MET CS699 – Project Assignment

Final Report

Brandon Bartol – Chuong Nguyen

1. **Introduction**

Data mining stands as a crucial step in data analysis, particularly in today's era of abundant data availability. Throughout history, humanity has drawn insights from past events to anticipate and differentiate between impending threats and opportunities for societal betterment. However, this process has often been passive, lacking the immediacy needed for timely action. Take, for instance, weather forecasting: once regarded as unpredictable disasters attributed to supernatural forces, technological advancements now enable us to define, explain, and predict natural events with a considerable lead time. This capability hinges on our ability to detect and analyze patterns in data, unveiling meaningful insights.

The primary aim of this project is to develop and assess the performance of various classification models tailored to predict depressive disorders. Our approach involves analyzing a comprehensive dataset featuring a diverse array of demographic, lifestyle, and health-related variables. Utilizing advanced machine learning techniques—such as naïve Bayes, logistic regression, decision trees, random forests, K-nearest neighbors (KNN), and neural networks—we seek to construct robust predictive models adept at accurately discerning individuals with depressive disorders from those without.

Beyond model construction lies a comprehensive comparison of these algorithms. By systematically evaluating performance metrics—such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC)—our aim is to pinpoint the most effective classifier for this task. This holistic analysis promises valuable insights into each model's strengths and weaknesses, facilitating informed recommendations for real-world application.

Additionally, we are eager to explore diverse feature selection techniques and sampling methods, particularly in the context of imbalanced datasets. Class distribution imbalance—wherein one class (e.g., individuals with depressive disorders) disproportionately outweighs the other—poses distinct challenges. To address this, we will investigate oversampling, undersampling, and bootstrap data generation methods to rebalance the dataset and alleviate the impact of class imbalance on model performance.

In summary, our project endeavors to develop and assess multiple classification models for predicting depressive disorders based on demographic, lifestyle, and health-related factors. By leveraging machine learning algorithms and exploring various techniques, we aim to enhance the accuracy and efficacy of predictive models, ultimately contributing to early detection and intervention strategies in mental healthcare.

1. **Data Preprocessing**

* Initial data inspection and summary statistics
* Handling missing data
* Removing columns with high missing values
* Imputing missing values using mode or median, stratified by the class column
* Removing columns with near-zero variance
* Removing highly correlated columns
* Scaling the data for model compatibility

1. **Feature Selection and Imbalanced Data Treatment**
   1. Splitting the data into training and testing sets
      1. Boruta
      2. Information Gain
      3. Principal Component Analysis (PCA)
   2. Imbalanced data treatment methods:
      1. Under-sampling and Over-sampling from ROSE library

This severe class imbalance in the dataset presents significant challenges for the classification models to effectively learn and generalize from the data. With a ratio of 1 to 100 between participants with depressive disorder and those without the condition, the minority class (participants with depressive disorder) is heavily underrepresented, leading to several issues. Firstly, models trained on imbalanced data tend to exhibit a bias towards the majority class, as they prioritize maximizing overall accuracy without adequately capturing patterns within the minority class. This results in poor classification performance, especially in accurately identifying instances belonging to the minority class. Additionally, imbalanced data can lead to decision boundaries being skewed towards the majority class, making it challenging for models to distinguish between the two classes accurately. As a result, the predictive power of classification models is significantly compromised, highlighting the critical importance of addressing class imbalance through appropriate data sampling techniques and model evaluation strategies. (Aguiar, 2024)

To tackle the issue discussed above, there are several balancing treatment methods, including oversampling, undersampling, a combination of both, and bootstrap resampling were implemented. Each of these methods aims to mitigate the challenges posed by class imbalance and enhance the performance of classification models in predicting depressive disorders based on demographic, lifestyle, and health-related factors.

The first method, over-sampling, involves increasing the number of instances in the minority class to achieve a balanced class distribution. In our implementation, the over\_balance function utilizes the ROSE package's ovun.sample function with the "over" method to generate additional instances for the minority class, effectively increasing its representation in the dataset.

Conversely, undersampling reduces the number of instances in the majority class to achieve balance. The under\_balance function employs the "under" method in the ovun.sample function to randomly select a subset of instances from the majority class. While effective in balancing the dataset, this method may lead to the loss of valuable information present in the majority class.

To comprehensively address class imbalance, a combination of over-sampling and under-sampling techniques is employed. The combine\_sampling\_balance function simultaneously generates synthetic instances for the minority class and downsamples the majority class using the "both" method in the ovun.sample function. By leveraging the strengths of both methods, this approach aims to create a balanced dataset while minimizing information loss.One note is that all three above methods, over-sampling, under-sampling, and combination of both utilized ROSE package in R in R 4.2.1 environment.

* + 1. Bootstrap Resampling

Finally, bootstrap resampling involves creating balanced datasets by repeatedly sampling instances from the minority class with replacement. The bootstrap\_balance function implements this method by randomly selecting instances from the minority class and combining them with instances from the majority class. Unlike traditional over-sampling techniques, bootstrap balancing relies on resampling existing instances, potentially preserving the original distribution more accurately. There is one parameter to determine the target ratio between the majority and minority and it was set to 1 to 1 in this project.

These balancing treatment methods are essential in mitigating the impact of class imbalance on classification model performance. By addressing class distribution disparities, these techniques enable more robust and unbiased model training, ultimately improving the model's predictive power in identifying individuals with depressive disorders.

1. **Model Building**
   1. Splitting the training and testing sets

Splitting data into training and testing sets is crucial in machine learning for several reasons. Firstly, it allows us to evaluate the performance of our model on unseen data, simulating how it would perform in real-world scenarios. By reserving a portion of the data for testing, we can assess how well the model generalizes to new instances, providing insights into its ability to make accurate predictions beyond the training data. Moreover, splitting the data helps prevent overfitting, where the model learns to memorize the training data rather than capturing underlying patterns. By training the model on one set of data and testing it on another independent set, we can identify and mitigate overfitting, ensuring that the model's performance is more robust and reliable.

In particular, the data was split twice in this project. In the first split, we aimed to preserve one version of the test dataset for final evaluation. This was done before passing the data into any machine learning models, including feature selections, balancing methods, and classification algorithms. Due to the heavy imbalance of the given dataset, the first split was done with 20% in the testing set and 80% in the training sets to preserve as much information as possible for later data training. In the code, these sets of data were named *main.train.df* and *main.test.df*.

The second split was performed right before triggering all the data training in the pipeline with six classification models. The initial training data was then split again in a ratio of 66% to 33%. This split provided immediate training and testing data to validate and tune parameters in each classification algorithm. In the code, these sets of data were named *split.train* and *split.test*.

* 1. Building and evaluating multiple classification models:
     1. Naive Bayes

The first model used in the project is Naïve Bayes. It is wrapped in a function with three parameters including the immediate training dataset, the immediate testing dataset, and the testing dataset from the initial splitting. The function first builds a Naive Bayes model running on the training dataset, then, it predicts the class labels for the instances in the testing dataset using the trained model. It calculates performance metrics, such as accuracy, precision, recall, and F1-score, based on the predictions compared to the actual class labels in the testing dataset. Additionally, the function tests the model on a separate testing dataset, initaldata\_test, and computes performance metrics for this dataset as well. Finally, the function returns a list containing the performance metrics for both the testing dataset used for model training (inner\_split\_test) and the separate testing dataset (inital\_split\_test). This approach allows for the evaluation of the model's performance on both the immediate testing data and a separate validation dataset, providing insights into its generalization ability.

* + 1. Logistic Regression

The second model utilized in this project is the logistic regression algorithm. The process commences with training an initial logistic regression model (*logitModel1*) using the provided training dataset. All logistic models are fitted using the *glm* function with the binomial family. A summary of this model's coefficients and statistical information is generated, providing insights into the relationships between predictors and the target variable.

Once trained, the model is applied to the test dataset to predict class probabilities (*logitmdl.pred.prob*). We implemented a function to determine the optimal threshold called the findOptimalThreshold function. The calculation prioritizes a combined metric balancing sensitivity and specificity. Since we are detecting people with depressive disorder and potentially developing preventive actions against the development of this condition, the thresholds are calculated weighing more on sensitivity. There is a balanced trade-off between total accuracy and recall, but we decided to put more focus on minimizing Type 2 errors.

Subsequently, variable importance is assessed using the *varImp* function, ranking predictors based on their contribution to the model's performance. The top-ranked variables are selected for further analysis and used to train a secondary logistic regression model (new\_logitModel). This refined model focuses on a subset of the most influential predictors, potentially improving interpretability and computational efficiency.

Finally, the performance of both models is evaluated on the original test dataset (*original\_test*). Predictions from the secondary model (*logitmdl\_main\_pred*) are generated using the selected top variables, and performance metrics are computed and compared to those of the initial model. This comprehensive approach not only enhances model interpretability through variable selection but also iteratively refines predictive accuracy, ensuring robust performance in real-world classification tasks.

* + 1. Decision Tree

The classification model in this project utilized the decision tree algorithm and it was built with two distinct functions in R: *J48* and *rpart*. The process begins by configuring the training control parameters, specifying a repeated 10-fold cross-validation approach to robustly assess model performance. For the J48 model, a grid of hyperparameters (C and M) is defined to tune the model during training. This grid facilitates the exploration of different parameter combinations to identify the optimal configuration that maximizes predictive accuracy.

Subsequently, the *J48* model is trained using the specified method and hyperparameter grid, leveraging the train function from the caret package. Once trained, the model is applied to the provided test dataset to generate predictions. These predictions are evaluated using predefined performance metrics (the function named *compute\_metrics\_and\_build\_table*).

Similarly, the *rpart* model is trained using the *rpart* algorithm, following the same cross-validation methodology and parameter tuning approach as the J48 model. The train function is again employed to train the *rpart* model, with the number of tuning iterations specified through the *tuneLength* parameter. Predictions are generated for the test dataset using the trained *rpart* model, and performance metrics are computed to evaluate its predictive capabilities.

Finally, this classification model outputs two sets of results for each decision tree method it employs.

* + 1. Random Forest
    2. K-Nearest Neighbors (KNN)
    3. Neural Network (NN)
* Summary of parameter tuning and performance evaluation for each model

1. **Model Evaluation & Interpretation**
   1. Evaluation metrics computation and interpretation
   2. Comparison of model performance using various balancing methods
   3. Discussion on the effectiveness of each classification model
   4. Interpretation of the results and implications for predicting depressive disorders
2. **Conclusion**
   1. Summary of key findings and insights from the project
   2. Reflection on the effectiveness of different classification models
   3. Suggestions for future research or improvements in methodology
   4. Closing remarks
3. **VII. References**

* List of all sources referenced in the paper, including datasets, packages, and methodologies used