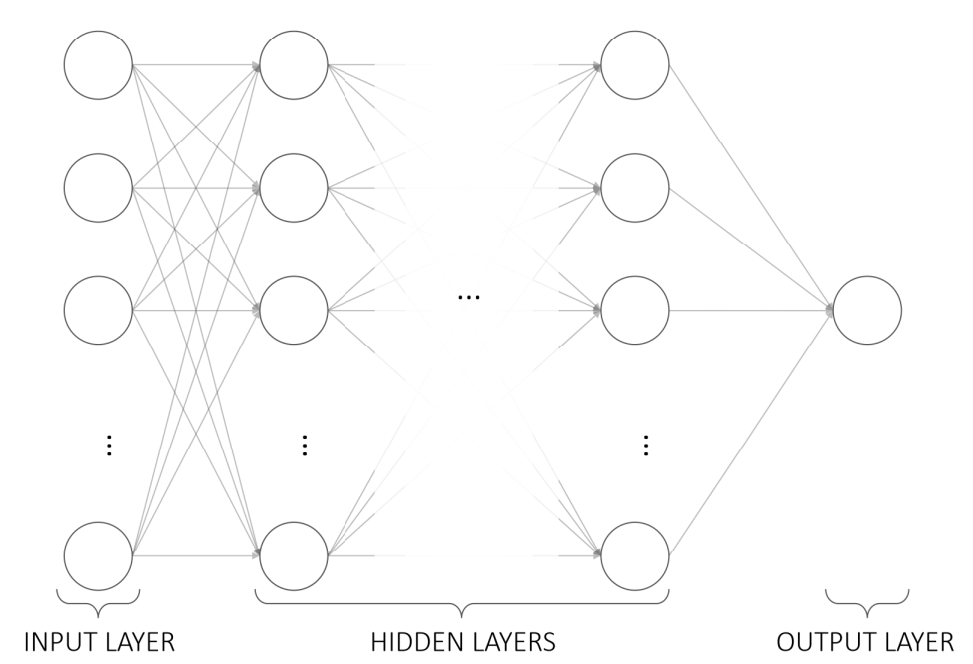
**2.2.4 Relu neural network**

The *relu* abbreviation stands for Rectified Linear Unit. In essence, the relu neural network is a linear function nested within a non-linear (i.e. rectifier) activation function. Indeed, suppose the sigmoid function is substituted for relu. In that case, logistic regression assumes that the neural network is single-layer only. The *vanishing gradient problem* associated with sigmoid activation functions results in multilayer neural networks starting using the relu activation function by default. The activation function of a rectifier (relu) is given by:

*f* (*x*) = *x*+ = max (0, *x*) (2.37)

where *x* is the product of node’s inputs and its weights.

The following diagram illustrates the structure of a neural network consisting of *layers* and *nodes*:



**Figure 2.9: Structure of a multilayer neural network**

[Source: Reprinted with permission from Kaszyński et al. (2020, p. 121)]

The neural network presented above has defined four layers (input layer, two hidden layers and output layer), with the set of 4, 4, 4 and 1 nodes corresponding to each layer. The presented architecture is universal (i.e. more layers, more nodes). For each layer in our example, the activation function is assumed to be relu.

The output is just the activation function evaluated on the node’s summed inputs multiplied by the node’s weights at the node level. Unfortunately, that mapping does not allow for modelling complicated, non-linear patterns in the data. As a non-linear activation function, the relu activation function is favoured because it enables nodes to express more complicated patterns in the data. The neural network is trained by maximising the weights as shown as the edges in Figure 2.9 in terms of the loss function. The advantages of relu neural network are:

* As the loss function is convex, the uniqueness of the solution (unimodality) is guaranteed.
* Absence of the vanishing gradient problem and the possibility to generate fast predictive algorithms.
* Absence of the randomness in the weights initialization and the problems with escaping from the local optima in terms of the model’s parameters (Kaszyński et al., 2020).

**2.3 Problem of Multicollinearity**

**2.3.1 Concept**

Multicollinearity of explanatory variables occurs when a significant linear relationship exists between them in multiple regression. The factors responsible for multicollinearity may be categorised as follows:-

* **Input data**: using explanatory variables with high linear correlation, a large number of explanatory variables, a small number of observations, badly-designed data collection methods, etc.,
* **Incorrect model specifications**: including the same number of binary variables as categories when converting categorical variables into binary variables, including an explanatory variable obtained as a result of calculations on other explanatory variables, etc.

The issue of multicollinearity may be explained intuitively in a graphical way using Venn diagrams through the following example.

# Basic Linear Regression Model: a graphical example

For this example, we simulate correlated data in continuous variables using the Cholesky decomposition method. This method reduces a symmetric matrix into a lower-triangular matrix which, when multiplied by its transpose, produces the original symmetric matrix

cholesky

Intial Matrix (A)

[ [36 30 18]

[30 41 23]

[18 23 14] ]

Cholesky decomposition takes the form: A = L x L\*

Lower triangular matrix L: Conjugate transpose of L (aka L\*):

[ [6. 5. 3.] [ [6. 0. 0.]

[5. 4. 0.] [0. 4. 2.]

[0. 0. 1.] ] [3. 2. 1.] ]

Multiplication of L x L\* that produces matrix A

[ [36. 30. 18.]

[30. 41. 23.]

[18. 23. 14.] ]

Now we simulate our data for our example. For this, we find the Cholesky decomposition of the covariance matrix. In this case, the covariance matrix values are the same as the desired correlation matrix. Once calculated, we multiply the Cholesky decomposition by the matrix of uncorrelated random variables to create correlated variables.

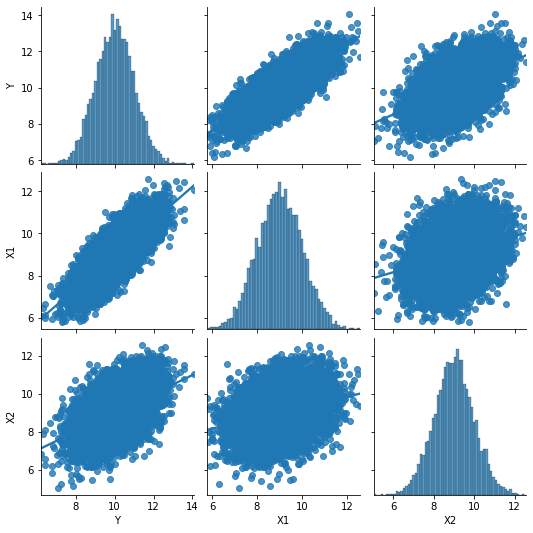
Desired correlation matrix:

0 1 2

0 1.0 0.8 0.5

1 0.8 1.0 0.3

2 0.5 0.3 1.0

Final correlation matrix from simulated data and the pair-wise correlation plot:

|  | **Y** | **X1** | **X2** |
| --- | --- | --- | --- |
| **Y** | 1.0 | 0.80 | 0.50 |
| **X1** | 0.8 | 1.00 | 0.28 |
| **X2** | 0.5 | 0.28 | 1.00 |

**Figure 2.10: Pair-wise Correlation**

(Source: Own work)

Now we have the following dataset with correlated variables and their descriptive statistics:

|  |  |  |  |
| --- | --- | --- | --- |
| Table 2.1: Data set | | | |
|  | **Y** | **X1** | **X2** |
| **0** | 8.914369 | 7.386914 | 9.540699 |
| **1** | 10.997345 | 9.610108 | 10.682766 |
| **2** | 10.282978 | 8.717015 | 9.364432 |
| **3** | 8.493705 | 9.221736 | 8.630311 |
| **4** | 9.421400 | 8.931620 | 6.894046 |
| **5** | 11.651437 | 10.449001 | 9.220025 |
| **6** | 7.573321 | 6.764074 | 9.133948 |
| **7** | 9.571087 | 8.007964 | 8.831282 |
| **8** | 11.265936 | 10.015630 | 9.936395 |
| **9** | 9.133260 | 8.090130 | 8.821305 |

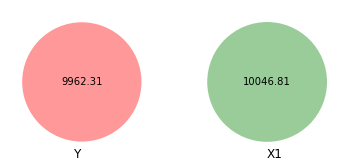
|  |  |  |  |
| --- | --- | --- | --- |
| Table 2.2: Descriptive statistics | | | |
|  | **Y** | **X1** | **X2** |
| **count** | 10000.000 | 10000.000 | 10000.000 |
| **mean** | 10.009712 | 9.013995 | 9.002930 |
| **std** | 0.998164 | 1.002388 | 0.996124 |
| **min** | 6.198622 | 5.793856 | 5.041916 |
| **25%** | 9.330154 | 8.330309 | 8.347290 |
| **50%** | 10.011846 | 9.008629 | 9.009639 |
| **75%** | 10.686122 | 9.703213 | 9.652050 |
| **max** | 14.068097 | 12.571978 | 12.575859 |

In the following regression diagrams, each variable is shown as a circle sized according to its variance, or:

According to Ip (2001), you can also size the circles based on just the numerator of that equation, or the sum-of-squares:

(2.39)

## **Relationship between 2 variables (Simple Regression Example)**

Now we look at the relationship between Y and X1. We will use the sum of squares to calculate the size of each circle.

**Figure 2.11: Variances of variables**

Source: Own work)

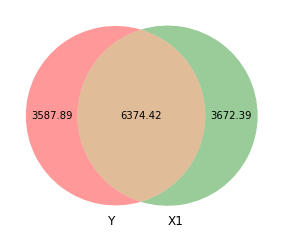
Looking at the above Venn diagram, we can see that Y has more minor variation than X1.

These two variables are related to each other and have some covariance. We can calculate the shared covariance using ANOVA. The nice thing about using the sum of squares values for these diagrams rather than variance values is that the function anova\_lm from python reports its results as sums of squares to use those results directly.

Let us see how much variation between Y and X1 is shared.

| **Table 2.3: ANOVA** | | | | |
| --- | --- | --- | --- | --- |
|  | **sum\_sq** | **df** | **F** | **PR(>F)** |
| **X1** | 6374.423827 | 1.0 | 17762.94982 | 0.0 |
| **Residual** | 3587.888840 | 9998.0 | NaN | NaN |

If we look at the sum of squares row, we can see that 6374.42 sum of squares units (whatever those mean) are shared between the two variables, with 3587.88 not shared (or residual).

****For visualising the shared space (variation) between both variables, we need to subtract the shared space from both Y and X1 as follows:

.

**Figure 2.12: Covariance between two variables**

(Source: Own work)

The above Venn diagram helps us to visualise the covariance between these two variables.

Area C here represents the amount of variation in Y explained by X1. In contrast, Area A represents the unexplained portion of Y. In regression language, A is the error term:

(2.40)

The great thing about visualizing this is that C also represents the *R*2. In general, *R*2 is the ratio between explained and total variance:

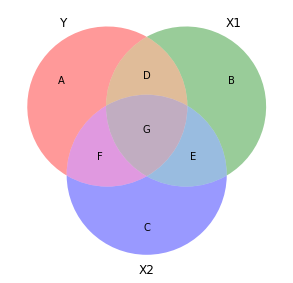
Based on this diagram, we can write this as:

Using actual numbers, we get:

According to this, X1 explains 64% of the variation in Y. That is apparent visually—the overlapping space covers about 64% of the total Y circle.

## **Relationship between three variables**

Now we look at the overlapping space between three different variables. Unfortunately, it is also a little more complicated since we need to calculate different shared variances and do some set theory calculations to find the exact size.

For example, the following diagram is exposed for illustration issues:

**Figure 2.13: Covariance between three variables**

(Source: Own work)

Here the same principles apply. The entire circles for Y, X1, and X2 represent each variable’s total variance. Overlapping areas represent shared variance. For instance, the combination of D and G here is the previously calculated covariance.

That means we can calculate *R*2 like this:

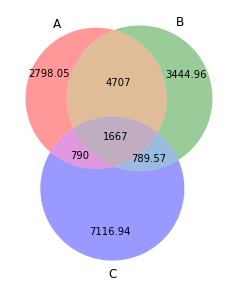
This visualization also helps with the intuition of *R*2. Generally, adding additional variables to a regression model increases the *R*2. That is because one is adding another circle to the diagram and absorbing more of the variation in the outcome.

For instance, even though the numbers in this diagram are not to scale at all, you can see that (D + G) (the *R*2 that we calculated in the two-variable diagram) is smaller than (D + E + G). So there is more explained variance here.

The following table helps us to understand the meaning of each segment:

|  |  |
| --- | --- |
| **Table 2.4: Variance** | |
| **Segment** | **Explanation** |
| A + D + E + G | Total variation in Y |
| B + D + F + G | Total variation in X1 |
| C + E + F + G | Total variation in X2 |
| A | Unexplained variation in Y after accounting for X1 and X2 |
| B | Unexplained variation in X1 after accounting for Y and X2 |
| C | Unexplained variation in X2 after accounting for Y and X1 |
| D + G | Variance shared by Y and X1 |
| E + G | Variance shared by Y and X2 |
| D + G | Variance shared by X1 and X2 |
| D | Variance only between Y and X1, without influence from X2 |
| E | Variance only between Y and X2, without influence from X1 |
| F | Variance only between X1 and X2, without influence from Y |
| G | Variance shared by Y, X1, and X2 |

(Source: Own work)

Here the same principles apply. The entire circles for Y, X1, and X2 represent each variable’s total variance. Overlapping areas represent shared variance. Now we calculate all the segments of the table, and based on the calculated areas (segments), we can plot them in a diagram:

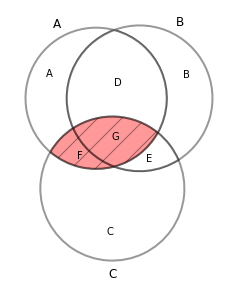
**Figure 2.14: Covariance between three variables (calculated values)**

(Source: Own work)

Once again, we calculate the *R*2 :

The calculated value *R*2 = 0.72 is confirmed with the regression model.

## **Multicollinearity Issue**

Another neat feature of this kind of diagram is that it helps visualize multicollinearity or the issues that arise when you control for explanatory variables that explain the same variation in the outcome. Multicollinearity leads to strange coefficient estimates and variance inflation because, mathematically, the regression model has no way of telling which of the highly correlated explanatory variables explain which parts of the outcome.

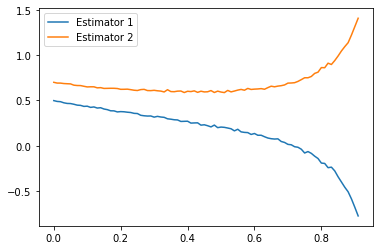
**Figure 2.15: Multicollinearity**

(Source: Own work)

In the above diagram, areas D and E are uniquely accounted for by X1 and X2, respectively. However, G is overlapped, making it impossible to know if X1 or X2 explains that portion of the variation in Y. Similarly, area F shows the variation shared by both X1 and X2. Again it is impossible to know which parts are unique. As a result, the area (F + G) represents the total multicollinearity in the model.

## **Experiment on regression coefficient under multicollinearity**

We repeat the same simulation for correlated data using the Cholesky decomposition method. We create a for loop that iterates for different values of *i* regarding the correlation coefficient in the correlation matrix (r). The different correlation values are for Estimator 1, and Estimator 2, from 0 to 93% of correlation are shown in the diagram below:



**Figure 2.16: Regression coefficient under multicollinearity**

(Source: Own work)

**2.3.2 Detection of multicollinearity**

The literature and practice propound two practical approaches for detecting multicollinearity: the Variance Inflation Factor (VIF) and the Condition Index (CI).

**(A) Variance Inflation Factor (VIF)** measures the increase of an explanatory variable’s variance based on its linear correlation with other explanatory variables. The value of the VIF for the *ith* variable is defined as:

where refers to the coefficient of determination for the linear regression of the explanatory variable *Xi* over the other explanatory variables *j* ∈{1,…, *m*}− {*i*}. The existing literature prescribes different levels of acceptance of VIF: from 5 (James et al., 2013) up to 10 (Hair et al., 1998). Welfe (2018) suggests using the regression models so long as > .

**(B) Condition Index (CI)**is calculated as:

where refers to the *i*-th eigenvalue of the normalized features matrix (Liao & Valliant, 2012).

In terms of interpretation, Kennedy (2003) recommends as a rule of thumb the usage of two limits for : greater than 15 may indicate collinearity problems while greater than 30 indicates strong collinearity. The table below presents an example of the calculation and values of CI (Kaszyński et al., 2020).

|  |  |  |
| --- | --- | --- |
| **Table 2.5: Calculation of Condition Index** | | |
| **Dimension** | **Eigenvalue** | **Condition Index** |
| 1 | 7.772 | 1.000 |
| 2 | 0.453 | 4.142 |
| 3 | 0.156 | 7.058 |
| 4 | 0.033 | 15.347 |
| 5 | 0.065 | 10.910 |
| 6 | 0.003 | 50.899 |
| 7 | 0.002 | 60.835 |

[Source: Adapted with permission from Kaszyński et al. (2020, p. 157)]

The above-described metrics, VIF and Condition Index, are sometimes referred to as alternatives. Their usage in conjunction is encouraged. The primary distinction in interpretation and application is included within the formulae. The VIF is computed at the single variable level. For example, the *i*-th variable is explained by the remaining features. At the same time, the Condition Index is computed at the model level.

**2.4 Omitted Variables Problem**

**2.4.1 The problem**

A significant issue that might result in bias in regression estimations involves omitting a necessary variable from the regression model. Let us consider the following population model

However, only the following regression is run:

Due to model misspecification, the bias may be defined as:

(2.50)

where  are slope estimators from the regression y*i* on  . is the slope from the regression *xi* two on *xi* 1.

may be treated as fixed quantity when computing E [] as depends only on the explanatory variables.

E [] = *E*[] = *E*[] = *β*1 + *β 2* (2.51)

thus implying that the omitted variable bias is *β 2*.

The bias in the model will be zero if *β 2* = = 0, which occurs if and only if *x*1 and *x*2 are uncorrelated in the sample (Stephenson, 2018).

**2.4.2 The remedies**

**2.4.2.1 Instrumental Variable Approach and 2SLS estimator**

The instrument variable technique and the two-stage least squares estimator are often employed to address omitted variable bias and other types of endogeneity difficulties (Angrist & Krueger, 2001; Wooldridge, 2008; Antonakis et al., 2010). It is relevant to observational research and experimental designs. The independent variables cannot be changed directly, for example, attitudes, emotions, expectations, choices, and behaviours (Sajons, 2020). Assuming the linear regression, the endogeneity issue appears when variable *q* is omitted since it shares variance with variables *x* and *y*. As a result, *x* comprises both - variations independent of q or exogenous variance and variance caused by *q* or endogenous variance. The instrument variable method is based on the premise that an instrument may forecast the exogenous variance of *x* and generate an exogenous independent variable, *x*. Thus, a consistent estimate of the relationship between *x* and y may be obtained. It is also feasible to instrument several endogenous independent variables using two-stage least squares (2SLS) estimator rather than the instrument variable estimator, albeit this needs at least as many instruments as endogenous variables (Angrist and Krueger, 2001; Wooldridge, 2008; Wilms et al., 2021).

**2.4.2.2 Regression discontinuity**

In experiments, the group assignment is carefully modelled in studies, with each participant having an equal chance of being allocated to a specific group or not. Regression discontinuity designs (RDD; Thistlethwaite and Campbell, 1960) also accurately reflect the process of group selection, albeit in a non-random fashion. According to Hahn et al. (2001), the RDD is a quasi-experimental design in which the probability of receiving treatment is related to and one or more underlying variables in a discontinuous manner. This is shown with an example. Assume we wish to determine the impact of a scholarship on future earnings. Regrettably, scholarships are not distributed randomly, so we cannot experiment. Rather than that, students are awarded scholarships depending on their academic performance. Thus, when a third variable, for example, student’s grade, exceeds a specific cut-off or threshold, *c*, say the 85th percentile, the treatment, for example, a scholarship is administered, and RDDs use the detailed knowledge of the treatment rules (Angrist & Pischke, 2008) to estimate a consistent treatment effect, for example, increase in future wages. Suppose the treatment has an effect on the dependent variable, a jump or discontinuity in the dependent variable at the location of the thirvariable’s cut-off, for example. In that case, students who did not receive the scholarship, that is, grades below the cut-off should have lower future wages than students who did receive the scholarship, that is, grades above the cut-off (Wilms et al., 2021).

**2.5 Negative betas problem**

**2.5.1 The Problem**

Considering a random dependent variable *Yn* associated with the status of the event *default* having the value *yn* , where *n* is the unique observation index, we are interested in the model explaining the relationship between the probability of the event *default* mathematically written as *pn* = *P*(*Yn* = 1) and predictors denoted as a sequence of explanatory variables where *m* is the number of variables. In the beginning, the regression part is defined, namely a combination of predictors:

where and ***β*** are the vectors of the values of the explanatory variables and the regression coefficients, respectively.

The logit function, as discussed earlier, is then given by:

Logit (*pn*) = (2.53)

where *pn* refers to the theoretical probability values of the event *default* for the *n*th observation.

The WoE transformation is a well-known notion called Weight of Evidence (Siddiqi, 2012). It is very similar to logit value because:

where

*k* means any variable category;

*Gk* and *Bk* refer to the number of good and bad customers respectively in the category and *G* and *B* in the entire population.

The Weight of Evidence (WoE) for a category is thus the difference between the entire population logit and the category logit.

Beta coefficients may be negative in the case of the WoE transformation. The negative betas show the undetected interactions between factors in the single factor analysis used to calculate the WoE. If negative betas are calculated, it suggests that the variable positively influences the target when considered alone, but has a negative effect when combined with other variables, a phenomenon known as Simpson’s Paradox.

### 2.5.1.1 The Simpson’s Paradox

**2.5.1.1.1 The paradox**

What happens when we allow data to speak for itself? Indeed, in many instances, everything is possible. However, the researcher may acquire contradictory findings depending on how the data are organised and which factors are included in the model (Simpson, 1951). Simpson’s Paradox is a statistical phenomenon in which a trend occurs in multiple distinct data groups but vanishes or reverses when these groups are joined. It has presented itself for decades in various scientific areas (Selvitella, 2017). To begin, let us suppose that the analyst has obtained the following contingency table during exploratory data analysis:

**Table 2.6: Contingency Table1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Particulars | | Default | | Default Rate |
| 1 | 0 |
| Self-employed | Yes | 41 | 1,996 | 2.01% |
| No | 48 | 2,114 | 2.22% |

[Source: Adapted from Kaszyński et al. (2020, p. 162)]

Based on the above contingency table, one would infer that full-time employees are riskier since the default rate for salaried persons is 2.22% and just 2.01% for self-employed customers. Unfortunately, such a straightforward one-dimensional connection seldom has sufficient predicting potential. Numerous real-world challenges in credit scoring or any other industry need the development of multivariate models to understand the complexity of the data generation process. To do this, further stratification is conducted, resulting in two more contingency tables. The first group includes just those with an income of less than or equal to $4,000, while the second group includes everyone else.

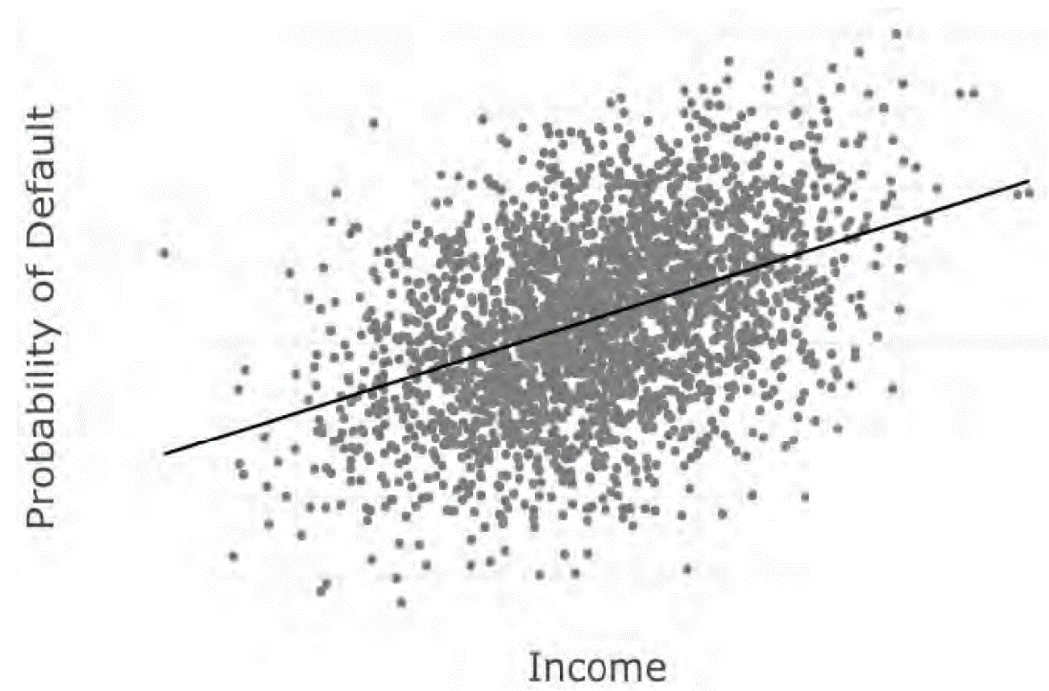
**Table 2.7: Contingency Table2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Income ≤ $4,000 | | Default | | Default Rate |
| 1 | 0 |
| Self-employed | Yes | 8 | 177 | 4.32% |
| No | 33 | 1,051 | 3.04% |
| Income > $4,000 | | Default | | Default Rate |
| 1 | 0 |
| Self-employed | Yes | 33 | 1,817 | 1.78% |
| No | 15 | 1,065 | 1.39% |

[Source: Adapted from Kaszyński et al. (2020, p. 163)]

The contents of these tables may surprise the analyst. Although, in contrast, full-time employees seemed to be riskier customers in general, additional stratification reveals the reverse pattern. Entrepreneurs have a greater default rate (4.32%) than full-time employees (3.04%) among low-income persons. The same is true for higher-income individuals, with default rates being 1.78% and 1.39%, respectively.

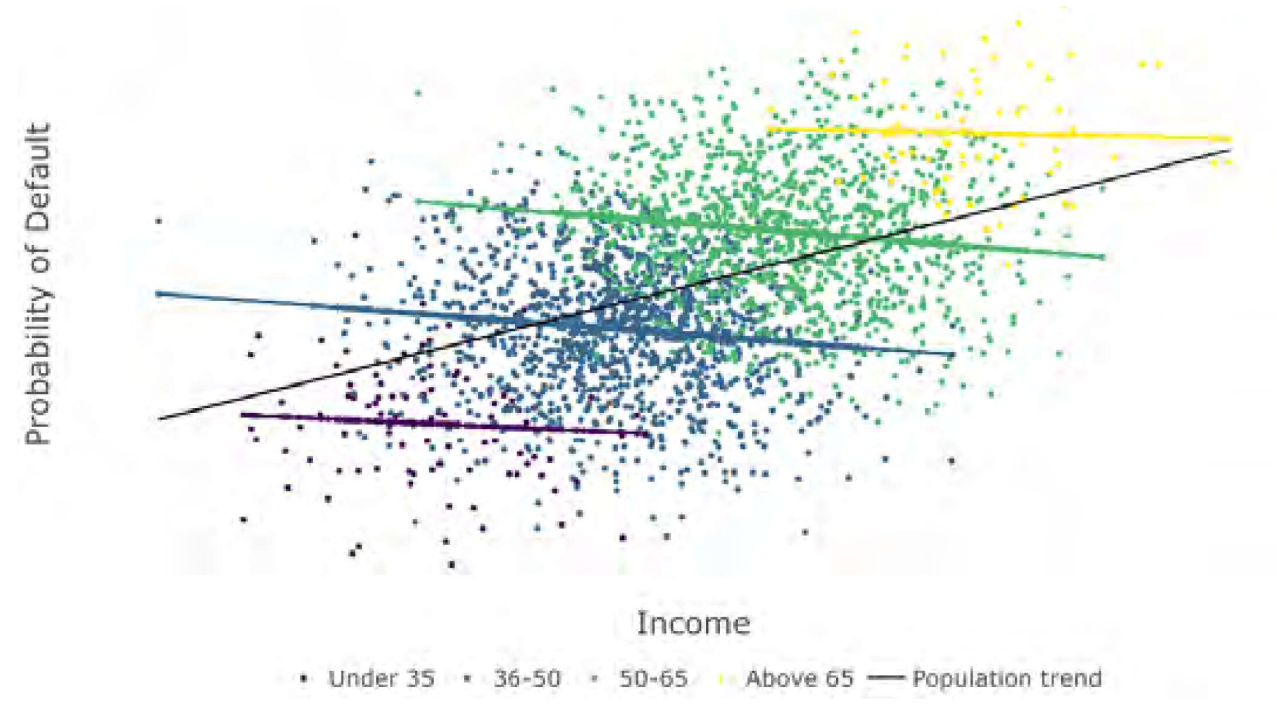
This behaviour is mathematically explicable. Each default rate in the stratified tables represents a proportion of loans defaulted within a particular category. They are then compared across sample sizes. In our case, the sample size mismatch is demonstrated by the fact that the full-time working group is split relatively evenly between lower- and higher-income groups in terms of clientele, but not equally in terms of bad loans, with the high-income group receiving the majority. The situation was very different for self-employed persons, with very few customers falling into the low-income category, and a significant number of them defaulted on their loans (Kaszyński et al., 2020). Simpson’s Paradox is not exclusive to categorical data; it may arise in various other situations as well. Let us consider another illustrative example of a scatterplot plotting the continuous variable Income versus Probability of Default depicted in the figure below.



**Figure 2.17: Probability of Default vs. Income level distribution**

[Source: Reprinted with permission from Kaszyński et al. (2020, p. 164)]

The scatterplot above clearly supports a positive association between income and default probability, implying that as income increases, the likelihood of default increases as well. Such a finding would seem contradictory to any credit analyst since more income often results in more consistent repayment of creditors. However, after adding another layer and estimating trends within age groups, not a single slope is positive, as seen in Figure 2.18 below; all the relationships are either negative or non-existent, which fits expectations (Kaszyński et al., 2020).



**Figure 2.18: Probability of Default vs. Income level stratified distribution**

[Source: Reprinted with permission from Kaszyński et al. (2020, p. 165)]

Simpson’s Paradox can arise in the case of continuous data either due to the data being clustered into distinct groups (as in the example above if we only had the categories of aged under 35 and over 65) or due to an underlying stratification within this group. These arguments are regularly seen in practice (Kaszyński et al., 2020).

**2.5.1.1.2 Tackling the paradox**

Let us return to the original contingency table example. What is the truth in this situation? Are self-employed individuals genuinely more dependable in general, despite being riskier consumers in both income groups? Regrettably, no general rule can be applied here. The same data may be used in this example to demonstrate either a high or low risk of self-employed customers, depending on the model utilised. As a result, statistics alone may not provide answers; a causal framework must be integrated into the research. Perhaps the data’s causal model looks like the income stratified tables and the conditional linkages are correct.

Additionally, the analyst may decide not to maintain the stratification because the income level was picked improperly, there should be many thresholds, or income plays no role in this process at all. There is an infinite number of options. This is why it would be simple for a human being assessing numerous stratifications one by one to overlook the proper choice or believe in an inaccurate one instead. While individuals are more aware of the data generation process than statistics, machine learning algorithms can evaluate substantially more data divisions than humans while remaining impartial. Occasionally people’s ideas are subjective or not mature or profound enough. This problem may be addressed by using techniques like decision trees or random forests, which can generate sophisticated rules much beyond the capabilities of credit analysts. This is why, in many domains, mathematical models finally surpassed expert knowledge in appeal. As a result, the analyst’s responsibility is to evaluate and relate the rules to reality, which requires far less work, time, and resources and results in more dependable models (Kaszyński et al., 2020).

**References:**

Anderson, R. (2007). *The Credit Scoring Toolkit: Theory and Practice for Retail Credit Risk Management and Decision Automation*. Oxford University Press

Angrist, J., & Pischke, J. (2008). *Mostly Harmless Econometrics*. Princeton University.

Antonakis, J., Bendahan, S., B., Jacquart, P., & Lalive, R. (2010). On making causal claims: A review and recommendations. *The Leadership Quarterly*, *21*(6), 1086-1120.

<https://doi.org/https://doi.org/10.1016/j.leaqua.2010.10.010>

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Breiman, L. (1996). Some properties of splitting criteria. *Machine Learning*, *24*(1), 41-47.

<https://doi.org/10.1023/A:1018094028462>

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5-32. <http://dx.doi.org/10.1023/A:1010950718922>

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. 22nd acm sigkdd international conference on knowledge discovery and data mining,

Cramer, J. S. (2002). *The Origins of Logistic Regression* (02-119/04). Tinbergen Institute.

Crook, J., Hamilton, R., & Thomas, L.C. (1992). The Degradation of the Scorecard over the Business Cycle. *IMA Journal of Mathematics Applied in Business and Industry 4*, 111-123. <https://doi.org/https://doi.org/10.1093/imaman/4.1.111>

Finance, Y. (2013). *How FICO became ’the’ credit score*. Retrieved 21.11.2021 from <https://finance.yahoo.com/news/fico-became-credit-score-100000037.html>

Freund, Y., Schapire, R., & Abe, N. . (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, *14*(5), 771-780.

<https://cseweb.ucsd.edu/~yfreund/papers/IntroToBoosting.pdf>

Hahn, J., Todd, P., & Klaauw, W. (2001). Identification and estimation of treatment effects with a regression-discontinuity design. *Econometrica*, *69*(1), 201-209.

<https://doi.org/https://www.jstor.org/stable/2692190>

Hair, J. F., Black, W. C., Babin, B. J., Anderson, R. E., & Tatham, R. L. (1998). *Multivariate Data Analysis* (Vol. 5). Prentice Hall

Hastie, T., Tibshirani, R., & Friedman, J. . (2009). *The elements of statistical learning: Data mining, inference, and prediction.* . Springer Science& Business Media.

Ip, E. H. S. (2001). Visualizing Multiple Regression. *Journal of Statistics Education 9*(1), 1-10.

<https://doi.org/https://doi.org/10.1080/10691898.2001.11910646>

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R* (Vol. 112). Springer.

<https://doi.org/https://link.springer.com/book/10.1007/978-1-4614-7138-7>

Johnston, J., & DiNardo, J. (1997). *Econometric methods*. McGraw-Hill Companies, Inc.

Kamiński, B., & Zawisza, M. (2012). *Receptury w R: podręcznik dla ekonomistów*. Szkoła Główna Handlowa. Oficyna Wydawnicza. .

Kass, G. V. (1980). An exploratory technique for investigating large quantities of categorical data. *Journal of the Royal Statistical Society:Series C (Applied Statistics)*, *29*(2), 119-127. <https://doi.org/https://doi.org/10.2307/2986296>

Kennedy, P. (2003). *A Guide to Econometrics*. MIT Press.

Liao, D., & Valliant, R. (2012). Condition indexes and variance decompositions for diagnosing collinearity in linear model analysis of survey data. *Survey Methodology*,Understanding variable importances in forests of randomized trees. NIPS,

Molnar, C. (2019). *Interpretable Machine Learning: A guide for making black box models explainable*. <https://christophm.github.io/interpretable-ml-book/index.html>

Mohanty, N., John, A. L.-S., Manmatha, R., Rath, T.M. (2013). Shape-Based Image Classifica- tion and Retrieval. *Handbook of Statistics*, *31*, 249-267.

<https://doi.org/http://dx.doi.org/10.1016/B978-0-444-53859-8.00010-2>

Nielsen, D. (2016). *Tree boosting with xgboost-why does xgboost win every machine learning competition?* [Master’s thesis]. NTNU.

Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

Wilms, R., Mäthner, E., Winnen, L., & Lanwehr, R. (2021). Omitted variable bias: A threat to estimating causal relationships *Methods in Psychology 5*(10075), 1-10.

<https://doi.org/https://doi.org/10.1016/j.metip.2021.100075>

Sajons, G. B. (2020). Estimating the causal effect of measured endogenous variables: A tutorial on experimentally randomized instrumental variables. *The Leadership Quarterly*, *31*(5), 101348. <https://doi.org/https://doi.org/10.1016/j.leaqua.2019.101348>

Selvitella, A. (2017). The ubiquity of the Simpson’s Paradox. *Journal of Statistical Distributions and Applications*, *4*(1), 1-16. <https://doi.org/https://doi.org/10.1186/s40488-017-0056-5>

Siddiqi, N. (2012). *Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring* (Vol. 3). John Wiley & Sons.

Simpson, E. H. (1951). The Interpretation of Interaction in Contingency Tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, *13*(2), 238-241.

<https://doi.org/https://www.jstor.org/stable/2984065>

Stephenson, A. (2018). *Econometrics*. Retrieved 03-12-2021 from

<https://www.alexstephenson.me/courses/econometrics/>

Kaszyński, D., Kamiński, B., & Szapiro, T., (Ed.). (2020). *CREDIT SCORING IN CONTEXT OF INTERPRETABLE MACHINE LEARNING*. SGH Publishing House.

Thistlethwaite, D. L., & Campbell, D. T. (1960). Regression-discontinuity analysis: An alternative to the ex post facto experiment. *Journal of Educational Psychology*, *51*(6), 309-317. https://doi.org/ <https://doi.org/10.1037/h0044319>

Thomas, L., Crook, J., & Edelman, D. (Ed.). (2017). *Credit Scoring and its Applications* Society for Industrial and Applied Mathematics (SIAM).

<https://doi.org/https://doi.org/10.1137/1.9781611974560>

Welfe, A. (2018). *Ekonometria*. Polskie Wydawnictwo Ekonomiczne.

Wooldridge, J. M. (2008). *Introductory Econometrics: A Modern Approach*. Cengage Learning.