Projectreport Machine Learning 2

Maluna Menke, Ari (Sara) Wahl, Pavlo Kravets

2024-01-08

Contents

1. Introduction

Our general idea was to work with LGBT-related data. This was not as easy as expected, since it seems there are not a lot of datasets openly available that have that kind of information. Finally, we found a US survey by the CDC, that regularly monitors the country's youth in a lot of dimensions, but among other questions also asks for sexual experiences and identification.

2. The Dataset

"The *Youth Risk Behavior Survey (YRBS)* measures health-related behaviors and experiences that can lead to death and disability among youth and adults.[...] Some of the health-related behaviors and experiences monitored are:

- Student demographics: sex, sexual identity, race, ethnicity, and grade
- Youth health behaviors and conditions: sexual, injury and violence, bullying, diet and physical activity, obesity, and mental health, including suicide
- Substance use behaviors: electronic vapor product and tobacco product use, alcohol use, and other drug use
- Student experiences: parental monitoring, school connectedness, unstable housing, and exposure to community violence [1].

It is a national survey conducted by the CDC (Center for Disease Control and Prevention) and includes high school students from both private and public schools within the U.S. Data is collected from 1991 through 2021, we are only using the most recent data from 2021. If you want to learn more about the data there is an accompanying Data User Guide.[2].

2.1 Preprocessing of the dataset

To preprocess the dataset, we first ran a summary of our dataset. The number of NAs seems to depend very much on the question. The variable "orig_rec" only contained NAs and has therefore been removed, as well as the variable "site" which only contained "XX" entries. Variables q4 and q5 are already aggregated in "raceeth" and have also been deleted. The variable "record" seems to be an ID for the observations. This also has to be deleted since IDs do not hold predictive information that can be used for machine learning.

2.2 Missing Data

We will first exclude all the observations with NAs in all the target-related variables q25 to q29. Since we want to build our target variable on these questions, the target variable cannot all be empty. Since our target variable will be a score indicating the risk of suicide among teenagers, unfortunately, it is probably not helpful for any relevant application of our Machine Learning model to define a specific missing category for the NAs related in the target variable, as discussed in the lecture.
The amount of data available should be enough to just exclude these observations. After removing the observations that have NAs in all the variables, that are used to create our target variable, we still have around 13.7% NAs in the dataset.
What if we had just excluded every NA in the dataset? We will try and see if this is a viable option since this would not just be quick and easy, but we would also just have "real" answers. The exclusion of NAs

leads to a severe reduction in the number of observations. The original data consisted of 17232 observations, if we reduce the target-related NAs only, we have 11753 observations left. If we omit all NAs, the reduced dataframe has 4334 observations.
In this case need to assess the loss of information foremost about our target variable. The important question is if there is a pattern to the missingness in our data, not just, but especially about our target variable.

2.2.2 Discussion on omitting NAs vs data imputation for the target variable

Whether we can omit the NAs or if it may be necessary to impute the missing data points, in general, depends on the type of missingness. If data is missing completely at random (MCAR), we can omit the NAs, if it is just missing at random (MAR) we would rather impute the data. If the data is missing not at random (MNAR), it would be a quite difficult problem because we cannot easily impute the missing data then. To find out if we can just omit the data without introducing any bias, an MCAR test was applied. We test the target-related variables q25 to q29 for potential pattern(s) in the target related missing data. This results in a p-value of 0, which indicates, that the data is not missing completely at random. Omitting all NAs can be problematic and lead to bias.
But imputing the target variable related data may be an arguable choice and quite challenging to do well. We therefore decide to use the information at hand to decide for a class of the target variable, starting with the most severe question (q29 to q25). This means we check if q29 ("If you attempted suicide…") was answered positively. If yes, we use the answer for the labeling, if not, we check the next less severe question to base our choice of class label on and so on. This means that NAs are treated similar than answering negatively to the question. We hope to introduce less bias this way.

We will use a rule base approach as described to create the target variable and impute the predictive variables afterwards. To ensure a good imputation, we need to impute our NAs before reducing the dataset to 2000 observations. To run the imputation properly we need to factorize our nominal and ordinal variables first.

2.3 Target Variable

As a target variable, we decided to calculate a score from 5 questions that reflect the suicide risk of the person (observation) in question. This score is aggregated with a rule-based approach.

After creating the target variable we need to exclude the variables q25 to q29, which were used for creating it, from our dataset. After originally starting with 5 classes (no risk, low risk, moderate risk, high risk, very high risk) for our target variable we reduced it to 3 classes (no risk, low or moderate risk, high risk).

2.4 Imputation

For the imputation of the missing data, we used a random forest approach with 500 trees and a predictive mean matching with observed values of 200 similar cases.

2.5 Reducing and balancing the dataset to 2000 observations

We need to reduce our data to the maximum allowed size of 2000 observations. To ensure the best possible data quality, we want to balance our dataset. We therefore use stratified sampling to create a dataset, that has the same number of observations for all classes of the target variable.

2.6 Simple Synopsis of the Dataset

- number of observations: 2000
- number of variables: 100
- datatypes:
  - factor: 96
    - nominal variables: 29
    - ordinal variables: 67
  - numeric variables: 4
    - discrete variables: 96 (here all factor variables)
    - continuous variables: 4 (here all numeric variables)

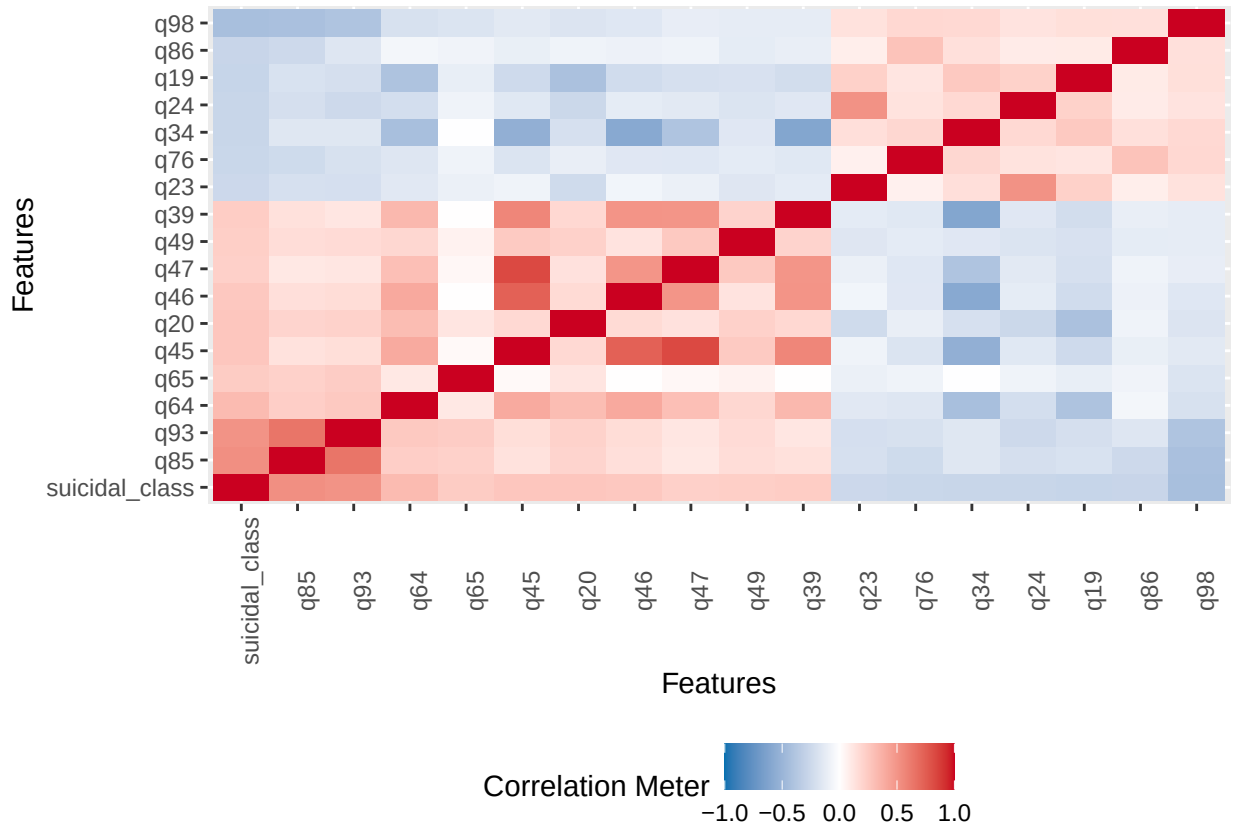
3. Additional Data Preparation

3.1 Feature Reduction

Since our dataset has lots of variables, we decided to start by excluding some variables depending on the estimated feature importance.

3.1.1 Correlations

Unfortunately at this point we have to many variables to do a pairs plot or correlation plot with a visually usable outcome. We will therefore perform a correlation analysis only with respect to the target variable and in numeric format instead of any visual plot.



The 17 variables with high correlations (>0.25) with our target variable are: q19, q20, q23, q24, q34, q39, q45, q46, q47, q49, q64, q65, q76, q85, q86, q93, q98.

3.1.2 Feature Selection Algorithm

Since the data still has a lot of variables, we need to use a feature selection technique to reduce the features before using a machine learning method. We chose to use model agnostic methods, because the feature selection should be valid for all methods that are later compared. In an earlier step the variables most correlated with the target variable were already identified. Unfortunately this captures only linear monotonous relationships in the data and does not work well for our nominal categorical features.

To capture non-linear relationships as well, information gain between the target and the predictor variables is measured as well. The variables with high information gain with respect to the target variable are kept, because they can contribute more in predicting the target variable.

3.1.3 Information gain for feature selection

Information Gain can be used in the construction of decision trees but also for feature selection [3]. Therefore the information gain for each variable with regard to the target variable is calculated.

3.1.4 Domain knowledge for final feature selection

```
## [1] "psu"    "weight" "q2"     "q40"
```

```
## [1] "suicidal_class" "q76"
```

If we compare the features that came up in the correlation analysis and in the information gain calculations, we find only five meaningful variables that come up, “q40”, “psu”, “weight”,“q2” and “q76” . According to the data manual, q40 encodes the range of age when a student first got into contact with drinking alcohol. This might be an indicator for a negligent social surrounding if someone is exposed to an alcoholic drink in an early age and therefore also could be a valid predictor variable in our case. “q2” is the sex of the student, “psu” is a regional code of the school the student attends and “q76” questions how regularly the student had breakfast. We will keep all the variables that came up, since they could carry helpful information.

3.2 Naming the variables and the factor levels

For an easy understanding of their values, the variables and levels are named according to their content. This is an optional step. It can lead to a better readability of our data frame.

- alcohol_age
- sex
- psu
- weight
- times_breakfast_7d
- suicidal_class
- Current_mental_health
- Mental_health_during_pandemic
- Difficulty_concentrating
- Sexual_identity
- Sex_of_sexual_contacts
- sexual_violence_12m
- bullied_electronically_12m
- bullied_at_school_12m
- forced_sex
- vapor_tried
- marijuana_age
- avg_h_sleep
- marijuana
- times_pain_meds_without_perscription
- marijuana_30d
- tried_stop_tobacco_12m

The above printed variable names, are the ones left from feature selecting and name-mapping.

3.3 Splitting the Data

According to the project requirements we split our data in 60% Training, 20% Validation and 20% Testing Data. Since we want to do a cross validation we will split into 80% Training and 20% Testing Data [4].

4. Machine Learning Models

4.1 Short Mathematical Overview on the used Methods

In the following section we describe the two methods used for our machine learning project.

4.1.1 Naive Bayes Classification

Ideally, a classifier is able to detect the class k which maximizes the conditional probability $P(Y = k|X = x_1, ..., x_p)$. A Bayes Classifier would calculate these probabilities for each of the classes exactly, but usually it is only possible to approximate those. The Naive Bayes Classifier is one method of approximation. It approximates by "naively" assuming the conditional independence of predictor variables. This leads to simpler calculations. The joint probability of two events A and B $P(A \cap B) = P(A \mid B) * P(B)$ can be simplified to $P(A \cap B) = P(A) * P(B)$ under the independence assumption, since conditional independence means $P(A \mid B) = P(A)$. The conditional probabilities of a class k can be calculated with the Bayes Theorem. It states that:

$$P(k \mid X) = \frac{P(X \mid k) \cdot P(k)}{P(X)}$$

Since the denominator only uses our predictor variables, we only need to focus on the nominator and find the maximum class for each observation to be classified. We can express this relationship with the proportionality operator:

$$P(k|X) \propto P(X|k) \cdot P(k)$$

At this point, the assumption of independent predictor variables simplifies the calculations if there is more than one predictor variable. Instead of calculating $P(k|x_1, ..., x_p) \propto P(x_1, ..., x_p|k) \cdot P(k)$ where $P(x_1, ..., x_p|k)$ is quite complicated to calculate because of all the possible dependencies among the variables, the independence assumption leads to:

$$P(k|x_1, ..., x_p) \propto P(k) \cdot \prod_{i=1}^{p} P(x_i|k)$$

Often this yields good results even if the quite strong assumption of conditional independence is not met. If the dependencies do not contribute that much to the outcome, the approximation is still quite good.

4.1.2 Support Vector Machines (SVM)

The name Support Vector Machines already describes some elements of this method. A certain number of data points will define the (linear) boundary between two classification regions, these are called the support vectors. The support vectors are the datapoints (observations) that lie closest to our decision boundary. The boundary in two dimensional space is a line, in three dimensions a plane and in more than three dimensional space a hyperplane. For our dataset, we need a multidimensional hyperplane. The number of dimensions depend on the number of our predictor variables. We need to find the hyperplane, that separate our data into the classes of our target variable best. The best hyperplane is the one that maximizes the margin between the support vectors of the different classes. The margin is a strip on each of the boundaries sides. In the case of a hard classifier this strip does not contain any points. But we will have a soft classifier with the cost C as a hyperparameter. This cost C describes a budget that we allow for points within the margin or on the other side of the boundary. Depending on the position of the point, the amount it attributes to the total cost changes. A point on the correct side of the margin will not attribute to the total cost at all. If it is in the margin, but on the correct side of the boundary it will attribute between zero and one, if on the wrong side of the boundary but within the margin it will attribute with one to two and if on the wrong side of the margin it will cost more than two. Mathematically we need to maximize the Margin M with respect to α_0, α_i and ϵ_i in the following objective function:

$$y_i \left(\alpha_0 + \sum_{j=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}_j) \right) \geq M(1 - \epsilon_i)$$

The following constraints are given: $\sum_{i=1}^{n} \alpha_i^2 = 1$, $\epsilon_i \geq 0$ and $\sum_{i=1}^{n} \epsilon_i < C$ with C ≥ 0 and $i = 1, ..., n$

For our SVM approach, we will try different kernels K as hyperparameters:

A linear Kernel

$$K(u, v) = \langle \mathbf{u}, \mathbf{v} \rangle = \sum_{j=1}^{p} u_j v_j$$

a polynomial Kernel

$$K(u, v) = (c + \langle \mathbf{u}, \mathbf{v} \rangle)^d, \ with \ d > 1$$

and a radial Kernel

$$K(u, v) = \exp\left(-\gamma \sum_{i=1}^{p} (u_i - v_i\right)$$

4.2 Preprocessing

We scale the data as part of pre-processing. Scaling transforms the data to have unit variance, further contributing to uniformity across different scales and improving algorithm performance. After that we check for constant variables.

4.3 Hyperparameter Optimization

Hyperparameter Optimization (HPO) enables the testing of various hyperparameter combinations to identify the optimal settings that maximize our target evaluation metric, namely the model's accuracy. The grid search method allows for the exhaustive pairing of each hyperparameter with every other, albeit at a significant computational cost. This approach, however, offers the advantage of explicitly specifying the values for testing.

4.3.1 HPO of Naive Bayes

Naive Bayes is quite simple and only has one hyperparameter that we need to consider: the use of Laplace smoothing. The type of Naive Bayes model is automatically estimated by our package (naivebayes). Laplace smoothing is used to obtain "more smooth" class distribution and avoid zero probabilities. It is a numeric parameter. We examine three common values for that parameter: 0 (leading to Max-likelihood estimation), 0.5 and 1.

Given the list of all resulting models, coming from the HPO of the Naive Bayes, we extract the best model for further inspection and results using Accuracy as our focused metric. Since our classes are balanced, Accuracy is the preferred testing metric.

According to our HPO the best value for the Laplace Smoothing is 1. This means that a smoothing term is helpful in our case, most probably for preventing overfitting and adjusting to the unseen data. The automatic bayes model type configuration gives us a gaussian naive bayes model, with the most of our features being multinomial.

4.3.2 HPO of SVM

For the SVM we examine three distinct kernel types - linear, radial, and polynomial - each characterized by unique parameters, in addition to the common cost parameter.
Kernel and the cost parameter C have a significantly impact on the model's performance and need to be tuned carefully.

Numerical values were already scaled, if they exist in the database, we set scale to FALSE. Using 5-fold cross validation testing different hyperparameters for each kernel with grid search.
A large value of C leads to points within the margin or on the other side of the boundary being heavily penalized so there are few points in the margin/missclassified (Lecture 7, p.5)

Here we also create a list of all resulting models, coming from the HPO of the SVM, the best model is identified and evaluated with different performance metrics. The tuned models contain the parameter $best.performance, which gives the classification error. The Accuracy can be calculated with 1 - classification eror.

The best model was trained with a polynomial kernel, cost factor of 1 (which is also the default and a balanced approach) and gamma value of 0.0312 suggests that the influence of each training example is moderate. Higher values would lead to narrower, more complex decision boundaries, while lower values result in broader, simpler decision boundaries. The degree of the polynomial kernel is set to 1. This means the polynomial kernel effectively becomes linear as it's raised only to the first power. Taking the best linear kernel having nearly the same accuracy, this is not surprising. With a degree of 1, the polynomial kernel equation simplifies to a linear relationship between the features. This suggests that, despite choosing a polynomial kernel, the model is using a linear decision boundary.

5. Comparison of the Models / Model's Performance on Test Data

5.1 Metrics

5.1.1 Confusion Matrix

A confusion matrix is a fundamental tool in statistical classification and machine learning, used to visualize the performance of an algorithm. It's a table used to describe the performance of a classification model on a set of test data for which the true values are known. Here's a brief overview of its components:

From the confusion matrix, various metrics such as accuracy, precision, recall (sensitivity), and F1-score can be calculated to evaluate the model's performance. These metrics provide insights into aspects like how many instances are correctly classified, the balance between the sensitivity and precision of the model, and the overall effectiveness of the classifier.

5.1.1.2 Naive Bayes

```
      Actual
```

Prediction l1 l2 l3 l1 106 30 12 l2 22 62 45 l3 6 42 77 Confusion Matrix and Statistics

```
  l1  l2  l3
```

l1 106 30 12 l2 22 62 45 l3 6 42 77

Overall Statistics

```
           Accuracy : 0.6095
             95% CI : (0.5599, 0.6574)
No Information Rate : 0.3333
P-Value [Acc > NIR] : <2e-16

              Kappa : 0.4142
```

Mcnemar's Test P-Value : 0.3429

Statistics by Class:

```
                     Class: l1 Class: l2 Class: l3
```

Sensitivity 0.7910 0.4627 0.5746 Specificity 0.8433 0.7500 0.8209 Pos Pred Value 0.7162 0.4806 0.6160 Neg Pred Value 0.8898 0.7363 0.7942 Prevalence 0.3333 0.3333 0.3333 Detection Rate 0.2637 0.1542 0.1915 Detection Prevalence 0.3682 0.3209 0.3109 Balanced Accuracy 0.8172 0.6063 0.6978
Analyzing the performance of the Naive Bayes model, the Confusion Matrix reveals an Accuracy of approximately 60.95% for the test data. The 95% confidence interval indicates that the true Accuracy of the model lies between 55.99% and 65.74%. Remarkably, the p-value is less than 2.2e−16 for the hypothesis stating 'Accuracy is greater than No Information Rate'. This presents an extremely strong statistical evidence confirming that the accuracy of the Naive Bayes model significantly surpasses the baseline of random guessing (33%). It's clear that this model is effectively learning from the dataset's features, especially excelling in identifying cases with no suicide risk and high suicide risk.
5.1.1.2 SVM

```
      Actual
```

Prediction l1 l2 l3 l1 105 21 6 l2 22 73 46 l3 7 40 82 Confusion Matrix and Statistics

svm_best_model_pred l1 l2 l3 l1 105 21 6 l2 22 73 46 l3 7 40 82

Overall Statistics

```
           Accuracy : 0.6468
             95% CI : (0.5979, 0.6935)
No Information Rate : 0.3333
P-Value [Acc > NIR] : <2e-16

              Kappa : 0.4701
```

Mcnemar's Test P-Value : 0.9147

Statistics by Class:

```
                     Class: l1 Class: l2 Class: l3
```

Sensitivity 0.7836 0.5448 0.6119 Specificity 0.8993 0.7463 0.8246 Pos Pred Value 0.7955 0.5177 0.6357 Neg Pred Value 0.8926 0.7663 0.8095 Prevalence 0.3333 0.3333 0.3333 Detection Rate 0.2612 0.1816 0.2040 Detection Prevalence 0.3284 0.3507 0.3209 Balanced Accuracy 0.8414 0.6455 0.7183

Evaluating the SVM model, the Confusion Matrix shows a higher Accuracy, around 64.68%, for the test data compared to the Naive Bayes model. The 95% confidence interval for the SVM model suggests its true Accuracy likely ranges between 59.79% and 69.35%. Similar to the Naive Bayes model, the p-value here is also less than 2.2e−16, supporting the hypothesis that 'Accuracy is greater than No Information Rate'. This indicates a very strong statistical evidence that the SVM model's accuracy significantly exceeds mere chance (33%). This points to the SVM's robust predictive capabilities and its effective utilization of dataset features, marking it as a potentially more reliable model in this context.
Call: best.svm(x = suicidal_class ~ ., data = scaledRISK_train, degree = c(1, 2, 3), gamma = gamma.range, cost = C, method = "svmPoly", cross = 5, scale = FALSE, probability = TRUE)

Parameters: SVM-Type: C-classification SVM-Kernel: radial cost: 1

Number of Support Vectors: 1257

(413 523 321)

Number of Classes: 3

Levels: l1 l2 l3

5-fold cross-validation on training data:

Total Accuracy: 61.84211 Single Accuracies: 60.50157 60.50157 64.89028 60.50157 62.8125

5.1.2 Precision, Recall and F1-Score

Additionally, to the accuracy score, we also evaluate our model for Precision, Recall, and F1-score. These metrics are especially helpful for classification in medical-related fields, where it is important to look at Precision and Recall specifically. We therefore use multi-class metrics [5].

5.1.2.1 Naive Bayes

Class: no risk (1) Recall: 0.791044776119403 Precision: 0.716216216216216 F1: 0.75177304964539

Class: low to moderate risk (2) Recall: 0.462686567164179 Precision: 0.48062015503876 F1: 0.47148288973384

Class: high risk (3) Recall: 0.574626865671642 Precision: 0.616 F1: 0.594594594594595
As shown before, the classifier performs better on the extreme classes, especially "no-risk" class. The F1 scores for all classes reflect a balance between recall and precision. Suicide prevention is a topic that tends to favor, for understandable reasons, more sensitive toward suicide risk class classifiers over more specific ones. This implies that the model must either be simplified, not differentiating between low and high risk, or be further enhanced using better features or methods. Naive Bayes in general tends to serve as a baseline model, as it, by design, cannot capture the relationships between features. As such, other options, such as SVM should be better than it.

5.1.2.2 SVM

Since we have a multi-class classification problem, we also evaluate our metrics separately for each of our classes.
Class: no risk (1) Recall: 0.783582089552239 Precision: 0.795454545454545 F1: 0.789473684210526

Class: low to moderate risk (2) Recall: 0.544776119402985 Precision: 0.517730496453901 F1: 0.530909090909091

Class: high risk (3) Recall: 0.611940298507463 Precision: 0.635658914728682 F1: 0.623574144486692

The classifier performs best for the "no risk" category, showing high effectiveness in both identifying actual no-risk cases and in ensuring that most of its no-risk predictions are correct. For "low to moderate risk" and "high risk" categories, the performance is moderate. The model is less effective in correctly identifying actual cases and ensuring the correctness of its predictions in these risk categories compared to the no-risk category. The F1 scores for all classes reflect a balance between recall and precision, but they highlight that the model is more reliable in identifying "no risk" cases than "low to moderate risk" or "high risk" cases. By highly sensible topics like suicide, these results indicate a need for improvement in the classifier's ability to identify and correctly classify individuals at various levels of suicide risk. While the model is relatively effective in identifying individuals with no risk, enhancing its capability to accurately identify and classify individuals at low to moderate and high risk is crucial for effective intervention and support. But also shows, that maybe even more features or more detailed answer options (more levels or non categorical could help the model to learn, when someone might in some kind of risk of suicide.

5.2 Overfitting

Overfitting is a frequent problem in machine learning. It happens when a model learns the training data too well, including all its quirks and noise. As a result, its ability to generalize weakens. When the model is tested with new data, its performance often drops significantly. This is because it's overly tuned to the training data and doesn't adapt well to new, unseen data. We take several measures trying to prevent overfitting on our training data.
By employing Cross-Validation (CV) and utilizing a range of performance metrics such as recall, precision, and F1-score, we try to avoid overfitting in our models.

5.3.2 SVM

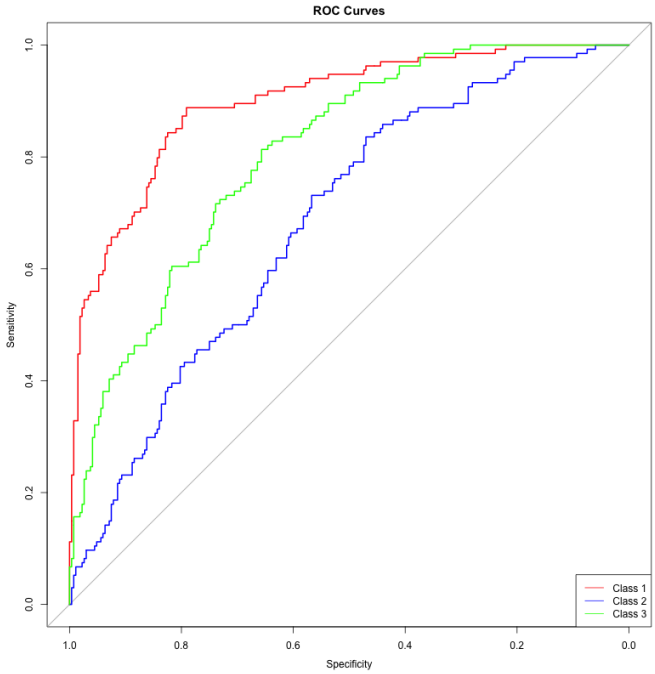
Our experiments with different kernels reveal that the polynomial, which is with degree 1 behaving like a linear kernel maintains robust performance, even when compared to the more complex radial and higher-degree polynomial kernels. Before modeling, we also took the precaution of reducing the number of features, which further diminishes the risk of overfitting. Moreover, setting the cost parameter C to a moderate level – in our case, 1, which is the default value – lessens the likelihood of overfitting.

Considering these factors collectively, we are confident that our models are unlikely to overfit on our training data.

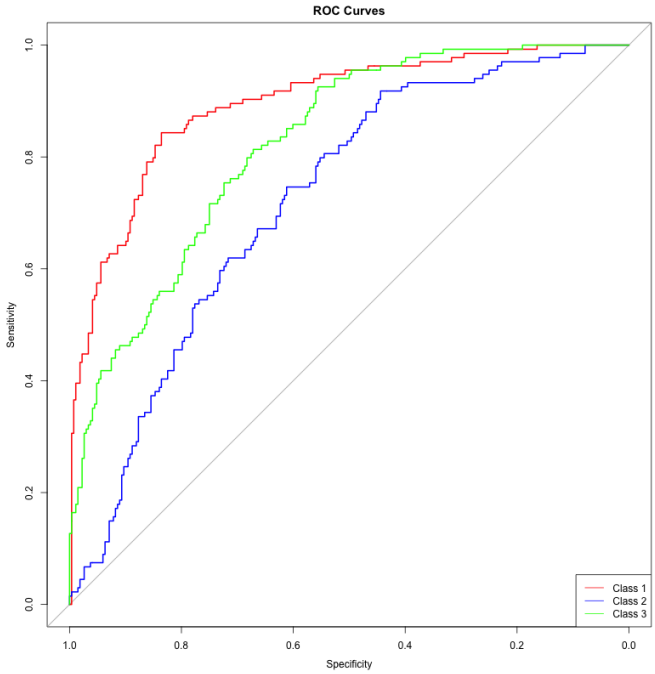
5.3 ROC

To further visually assess our classifier we plot the ROC curves for our three classes. To plot the ROC curves, we need to get the probabilities for our model predictions. Since we set a seed, we can use our best model and its hyperparameters to get the probabilities.

5.3.2 ROC for best Naive Bayes Model



5.3.2 ROC for best SVM model



Both models produce slightly similar ROC-Curves for each class. Like already seen in the confusion matrix, both Naive Bayes and SVM can produced best results in Class 1 - the "no suicide risk" Class. Class 2 performs worst in both cases. But all show that there are above the diagnoal grey line - which is the baseline / null model.

6. Final Discussion

We can say after the comparison of our models that the SVM performs slightly better than the Naive Bayes model, but both models only have an accuracy of between 60 and 65%. After closer inspection, one can see that the models can predict the low-risk class best, followed by the high-risk and the moderate-risk class. The fact that the results are so similar may be a hint, that the key to improving the results might be foremost related to the input data.

It becomes evident that these models do not achieve the necessary accuracy for practical application, particularly given the critical nature of the target variable.

Misclassifying individuals, especially those at high risk of suicide, can lead to dire consequences. When someone at moderate to high risk (Class 3) is incorrectly categorized as low to moderate risk (Class 2) or no risk (Class 1), they may not receive the urgent care and intervention they need.

Furthermore, it's crucial to understand that accuracy alone is not a sufficient metric in this context. Both Type I (False Positive) and Type II (False Negative) Errors need to be carefully considered. Here, Type II Errors are of greater significance than Type I Errors. While Type I errors might result in over-treatment or undue anxiety, Type II errors are more serious as they could lead to a lack of necessary support for individuals at high risk of suicide. Therefore, it is essential to particularly focus on minimizing Type II errors in predictive models dealing with mental health and suicide risk.
Given that the current models are not meeting the necessary standards, it's imperative to investigate other machine-learning models. Each model has its strengths and weaknesses, and finding the right fit for your specific dataset is crucial. This exploration could include models that are potentially more adept at handling imbalanced datasets or complex patterns.
The method of handling missing data can significantly impact model performance. Current imputation techniques may be leading to the selection of non-representative features, adversely affecting the model's predictive ability. Testing different imputation methods could help in identifying more descriptive features, thereby enhancing model accuracy. This aspect might warrant a separate project due to its complexity and potential impact.
Introducing an additional category in the target variable for missing responses is a thoughtful consideration. However, the underlying reasons for non-responses to questions about suicide need careful analysis. These missing responses could indicate a range of possibilities, from a lack of concern about the subject (possibly classifiable as low risk) to a reluctance to discuss a sensitive topic (potentially indicative of moderate to high risk). Understanding the nuances behind these non-responses is crucial for accurate classification.
The current dataset size of 2000 observations may be insufficient for training robust models. A larger and more comprehensive dataset could significantly improve model performance. However, this expansion was not feasible in your current project due to computational limitations and project scope. For future iterations of the project, seeking larger datasets or finding ways to augment the existing dataset could be highly beneficial.
Overall, these proposed measures aim to address the current limitations and enhance the predictive power of your models, which is vital given the serious implications of misclassifying individuals at risk of suicide.

7. Use of AI in the project

For the project we used Grammarly and ChatGPT to correct the grammar and spelling in our texts. Also some code snippets were created with ChatGPT (see annotations), which was also used for debugging.

8. References

[1] https://www.cdc.gov/healthyyouth/data/yrbs/overview.htm

[2] https:
//www.cdc.gov/healthyyouth/data/yrbs/pdf/2021/2021_YRBS_Data_Users_Guide_508.pdf

[3] https://machinelearningmastery.com/information-gain-and-mutual-information/

[4] https://cran.r-project.org/web/packages/splitTools/vignettes/splitTools.html

[5] https://rdrr.io/cran/mltest/man/ml_test.html