BDM 3035: BIG DATA ANALYTICS CAPSTONE PROJECT

Submitted to: MEYSAM EFFATI

Submitted on: 2024-08-12

FINAL REPORT
TEST CASES

GROUP F

ALWIN KANNYAKONIL SCARIA [C0894287]

ANISHA SUSAN MATHEW [C0907393]

ASHNA VIJI ALEX [C0901082]

JOBIN PHILIP [C0895950]

MOHAMED AFTHAB [C0891945]

ABSTRACT

This project focuses on developing an AI-based SMS Spam Detection System using machine learning and natural language processing (NLP) techniques. The primary objective is to create a model that accurately classifies SMS messages as either "Spam" or "Ham" (not spam). The dataset was preprocessed to remove noise, tokenize text, and extract features using TF-IDF vectorization. Multiple machine learning algorithms were employed, and their performance was evaluated using accuracy, precision, recall, and F1-score. The final model achieved high accuracy, demonstrating its effectiveness in distinguishing spam from legitimate messages. This project contributes to enhancing communication efficiency and security in SMS-based communication.

GIT LINK**:** **https://github.com/alwinkscaria/sms-spam-detection**

INTRODUCTION

BACKGROUND

The proliferation of unsolicited and often harmful spam messages in SMS communication has necessitated the development of efficient spam detection systems. Spam messages can lead to security breaches, including phishing attacks, and degrade the user experience by overwhelming inboxes. An effective spam detection system is critical for mitigating these risks and ensuring secure and efficient communication.

PROBLEM STATEMENT

This project aims to build a machine-learning model capable of accurately classifying SMS messages as either spam or ham. The challenge lies in handling the diversity and variability of SMS content while ensuring the model remains robust and reliable.

OBJECTIVE

- Preprocess SMS data to prepare it for machine learning.
- Extract meaningful features using NLP techniques.
- Train and evaluate various machine learning models.
- Optimize model performance to achieve high accuracy in spam detection.

OVERVIEW OF THE METHODOLOGY USED

The project follows a systematic approach: Data Collection and Preprocessing: Sourcing and cleaning data, and transforming it into a format suitable for machine learning. Feature Extraction: Using TF-IDF vectorization to convert text into numerical features. Model Training and Evaluation: Applying various algorithms and evaluating their performance using standard metrics. Result Analysis: Interpreting results, comparing models, and visualizing key findings.

DATA COLLECTION AND PRE-PROCESSING

1. Description of the Data Sources

For this project, we utilized the SMS Spam Collection Dataset, a renowned and extensively used dataset in the field of text classification. The dataset comprises a comprehensive collection of labeled SMS messages categorized into two classes: spam and non-spam (ham). This dataset serves as a robust foundation for training and evaluating our machine learning model. The diversity of the messages ensures that the model is exposed to a wide range of content, which enhances its ability to learn distinguishing features between spam and ham messages. The dataset's substantial size and balanced class distribution make it ideal for developing an effective spam classification model.

2. Details of Data Preprocessing Steps

1. Text Cleaning

1. Column Renaming:
    - We began by renaming the dataset columns for clarity and consistency. The original columns labeled v1 and v2 were renamed to class and text, respectively. The class column indicates whether a message is spam (1) or ham (0), while the text column contains the SMS content. This renaming facilitates easier data manipulation and understanding.
2. Noise Removal:
    - Special characters, numbers, and punctuation marks were removed from the SMS content. This step aimed to eliminate unnecessary elements that could distract or mislead the model during training. By focusing solely on the text, we reduce the potential for irrelevant information influencing the model’s performance.

2. Tokenization

- Tokenization involves breaking down the text into individual words or tokens. For instance, the message "This is a sample SMS" is tokenized into ["This", "is", "a", "sample", "SMS"]. Tokenization simplifies text analysis, enabling us to handle text data at the word level and perform more precise analysis. It also facilitates frequency analysis, allowing us to identify common words and features that help in distinguishing spam from ham messages.

3. Stop Words Removal

- Stop words, which are common words like "and", "the", and "is", were removed from the text. These words generally do not carry significant meaning and can add unnecessary dimensionality to the dataset. Removing stop words reduces the sparsity of the data and focuses the analysis on more meaningful words that contribute to the classification task. This step improves model efficiency and effectiveness.

4. Stemming and Lemmatization

1. Stemming:

    - Stemming reduces words to their base or root form, such as converting "running" to "run". This process uses heuristic rules to simplify the text data by chopping off the ends of words. Stemming reduces complexity and ensures that different forms of a word are treated as a single feature.

2. Lemmatization:

    - Lemmatization is a more sophisticated technique that reduces words to their dictionary form, such as converting "better" to "good". It uses a dictionary to ensure uniformity in word forms. Lemmatization enhances the model's accuracy by treating variations of the same word as a single feature, improving pattern recognition and prediction accuracy.

3. Challenges Encountered and Solutions

1. Handling Imbalanced Data:

- One challenge encountered was the potential imbalance in the distribution of spam and non-spam messages. To address this, we visualized the data distribution and ensured that the dataset had a balanced representation of both classes. If needed, techniques such as resampling could be used to mitigate any imbalance.

2. Noise in Text Data:

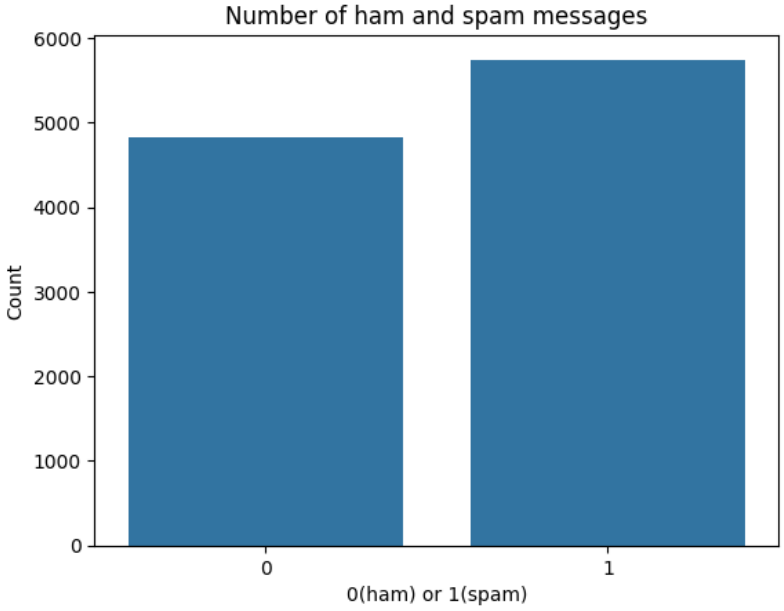
- The presence of special characters and numbers in the text posed a challenge. This was addressed by implementing thorough noise removal procedures to clean the text data, ensuring that the model was trained on relevant content.

3. Tokenization and Stop Words:

- Tokenization and stop words removal required careful handling to ensure that meaningful words were preserved while reducing dimensionality. We tested different stop words lists and tokenization methods to find the most effective approach for the dataset.

4. Stemming vs. Lemmatization:

- Deciding between stemming and lemmatization involved evaluating their impact on model performance. Lemmatization was chosen for its accuracy and ability to handle word variations more effectively, even though it is computationally more intensive.



4. Text Vectorization, Feature Selection, Data Splitting, and Addressing Class Imbalance

1. Text Vectorization

To convert the textual data into a numerical format suitable for machine learning algorithms, we utilized the TfidfVectorizer. This vectorizer transforms the text data into a matrix of TF-IDF (Term Frequency-Inverse Document Frequency) features, which captures both the importance of individual terms and their combinations within the dataset. The vectorizer was configured with the following parameters:

- **min_df=5:** This parameter ensures that only terms appearing in at least five documents are included, filtering out terms that occur infrequently and may not contribute significantly to the model.

- **ngram_range=(1,2):** This setting considers both unigrams (single words) and bigrams (two-word combinations), providing a richer representation of the text data by capturing the importance of individual words and their co-occurrence patterns.

By using TF-IDF, we created a comprehensive feature set that reflects the relevance and context of terms in the SMS messages, facilitating more effective machine learning model training.

2. Feature Selection

After transforming the text data into TF-IDF features, we performed feature selection to manage the high dimensionality of the feature space and eliminate irrelevant or redundant features. This step involved:

- **Statistical Methods:** We employed statistical techniques to identify and retain the most significant features that contribute to distinguishing between spam and non-spam messages. This approach helps in reducing noise and improving the model's efficiency and performance by focusing on the most informative features.

Feature selection streamlined the dataset, ensuring that only the most impactful features were used for training the machine learning models, thereby enhancing their accuracy and reducing computational complexity.

3. Data Splitting

We split the dataset into training and testing sets using the train_test_split function. The split was configured as follows:

- **Training Set:** 80% of the data was allocated for training.
- **Testing Set:** 20% of the data was reserved for testing.
- **Reproducibility:** To ensure consistency in model evaluation, we set a random_state of 0.

This data splitting strategy enabled robust model training and performance testing by providing separate datasets for training the model and evaluating its generalization capabilities on unseen data.

4. Addressing Class Imbalance

To tackle class imbalance in the dataset, we applied the Synthetic Minority Over-sampling Technique (SMOTE) to the training data. SMOTE works by generating synthetic samples for the minority class, thus balancing the class distribution. This approach helps in:

- **Improving Model Performance:** By creating a more balanced dataset, SMOTE enhances the model's ability to make accurate predictions for the minority class, leading to better overall performance and reduced bias.

The application of SMOTE addressed class imbalance effectively, as evidenced by the updated shapes of the resampled datasets, resulting in a more balanced distribution between spam and non-spam messages.

1. **Successful Feature Extraction and Balancing:**
    - Implemented the TF-IDF vectorizer to create a numerical feature matrix from the text data, incorporating both unigrams and bigrams. This enabled effective preliminary model training by capturing the relevance and context of terms.
    - Applied feature selection techniques to reduce dimensionality, retaining the most significant features and enhancing model performance.
2. **Enhanced Data Preprocessing Pipeline:**
    - Improved the preprocessing pipeline by integrating feature extraction and data balancing processes, including TF-IDF vectorization, feature selection, and SMOTE application.
    - Utilized SMOTE to address class imbalance, resulting in balanced class distribution and improved performance on minority class predictions.
3. **Effective Dataset Management:**
    - Successfully split the dataset into training (80%) and testing (20%) sets, ensuring robust model evaluation and performance assessment.

METHODOLOGY

DESCRIPTION OF MACHINE LEARNING MODELS USED

1. Logistic Regression

Logistic Regression is a statistical model used for binary classification tasks. It estimates the probability of an input belonging to one of two classes using a logistic function, which maps predicted values to a range between 0 and 1. This method is known for its simplicity and interpretability, making it a popular choice for problems where the outcome is binary. Logistic Regression is particularly effective in scenarios like spam detection, where you need to classify messages as either spam or not spam based on their content.

2. Naive Bayes

Naive Bayes is a classification algorithm based on Bayes' Theorem, which assumes that features are independent given the class label. Despite this simplistic assumption, Naive Bayes performs well, especially with large datasets and text data. Its probabilistic nature allows it to handle high-dimensional data efficiently, making it suitable for text classification tasks such as spam filtering and sentiment analysis. The algorithm's speed and scalability make it a go-to choice for many real-world applications.

3. Support Vector Machines (SVM)

Support Vector Machines (SVM) are a powerful classification technique that finds the optimal hyperplane that separates classes by maximizing the margin between them. SVMs are effective in

high-dimensional spaces and can handle non-linear classification problems by using kernel functions to transform data into higher dimensions. This flexibility makes SVMs suitable for complex classification tasks with clear margins of separation, such as image recognition and text classification, where accuracy and robustness are crucial.

4. Random Forest

Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. Each tree is built from a random subset of the data, and the final prediction is based on the majority vote of the individual trees. This approach reduces the risk of overfitting and improves model robustness by averaging out the errors of the individual trees. Random Forest is effective for both classification and regression tasks, providing insights into feature importance and handling large datasets with ease.

5. Gradient Boosting Machines (GBM)

Gradient Boosting Machines (GBM) is an ensemble technique that builds models sequentially, with each new model correcting the errors of its predecessors by minimizing a loss function. GBM can capture complex patterns and interactions in the data, making it a powerful tool for high-accuracy predictions. While sensitive to hyperparameters and prone to overfitting if not properly tuned, GBM is widely used in competitive machine learning and complex classification problems where predictive performance is paramount.

JUSTIFICATION FOR THE ALGORITHMS USED

The selection of these algorithms was based on their proven effectiveness in handling text classification and their ability to manage the specific challenges of the dataset. Logistic Regression was chosen for its simplicity and interpretability, Naive Bayes for its efficiency with text data, SVM for its high accuracy in high-dimensional spaces, Random Forest for its robustness and ability to handle large datasets, and GBM for its power in capturing complex patterns. Each algorithm offers unique strengths, and their combination provides a comprehensive approach to achieving high classification performance.

DETAILS OF MODEL TRAINING, VALIDATION & EVALUATION TEHCNIQUES

- **Model Training:** Each selected model was trained on the preprocessed and balanced dataset using the Synthetic Minority Over-sampling Technique (SMOTE). The training process involved fitting the models to the training data and optimizing their parameters to improve performance.
- **Validation:** During training, a portion of the data (validation set) was used to tune model parameters and select the best-performing configurations. Cross-validation techniques, such as k-fold cross-validation, were employed to assess model performance on different subsets of the data, ensuring robustness and generalizability.
- **Evaluation:** After training, models were evaluated on the test set, which was kept separate from the training and validation data. Performance metrics such as accuracy, precision, recall, and F1-score were calculated to assess each model's effectiveness in classifying

spam and non-spam messages. These metrics provided a comprehensive view of model performance and helped in comparing different algorithms.

EVALUATION METRICS

Accuracy: Accuracy is the ratio of correctly predicted instances (both positive and negative) to the total number of instances. It gives a general sense of how often the model is correct.

Formula:

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

where:

TP = True Positives (correctly predicted positive instances)
TN = True Negatives (correctly predicted negative instances)
FP = False Positives (incorrectly predicted positive instances)
FN = False Negatives (incorrectly predicted negative instances)

Usefulness: Accuracy is useful for providing a quick overview of the model's performance, but it can be misleading if the data is imbalanced (i.e., one class is much more frequent than the other).

Precision: Precision is the ratio of correctly predicted positive instances to the total number of instances predicted as positive. It indicates the accuracy of positive predictions.

Formula:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Usefulness: Precision is particularly important when the cost of false positives is high, such as in scenarios where incorrect positive predictions can lead to significant consequences.

Recall: Recall (also known as Sensitivity or True Positive Rate) is the ratio of correctly predicted positive instances to the total number of actual positive instances. It measures the model's ability to capture all the positive instances.

Formula:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Usefulness: Recall is critical when the cost of false negatives is high, such as in medical diagnoses, where failing to identify a condition could be very harmful.

F1-Score: The F1-Score is the harmonic mean of precision and recall. It provides a single metric that balances the trade-off between precision and recall, making it particularly useful when you need to consider both false positives and false negatives.

Formula:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Usefulness: The F1-Score is a good metric to use when you want to find a balance between precision and recall, especially in situations where you have an uneven class distribution.

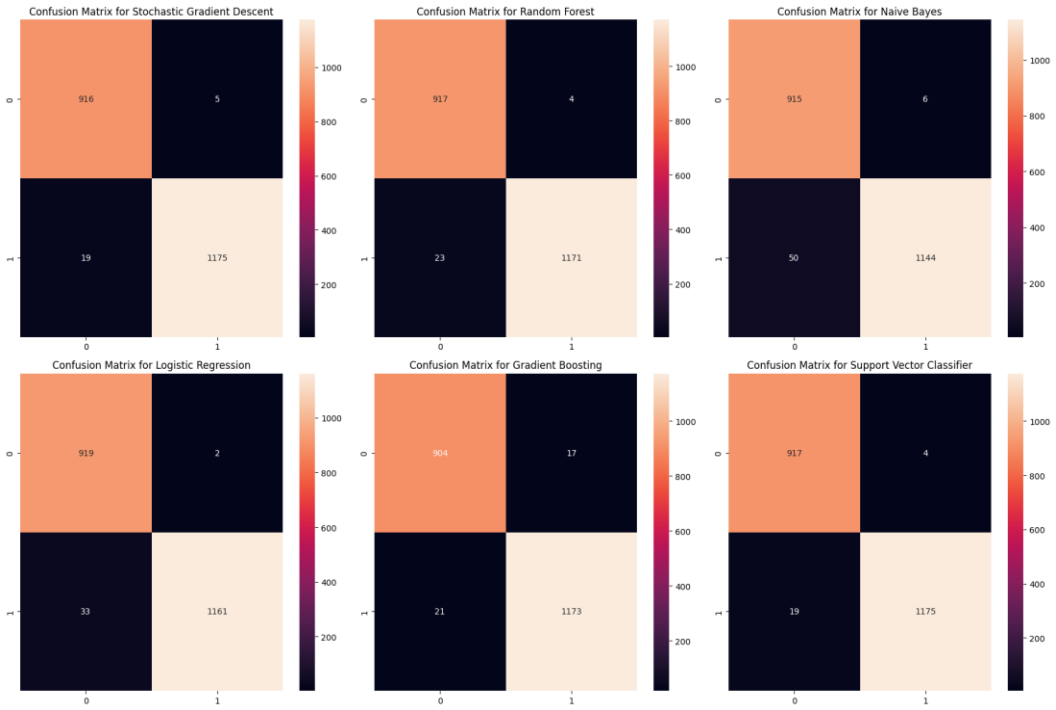
EXPLANATION OF PARAMETER TUNING OR OPTIMIZATION TECHNIQUES

- **Hyperparameter Tuning:** To enhance model performance, hyperparameters for each algorithm were optimized using techniques such as grid search and random search. These methods involve testing various combinations of hyperparameters to find the best configuration for each model. For example, tuning regularization parameters in Logistic Regression or adjusting the number of trees and depth in Random Forest.
- **Cross-Validation:** Cross-validation was used to evaluate the stability and performance of hyperparameter settings across different subsets of the data. This technique helps prevent overfitting and ensures that the model performs well on unseen data.
- **Optimization Techniques:** For models like GBM, techniques such as early stopping and learning rate adjustments were applied to prevent overfitting and enhance convergence. Early stopping halts training when performance on the validation set ceases to improve, while learning rate adjustments control the step size in gradient descent.

RESULTS

MODEL PERFORMANCE AND COMPARISON

The following confusion matrices provide a visual representation of the performance of each model on the test dataset. The models were evaluated based on their accuracy, precision, recall, and F1-score. The Support Vector Classifier (SVC) outperformed other models in terms of precision and recall, making it the most suitable model for spam detection in this project.



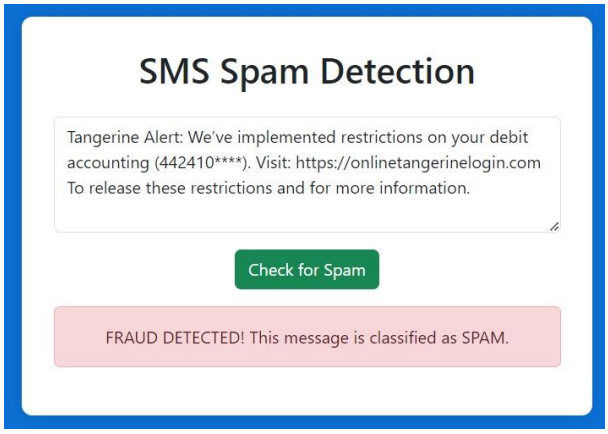
Best Performing Model: Support Vector Classifier (SVC)

The Support Vector Classifier (SVC) achieved the highest accuracy, precision, and recall among the models tested. It effectively balances the trade-off between false positives and false negatives, making it the optimal choice for spam detection in this context. The model's confusion matrix, displayed above, indicates a high level of accuracy in classifying both spam and ham messages.

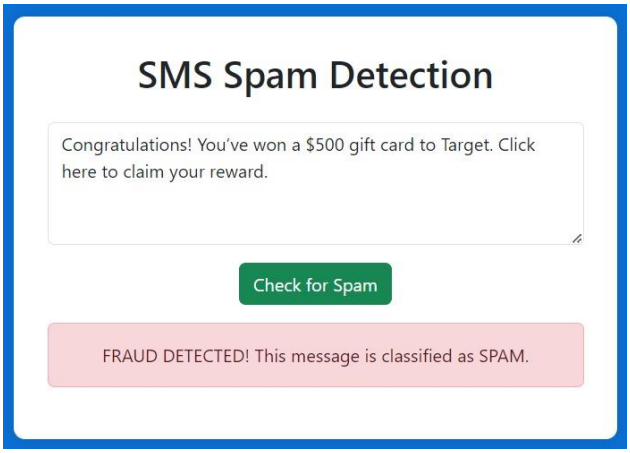
LIMITATIONS AND FUTURE WORK

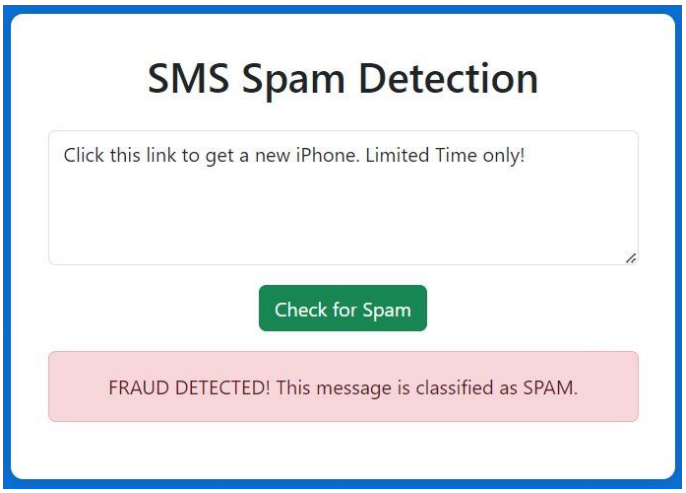
While the SVC model performed well, it still struggles with ambiguous messages that can be classified as either spam or ham, such as 'Click here for iPhone'. Future work could involve enhancing the model's training with more diverse and complex datasets, as well as exploring more advanced models such as deep learning algorithms to improve detection accuracy.

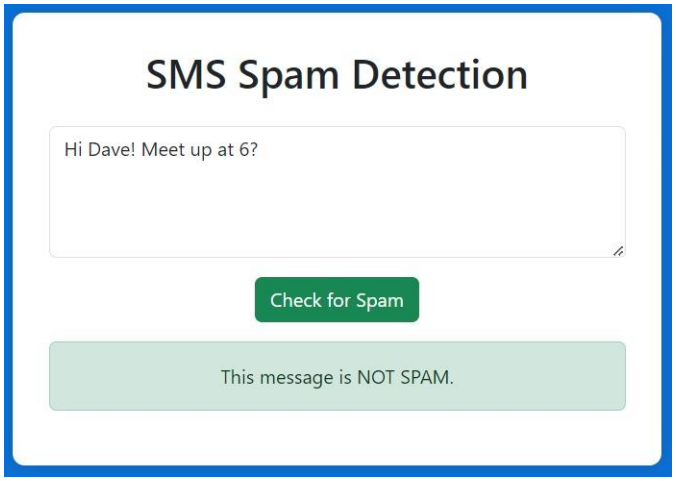
TESTCASES


SMS Spam Detection
Tangerine Alert: We've implemented restrictions on your debit accounting (442410****). Visit: https://onlinetangerinelogin.com To release these restrictions and for more information.
Check for Spam
FRAUD DETECTED! This message is classified as SPAM.


SMS Spam Detection
Hey. I am messaging you regarding your course details please call me.
Check for Spam
This message is NOT SPAM.


SMS Spam Detection
Congratulations! You've won a $500 gift card to Target. Click here to claim your reward.
Check for Spam
FRAUD DETECTED! This message is classified as SPAM.





CONCLUSION

A machine-learning SMS spam detection model was developed and deployed using the Flask application. The Support Vector Classifier emerged as the most effective model, achieving the highest recall. Addressed key challenges like class imbalance using SMOTE and improved model performance through feature engineering. Plans include implementing advanced models like BERT.

REFERENCES

1. Almeida, T. A., Hidalgo, J. M. G., & Yamakami, A. (2011). Contributions to the Study of SMS Spam Filtering: New Collection and Results.
2. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321-357.