Hindi Fake News Detection using Machine Learning

Shreyas Nagesh[1], Aditya N Sampath[2], Pradhumna Guruprasad[3], Salai Sanjay S[4] and V Ramana Murthy Oruganti[5,*]

[1,2,3,4,5]Department of Electrical and Electronics Engineering,
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham, India

[1]cb.en.u4elc19052@cb.students.amrita.edu,
[2]cb.en.u4elc19004@cb.students.amrita.edu,
[3]cb.en.u4elc19036@cb.students.amrita.edu,
[4]cb.en.u4elc19047@cb.students.amrita.edu,
[5]ovr_murthy@cb.amrita.edu

Abstract. Several solutions exist to detect whether English news articles are fake or not. However, in a country such as India, news articles are published in numerous regional languages too. There has been little research work done on regional languages such as Hindi. This work proposes a framework to detect Hindi fake news articles. A dataset of nearly 4500 news articles has been collected in Hindi, written using the Devanagari Script. These news articles are scraped using BeautifulSoup and Scrapy libraries. The IndicBERT and MuRIL libraries, developed earlier in Hindi, are used to generate the word embedding. SVM, Logistic Regression, and a simple CNN are used as classifiers. Extensive experiments have been methodically conducted in order to find the best combination to detect fake news in Hindi. The best combination was found to be MuRIL and SVM, yielding 98.58% classification accuracy. A standalone GUI running on a local host is built to demonstrate the working model.

Keywords: Hindi Fake News, Machine Learning, Natural Language Processing, IndicBERT, MuRIL, StanfordNLP

1 Introduction

Popular NLP applications include sentiment analysis [1] using COVID-19 Tweets [2], troll meme classification in Dravidian languages [3] (Tamil and Malayalam), identifying toxicity in comments [4]. Fake news detection is recent upcoming area of NLP application. Fake news is the purposeful or inadvertent dissemination of unconfirmed, falsified, and unauthentic information on digital media. The most infamous and impactful examples of the problems caused by fake news were clearly visible during the 2016 U.S. Presidential elections and the COVID-19 pandemic. During the COVID-19

[*] Corresponding author

pandemic, there were a lot of rumours and scares. Hardly any evidence-backed news articles were spread, and it had become very difficult to distinguish between what was fake and what was true. Furthermore, these fake news articles spread faster than the pandemic. In due course of time, fake news articles in regional languages such as Hindi have spread. This inspired an interest in Hindi false news detection studies. This paper proposes to build an ML model to distinguish between fake news articles and true news. Fake news articles usually have more of an emotional connection to the reader to make them believe strongly in their false claims. True news, on the other hand, is to the point, concise, with minimal emotions and only facts. This is what the ML model aims to use in order to disambiguate between the two. This article has the following contributions in the given context:

- Websites of two well-known media houses were scraped to acquire a labelled dataset of true and fake news dataset of around 4500 news articles.
- A performance comparison between two word embedding libraries for Hindi was conducted.
- Two classical ML models and a CNN were trained for each word embedding library to generate the final classification output.

The remainder of the article is divided as follows: Section II deals with the literature review on the Hindi news dataset. Section III contains the suggested approach used along with the contributions made in this article. Results and inferences are discussed in Section IV. Section V concludes the article and provides the future research possibilities.

2 Literature Review

Jain et al. [5] used Kaggle data and classifiers such as SVM, Naïve Bayes, Feedforward neural network, and LSTM to detect English news as true or fake. Jose et al. [6] proposed a fabricated content classifier using Bi-directional LSTM to detect English fake news on online social media networks. Garg et al. [7] tested numerous conventional machine learning models -– Naïve Bayes, SVM, K-NN, random forest, LR, decision trees, and deep learning models such as LSTM, Bi-LSTM on CFN-dataset-1 and CFN-dataset-2 to detect COVID-19 fake news. They reported that deep learning models outperform conventional machine learning models.

Sharma and Garg [8] collected a Hindi news dataset using Devanagari script collected using the Parsehub scrapping tool. They collected 1000 fake news articles from the Boomlive website and 2000 real news articles from the Jagran website.

Sudhanshu Kumar et. al. [9] collected “Hindi Fake and True Dataset” by handpicking 2100 news articles from various Hindi news channel websites, including BBC-Hindi and NDTV.

Kar et. al. [10] collected an Indic-covidemic Tweet dataset for the task of detecting fake Tweets on COVID-19. The dataset consists of 183 Bengali fake and 297 non-fake Tweets; 192 Hindi fake and 262 non-fake Tweets; 199 English fake and 305 non-fake Tweets. They separately extracted different textual and statistical information from

Tweet messages and user information to construct the features – Text Features, Twitter User Features, Fact Verification Score, and Bias Score. After comparing different feature combinations and classifiers, the mBERT classifier was found to yield the highest performance.

Most of the above regional datasets were not available publicly at the beginning of this work. A major contribution of this work is collecting Hindi data and making it available for the research community.

3 Proposed Methodology

The overall framework is shown in Fig. 1. The input news article is first pre-processed developed specifically for the Devanagari script. Using libraries built over Devanagari script, two indigenous features are extracted. Traditional machine learning models and deep learning models are being investigated as classifiers. Details of each stage are described as follows:

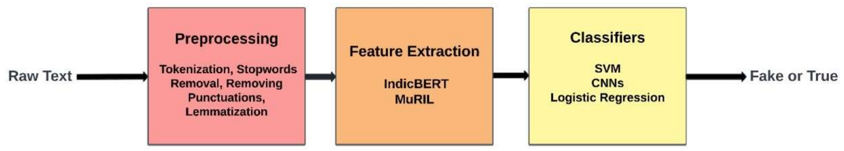


Fig. 1. Overall Framework

3.1 Dataset description

The dataset consists of fake and true news collated from two prominent sources, NDTV[1] and Boomlive[2]. True and fake news articles consisting of the headline followed by the first paragraph is collected from both the websites. A fact check has been performed by the websites where the news is labelled as true or fake. BeautifulSoup[3] and Scrapy[4] libraries have been used to extract the data from the websites. The categorization of news articles as true or fake were marked as per the labels in the respective websites.

The total data collected amounts to 4588 news pieces, in which 50% is fake and 50% is true. The dataset is collected in a way to maintain balance between true and fake news in order to avoid over-fitting of a particular class. The data collection process is summarized in Algorithm 1.

[1] https://ndtv.in/topic/fake-news

[2] https://www.boomlive.in/

[3] https://www.crummy.com/software/BeautifulSoup/

[4] https://scrapy.org/

Algorithm 1 Proposed Algorithm to scrape the News:

```
Input: Real_Websites, Fake_Websites
Output: None
   Extract_Real_Fake_Features(Real_Websites, Fake_Websites) :
1: for Each link in Real_Websites: do
2:     ContentTrue ←WebScrapping(Link)
3:     True←Label
4: end for
5: for Each link in Fake_Websites: do
6:     Content ←WebScrapping(Link)
7:     False ←Label
8: end for
9: return None
```

Real_Websites are the total websites containing the Real News. Fake_Websites are the total websites containing Fake News.

Algorithm 2 Proposed Algorithm to classify the News:

```
Input: News
Output: Classification
   Preprocessing(news):
1: News ← Tokenization(news)
2: News ← Stop_words_removal(News)
2: News ← Punctuation_removal(News)
3: News ← Lemmatization(News)
4: News ← News
5: return News
   Main Driver Algorithm:
6: news ← Input(Text)
7: News ← Preprocessing(news)
8: News ← Words_Embeddings(News)
9: Class ← Classifier(News)
10: Output ← Class(True / False)
```

The dataset and source code are made publicly available in the GitHub repository[5].

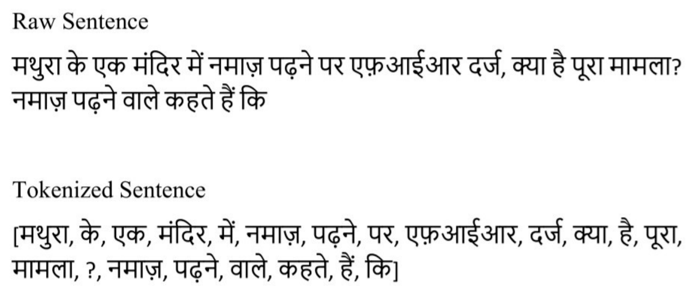
3.2 Preprocessing

The raw text collected often contains both Hindi and non-Hindi characters. Preprocessing is done before discriminative features are extracted. The following steps are performed in the pre-processing stage.

[5] https://github.com/TheGhoul27/Fake-News-Detection/tree/master/Hindi

1. Tokenization of words
2. Stopwords removal
3. Removal of Punctuations
4. Lemmatization

Tokenization. Splitting the raw text into smaller pieces called tokens is tokenization. Each character, word, or sentence is a token based on the requirement. In our use case, raw text is passed to the function to receive tokens in the form of words. For tokenization, the IndicNLP library is used which has a built-in function called 'trivial_tokenize'. An example of tokenization is given in Fig.2.



Raw Sentence

मथुरा के एक मंदिर में नमाज़ पढ़ने पर एफ़आईआर दर्ज, क्या है पूरा मामला?
नमाज़ पढ़ने वाले कहते हैं कि

Tokenized Sentence

[मथुरा, के, एक, मंदिर, में, नमाज़, पढ़ने, पर, एफ़आईआर, दर्ज, क्या, है, पूरा, मामला, ?, नमाज़, पढ़ने, वाले, कहते, हैं, कि]

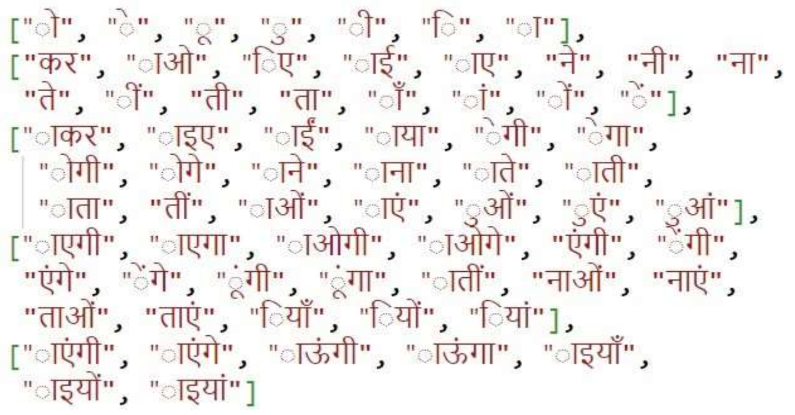
Fig. 2. Tokenization Example

Stopwords Removal. The non-important words are called stop words. They are of no use to providing any meaning in the data. The stopwords for Hindi were taken from the Mendeley dataset [11]. This dataset may not have the complete list of stopwords in Hindi, but it provides a good start. Some of these stopwords include, मैं, मुझको, मेरा, करना, करता है, कया, etc.,

Removal of Punctuation. Punctuations are unnecessary for model training as they aren't very beneficial in the terms of providing any extra information. Hence, the need for removal. This will increase the overall accuracy of the dataset. For example, in the sentence इंडोनेशिया में समुद्र के नीचे स्थित ये विष्णु भगवान का मंदिर क़रीब 5 हज़ार साल पुराना ना है।' the punctuation '।' does not add any value in text analysis.

Stemming and Lemmatization. Stemming is the process by which a word is reduced to its canonical/root form by removing any attached affixes to the terms. The suffixes to be removed were taken from the kcdon dataset[6]. Suffixes trimmed for stemming in hindi are given in Fig. 3.

[6] https://github.com/kcdon/Stemmer-Hindi-Language



```
["ो", "े", "ू", "ु", "ी", "ि", "ा"],
["कर", "ाओ", "िए", "ाई", "ाए", "ने", "नी", "ना",
 "ते", "ीं", "ती", "ता", "ाँ", "ां", "ों", "ें"],
["ाकर", "ाइए", "ाईं", "ाया", "ेगी", "ेगा",
  "ोगी", "ोगे", "ाने", "ाना", "ाते", "ाती",
  "ाता", "तीं", "ाओं", "ाएं", "ुओं", "ुएं", "ुआं"],
["ाएगी", "ाएगा", "ाओगी", "ाओगे", "एंगी", "ेंगी",
 "एंगे", "ेंगे", "ूंगी", "ूंगा", "ातीं", "नाओं", "नाएं",
 "ताओं", "ताएं", "ियाँ", "ियों", "ियां"],
["ाएंगी", "ाएंगे", "ाऊंगी", "ाऊंगा", "ाइयाँ",
 "ाइयों", "ाइयां"]
```

Fig. 3. Suffixes for Stemming

For example, the word ‘भारतीय’ is reduced to ‘भारत’.

Lemmatization is the process of converting each word to its canonical form in which the algorithms use a dictionary to find the root for each word in the dataset.

The basic function of both stemming and lemmatization are similar. Both the processes reduce a word to its ‘stem’. However, there is a subtle difference in the working of the processes. In stemming, a word is reduced to its stem based on a set of rules. However, the part of speech (POS) and the context in which the word is used are ignored. In contrast, lemmatizing converts a word to its ’lemma’ or its root form after looking at the POS and the context in which the word is used.

Due to the complex structure of word formations in the Hindi language, words such as ‘पिता’, ‘माता’ are reduced to 'पित' and ‘मात’ when stemmed, which is incorrect. This is not the case with lemmatization.

For lemmatization of Hindi language, the Stanza NLP [12] library is used. Stanza is a natural language analysis library built by the Stanford NLP group. This library can be used for text analysis and create neural network pipelines. Stanza also lemmatizes each word in a sentence to recover its canonical form.

The lemmatizer is takes in a list of words as input and returns a list with the canonical form of each word.

3.3 Word Embeddings

Word embeddings help to convert the pre-processed text into equal-length numeric vectors, satisfying certain conditions. For words with similar meanings, a similar representation is generated. This allows our ML model to not be biased or thrown off with other synonyms, and the meaning of the sentence can be preserved. In our case, for Hindi, two such libraries are compared which generate word embeddings – IndicBERT and MuRIL.

- IndicBERT [13] – It is a multilingual ALBERT model that is trained on 1.8 billion tokens from the Hindi language. This allows the embeddings for words with similar meanings to be almost the same. The huggingface transformers library is called to use its functions in the code. For each pre-processed text in our dataset, it generates a 724-length word embedding.
- MuRIL [14] – MuRIL stands for Multilingual Representations for Indian Languages. It is developed by Google specifically for Indian languages. It supports 17 Indian languages. It is also trained on the BERT language model.

3.4 Classification Models

Three different models for classification have been implemented – Convolutional Neural Network (CNN), Support Vector Machine (SVM) and Logistic Regression (LR).

CNN. CNN [15] is a supervised Deep Learning method. CNNs are a type of artificial neural network which have a convolution and pooling layer in addition to the layers of neurons already present. This type of neural network incorporates a sliding weights matrix over the original matrix of data. The values in the sliding weights matrix are convoluted with the data to extract important features from word embeddings. In our implementation, a 4x4 kernel is used to go over the input data matrix. However, this convolution operation greatly increases the dimension of the input matrix. Thus, after each convolution layer, a pooling layer is added to mathematically combine nearby values in the matrix. An average pooling layer takes the average of the neighborhood of an element in a matrix. Therefore, the size of the matrix is greatly reduced, which aids in faster computation. In addition to this, an LSTM layer was used as the first layer before the convolution and pooling layers. LSTMs are able to use information from previous inputs to bias the current output on that. This helps the network to achieve a certain degree of memory and not only work with the current inputs. It also helps to remedy the issues of vanishing and exploding gradients in recurrent neural networks. To further improve the neural network's ability, a dropout layer is introduced after the convolutions. This is to prevent overfitting of the model. This operation randomly discards 40% of the neurons while training and does not update weights. Therefore, the weights of the neurons do not exactly mimic that of the input and obtain general values which helps in the overall accuracy of the model. The final layer is a sigmoid layer which returns a value between 0 and 1. A value above 0.5, i.e., closer to 1 indicates that the news is true; while a value below 0.5 indicates that the news is false.

SVM. SVM [16] is a supervised learning technique that can perform nonlinear classification by transforming inputs directly into n-dimensional feature vector spaces. The value of a single feature shows the value of a specific coordinate and each data item is a point in an n-dimensional space, where n is the number of features. Classification is performed by locating the hyperplane that best separates the given classes. In our implementation, an RBF kernel [17] is used. This makes the decision boundary as a non-linear function instead of a linear hyperplane. The equation used by sklearn to implement the RBF function is given below.

$$K(x, x') = e^{-\gamma * |x - x'|^2} \tag{1}$$

γ denotes the amount of influence a point has on the classifier.[7]

LR. Logistic regression [16] is a statistical algorithm used for classification problems. The algorithm assigns observations to a distinct set of classes using the concept of probability. This regression algorithm dampens the cost function between 0 and 1. The sigmoid function converts the expected values to probabilities which gives a real value between 0 and 1. A threshold value is determined by classifying the probability of exceeding a certain amount.

3.5 Classification

Each news article is first preprocessed using the pipeline proposed in Sec III B. Following this, the cleaned news article is passed to the word embedding models to convert the text into numeric vectors. This is followed by passing the word embeddings to any one of the ML models stated above as per the user's choice. These models output the final predicted class of the input text.

4 Standalone app

Eel[8] is a Python library that eases the process of making offline web apps in Python. It acts as a bridge between the HTML and CSS frontend and the backend Python code. Using this library, an easy-to-use GUI was created where the user can type in a piece of news to be validated. The interface also has an option to choose which models to use for word embedding and classification, thus making it interactive. All the trained models are stored at the backend and, based on the user's choice, the appropriate embedding – model pair is selected to validate the user's data. The output is presented as a popup window where the piece of news is mentioned as true or false. The complete pipeline is packaged as an executable file and can be run on any Windows machine. The working of the application with the various options and a sample classification is shown in the link provided[9].

[7] https://towardsdatascience.com/svm-classifier-and-rbf-kernel-how-to-make-bettermodels-in-python-73bb4914af5b

[8] https://github.com/ChrisKnott/Eel

[9] https://github.com/TheGhoul27/Fake-News-Detection/tree/master/Hindi/Frontend/Software_Screenshots

5 Experiments and Results

The ML and DL models were trained on a local machine with a dedicated GPU. To implement the pre-processing methods, StanfordNLP, MuRIL, IndicBERT were used, and for training, scikit-learn and keras were called. Maximum features size generated by the word embedding models is 768. Training dataset size is 80%, Test dataset size is 10% and Cross Validation dataset size is 10%. TABLE II represents the confusion matrix of various ML models. Among IndicBERT and MuRIL word embedding libraries, the latter performs consistently better and that can be seen in the accuracy table too. Among the various ML Models, SVM was the best. As an overall combination, MuRIL in combination with an SVM classifier produces the best accuracy of 98.56%. This is the proposed pipeline to classify Hindi news articles into true and fake.

Table 1. Accuracy Table

Classifier	***MuRIL***	***IndicBERT***
SVM	**98.58%**	96.51%
Logistic Regression	98.26%	96.95%
CNN	97.82%	95.42%

Table 2. Confusion Matrices

Type	***TN***	***FN***	***FP***	***TP***
CNN + MuRIL	207	8	2	242
SVM + MuRIL	442	13	0	463
Logistic Regression + MuRIL	442	13	0	463
CNN + IndicBERT	209	13	8	229
SVM + IndicBERT	431	27	5	455
Logistic Regression + IndicBERT	436	22	6	454

TN – True Negative, FN – False Negative
FP – False Positive, TP – True Positive

6 Conclusion and Future Work

This paper proposes an end-to-end pipeline to detect fake news in Hindi. Two word embedding libraries and three classifier models were used for detecting fake news articles. The proposed algorithm yielded the highest accuracy of 98.58% on the dataset using MuRIL and SVM. This dataset could be used to detect, evolve, and mitigate disinformation. In the future the labelling of the dataset can be manually verified by the authors and the dataset size can be increased. This will help in better prediction and cross validation.

References

1. Nair, A.J., G, V., Vinayak, A.: Comparative study of twitter sentiment on covid - 19 tweets. In: 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), pp. 1773–1778 (2021). https://doi.org/10.1109/ICCMC51019.2021.9418320
2. Bl, M., Midha, S., Murthy Oruganti, V.R.: Sentiment analysis in indian subcontinent during covid-19 second wave using twitter data. In: 2021 IEEE 9th Region 10 Humanitarian Technology Conference (R10-HTC), pp. 01–06 (2021). https://doi.org/10.1109/R10-HTC53172.2021.9641559
3. B, P., Chakravarthi, B.R., Subramanian, M., B, B., Kp, S., V., D., K, S., Pandian, A., Kumaresan, P.K.: Findings of the shared task on multimodal sentiment analysis and troll meme classification in dravidian languages. Proceedings of the Second Workshop on Speech and Language Technologies for Dravidian Languages (2022)
4. Garlapati, A., Malisetty, N., Narayanan, G.: Classification of toxicity in comments using nlp and lstm. In: 2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS), vol. 1, pp. 16–21 (2022). https://doi.org/10.1109/ICACCS54159.2022.9785067
5. Jain, P., Sharma, S., Monica, Aggarwal, P.K.: Classifying fake news detection using svm, naive bayes and lstm. In: 2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence), pp. 460–464 (2022). https://doi.org/10.1109/Confluence52989.2022.9734129
6. Jose, X., Kumar, S.M., Chandran, P.: Characterization, classification and detection of fake news in online social media networks. In: 2021 IEEE Mysore Sub Section International Conference (MysuruCon), pp. 759–765 (2021). https://doi.org/10.1109/MysuruCon52639.2021.9641517
7. Garg, R., S, J.: Effective fake news classifier and its applications to covid-19. In: 2021 IEEE Bombay Section Signature Conference (IBSSC), pp. 1–6 (2021). https://doi.org/10.1109/IBSSC53889.2021.9673448
8. Sharma, D.K., Garg, S.: Machine learning methods to identify hindi fake news within social-media. In: 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), pp. 1–6 (2021). https://doi.org/10.1109/ICCCNT51525.2021.9580073
9. Kumar, S., Singh, T.D.: Fake news detection on hindi news dataset. Global Transitions Proceedings **3**(1), 289–297 (2022). https://doi.org/https://doi.org/10.1016/j.gltp.2022.03.014. International Conference on Intelligent Engineering Approach(ICIEA-2022)
10. Kar, D., Bhardwaj, M., Samanta, S., Azad, A.: No rumours please! a multi-indiclingual approach for covid fake-tweet detection. 2021 Grace Hopper Celebration India (GHCI) pp. 1–5 (2021)
11. Jha, V., Manjunath, N., Shenoy, P.D., Venugopal, K.R., Patnaik, L.M.: Homs: Hindi opinion mining system. In: 2015 IEEE 2nd International Conference on Recent Trends in Information Systems (ReTIS), pp. 366–371 (2015). https://doi.org/10.1109/ReTIS.2015.7232906

12. Qi, P., Zhang, Y., Zhang, Y., Bolton, J., Manning, C.D.: Stanza: A Python natural language processing toolkit for many human languages. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (2020). URL https://nlp.stanford.edu/pubs/qi2020stanza.pdf
13. Kakwani, D., Kunchukuttan, A., Golla, S., N.C., G., Bhattacharyya, A., Khapra, M.M., Kumar, P.: IndicNLPSuite: Monolingual Corpora, Evaluation Benchmarks and Pre-trained Multilingual Language Models for Indian Languages. In: Findings of EMNLP (2020)
14. Khanuja, S., Bansal, D., Mehtani, S., Khosla, S., Dey, A., Gopalan, B., Margam, D.K., Aggarwal, P., Nagipogu, R.T., Dave, S., Gupta, S., Gali, S.C.B., Subramanian, V., Talukdar, P.: Muril: Multilingual representations for indian languages (2021)
15. O'Shea, K., Nash, R.: An introduction to convolutional neural networks (2015). https://doi.org/10.48550/ARXIV.1511.08458. URL https://arxiv.org/abs/1511.08458
16. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
17. Thurnhofer-Hemsi, K., López-Rubio, E., Molina-Cabello, M.A., Najarian, K.: Radial basis function kernel optimization for support vector machine classifiers (2020). https://doi.org/10.48550/ARXIV.2007.08233. URL https://arxiv.org/abs/2007.08233