SCHIZOPHRENIA DETECTION WITH NLP

Prepared by:	**SİNEM FAİDE ALTUN**
Student No:	**090200360**
Submission Date:	**January 14, 2023**
Course:	**MAT 4901E**
Supervisor:	**PROF. DR. ATABEY KAYGUN**

Contents

Introduction

In the dynamic intersection of artificial intelligence and mental health research, this thesis embarks on a comprehensive exploration of predictive modeling for schizophrenia diagnosis, leveraging progressive Natural Language Processing (NLP) applications. The foundational framework of this project rests upon the Discussing Abstract Ideas in Schizophrenia Corpus (DAIS-C)[1], a psychological experiment involving a series of interviews with 15 individuals diagnosed with schizophrenia and 14 without psychiatric history. The DAIS-C, developed by researchers from Manchester Metropolitan University and healthcare professionals, provides a unique view into the linguistic refinements associated with schizophrenia.

Nonetheless, as we navigate the corridors of this dataset, it is important to recognize and embrace ethical considerations in the environment of psychology and artificial intelligence. We took careful consideration into maintaining the delicate balance between harnessing modern technologies for the betterment of human mental health and safeguarding ethical boundaries. The humanistic essence of psychology should not be overshadowed by the vast capabilities of AI; instead, they should harmonize to uplift and empower individuals facing mental health challenges. In an era where AI applications grow rapidly and carry the potential for misuse, our goal is not to impede technological growth but to guide it towards benevolent ends. From Naive Bayes to Support Vector and Logistic Regression, this thesis follows very experimental approaches on advanced modeling techniques, with the destination of contributing not only to the evolution of predictive modeling in mental health but also to the broader discourse on the ethical use of AI in psychiatric fields, hopefully introducing healthy trust bounds.

All of the code and detailed outputs for this project could be reached through the GitHub repository **Schizophrenia-Detection-with-NLP**.

Methodology

2.1 Data Collection

The Discussion Abstract Ideas in Schizophrenia Corpus (DAIS-C) zip file contained folders categorized as clinical (CL) and comparison (CO), representing data from participants diagnosed with and without schizophrenia.

Within these folders, files were organised into:

- Interactional: This includes transcripts involving both the speaker and interviewer, capturing the running dialogue.
- Speaker Only Raw: Each speaker's text was isolated, excluding interviewer speech and XML tags.
- Speaker only for analysis: Similar to "Speaker Only Raw," but retaining XML tags.
- Timestamped: This category involved speaker and interviewer timestamps, excluding speech and maintaining the running dialogue.[2]

Our primary focus, aligned with the project aim, centered on examining the content and structure of individual speech. Consequently, we chose to work primarily with the "Speaker Only Raw" data. It simplified our analyzing processs by eliminating external influences.

The data bundle accommodated not only the DAIS-C Corpus files but also a Metadata excel file, a resource for obtaining a comprehensive overview of the participants. This file included features, as presented in Table 2.1: Metadata Fields and Descriptions.

The Metadata folder served as a crucial source of participant information, while the "Speaker Only Raw" folder contained dialogue text files associated with these participants. The merging of these two datasets facilitated the creation of an initial dataset for analysis. This combined dataset included:

- The question posed to the participant *extracted from Metadata*
- The participant's response to the question *extracted from Speaker Only Raw*
- The participant's condition, categorized as clinical or comparison *extracted from Metadata*

The joining process involved correlating the information from these two datasets based on a common feature, the **participant ID**.

Field	Description
Participant ID	Unique identifier for each participant
Group	Clinical or comparison
Age	Age of the participant
Sex	Biological sex of the participant
Gender	Gender identity of the participant
Diagnosis	Schizophrenia diagnosis
Known EPSEs	Known extrapyramidal side effects
Medication	Medication information
Interview Only	Does the data correspond to an interview-only session.
Experiment Pathway	Describes the pathway of the experimental condition in the study.
Experiment Mode	Face to face, remote.
CLQT(+)	Presence or absence of CLQT (Cognitive Linguistic Quick Test) assessment.
CLQT Order	Order in which CLQT assessment was conducted, if applicable.
Handedness	Right-handed, left-handed.
Interview Mode	Specifies the mode of the interview session.
Face Mask (Interviewer)	Presence or absence of a face mask worn by the interviewer.
Face Mask (Participant)	Presence or absence of a face mask worn by the participant.
Interview Location	Location of the interview session.
Locale	Describes the general setting of the interview location.
Audio Quality	Reflects the quality of the audio recording during the session.
Duration	The duration of the interview.
Interviewer Device	The device used by the interviewer for the session.
Participant Device	The device used by the participant during the session.
Recording Setup	Describes the setup used for recording the session.
Recording Date	The date when the session was recorded.
Recording Time	The time at which the session recording started.
Education Level	The highest level of education attained by the participant.
GP Consent	Indicates whether consent was obtained from the participant's general practitioner.
Archival Consent	Specifies the participant's consent for archival purposes.
Initial Question Type	Describes the type of initial question posed to the participant.
Interview Question	The actual question asked during the interview.
Historic Clinical Observations	Any historical clinical observations noted for the participant.
Interviewer Observations	Observations made by the interviewer during the session.

Table 2.1: Metadata Fields and Descriptions

2.2 Text Data Preprocessing

Data preparation, particularly in the category of text data, is a crucial preliminary step before building classification models. Throughout each stage of preprocessing, a careful understanding of the data content was maintained, guiding the process of both standard techniques and more specialized functions. It is essential to emphasize that our text consists of transcripts, representing the spoken language of participants transcribed into written form. This unique nature of our data needed a thoughtful approach to ensure its suitability for meaningful analysis.

2.2.1 Pre-Processing Methodologies

Two distinct pre-processing methodologies were implemented to ascertain the most reliable predictions. This decision revolved around determining the optimal unit of data – whether to retain the text in its entirety or to segment it into smaller units through tokenization. A detailed discussion on tokenization, providing deeper insights into this process, will follow in later sections.

Following this crossroad, we acquired two separate datasets wherein the text data existed as either individual sentences or entire conversations. Both datasets underwent the same pre-modelling and modelling procedures outlined in later subsections, including data cleaning, word tokenization, spell checking, lemmatization, vectorization and subsequent training-testing of the classification models. The determination of the most optimal analysis unit was made after the modelling stages based on evaluation scores.

The two datasets' target class distribution is as follows.

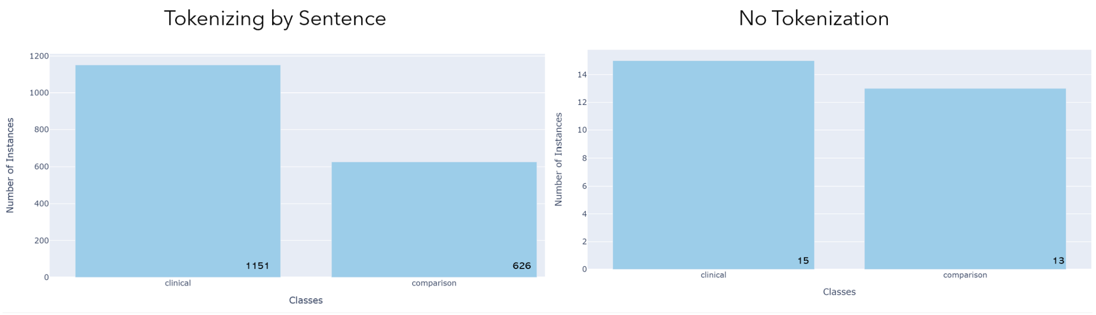


Figure 2.1: Class distribution: clinical versus comparison

Splitting by Sentences

Our objective was to dissect each participant's dialogue into individual sentences, utilizing NLTK's Tokenizer Package for sentence-based tokenization. This function operates by identifying punctuation marks that denote the end of a sentence.

However, it was observed that in the original transcript, sentence endings were indicated with a new line character, instead of punctuation marks. Leveraging Python's regular expression operations module, we efficiently replaced new lines with dots, formatting the text in a manner recognizable by the sentence tokenizer. The final data frame created in this stage contains 1777 sentences. The transformation of characters within a text chunk and tokenizer's working progress is illustrated below:

```
can you repeat that \n yes I do to some extent I I speak every day to
fellow residents in the hostel and I talk creatively from time to time
and I also write literature which isn't spoken but it is the written
word and I'm creative in that sense too an I'm saying that my main
form of creative speech is with residents in the hostel where I talk
originally from time to time \n yes I talk with residents in the
hostel and I speak originally \n

```

Listing 2.1: Original Text (Shortened Version)

```
can you repeat that .  yes I do to some extent I I speak every day to
fellow residents in the hostel and I talk creatively from time to time
and I also write literature which isn't spoken but it is the written
word and I'm creative in that sense too an I'm saying that my main
form of creative speech is with residents in the hostel where I talk
originally from time to time . yes I talk with residents in the
hostel and I speak originally .

```

Listing 2.2: Regular Expression Operations Module Replacing New Lines With Dots

```
[' can you repeat that .',

 "yes I do to some extent I I speak every day to fellow residents in the
 hostel and I talk creatively from time to time and I also write
 literature which isn't spoken but it is the written word and
 I'm creative in that sense too an I'm saying that my main form of
 creative speech is with residents in the hostel where I talk
 originally from time to time .",

 'yes I talk with residents in the hostel and I speak originally .']

```

Listing 2.3: Sentence Tokenizer Splitting The Text Into Individual Sentences

Working on Whole Texts

Working with whole texts, did not require any additional operations, allowing us to use the data frame created in section 2.1 (Data Collection) directly.

2.2.2 Standard Text Cleaning Techniques

Standard techniques of data cleaning included converting all letters to lowercase, transcribing digits into text, expanding contractions, eliminating extra spaces, and filtering out stop words. These procedures were performed via Python's NLTK (The Natural Language Toolkit) library.

2.2.3 Spell Checking

To address spelling mistakes, we employed the pyspellchecker module. This Python module, dedicated to pure spell checking, draws inspiration from Peter Norvig's blog post, "How to Write a Spelling Corrector" [3].

The spell checker module uses the Levenshtein Distance algorithm to measure the similarity between a given word and known words in its dictionary. The Levenshtein Distance is calculated by finding the minimum number of changes such as insertions, deletions, replacements or transpositions, needed to transform one word into another.The Levenshtein Distance between two strings s and t is given by the formula:

$$\text{lev}(s,t) = \begin{cases} \max(\text{len}(s), \text{len}(t)) & \text{if } \min(\text{len}(s), \text{len}(t)) = 0 \\ \min \begin{cases} \text{lev}(\text{tail}(s), t) + 1 \\ \text{lev}(s, \text{tail}(t)) + 1 \\ \text{lev}(\text{tail}(s), \text{tail}(t)) + \delta_{\text{head}(s) \neq \text{head}(t)} \end{cases} & \text{otherwise} \end{cases}$$

where $\text{tail}(s)$ represents the substring of s excluding its first character, $\text{head}(s)$ is the first character of s, and $\delta_{\text{head}(s)\neq\text{head}(t)}$ is the Kronecker delta function, which is 1 if $\text{head}(s) \neq \text{head}(t)$ and 0 otherwise.

2.2.4 NLTK Corpus Based Filtering

To ensure the utmost cleanliness of our text data, we incorporated the NLTK library's English corpus. With this extra step, after checking the spelling of each word, we compared them with NLTK's internal English dictionary, which holds a list of recognized and proper words. The goal was to filter out words that don't fit standard English language rules. This NLTK corpus-based filtering helped us keep our dataset clean from words that aren't part of the English language or don't make sense. This careful process, using the knowledge stored in the NLTK library, was essential in refining our text data for the next steps in our project.

2.2.5 Recovery Mechanism for Eliminated Words

NLTK corpus-based filtering, was an experimental approach, to gauge its impact and necessity NLTK corpus-based filtering underwent experimental testing. Two tests were orchestrated to assess its effectiveness and determine its necessity.

In Test 1, only the spell checker module operated on the tokens, while Test 2 included both spell checking and corpus-based filtering. Representing the results as sets, the set difference $(Test1)\backslash(Test2)$ contained words eliminated by corpus filtering.

This distinct set could be manually examined to ensure that potentially vital inputs were not wrongfully removed from the data. This approach, was feasible with our small dataset, on the other hand the need for new strategies might arise when dealing with larger data sets.

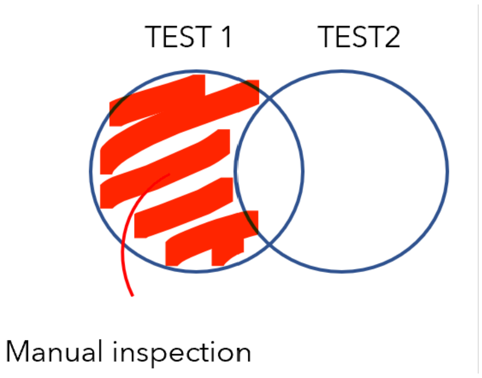


Figure 2.2: The colored distinct set represent words eliminated by corpus filtering.

2.2.6 Tokenization and Lemmatization

In the scope of text data preprocessing, tokenization and lemmatization play pivotal roles. The primary objective is to enable computers to comprehend human language efficiently and computationally. Natural Language Toolkit library was once again helpful as the word tokenizer breaks down our text data into distinct word forms.Simultaneously, the WordNet lemmatizer, a component of NLTK, streamlines words to their base or root form, referred to as the lemma.

With word representations, WordNet is a very popular, classical approach, currently hosting more than 150,000 words and more than 100,000 synonym groups. It depends on an external lexical knowledge base that captures details about a word's definition, synonyms, ancestry, descendants, and more.[4]

2.3 Vectorization Methods

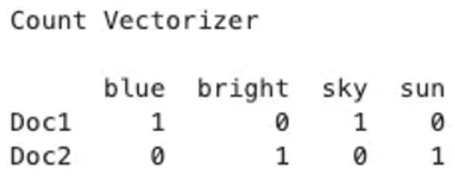
Transitioning from raw and cleaned text data to a form ready for classification algorithms, involves a crucial intermediary step: text vectorization. In this process, words are transformed into numerical representations, capturing semantic relationships and contextual nuances. To help enable models to recognize complex patterns, vectorizato is pivotal phase in natural language processing applications. Consequently, the accuracy and performance of text classification tasks are improved, portraying our predictions more reliable and informed.

2.3.1 Count Vectorization

Count vectorization stands as a straightforward approach to text vectorization, representing a vector by the count of distinct words within each unit of text, whether that is a sentence, a paragraph or an entire document. While its simplicity proves advantageous, count vectorization lacks a detailed analysis of words and their semantic relationships. This method may prioritize words abundant in a corpus, assuming them to be statistically significant due to higher counts.

In our specific case, this might be advantageous if certain repeated words, such as ”mhm” and ”ehm,” carry relevance in our predictions.

However count vectors are often not preferred as they can pose challenges related to memory and processing. These challenges arise from high vocabulary sizes or larger sparse representations. An example of a count vector is given below.



```
Count Vectorizer

      blue  bright  sky  sun
Doc1     1       0    1    0
Doc2     0       1    0    1
```

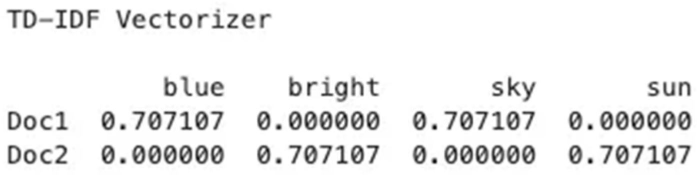
Figure 2.3: Sparse matrix of a count vectorizer
[5]

2.3.2 TF-IDF Vectorization

While maintaining simplicity, Term Frequency – Inverse Document Frequency (TF-IDF) vectorization takes a more nuanced approach by assigning weights to words, reducing the impact of common words. It has the benefit of helping identify terms that are significant to a particular document but not necessarily common across the entire corpus.

Nonetheless, this method still lacks the ability to capture semantic relations and word oderings.

The formula for TF-IDF vectorization will be introduced in the following section 3.3 (Design). An example of a TF-IDF vector is given below.



```
TD-IDF Vectorizer

          blue    bright       sky       sun
Doc1  0.707107  0.000000  0.707107  0.000000
Doc2  0.000000  0.707107  0.000000  0.707107
```

[5]

Figure 2.4: Sparse matrix of a tf-idf vectorizer

2.3.3 Word Embeddings

Word to Vector

In 2010, Tomáš Mikolov alongside co-authors initiated a project involving a simple recurrent neural network with a single hidden layer for language modelling [6]. This project was finalized and published in 2013 by a team of researchers led by Mikolov at Google over two papers.

The word2vec algorithm estimates vector representations of words with the aim of detecting their meanings and usage in the given text. Its input is a large corpus of text and the output is carefully drafted vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. These vectors capture the semantic and syntactic qualities of words, a basic mathematical formula of cosine similarity can tell us based on word2vec's mechanism how similar are words with their context. This process could be conducted with two main neural network architectures: Continuous Bag-Of-Words (CBOW) or continuously Skip-Gram.

The word2vec algorithm aims to discern the meanings and usage of words within a given text by estimating vector representations. Operating on a large corpus of text, it crafts a carefully designed vector space, typically spanning several hundred dimensions. Each unique word in the corpus is assigned a corresponding vector in this space, effectively capturing both semantic and syntactic qualities. The algorithm's mechanism employs a basic mathematical formula—cosine similarity—to quantify the similarity between words and their contexts. This intricate process can be executed through two primary neural network architectures: Continuous Bag-Of-Words (CBOW) or continuously Skip-Gram.

Within a specified context window size, Continuous Bag-Of-Words (CBOW), delves into estimating the relationship between words, detecting their influence on each other. It can be summarized as configuring a missing word in the text based on its neighboring words. The mathematical explanation of CBOW is as follows.

1. **Input Representation:**

$$\text{Input} = \frac{1}{C}\sum_{i=1}^{C} W \cdot w_i$$

2. **Hidden Layer:**

$$\text{Hidden Layer} = \text{Input} \cdot H$$

3. **Output Layer:**

$$\text{Output Layer} = \text{softmax}(\text{Hidden Layer})$$

where

$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

4. **Loss Function:**

$$\text{Loss} = -\sum_j y_j \cdot \log(\text{softmax}(\text{Hidden Layer})_j)$$

where y is the one-hot encoded vector for the true target word.

In contrast to CBOW, the skip-gram architecture operates by having the model predict the surrounding words within the defined window context, using the information from the current word. Below, is the detailed mathematical representation of skip-gram.

1. **Input Representation:**

$$\text{Input} = W \cdot w_i$$

2. **Hidden Layer:**

$$\text{Hidden Layer} = \text{Input} \cdot H$$

3. **Output Layer:**

$$\text{Output Layer} = \text{softmax}(\text{Hidden Layer})$$

where

$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

4. **Loss Function:**

$$\text{Loss} = -\sum_j y_j \cdot \log(\text{softmax}(\text{Hidden Layer})_j)$$

where y is the one-hot encoded vector for the true context word.

Bidirectional Encoder Representations from Transformers

The advent of Transformers[7] marked a significant shift in natural language processing, diverting attention from traditional mechanisms like recurrent neural networks. This preference was attributed to Transformers' parallel processing capabilities and the introduction of a self-attention mechanism. Building on the Transformers architecture, Google researchers introduced BERT (Bidirectional Encoder Representations from Transformers) in October 2018.

BERT operates through two tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP).

In MLM, a percentage of tokens in the input sequence are randomly replaced with a special token **"[MASK]"**. The model's task is to predict these masked tokens based on surrounding context, training on both left and right sides of the token. With random word and unchanged word replacements, BERT performs in-depth masking, showcasing its bidirectional processing capabilities. In addition to MLM, NSP -a binary classification task tests the relation between sentences and beyond by determining whether a given sentence can follow the current sentence. [8]

2.4 Modeling Approaches

2.4.1 Choice of Classification Algorithms

Logistic Regression

Logistic regression is a very classic and reliable algorithm with binary classification problems, because of its simplicity and relatively fast computation time. This algorithm, applied in the context of our project, has the capacity to discern intricate relationships within the feature space, providing valuable insights into the factors influencing the likelihood of a participant being diagnosed with schizophrenia.

The logistic regression formula for binary classification is given by the sigmoid function:

$$P(Y=1) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_n X_n)}} \tag{2.1}$$

where:

- Y is the binary outcome (e.g., 1 for the positive class, 0 for the negative class),
- $X_1, X_2, \ldots, X_n$ are the features,
- $\beta_0, \beta_1, \ldots, \beta_n$ are the coefficients,
- e is the base of the natural logarithm.

This formula transforms the linear combination of features into probabilities. The logistic regression model predicts $Y = 1$ if the probability is greater than a threshold (typically 0.5) and $Y = 0$ otherwise.

Naive Bayes

In the selection of the modeling approach for this project, another choice gravitated towards employing Naive Bayes classifiers, specifically Multinomial, Complement, and Bernoulli variants. The rationale behind this decision stems from the suitability of Naive Bayes algorithms for text classification tasks, given their efficiency and simplicity. Naive Bayes algorithm is built upon the Bayes' theorem, relating the conditional and marginal probabilities of random variables. It is expressed as:

$$P(A|B) = \frac{P(B|A)\cdot P(A)}{P(B)} \tag{2.2}$$

where:

- $P(A|B)$ is the conditional probability of event A given that event B has occurred.
- $P(B|A)$ is the conditional probability of event B given that event A has occurred.
- $P(A)$ is the probability of event A.
- $P(B)$ is the probability of event B.

The Naive Bayes algorithm's oversimplified assumption of feature independence align well with traditional text representations such as count or TF-IDF vectorizers. However, a pertinent question arises regarding its compatibility with word embedding methods like Word2Vec and BERT, which prioritize capturing semantic relations between words.

Support Vector Machine

A Support Vector Machine, is a type of large-margin classifier. This machine learning method operates in vector space, aiming to identify a decision boundary between two classes that maximizes the separation from all points in the training data. This approach considers the possibility of discounting certain points as outliers or noise during the determination of the decision boundary.[9]

The choice of kernel plays a pivotal role in influencing the hyperplane that separates features. The linear kernel operates under the assumption that the relationship between input features is approximately linear, making it suitable for linearly separable data. Conversely, polynomial kernels are effective in capturing non-linear relationships. The radial basis function (RBF) kernel, known for its versatility, is effective in tackling non-linear patterns and complex relationships within the data. Lastly, the sigmoid kernel is particularly suitable for neural network applications and binary classification problems. It maps non-linear data to a higher dimensional space. All these kernels were experimented within the Naive Bayes modeling framework, to avoid wrong assumptions about the linearity, complexity of the data and identify the best fit.

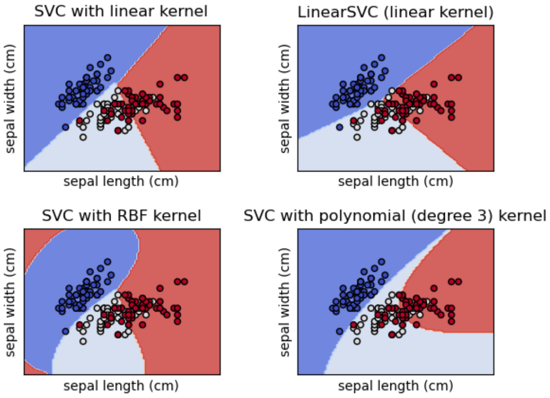


Figure 2.5: Different types of kernels on SVM models.
[10]

2.4.2 Principal Component Analysis (PCA) for Dimensionality Reduction

Principal component analysis (PCA) is a mathematical technique that is designed to reduce the dimensionality of the data while keeping most of the variation in the data set.[11]

This reduction is achieved by identifying directions along which the variation in the data is maximal, these directions are referred as principal components. Consequently, samples can be represented by a considerably smaller number of variables instead of values for thousands of variables.[12]

We applied principal component analysis to our TF-IDF and Count vectors to estimate how much time efficiency could be gained without losing impactful data. As count and TF-IDF vectorizer typically generate sparse matrices, reducing dimensions was a logical move. However, due to the dense, high-dimensional nature of word embeddings and their role in capturing semantic relationships between words, we chose not to apply PCA to Word2Vec and BERT embedded text.

The principal components $\mathbf{PC}_1, \mathbf{PC}_2, \ldots, \mathbf{PC}_p$ are formed by arranging the eigenvectors in decreasing order of their corresponding eigenvalues.

$$\mathbf{PC}_i = \mathbf{v}_i \tag{2.3}$$

The transformed data $\mathbf{Y}$ is obtained by projecting the original data onto the principal components:

$$\mathbf{Y} = \mathbf{X}\mathbf{V} \tag{2.4}$$

where:

$\mathbf{X}$: Original data matrix

$\mathbf{V}$: Matrix of principal components

Figure 2.6: Principal Component Analysis (PCA) Process

2.4.3 Evaluation Metrics

It is crucial to establish the criteria by which your model's performance will be assessed, determining its efficiency and the accuracy of its predictions. For classification problems, various evaluation metrics are available and we will review them in this section.

Accuracy

The accuracy metric represents the ratio of correctly predicted instances to the total number of instances. It is a proficient and reliable metric for balanced datasets. In our case, accuracy could be a good measure for the non-tokenized dataset left in dialog form. However, for our tokenized to sentence-level dataset, accuracy yields unreliable conclusions.

$$\textit{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \times 100 \tag{2.5}$$

where:

$$\begin{aligned} \text{Number of Correct Predictions} &: \text{The count of correctly classified instances} \\ \text{Total Number of Predictions} &: \text{The total number of instances} \end{aligned}$$

Figure 2.7: Accuracy Calculation Formula

Precision

Precision is the ratio of true positives to the sum of true positives and false positives. It is a smooth indicator of the model's efficiency if our priority is minimizing false positives. In our case false positives are equivalent to falsely diagnosing people as schizophrenic, thus precision is a matter we should be aware of.

$$\textit{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.6}$$

where:

$$\begin{aligned} \text{True Positives} &: \text{The count of correctly predicted positive instances} \\ \text{False Positives} &: \text{The count of incorrectly predicted positive instances} \end{aligned}$$

Figure 2.8: Precision Calculation Formula

Recall

Recall is the ratio of true positives to the sum of true positives and false negatives. It is important when capturing as many positive instances as possible is a priority. In our context, positive instances refer to individuals diagnosed with schizophrenia. Therefore, considering our specific goal, determining whether recall is our preferred metric becomes a nuanced decision.

$$\textit{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.7}$$

where:

$$\text{True Positives} : \text{The count of correctly predicted positive instances}$$
$$\text{False Negatives} : \text{The count of incorrectly predicted negative instances}$$

Figure 2.9: Recall Calculation Formula

F1 Score

F1 score proposes a harmonic mean of precision and recall, providing a balance perspective on false positives and false negatives. It is an adequate evalution metric in our case as there is an imbalance between positive and negative classes within our sentence-level tokenized data frame.

$$\textit{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.8}$$

Figure 2.10: F1 Score Calculation Formula

ROC-AUC (Receiver Operating Characteristic - Area Under the Curve)

This metric represents the area under the ROC curve, which plots the true positive rate against the false positive rate. It is used to evaluate the model's ability to distinguish between classes.

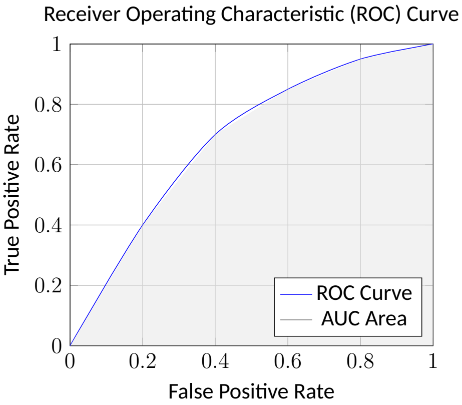


Figure 2.11: Receiver Operating Characteristic (ROC) Curve with Area Under the Curve (AUC) highlighted.

Time Evaluation

Runtime is a critical factor in software development. In NLP text classification problems, particularly those involving a large vocabulary, the choice of the best-fitted model is not solely based on correct prediction percentages but also on the computational time required for processing. Therefore, runtime is a crucial metric for us to consider.

2.4.4 Bootstrapping for Model Evaluation

Confidence intervals have become essential tools for applied statisticians, seamlessly merging point estimation and hypothesis testing in a single, intuitively appealing inferential statement.[13]. By adapting the bootstrapping methodology to derive an interval for the mean, standard deviation of evaluation metrics for each model, we aim to make a more generalized and stable statement regarding the success and reliability of the model.

The bootstrapping process in our project is depicted in the figure below.

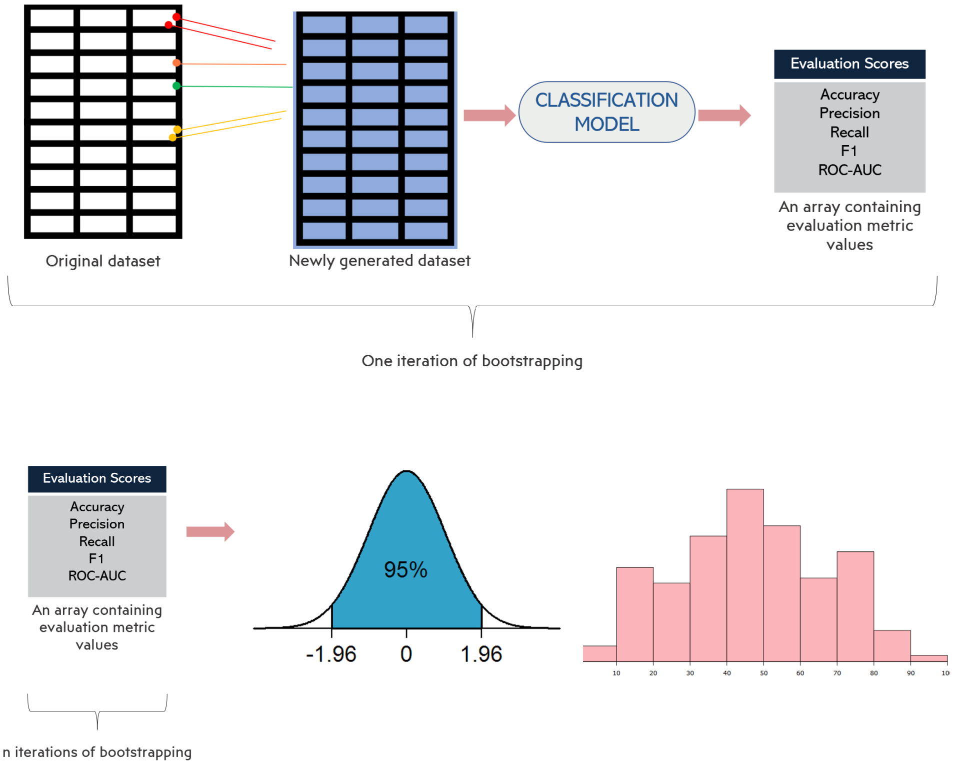
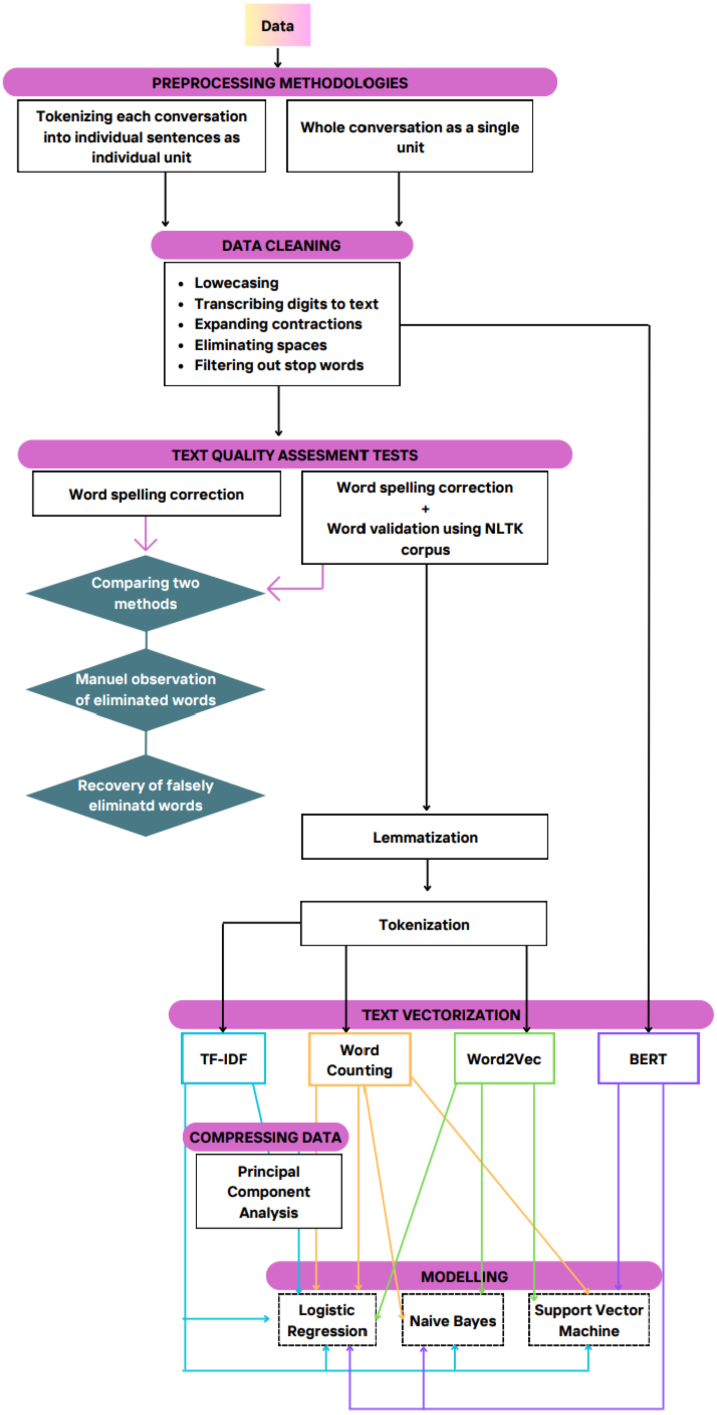


Figure 2.12: Bootstrapping for model evaluation.

Experiments

3.1 Overview of The Project



Data
PREPROCESSING METHODOLOGIES
Tokenizing each conversation into individual sentences as individual unit
Whole conversation as a single unit
DATA CLEANING
Lowecasing
Transcribing digits to text
Expanding contractions
Eliminating spaces
Filtering out stop words
TEXT QUALITY ASSESMENT TESTS
Word spelling correction
Word spelling correction + Word validation using NLTK corpus
Comparing two methods
Manuel observation of eliminated words
Recovery of falsely eliminatd words
Lemmatization
Tokenization
TEXT VECTORIZATION
TF-IDF
Word Counting
Word2Vec
BERT
COMPRESSING DATA
Principal Component Analysis
MODELLING
Logistic Regression
Naive Bayes
Support Vector Machine

Figure 3.1: Flowchart of the project

3.2 Data Selection

In the study we used only the patient part of the conversation data.

We applied two different pre-processing methodologies:

1. We tokenized each conversation into individual sentences, and consider each sentence as an individual unit.

2. We took the whole conversation as a single data unit.

After these distinct preprocessing methods, we followed the same steps: data cleaning, word tokenization, spell checking, lemmatization, and finally vectorization.

3.3 Design

During the data cleaning phase we employed various cleaning techniques. We recognized that the available text data constitutes a transcript that represents the participants' spoken language in written form. So early on, it became evident that misspellings could be ignored and would not be significantly impact the target variable, as they were not inherently tied to the participant. Hence, we used the standard techniques: we converted all letters to lowercase, transcribed digits into text, expanded contractions, eliminated extra spaces, and filtered out stop words. We used python's NLTK to perform these tasks to obtain structured word forms.

We must mention that detecting misspelled or non-English words represented a challenge. During this step, we conducted two tests to identify the optimal approach for Text Quality Assessment

For the first test, we used only the spell checker function to recover potentially relevant data. The second test involved verifying words through NLTK words corpus which contains words from various sources, dictionaries, and word lists. We found that the second approach seemed more fitting as spell checker alone was not capable of detecting and removing non-English words. Introducing NLTK's corpus, allowed us to perform word validation.

Unfortunately, we observed that corpus based validation was not

In the next step of our data processing pipeline, we used lemmatization to derive the final normal form of words before we vectorized our text data.

With the text vectorization phase we employed four different vectorization methods on the prepared tokens: a TF-IDF vectorizer, a pure counting vectorizer, a Word2Vec vectorizer, and a BERT vectorizer.

The TF-IDF vectorization method estimates the 'weight' of each token through mathematical calculations, establishing a relationship between the token and the entire document.

$$TFIDF(t,d) = TF(t,d) \times IDF(t,D)$$

$$TF(t,d) = \frac{\text{Number of times token } t \text{ appears in document } d}{\text{Total number of tokens in document } d}$$

$$IDF(t,D) = \log\left(\frac{\text{Total number of documents in the corpus } N}{\text{Number of documents containing token } t}\right)$$

In contrast, the pure counting vectorizer employs a simpler technique, counting the occurrence of unique tokens within a text. Additionally, we used Hugging face library's Word2Vec and BERT (Bidirectional Encoder Representations from Transformers) for embedding for comparison purposes, each contributing unique strengths to the analysis.

Finally, during the modelling phase classification based algorithms seemed appropriate for our text classification problem. Also Compression by Principal Component Analysis was possible on the TF-IDF and count vector.Logistic Regression, Naive Bayes and Suppor Vector Machine algorithms were triggered by bootstrapping with an iteration of 50, to get a confidence interval of evaluation metrics such as F1, ROC-AUC. Histograms of bootstrapping results were displayed along with the time to run the program for each individual model. It is important to note that

The modeling phase aimed to address the text classification problem through classification-based algorithms: Logistic Regression, Naive Bayes, and Support Vector Machine. Additionally, Principal Component Analysis (PCA) was applied to compress TF-IDF and count vectors so as to observe if a well-working model could be generated with reduced data.

All the algorithms were utilized by bootstrapping with 50 iterations, essentially creating multiple datasets of the same size as our original dataset by randomly selecting samples from the original dataset, with replacement. Each of these newly created datasets a.k.a bootstrap samples, is used to train and evaluate the classification models. This results in multiple performance estimates such as ROC-AUC and F1-score, from different iterations of the model trained on slightly different datasets. By aggregating the results from these multiple iterations, bootstrapping allowed us to estimate the variability or uncertainty associated with the model's performance metrics. Histograms depicting the results from bootstrapping were generated, accompanied by runtime details for each individual model.

Analysis

So far, we have progressed through the stages of collecting, preprocessing, cleaning, vectorizing and finally, modeling the text data. As we approach the end of the project, it is crucial to thoroughly review and analyze the calculated metrics. This analysis will guide us in determining the most effective approach for predicting Schizophrenic patients.

As we aggregate the metrics calculated from various input-model combinations, we find ourselves with approximately 576 data points for analysis.

$$\begin{aligned}
a &= \text{6 Input Types}\\
b &= \text{1 Logistic Regression Models}\\
c &= \text{3 Naive Bayes Models}\\
d &= \text{4 Support Vector Machine Models}\\
e &= \text{6 Evaluation Metrics}\\
f &= \text{2 Data Frames}
\end{aligned}$$

The result of this expression is:

$$[(a * b) + (a * c) + (a * d)] * e * f = 576$$

In the Appendix section, you will find two data frames containing the results of our evaluations for tokenized and untokenized versions of the data. It is important to note that manually examining these results is not ideal. A simple algorithm, guided by our preferred evaluation metric-time efficiency, could help us determine the ideal model. Nonetheless, for now, we highlighted some of the results in the following subsections.

4.1 Analyzing Results of Text Classification on Sentence-Level Tokenized Data

The histograms presented below depict the results of each evaluation metric obtained through 100 rounds of bootstrapping. From our examination, the following conclusions emerge:

For sentence-level tokenized text data,

1. Logistic Regression models consistently perform well across different evaluation metrics.
2. BERT embeddings and count vectorization prove to be effective input representations.
3. The Naive Bayes algorithm's performance with word embeddings is suboptimal, aligning with our initial suspicion. Naive Bayes' assmption of independent features contradicted with word embedding's nature of capturing relations between words.
4. Word2Vec embeddings visibly underperformed compared to BERT. Upon reflection, we attribute this to Word2Vec's adherence to a classical tokenization approach, whereas BERT employs its own. This distinction likely contributes to BERT's ability to understand and capture the essence of text data, detecting crucial semantic relations. While Word2Vec focuses solely on the local context of a word, BERT's bidirectional functionality allows it to gain a broader perspective on the complete context.
5. The compressed versions of both count and TF-IDF vectorizations demonstrated the creation of well-equipped models, emphasizing that valuable predictions can be achieved with significantly reduced computation time when applied to our dataset. This finding underscores the efficiency gains associated with compressed representations, without compromising the predictive capabilities of the models.

The selection of the best model, along with the optimal tokenization method, depends on the priorities:

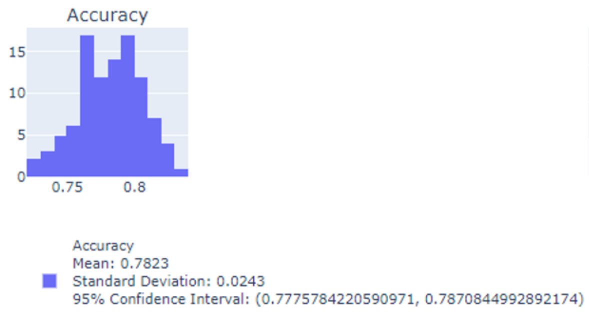
- If Overall Performance is Critical: Opt for the Logistic Regression Model with BERT Embedding for its consistent excellence across multiple metrics.
- If Efficiency is a Top Priority: Consider Complementing Naive Bayes with Count Vectorization for its short runtime and competitive recall.
- If a Balanced Approach is Desired: Lean towards the Logistic Regression Model with Compressed Count Vectorization for its efficiency and competitive performance across various metrics.

4.1.1 The Highest Mean of Accuracy Scores

As mentioned earlier, accuracy proves to be an unreliable metric for imbalanced datasets, and our dataset for sentence-level tokenization is indeed highly imbalanced. This imbalance can significantly impact the interpretation of model performance, emphasizing the need for alternative evaluation metrics.

SVM Model

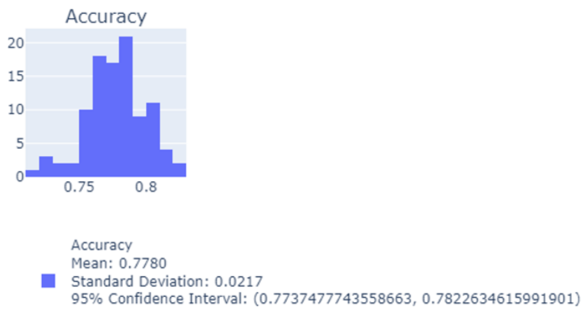
- Input: TF-IDF Vector
- Kernel: RBF
- Runtime: 2747.84 seconds $\approx$ 45 minutes



Accuracy
Mean: 0.7823
■ Standard Deviation: 0.0243
95% Confidence Interval: (0.7775784220590971, 0.7870844992892174)

Logistic Regression Model

- Input: BERT embedding
- Runtime: 61.44 seconds



Accuracy
Mean: 0.7780
■ Standard Deviation: 0.0217
95% Confidence Interval: (0.7737477743558663, 0.7822634615991901)

SVM Model

- Input: Compressed TF-IDF vector
- Kernel: RBF
- Runtime: 184.84 seconds $\approx$ 2 minutes

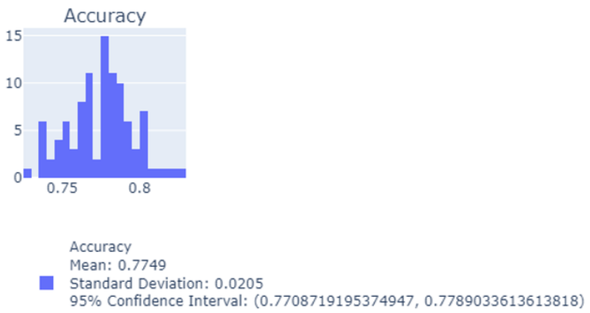
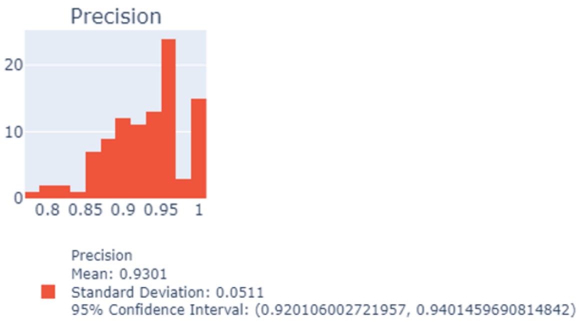


Figure 4.1: Your figure caption here.

4.1.2 The Highest Mean of Precision Values

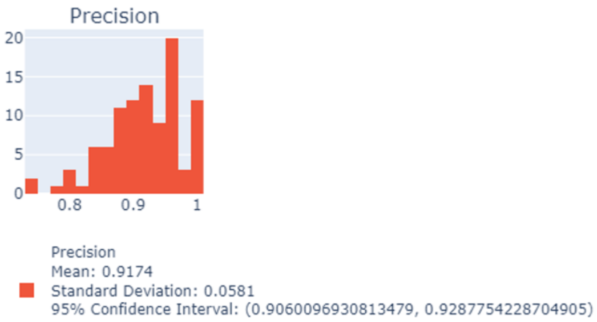
SVM Model

- Input: Count Vector
- Kernel: RBF
- Runtime: 2562.54 seconds $\approx$ 48 minutes



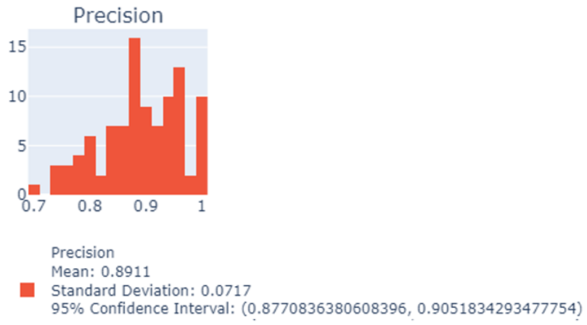
SVM Model

- Input: Compressed Count Vector
- Kernel: RBF
- Runtime: 679.47 seconds $\approx$ 11 minutes



Multinomial Naive Bayes Model

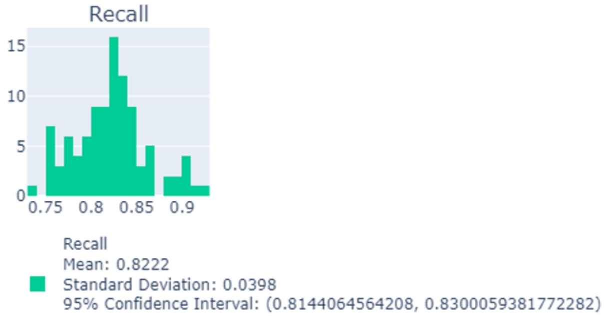
- Input: TF-IDF Vector
- Runtime: 17.09 seconds



4.1.3 The Highest Mean of Recall Values

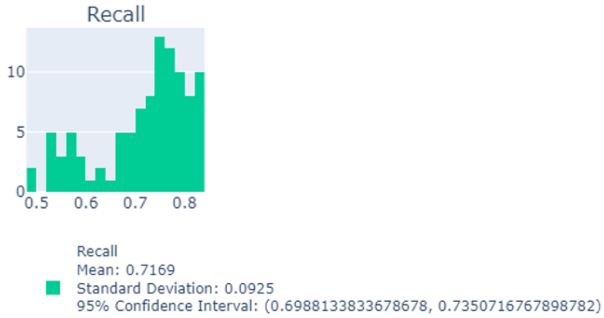
Complement Naive Bayes Model

- Input: Count Vector
- Runtime: 23 seconds



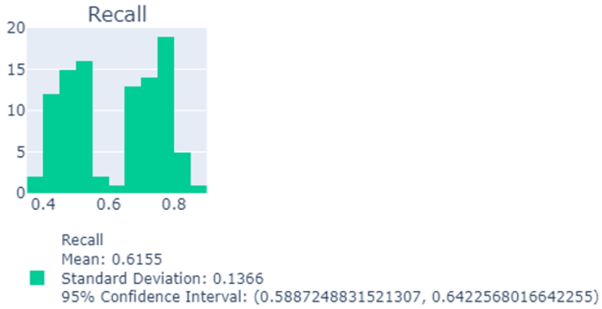
Logistic Regression Model

- Input: BERT embedding
- Runtime: 61.44 seconds



SVM Model

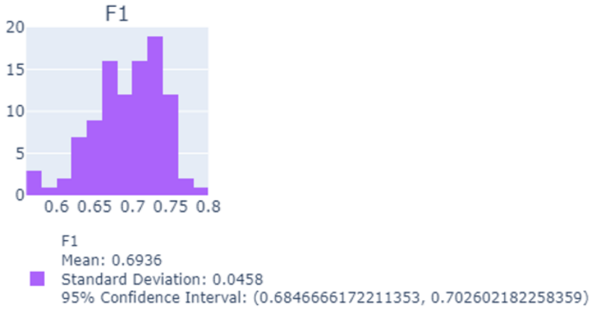
- Input: TF-IDF Vector
- Kernel: Linear
- Runtime: 2367.08 seconds $\approx$ 40 minutes



4.1.4 The Highest Mean of F1 Scores

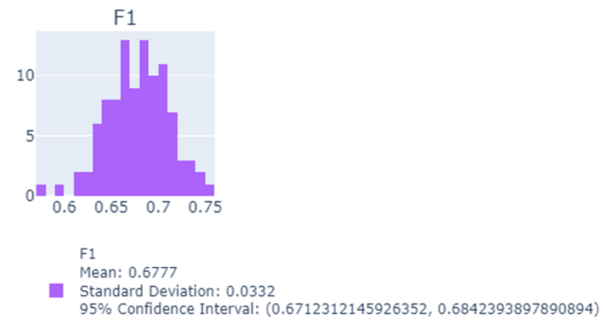
Logistic Regression Model

- Input: BERT embedding
- Runtime: 61.44 seconds



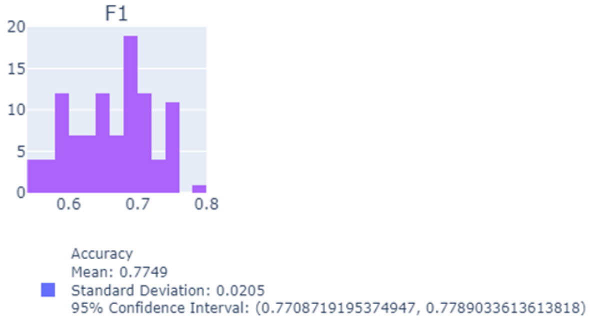
Complement Naive Bayes Model

- Input: Count Vector
- Runtime: 23.02 seconds



SVM Model

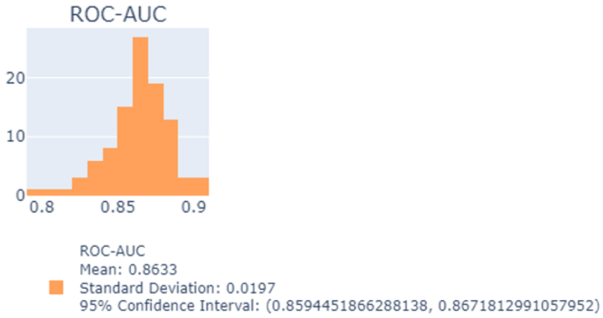
- Input: TF-IDF Vector
- Kernel: RBF
- Runtime: : 2747.83 seconds $\approx$ 45.8 minutes



4.1.5 The Highest Mean of ROC-AUC Scores

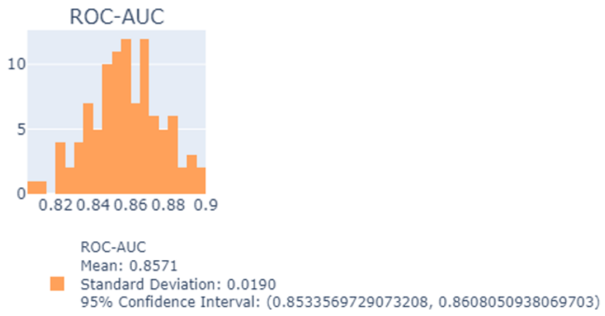
Logistic Regression Model

- Input: BERT embedding
- Runtime: 64.44 seconds



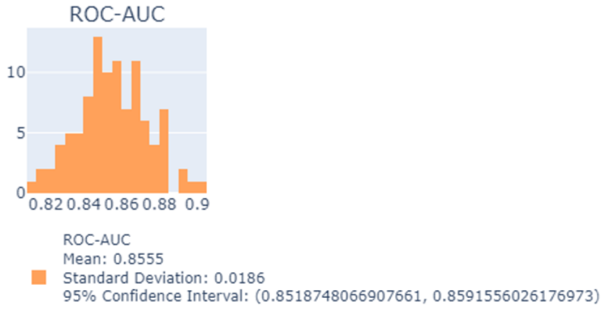
Logistic Regression Model

- Input: Count Vector
- Runtime: 45.16 seconds



Logistic Regression Model

- Input: Compressed Count Vector
- Runtime: 20 seconds



4.2 Analyzing Results of Text Classification on Untokenized Full Dialog Data

Similar calculations and model evaluations were executed on the untokenized full dialog data. However, due to its limited size (consisting of only 28 full texts), we chose not to place full trust in its evaluation scores, as we believed that creating an efficient model under these circumstances might be challenging. Nonetheless, the result plots can be found on the GitHub repository of the project if desired. The numeric score tableau is given in the Appendix section

Conclusion

Reaching the end of this project, we explored promising insights into the intersection of Artificial Intelligence and psychology, highlighting the potential of deriving meaningful innovations for human mental health studies through AI.

5.1 Difficulties

The cleaning process of text data served as an important part of the project, the caution towards potentially removing what-could be insightful information from the data, made this stage more difficult to manage. Despite the relatively small size of our dataset, the question of how to best clean data became a significant consideration. With larger datasets, manual inspections and manipulations might not be feasible, this brings attention to optimal approaches to data cleaning. Along the way, we realized that the answer to this question lay in refraining from making judgmental assumptions and treating information within the dataset as valuable until proven otherwise, a determination validated through calculated metrics.

Another challenge surfaced in the experimental nature of this project. Although this is an experimental project hoping to introduce many methodologies and compare them, the diversity of types of inputs and models made it more difficult to focus on direct techniques, giving us a comprehensive evaluation table, as presented in the appendix section. While this approach was helpful as a learning curve to NLP strategies, it was not straightforward in reaching a conclusive result.

5.2 Future Work

Even though our primary focus was centered on insights from participants' responses to questions, the metadata from the corpus contained many features, as detailed in section 2.1 (Data Collection). The environment of the experiment, the specific questions asked, the behaviour and comfort level of the participants and the mutual dialog between the participant and the interviewer could contribute greatly in predicting signs of schizophrenia.

References

[1] Oliver Delgaram-Nejad, Dawn Archer, Gerasimos Chatzidamianos, Alex Bartha, and Louise Robinson. Discussing abstract ideas in schizophrenia corpus, 2017-2023. https://reshare.ukdataservice.ac.uk/855021/, 2023. [Data Collection]. Colchester, Essex: UK Data Service. DOI: https://doi.org/10.5255/UKDA-SN-855021. Accessed on: 2024.

[2] Oliver Delgaram-Nejad, Dawn Archer, Gerasimos Chatzidamianos, Alex Bartha, and Louise Robinson. Discussing abstract ideas in schizophrenia corpus, 2017-2023, 2023. [Data Collection]. Colchester, Essex: UK Data Service, 2023. DOI: https://doi.org/10.5255/UKDA-SN-855021. Accessed on: 2023 (Reference from the README file).

[3] Peter Norvig. How to write a spelling corrector, 2007-2016. [Blog Post]. Retrieved from https://norvig.com/spell-correct.html. Accessed on: 2024.

[4] Thushan Ganegedara. *Natural Language Processing with TensorFlow: Teach language to machines using Python's deep learning library*. Packt Publishing Ltd, 2018.

[5] Mukesh Chandra Maurya. Tf-idf vectorizer - scikit learn, 2021. Accessed on: 2024.

[6] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Proc. Interspeech 2010*, pages 1045–1048, 2010.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, Illia Polosukhin, et al. Advances in neural information processing systems. *Attention is All you Need*, 2017.

[8] Mohammad Taher Pilehvar and Jose Camacho-Collados. *Embeddings in natural language processing: Theory and advances in vector representations of meaning*. Morgan & Claypool Publishers, 2020.

[9] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge, 2008.

[10] scikit-learn library 1.4. support vector machines. https://scikit-learn.org/stable/modules/svm.html. Accessed on: 2024.

[11] Ian T Jolliffe. *Principal component analysis for special types of data*. Springer, 2002.

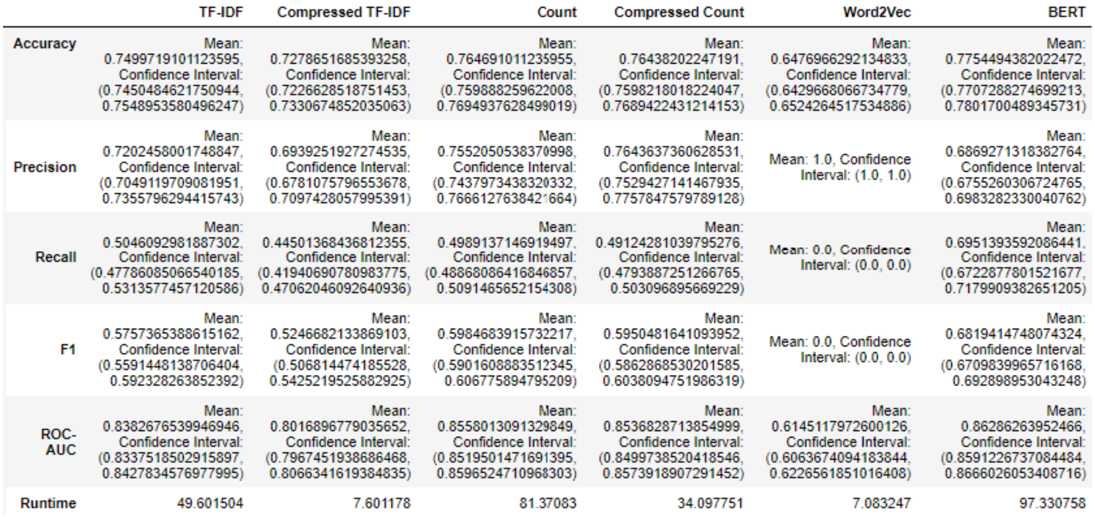
[12] Markus Ringnér. What is principal component analysis? *Nature biotechnology*, 26(3):303–304, 2008.

[13] Thomas J DiCiccio and Bradley Efron. Bootstrap confidence intervals. *Statistical science*, 11(3):189–228, 1996.

Appendix

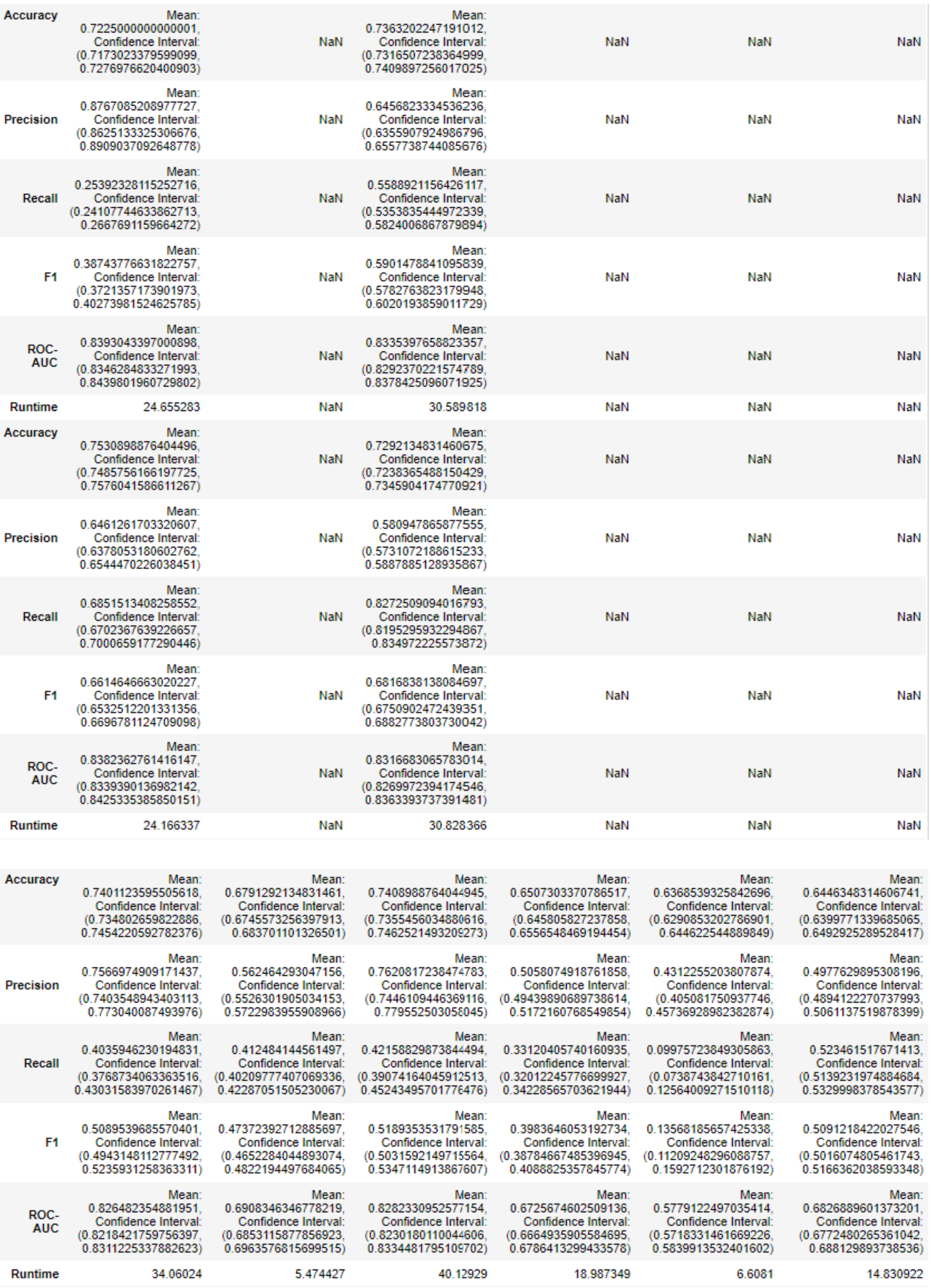
A.1 Evaluation Score Tables

Sentence-Level Tokenized Data Frame



	TF-IDF	Compressed TF-IDF	Count	Compressed Count	Word2Vec	BERT
Accuracy	Mean: 0.7499719101123595, Confidence Interval: (0.7450484621750944, 0.7548953580496247)	Mean: 0.7278651685393258, Confidence Interval: (0.7226628518751453, 0.7330674852035063)	Mean: 0.764691011235955, Confidence Interval: (0.759888259622008, 0.7694937628499019)	Mean: 0.76438202247191, Confidence Interval: (0.7598218018224047, 0.7689422431214153)	Mean: 0.6476966292134833, Confidence Interval: (0.6429668066734779, 0.6524264517534886)	Mean: 0.7754494382022472, Confidence Interval: (0.7707288274699213, 0.7801700489345731)
Precision	Mean: 0.7202458001748847, Confidence Interval: (0.7049119709081951, 0.7355796294415743)	Mean: 0.6939251927274535, Confidence Interval: (0.6781075796553678, 0.7097428057995391)	Mean: 0.7552050538370998, Confidence Interval: (0.7437973438320332, 0.7666127638421664)	Mean: 0.7643637360628531, Confidence Interval: (0.7529427141467935, 0.7757847579789128)	Mean: 1.0, Confidence Interval: (1.0, 1.0)	Mean: 0.6869271318382764, Confidence Interval: (0.6755260306724765, 0.6983282330040762)
Recall	Mean: 0.5046092981887302, Confidence Interval: (0.47786085066540185, 0.5313577457120586)	Mean: 0.44501368436812355, Confidence Interval: (0.41940690780983775, 0.47062046092640936)	Mean: 0.4989137146919497, Confidence Interval: (0.48868086416846857, 0.5091465652154308)	Mean: 0.49124281039795276, Confidence Interval: (0.4793887251266765, 0.503096895669229)	Mean: 0.0, Confidence Interval: (0.0, 0.0)	Mean: 0.6951393592086441, Confidence Interval: (0.6722877801521677, 0.7179909382651205)
F1	Mean: 0.5757365388615162, Confidence Interval: (0.5591448138706404, 0.592328263852392)	Mean: 0.5246682133869103, Confidence Interval: (0.506814474185528, 0.5425219525882925)	Mean: 0.5984683915732217, Confidence Interval: (0.5901608883512345, 0.606775894795209)	Mean: 0.5950481641093952, Confidence Interval: (0.5862868530201585, 0.6038094751986319)	Mean: 0.0, Confidence Interval: (0.0, 0.0)	Mean: 0.6819414748074324, Confidence Interval: (0.6709839965716168, 0.692898953043248)
ROC-AUC	Mean: 0.8382676539946946, Confidence Interval: (0.8337518502915897, 0.8427834576977995)	Mean: 0.8016896779035652, Confidence Interval: (0.7967451938686468, 0.8066341619384835)	Mean: 0.8558013091329849, Confidence Interval: (0.8519501471691395, 0.8596524710968303)	Mean: 0.8536828713854999, Confidence Interval: (0.8499738520418546, 0.8573918907291452)	Mean: 0.6145117972600126, Confidence Interval: (0.6063674094183844, 0.6226561851016408)	Mean: 0.86286263952466, Confidence Interval: (0.8591226737084484, 0.8666026053408716)
Runtime	49.601504	7.601178	81.37083	34.097751	7.083247	97.330758

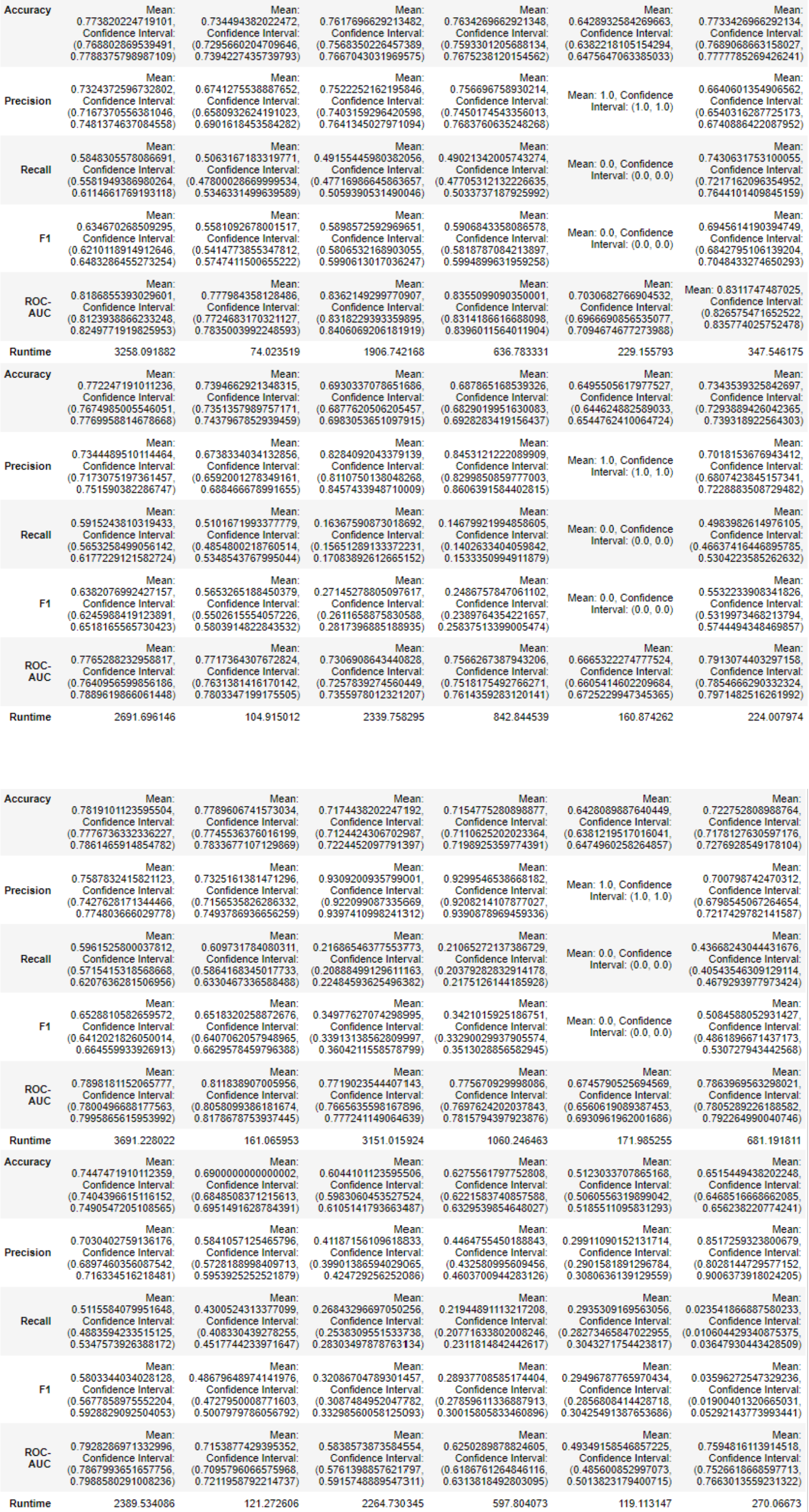
Figure A.1: Logistic Regression



Accuracy	Mean: 0.7225000000000001, Confidence Interval: (0.7173023379599099, 0.7276976620400903)	NaN	Mean: 0.7363202247191012, Confidence Interval: (0.7316507238364999, 0.7409897256017025)	NaN	NaN	NaN
Precision	Mean: 0.8767085208977727, Confidence Interval: (0.8625133325306676, 0.8909037092648778)	NaN	Mean: 0.6456823334536236, Confidence Interval: (0.6355907924986796, 0.6557738744085676)	NaN	NaN	NaN
Recall	Mean: 0.25392328115252716, Confidence Interval: (0.24107744633862713, 0.2667691159664272)	NaN	Mean: 0.558892115642611 7, Confidence Interval: (0.5353835444972339, 0.5824006867879894)	NaN	NaN	NaN
F1	Mean: 0.38743776631822757, Confidence Interval: (0.3721357173901973, 0.40273981524625785)	NaN	Mean: 0.5901478841095839, Confidence Interval: (0.5782763823179948, 0.6020193859011729)	NaN	NaN	NaN
ROC-AUC	Mean: 0.8393043397000898, Confidence Interval: (0.8346284833271993, 0.8439801960729802)	NaN	Mean: 0.8335397658823357, Confidence Interval: (0.8292370221574789, 0.8378425096071925)	NaN	NaN	NaN
Runtime	24.655283	NaN	30.589818	NaN	NaN	NaN
Accuracy	Mean: 0.7530898876404496, Confidence Interval: (0.7485756166197725, 0.7576041586611267)	NaN	Mean: 0.7292134831460675, Confidence Interval: (0.7238365488150429, 0.7345904174770921)	NaN	NaN	NaN
Precision	Mean: 0.6461261703320607, Confidence Interval: (0.6378053180602762, 0.6544470226038451)	NaN	Mean: 0.580947865877555, Confidence Interval: (0.5731072188615233, 0.5887885128935867)	NaN	NaN	NaN
Recall	Mean: 0.6851513408258552, Confidence Interval: (0.6702367639226657, 0.7000659177290446)	NaN	Mean: 0.8272509094016793, Confidence Interval: (0.8195295932294867, 0.834972225573872)	NaN	NaN	NaN
F1	Mean: 0.6614646663020227, Confidence Interval: (0.6532512201331356, 0.6696781124709098)	NaN	Mean: 0.6816838138084697, Confidence Interval: (0.6750902472439351, 0.6882773803730042)	NaN	NaN	NaN
ROC-AUC	Mean: 0.8382362761416147, Confidence Interval: (0.8339390136982142, 0.8425335385850151)	NaN	Mean: 0.8316683065783014, Confidence Interval: (0.8269972394174546, 0.8363393737391481)	NaN	NaN	NaN
Runtime	24.166337	NaN	30.828366	NaN	NaN	NaN

Accuracy	Mean: 0.7401123595505618, Confidence Interval: (0.734802659822886, 0.7454220592782376)	Mean: 0.6791292134831461, Confidence Interval: (0.6745573256397913, 0.683701101326501)	Mean: 0.7408988764044945, Confidence Interval: (0.7355456034880616, 0.7462521493209273)	Mean: 0.6507303370786517, Confidence Interval: (0.645805827237858, 0.6556548469194454)	Mean: 0.6368539325842696, Confidence Interval: (0.6290853202786901, 0.644622544889849)	Mean: 0.6446348314606741, Confidence Interval: (0.6399771339685065, 0.6492925289528417)
Precision	Mean: 0.7566974909171437, Confidence Interval: (0.7403548943403113, 0.773040087493976)	Mean: 0.562464293047156, Confidence Interval: (0.5526301905034153, 0.5722983955908966)	Mean: 0.7620817238474783, Confidence Interval: (0.7446109446369116, 0.779552503058045)	Mean: 0.5058074918761858, Confidence Interval: (0.4943989068973861 4, 0.5172160768549854)	Mean: 0.4312255203807874, Confidence Interval: (0.405081750937746, 0.4573692898238287 4)	Mean: 0.4977629895308196, Confidence Interval: (0.4894122270737993, 0.5061137519878399)
Recall	Mean: 0.4035946230194831, Confidence Interval: (0.3768734063363516, 0.43031583970261467)	Mean: 0.412484144561497, Confidence Interval: (0.40209777407069336, 0.42287051505230067)	Mean: 0.42158829873844494, Confidence Interval: (0.39074164045912513, 0.45243495701776476)	Mean: 0.33120405740160935, Confidence Interval: (0.32012245776699927, 0.34228565703621944)	Mean: 0.09975723849305863, Confidence Interval: (0.0738743842710161, 0.12564009271510118)	Mean: 0.523461517671413, Confidence Interval: (0.5139231974884684, 0.5329998378543577)
F1	Mean: 0.5089539685570401, Confidence Interval: (0.4943148112777492, 0.5235931258363311)	Mean: 0.47372392712885697, Confidence Interval: (0.4652284044893074, 0.4822194497684065)	Mean: 0.5189353531791585, Confidence Interval: (0.5031592149715564, 0.5347114913867607)	Mean: 0.3983646053192734, Confidence Interval: (0.38784667485396945, 0.4088825357845774)	Mean: 0.13568185657425338, Confidence Interval: (0.11209248296088757, 0.1592712301876192)	Mean: 0.5091218422027546, Confidence Interval: (0.5016074805461743, 0.5166362038593348)
ROC-AUC	Mean: 0.826482354881951, Confidence Interval: (0.8218421759756397, 0.8311225337882623)	Mean: 0.6908346346778219, Confidence Interval: (0.6853115877856923, 0.6963576815699515)	Mean: 0.8282330952577154, Confidence Interval: (0.8230180110044606, 0.8334481795109702)	Mean: 0.6725674602509136, Confidence Interval: (0.6664935905584695, 0.6786413299433578)	Mean: 0.5779122497035414, Confidence Interval: (0.5718331461669226, 0.5839913532401602)	Mean: 0.6826889601373201, Confidence Interval: (0.6772480265361042, 0.688129893738536)
Runtime	34.06024	5.474427	40.12929	18.987349	6.6081	14.830922

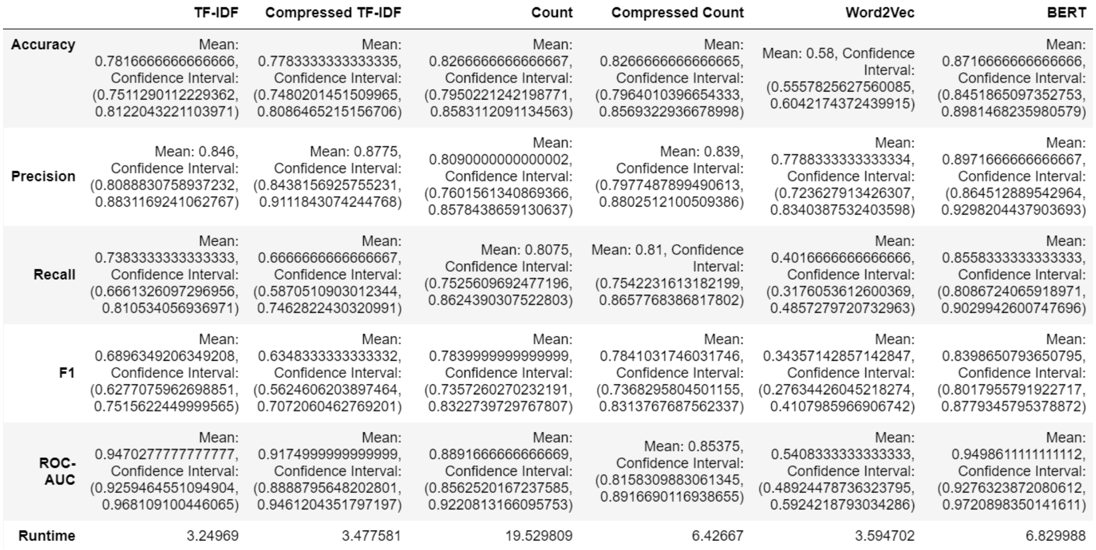
Figure A.2: Naive Bayes Multinomial/Complement/Bernoulli



Accuracy	Mean: 0.773820224719101, Confidence Interval: (0.768802869539491, 0.7788375798987109)	Mean: 0.734494382022472, Confidence Interval: (0.7295660204709646, 0.7394227435739793)	Mean: 0.7617696629213482, Confidence Interval: (0.7568350226457389, 0.7667043031969575)	Mean: 0.7634269662921348, Confidence Interval: (0.7593301205688134, 0.7675238120154562)	Mean: 0.6428932584269663, Confidence Interval: (0.6382218105154294, 0.6475647063385033)	Mean: 0.7733426966292134, Confidence Interval: (0.7689068663158027, 0.7777785269426241)
Precision	Mean: 0.7324372596732802, Confidence Interval: (0.7167370556381046, 0.7481374637084558)	Mean: 0.6741275538887652, Confidence Interval: (0.6580932624191023, 0.6901618453584282)	Mean: 0.7522252162195846, Confidence Interval: (0.7403159296420598, 0.7641345027971094)	Mean: 0.756696758930214, Confidence Interval: (0.7450174543356013, 0.7683760635248268)	Mean: 1.0, Confidence Interval: (1.0, 1.0)	Mean: 0.6640601354906562, Confidence Interval: (0.6540316287725173, 0.6740886422087952)
Recall	Mean: 0.5848305578086691, Confidence Interval: (0.5581949386980264, 0.6114661769193118)	Mean: 0.5063167183319771, Confidence Interval: (0.47800028669999534, 0.5346331499639589)	Mean: 0.49155445980382056, Confidence Interval: (0.47716986645863657, 0.5059390531490046)	Mean: 0.49021342005743274, Confidence Interval: (0.47705312132226635, 0.5033737187925992)	Mean: 0.0, Confidence Interval: (0.0, 0.0)	Mean: 0.7430631753100055, Confidence Interval: (0.7217162096354952, 0.7644101409845159)
F1	Mean: 0.634670268509295, Confidence Interval: (0.6210118914912646, 0.6483286455273254)	Mean: 0.5581092678001517, Confidence Interval: (0.5414773855347812, 0.5747411500655222)	Mean: 0.5898572592969651, Confidence Interval: (0.5806532168903055, 0.5990613017036247)	Mean: 0.5906843358086578, Confidence Interval: (0.5818787084213897, 0.5994899631959258)	Mean: 0.0, Confidence Interval: (0.0, 0.0)	Mean: 0.6945614190394749, Confidence Interval: (0.6842795106139204, 0.7048433274650293)
ROC-AUC	Mean: 0.8186855393029601, Confidence Interval: (0.8123938866233248, 0.8249771919825953)	Mean: 0.777984358128486, Confidence Interval: (0.7724683170321127, 0.7835003992248593)	Mean: 0.8362149299770907, Confidence Interval: (0.8318229393359895, 0.8406069206181919)	Mean: 0.8355099090350001, Confidence Interval: (0.8314186616688098, 0.8396011564011904)	Mean: 0.7030682766904532, Confidence Interval: (0.6966690856535077, 0.7094674677273988)	Mean: 0.8311747487025, Confidence Interval: (0.826575471652522, 0.835774025752478)
Runtime	3258.091882	74.023519	1906.742168	636.783331	229.155793	347.546175
Accuracy	Mean: 0.772247191011236, Confidence Interval: (0.7674985005546051, 0.7769958814678668)	Mean: 0.7394662921348315, Confidence Interval: (0.7351357989757171, 0.7437967852939459)	Mean: 0.6930337078651686, Confidence Interval: (0.6877620506205457, 0.6983053651097915)	Mean: 0.687865168539326, Confidence Interval: (0.6829019951630083, 0.6928283419156437)	Mean: 0.6495505617977527, Confidence Interval: (0.644624882589033, 0.6544762410064724)	Mean: 0.7343539325842697, Confidence Interval: (0.7293889426042365, 0.739318922564303)
Precision	Mean: 0.7344489510114464, Confidence Interval: (0.7173075197361457, 0.751590382286747)	Mean: 0.6738334034132856, Confidence Interval: (0.6592001278349161, 0.688466678991655)	Mean: 0.8284092043379139, Confidence Interval: (0.8110750138048268, 0.8457433948710009)	Mean: 0.8453121222089909, Confidence Interval: (0.8299850859777003, 0.8606391584402815)	Mean: 1.0, Confidence Interval: (1.0, 1.0)	Mean: 0.7018153676943412, Confidence Interval: (0.6807423845157341, 0.7228883508729482)
Recall	Mean: 0.5915243810319433, Confidence Interval: (0.5653258499056142, 0.6177229121582724)	Mean: 0.5101671993377779, Confidence Interval: (0.4854800218760514, 0.5348543767995044)	Mean: 0.16367590873018692, Confidence Interval: (0.15651289133372231, 0.17083892612665152)	Mean: 0.14679921994858605, Confidence Interval: (0.1402633404059842, 0.1533350994911879)	Mean: 0.0, Confidence Interval: (0.0, 0.0)	Mean: 0.4983982614976105, Confidence Interval: (0.46637416446895785, 0.5304223585262632)
F1	Mean: 0.6382076992427157, Confidence Interval: (0.6245988419123891, 0.6518165565730423)	Mean: 0.5653265188450379, Confidence Interval: (0.5502615554057226, 0.5803914822843532)	Mean: 0.27145278805097617, Confidence Interval: (0.2611658875830588, 0.2817396885188935)	Mean: 0.2486757847061102, Confidence Interval: (0.2389764354221657, 0.25837513399005474)	Mean: 0.0, Confidence Interval: (0.0, 0.0)	Mean: 0.5532233908341826, Confidence Interval: (0.5319973468213794, 0.5744494348469857)
ROC-AUC	Mean: 0.7765288232958817, Confidence Interval: (0.7640956599856186, 0.7889619866061448)	Mean: 0.7717364307672824, Confidence Interval: (0.7631381416170142, 0.7803347199175505)	Mean: 0.7306908643440828, Confidence Interval: (0.7257839274560449, 0.7355978012321207)	Mean: 0.7566267387943206, Confidence Interval: (0.7518175492766271, 0.7614359283120141)	Mean: 0.6665322274777524, Confidence Interval: (0.6605414602209684, 0.6725229947345365)	Mean: 0.7913074403297158, Confidence Interval: (0.7854666290332324, 0.7971482516261992)
Runtime	2691.696146	104.915012	2339.758295	842.844539	160.874262	224.007974
Accuracy	Mean: 0.7819101123595504, Confidence Interval: (0.7776736332336227, 0.7861465914854782)	Mean: 0.7789606741573034, Confidence Interval: (0.7745536376016199, 0.7833677107129869)	Mean: 0.7174438202247192, Confidence Interval: (0.7124424306702987, 0.7224452097791397)	Mean: 0.7154775280898877, Confidence Interval: (0.7110625202023364, 0.7198925359774391)	Mean: 0.6428089887640449, Confidence Interval: (0.6381219517016041, 0.6474960258264857)	Mean: 0.722752808988764, Confidence Interval: (0.7178127630597176, 0.7276928549178104)
Precision	Mean: 0.7587832415821123, Confidence Interval: (0.7427628171344466, 0.774803666029778)	Mean: 0.7325161381471296, Confidence Interval: (0.7156535826286332, 0.7493786936656259)	Mean: 0.9309200935799001, Confidence Interval: (0.922099087335669, 0.9397410998241312)	Mean: 0.9299546538668182, Confidence Interval: (0.9208214107877027, 0.9390878969459336)	Mean: 1.0, Confidence Interval: (1.0, 1.0)	Mean: 0.700798742470312, Confidence Interval: (0.6798545067264654, 0.7217429782141587)
Recall	Mean: 0.5961525800037812, Confidence Interval: (0.5715415318568668, 0.6207636281506956)	Mean: 0.609731784080311, Confidence Interval: (0.5864168345017733, 0.6330467336588488)	Mean: 0.21686546377553773, Confidence Interval: (0.20888499129611163, 0.22484593625496382)	Mean: 0.21065272137386729, Confidence Interval: (0.20379282832914178, 0.2175126144185928)	Mean: 0.0, Confidence Interval: (0.0, 0.0)	Mean: 0.43668243044431676, Confidence Interval: (0.40543546309129114, 0.4679293977973424)
F1	Mean: 0.6528810582659572, Confidence Interval: (0.6412021826050014, 0.664559933926913)	Mean: 0.6518320258872676, Confidence Interval: (0.6407062057948965, 0.6629578459796388)	Mean: 0.34977627074298995, Confidence Interval: (0.33913138562809997, 0.3604211558578799)	Mean: 0.3421015925186751, Confidence Interval: (0.33290029937905574, 0.3513028856582945)	Mean: 0.0, Confidence Interval: (0.0, 0.0)	Mean: 0.5084588052931427, Confidence Interval: (0.4861896671437173, 0.530727943442568)
ROC-AUC	Mean: 0.7898181152065777, Confidence Interval: (0.7800496688177563, 0.7995865615953992)	Mean: 0.811838907005956, Confidence Interval: (0.8058099386181674, 0.8178678753937445)	Mean: 0.7719023544407143, Confidence Interval: (0.7665635598167896, 0.777241149064639)	Mean: 0.775670929998086, Confidence Interval: (0.7697624202037843, 0.7815794397923876)	Mean: 0.6745790525694569, Confidence Interval: (0.6560619089387453, 0.6930961962001686)	Mean: 0.7863969563298021, Confidence Interval: (0.7805289226188582, 0.792264990040746)
Runtime	3691.228022	161.065953	3151.015924	1060.246463	171.985255	681.191811
Accuracy	Mean: 0.7447471910112359, Confidence Interval: (0.7404396615116152, 0.7490547205108565)	Mean: 0.6900000000000002, Confidence Interval: (0.6848508371215613, 0.6951491628784391)	Mean: 0.6044101123595506, Confidence Interval: (0.5983060453527524, 0.6105141793663487)	Mean: 0.6275561797752808, Confidence Interval: (0.6221583740857588, 0.6329539854648027)	Mean: 0.5123033707865168, Confidence Interval: (0.5060556319899042, 0.5185511095831293)	Mean: 0.6515449438202248, Confidence Interval: (0.6468516668662085, 0.656238220774241)
Precision	Mean: 0.7030402759136176, Confidence Interval: (0.6897460356087542, 0.716334516218481)	Mean: 0.5841057125465796, Confidence Interval: (0.5728188998409713, 0.5953925252521879)	Mean: 0.41187156109618833, Confidence Interval: (0.39901386594029065, 0.424729256252086)	Mean: 0.4464755450188843, Confidence Interval: (0.432580995609456, 0.4603700944283126)	Mean: 0.29911090152131714, Confidence Interval: (0.2901581891296784, 0.3080636139129559)	Mean: 0.8517259323800679, Confidence Interval: (0.8028144729577152, 0.9006373918024205)
Recall	Mean: 0.5115584079951648, Confidence Interval: (0.4883594233515125, 0.5347573926388172)	Mean: 0.4300524313377099, Confidence Interval: (0.408330439278255, 0.4517744233971647)	Mean: 0.26843296697050256, Confidence Interval: (0.2538309551533738, 0.28303497878763134)	Mean: 0.21944891113217208, Confidence Interval: (0.20771633802008246, 0.2311814842442617)	Mean: 0.2935309169563056, Confidence Interval: (0.28273465847022955, 0.3043271754423817)	Mean: 0.023541866887580233, Confidence Interval: (0.010604429340875375, 0.03647930443428509)
F1	Mean: 0.5803344034028128, Confidence Interval: (0.5677858975552204, 0.5928829092504053)	Mean: 0.48679648974141976, Confidence Interval: (0.4727950008771603, 0.5007979786056792)	Mean: 0.32086704789301457, Confidence Interval: (0.3087484952047782, 0.33298560058125093)	Mean: 0.28937708585174404, Confidence Interval: (0.27859611336887913, 0.30015805833460896)	Mean: 0.29496787765970434, Confidence Interval: (0.2856808414428718, 0.30425491387653686)	Mean: 0.03596272547329236, Confidence Interval: (0.01900401320665031, 0.05292143773993441)
ROC-AUC	Mean: 0.7928286971332996, Confidence Interval: (0.7867993651657756, 0.7988580291008236)	Mean: 0.7153877429395352, Confidence Interval: (0.7095796066575968, 0.7211958792214737)	Mean: 0.5838573873584554, Confidence Interval: (0.5761398857621797, 0.5915748889547311)	Mean: 0.6250289878824605, Confidence Interval: (0.6186761264846116, 0.6313818492803095)	Mean: 0.49349158546857225, Confidence Interval: (0.485600852997073, 0.5013823179400715)	Mean: 0.7594816113914518, Confidence Interval: (0.7526618668597713, 0.7663013559231322)
Runtime	2389.534086	121.272606	2264.730345	597.804073	119.113147	270.06673

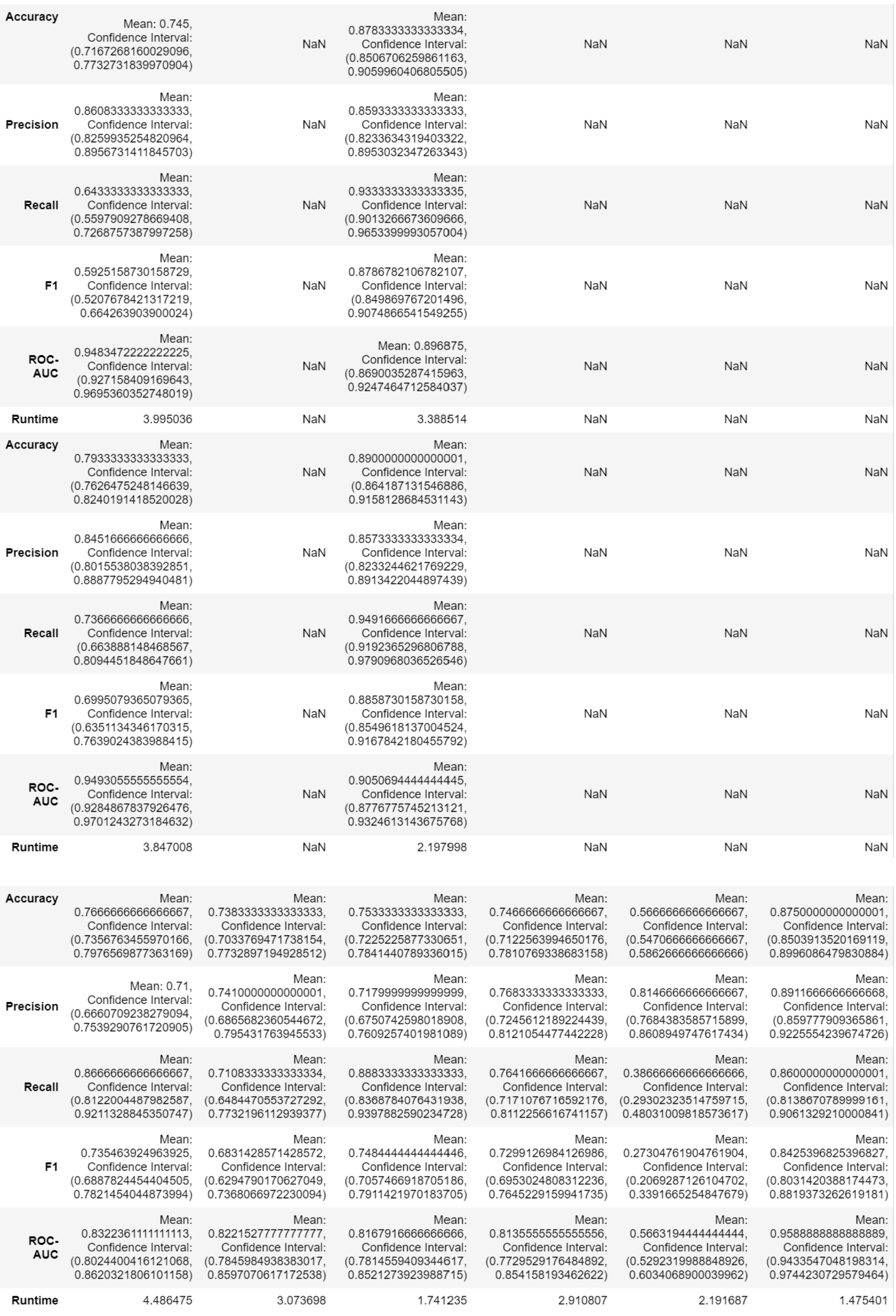
Figure A.3: SVM Linear/Poly/Rbf/Sigmoid

Untokenized Data Frame



	TF-IDF	Compressed TF-IDF	Count	Compressed Count	Word2Vec	BERT
Accuracy	Mean: 0.7816666666666666, Confidence Interval: (0.7511290112229362, 0.8122043221103971)	Mean: 0.7783333333333335, Confidence Interval: (0.7480201451509965, 0.8086465215156706)	Mean: 0.8266666666666667, Confidence Interval: (0.7950221242198771, 0.8583112091134563)	Mean: 0.8266666666666665, Confidence Interval: (0.7964010396654333, 0.8569322936678998)	Mean: 0.58, Confidence Interval: (0.5557825627560085, 0.6042174372439915)	Mean: 0.8716666666666666, Confidence Interval: (0.8451865097352753, 0.8981468235980579)
Precision	Mean: 0.846, Confidence Interval: (0.8088830758937232, 0.8831169241062767)	Mean: 0.8775, Confidence Interval: (0.8438156925755231, 0.9111843074244768)	Mean: 0.8090000000000002, Confidence Interval: (0.7601561340869366, 0.8578438659130637)	Mean: 0.839, Confidence Interval: (0.7977487899490613, 0.8802512100509386)	Mean: 0.7788333333333334, Confidence Interval: (0.723627913426307, 0.8340387532403598)	Mean: 0.8971666666666667, Confidence Interval: (0.864512889542964, 0.9298204437903693)
Recall	Mean: 0.7383333333333333, Confidence Interval: (0.6661326097296956, 0.810534056936971)	Mean: 0.6666666666666667, Confidence Interval: (0.5870510903012344, 0.7462822430320991)	Mean: 0.8075, Confidence Interval: (0.7525609692477196, 0.8624390307522803)	Mean: 0.81, Confidence Interval: (0.7542231613182199, 0.8657768386817802)	Mean: 0.4016666666666666, Confidence Interval: (0.3176053612600369, 0.4857279720732963)	Mean: 0.8558333333333333, Confidence Interval: (0.8086724065918971, 0.9029942600747696)
F1	Mean: 0.6896349206349208, Confidence Interval: (0.6277075962698851, 0.7515622449999565)	Mean: 0.6348333333333332, Confidence Interval: (0.5624606203897464, 0.7072060462769201)	Mean: 0.7839999999999999, Confidence Interval: (0.7357260270232191, 0.8322739729767807)	Mean: 0.7841031746031746, Confidence Interval: (0.7368295804501155, 0.8313767687562337)	Mean: 0.34357142857142847, Confidence Interval: (0.27634426045218274, 0.4107985966906742)	Mean: 0.8398650793650795, Confidence Interval: (0.8017955791922717, 0.8779345795378872)
ROC-AUC	Mean: 0.9470277777777777, Confidence Interval: (0.9259464551094904, 0.968109100446065)	Mean: 0.9174999999999999, Confidence Interval: (0.8888795648202801, 0.9461204351797197)	Mean: 0.8891666666666669, Confidence Interval: (0.8562520167237585, 0.9220813166095753)	Mean: 0.85375, Confidence Interval: (0.8158309883061345, 0.8916690116938655)	Mean: 0.5408333333333333, Confidence Interval: (0.48924478736323795, 0.5924218793034286)	Mean: 0.9498611111111112, Confidence Interval: (0.9276323872080612, 0.9720898350141611)
Runtime	3.24969	3.477581	19.529809	6.42667	3.594702	6.829988

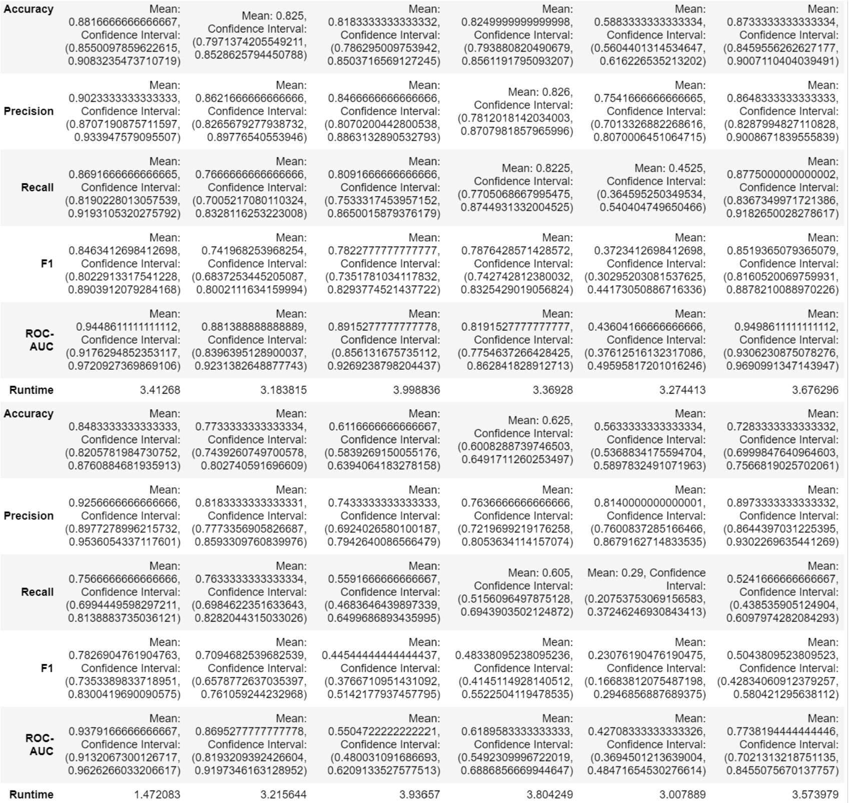
Figure A.4: Logistic Regression



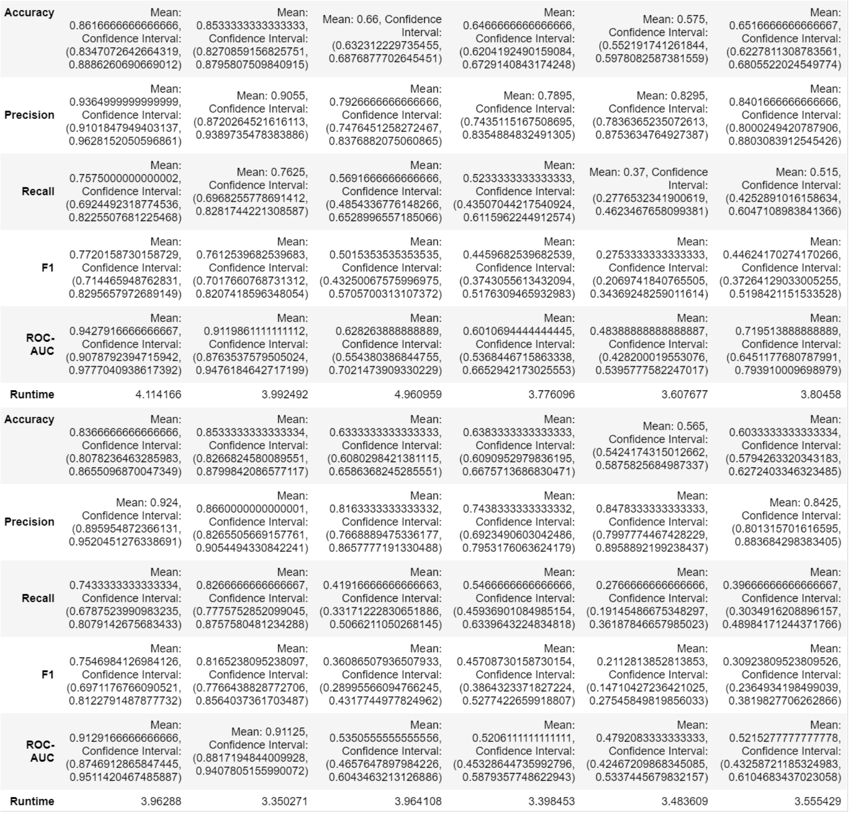
Accuracy	Mean: 0.745, Confidence Interval: (0.7167268160029096, 0.7732731839970904)	NaN	Mean: 0.8783333333333334, Confidence Interval: (0.8506706259861163, 0.9059960406805505)	NaN	NaN	NaN
Precision	Mean: 0.8608333333333333, Confidence Interval: (0.8259935254820964, 0.8956731411845703)	NaN	Mean: 0.8593333333333333, Confidence Interval: (0.8233634319403322, 0.8953032347263343)	NaN	NaN	NaN
Recall	Mean: 0.6433333333333333, Confidence Interval: (0.5597909278669408, 0.7268757387997258)	NaN	Mean: 0.9333333333333335, Confidence Interval: (0.9013266673609666, 0.9653399993057004)	NaN	NaN	NaN
F1	Mean: 0.5925158730158729, Confidence Interval: (0.5207678421317219, 0.664263903900024)	NaN	Mean: 0.8786782106782107, Confidence Interval: (0.849869767201496, 0.9074866541549255)	NaN	NaN	NaN
ROC-AUC	Mean: 0.9483472222222225, Confidence Interval: (0.927158409169643, 0.9695360352748019)	NaN	Mean: 0.896875, Confidence Interval: (0.8690035287415963, 0.9247464712584037)	NaN	NaN	NaN
Runtime	3.995036	NaN	3.388514	NaN	NaN	NaN
Accuracy	Mean: 0.7933333333333333, Confidence Interval: (0.7626475248146639, 0.8240191418520028)	NaN	Mean: 0.8900000000000001, Confidence Interval: (0.864187131546886, 0.9158128684531143)	NaN	NaN	NaN
Precision	Mean: 0.8451666666666666, Confidence Interval: (0.8015538038392851, 0.8887795294940481)	NaN	Mean: 0.8573333333333334, Confidence Interval: (0.8233244621769229, 0.8913422044897439)	NaN	NaN	NaN
Recall	Mean: 0.7366666666666666, Confidence Interval: (0.663888148468567, 0.8094451848647661)	NaN	Mean: 0.9491666666666667, Confidence Interval: (0.9192365296806788, 0.9790968036526546)	NaN	NaN	NaN
F1	Mean: 0.6995079365079365, Confidence Interval: (0.6351134346170315, 0.7639024383988415)	NaN	Mean: 0.8858730158730158, Confidence Interval: (0.8549618137004524, 0.9167842180455792)	NaN	NaN	NaN
ROC-AUC	Mean: 0.9493055555555554, Confidence Interval: (0.9284867837926476, 0.9701243273184632)	NaN	Mean: 0.9050694444444445, Confidence Interval: (0.8776775745213121, 0.9324613143675768)	NaN	NaN	NaN
Runtime	3.847008	NaN	2.197998	NaN	NaN	NaN

Accuracy	Mean: 0.7666666666666667, Confidence Interval: (0.7356763455970166, 0.7976569877363169)	Mean: 0.7383333333333333, Confidence Interval: (0.7033769471738154, 0.7732897194928512)	Mean: 0.7533333333333333, Confidence Interval: (0.7225225877330651, 0.7841440789336015)	Mean: 0.7466666666666667, Confidence Interval: (0.7122563994650176, 0.7810769338683158)	Mean: 0.5666666666666667, Confidence Interval: (0.5470666666666667, 0.5862666666666666)	Mean: 0.8750000000000001, Confidence Interval: (0.8503913520169119, 0.8996086479830884)
Precision	Mean: 0.71, Confidence Interval: (0.6660709238279094, 0.7539290761720905)	Mean: 0.7410000000000001, Confidence Interval: (0.6865682360544672, 0.795431763945533)	Mean: 0.7179999999999999, Confidence Interval: (0.6750742598018908, 0.7609257401981089)	Mean: 0.7683333333333333, Confidence Interval: (0.7245612189224439, 0.8121054477442228)	Mean: 0.8146666666666667, Confidence Interval: (0.7684383585715899, 0.8608949747617434)	Mean: 0.8911666666666668, Confidence Interval: (0.859777909365861, 0.9225554239674726)
Recall	Mean: 0.8666666666666667, Confidence Interval: (0.8122004487982587, 0.9211328845350747)	Mean: 0.7108333333333334, Confidence Interval: (0.6484470553727292, 0.7732196112939377)	Mean: 0.8883333333333333, Confidence Interval: (0.8368784076431938, 0.9397882590234728)	Mean: 0.7641666666666667, Confidence Interval: (0.7171076716592176, 0.8112256616741157)	Mean: 0.3866666666666666, Confidence Interval: (0.29302323514759715, 0.48031009818573617)	Mean: 0.8600000000000001, Confidence Interval: (0.8138670789999161, 0.9061329210000841)
F1	Mean: 0.735463924963925, Confidence Interval: (0.6887824454404505, 0.7821454044873994)	Mean: 0.6831428571428572, Confidence Interval: (0.6294790170627049, 0.7368066972230094)	Mean: 0.7484444444444446, Confidence Interval: (0.7057466918705186, 0.7911421970183705)	Mean: 0.7299126984126986, Confidence Interval: (0.6953024808312236, 0.7645229159941735)	Mean: 0.27304761904761904, Confidence Interval: (0.2069287126104702, 0.3391665254847679)	Mean: 0.8425396825396827, Confidence Interval: (0.8031420388174473, 0.8819373262619181)
ROC-AUC	Mean: 0.8322361111111113, Confidence Interval: (0.8024400416121068, 0.8620321806101158)	Mean: 0.8221527777777777, Confidence Interval: (0.7845984938383017, 0.8597070617172538)	Mean: 0.8167916666666666, Confidence Interval: (0.7814559409344617, 0.8521273923988715)	Mean: 0.8135555555555556, Confidence Interval: (0.7729529176484892, 0.854158193462622)	Mean: 0.5663194444444444, Confidence Interval: (0.5292319988848926, 0.6034068900039962)	Mean: 0.9588888888888889, Confidence Interval: (0.9433547048198314, 0.9744230729579464)
Runtime	4.486475	3.073698	1.741235	2.910807	2.191687	1.475401

Figure A.5: Naive Bayes Multinomial/Complement/Bernoulli



Accuracy	Mean: 0.8816666666666667, Confidence Interval: (0.8550097859622615, 0.9083235473710719)	Mean: 0.825, Confidence Interval: (0.7971374205549211, 0.8528625794450788)	Mean: 0.8183333333333332, Confidence Interval: (0.786295009753942, 0.8503716569127245)	Mean: 0.8249999999999998, Confidence Interval: (0.793880820490679, 0.8561191795093207)	Mean: 0.5883333333333334, Confidence Interval: (0.5604401314534647, 0.616226535213202)	Mean: 0.8733333333333334, Confidence Interval: (0.8459556262627177, 0.9007110404039491)
Precision	Mean: 0.9023333333333333, Confidence Interval: (0.8707190875711597, 0.933947579095507)	Mean: 0.8621666666666666, Confidence Interval: (0.8265679277938732, 0.89776540553946)	Mean: 0.8466666666666666, Confidence Interval: (0.8070200442800538, 0.8863132890532793)	Mean: 0.826, Confidence Interval: (0.7812018142034003, 0.8707981857965996)	Mean: 0.7541666666666665, Confidence Interval: (0.7013326882268616, 0.8070006451064715)	Mean: 0.8648333333333333, Confidence Interval: (0.8287994827110828, 0.9008671839555839)
Recall	Mean: 0.8691666666666665, Confidence Interval: (0.8190228013057539, 0.9193105320275792)	Mean: 0.7666666666666666, Confidence Interval: (0.7005217080110324, 0.8328116253223008)	Mean: 0.8091666666666666, Confidence Interval: (0.7533317453957152, 0.8650015879376179)	Mean: 0.8225, Confidence Interval: (0.7705068667995475, 0.8744931332004525)	Mean: 0.4525, Confidence Interval: (0.364595250349534, 0.540404749650466)	Mean: 0.8775000000000002, Confidence Interval: (0.8367349971721386, 0.9182650028278617)
F1	Mean: 0.8463412698412698, Confidence Interval: (0.8022913317541228, 0.8903912079284168)	Mean: 0.741968253968254, Confidence Interval: (0.6837253445205087, 0.8002111634159994)	Mean: 0.7822777777777777, Confidence Interval: (0.7351781034117832, 0.8293774521437722)	Mean: 0.7876428571428572, Confidence Interval: (0.742742812380032, 0.8325429019056824)	Mean: 0.3723412698412698, Confidence Interval: (0.30295203081537625, 0.44173050886716336)	Mean: 0.8519365079365079, Confidence Interval: (0.8160520069759931, 0.8878210088970226)
ROC-AUC	Mean: 0.9448611111111112, Confidence Interval: (0.9176294852353117, 0.9720927369869106)	Mean: 0.881388888888889, Confidence Interval: (0.8396395128900037, 0.9231382648877743)	Mean: 0.8915277777777778, Confidence Interval: (0.856131675735112, 0.9269238798204437)	Mean: 0.8191527777777777, Confidence Interval: (0.7754637266428425, 0.862841828912713)	Mean: 0.4360416666666666, Confidence Interval: (0.37612516132317086, 0.49595817201016246)	Mean: 0.9498611111111112, Confidence Interval: (0.9306230875078276, 0.9690991347143947)
Runtime	3.41268	3.183815	3.998836	3.36928	3.274413	3.676296
Accuracy	Mean: 0.8483333333333333, Confidence Interval: (0.8205781984730752, 0.8760884681935913)	Mean: 0.7733333333333334, Confidence Interval: (0.7439260749700578, 0.802740591696609)	Mean: 0.6116666666666667, Confidence Interval: (0.5839269150055176, 0.6394064183278158)	Mean: 0.625, Confidence Interval: (0.6008288739746503, 0.6491711260253497)	Mean: 0.5633333333333334, Confidence Interval: (0.5368834175594704, 0.5897832491071963)	Mean: 0.7283333333333332, Confidence Interval: (0.6999847640964603, 0.7566819025702061)
Precision	Mean: 0.9256666666666666, Confidence Interval: (0.8977278996215732, 0.9536054337117601)	Mean: 0.8183333333333331, Confidence Interval: (0.7773356905826687, 0.8593309760839976)	Mean: 0.7433333333333333, Confidence Interval: (0.6924026580100187, 0.7942640086566479)	Mean: 0.7636666666666666, Confidence Interval: (0.7219699219176258, 0.8053634114157074)	Mean: 0.8140000000000001, Confidence Interval: (0.7600837285166466, 0.8679162714833535)	Mean: 0.8973333333333332, Confidence Interval: (0.8644397031225395, 0.9302269635441269)
Recall	Mean: 0.7566666666666666, Confidence Interval: (0.6994449598297211, 0.8138883735036121)	Mean: 0.7633333333333334, Confidence Interval: (0.6984622351633643, 0.8282044315033026)	Mean: 0.5591666666666667, Confidence Interval: (0.4683646439897339, 0.6499686893435995)	Mean: 0.605, Confidence Interval: (0.5156096497875128, 0.6943903502124872)	Mean: 0.29, Confidence Interval: (0.20753753069156583, 0.37246246930843413)	Mean: 0.5241666666666667, Confidence Interval: (0.438535905124904, 0.6097974282084293)
F1	Mean: 0.7826904761904763, Confidence Interval: (0.7353389833718951, 0.8300419690090575)	Mean: 0.7094682539682539, Confidence Interval: (0.6578772637035397, 0.761059244232968)	Mean: 0.44544444444444437, Confidence Interval: (0.3766710951431092, 0.5142177937457795)	Mean: 0.48338095238095236, Confidence Interval: (0.4145114928140512, 0.5522504119478535)	Mean: 0.23076190476190475, Confidence Interval: (0.16683812075487198, 0.2946856887689375)	Mean: 0.5043809523809523, Confidence Interval: (0.42834060912379257, 0.580421295638112)
ROC-AUC	Mean: 0.9379166666666667, Confidence Interval: (0.9132067300126717, 0.9626266033206617)	Mean: 0.8695277777777778, Confidence Interval: (0.8193209392426604, 0.9197346163128952)	Mean: 0.5504722222222221, Confidence Interval: (0.480031091686693, 0.6209133527577513)	Mean: 0.6189583333333333, Confidence Interval: (0.5492309996722019, 0.6886856669944647)	Mean: 0.42708333333333326, Confidence Interval: (0.3694501213639004, 0.48471654530276614)	Mean: 0.7738194444444446, Confidence Interval: (0.7021313218751135, 0.8455075670137757)
Runtime	1.472083	3.215644	3.93657	3.804249	3.007889	3.573979



Accuracy	Mean: 0.8616666666666666, Confidence Interval: (0.8347072642664319, 0.8886260690669012)	Mean: 0.8533333333333333, Confidence Interval: (0.8270859156825751, 0.8795807509840915)	Mean: 0.66, Confidence Interval: (0.632312229735455, 0.6876877702645451)	Mean: 0.6466666666666666, Confidence Interval: (0.6204192490159084, 0.6729140843174248)	Mean: 0.575, Confidence Interval: (0.552191741261844, 0.5978082587381559)	Mean: 0.6516666666666667, Confidence Interval: (0.6227811308783561, 0.6805522024549774)
Precision	Mean: 0.9364999999999999, Confidence Interval: (0.9101847949403137, 0.9628152050596861)	Mean: 0.9055, Confidence Interval: (0.8720264521616113, 0.9389735478383886)	Mean: 0.7926666666666666, Confidence Interval: (0.7476451258272467, 0.8376882075060865)	Mean: 0.7895, Confidence Interval: (0.7435115167508695, 0.8354884832491305)	Mean: 0.8295, Confidence Interval: (0.7836365235072613, 0.8753634764927387)	Mean: 0.8401666666666666, Confidence Interval: (0.8000249420787906, 0.8803083912545426)
Recall	Mean: 0.7575000000000002, Confidence Interval: (0.6924492318774536, 0.8225507681225468)	Mean: 0.7625, Confidence Interval: (0.6968255778691412, 0.8281744221308587)	Mean: 0.5691666666666666, Confidence Interval: (0.4854336776148266, 0.6528996557185066)	Mean: 0.5233333333333333, Confidence Interval: (0.43507044217540924, 0.6115962244912574)	Mean: 0.37, Confidence Interval: (0.2776532341900619, 0.4623467658099381)	Mean: 0.515, Confidence Interval: (0.4252891016158634, 0.6047108983841366)
F1	Mean: 0.7720158730158729, Confidence Interval: (0.714465948762831, 0.8295657972689149)	Mean: 0.7612539682539683, Confidence Interval: (0.7017660768731312, 0.8207418596348054)	Mean: 0.5015353535353535, Confidence Interval: (0.43250067575996975, 0.5705700313107372)	Mean: 0.4459682539682539, Confidence Interval: (0.3743055613432094, 0.5176309465932983)	Mean: 0.2753333333333333, Confidence Interval: (0.2069741840765505, 0.34369248259011614)	Mean: 0.44624170274170266, Confidence Interval: (0.37264129033005255, 0.5198421151533528)
ROC-AUC	Mean: 0.9427916666666667, Confidence Interval: (0.9078792394715942, 0.9777040938617392)	Mean: 0.9119861111111112, Confidence Interval: (0.8763537579505024, 0.9476184642717199)	Mean: 0.628263888888889, Confidence Interval: (0.554380386844755, 0.7021473909330229)	Mean: 0.6010694444444445, Confidence Interval: (0.5368446715863338, 0.6652942173025553)	Mean: 0.4838888888888887, Confidence Interval: (0.428200019553076, 0.5395777582247017)	Mean: 0.719513888888889, Confidence Interval: (0.6451177680787991, 0.793910009698979)
Runtime	4.114166	3.992492	4.960959	3.776096	3.607677	3.80458
Accuracy	Mean: 0.8366666666666666, Confidence Interval: (0.8078236463285983, 0.8655096870047349)	Mean: 0.8533333333333334, Confidence Interval: (0.8266824580089551, 0.8799842086577117)	Mean: 0.6333333333333333, Confidence Interval: (0.6080298421381115, 0.6586368245285551)	Mean: 0.6383333333333333, Confidence Interval: (0.6090952979836195, 0.6675713686830471)	Mean: 0.565, Confidence Interval: (0.5424174315012662, 0.5875825684987337)	Mean: 0.6033333333333334, Confidence Interval: (0.5794263320343183, 0.6272403346323485)
Precision	Mean: 0.924, Confidence Interval: (0.895954872366131, 0.9520451276338691)	Mean: 0.8660000000000001, Confidence Interval: (0.8265505669157761, 0.9054494330842241)	Mean: 0.8163333333333332, Confidence Interval: (0.7668889475336177, 0.8657777191330488)	Mean: 0.7438333333333332, Confidence Interval: (0.6923490603042486, 0.7953176063624179)	Mean: 0.8478333333333333, Confidence Interval: (0.7997774467428229, 0.8958892199238437)	Mean: 0.8425, Confidence Interval: (0.801315701616595, 0.883684298383405)
Recall	Mean: 0.7433333333333334, Confidence Interval: (0.6787523990983235, 0.8079142675683433)	Mean: 0.8266666666666667, Confidence Interval: (0.7775752852099045, 0.8757580481234288)	Mean: 0.41916666666666663, Confidence Interval: (0.33171222830651886, 0.5066211050268145)	Mean: 0.5466666666666666, Confidence Interval: (0.45936901084985154, 0.6339643224834818)	Mean: 0.2766666666666666, Confidence Interval: (0.19145486675348297, 0.36187846657985023)	Mean: 0.3966666666666667, Confidence Interval: (0.3034916208896157, 0.48984171244371766)
F1	Mean: 0.7546984126984126, Confidence Interval: (0.6971176766090521, 0.8122791487877732)	Mean: 0.8165238095238097, Confidence Interval: (0.7766438828772706, 0.8564037361703487)	Mean: 0.36086507936507933, Confidence Interval: (0.28995566094766245, 0.4317744977824962)	Mean: 0.45708730158730154, Confidence Interval: (0.3864323371827224, 0.5277422659918807)	Mean: 0.2112813852813853, Confidence Interval: (0.14710427236421025, 0.27545849819856033)	Mean: 0.30923809523809526, Confidence Interval: (0.2364934198499039, 0.3819827706262866)
ROC-AUC	Mean: 0.9129166666666666, Confidence Interval: (0.8746912865847445, 0.9511420467485887)	Mean: 0.91125, Confidence Interval: (0.8817194844009928, 0.9407805155990072)	Mean: 0.5350555555555556, Confidence Interval: (0.4657647897984226, 0.6043463213126886)	Mean: 0.5206111111111111, Confidence Interval: (0.45328644735992796, 0.5879357748622943)	Mean: 0.4792083333333333, Confidence Interval: (0.42467209868345085, 0.5337445679832157)	Mean: 0.5215277777777778, Confidence Interval: (0.43258721185324983, 0.6104683437023058)
Runtime	3.96288	3.350271	3.964108	3.398453	3.483609	3.555429

Figure A.6: SVM Linear/Poly/Rbf/Sigmoid