# CREATE: Clinical REcords Analysis Technology Ensemble

Alex Dekhtyar[a], Skylar Durst[b,a], Vadim Kagan[b], Andrew Stevens[b], V.S. Subrahmanian[c,b], Joshua Terrell[b]

[a] *Department of Computer Science and Software Engineering, California Polytechnic State University, San Luis Obispo*
[b] *SentiMetrix, Inc*
[c] *Department of Computer Science, University of Maryland*

**Abstract**

In this paper, we describe the work that enabled us to win the psychiatric symptom severity prediction challenge that constituted Track 2 of the 2016 Centers of Excellence in Genomic Science (CEGS) Neuropsychiatric Genome-Scale and RDOC Individualized Domains (N-GRID) Shared Task in Clinical Natural Language Processing. In order to predict the severity of psychiatric symptoms on a 4-point scale, we used a novel stacked machine learning architecture in which (i) a base data ingestion/cleaning step was followed by the (ii) derivation of a base set of features defined using text analytics, after which (iii) association rule learning was used in a novel way to generate new features, followed by a (iv) feature selection step to eliminate irrelevant features, followed by a (v) classifier training algorithm in which a total of 22 classifiers including 2 new classifier variants defined by us were used on seven different data views, and (vi) finally an ensemble learning step, on which ensembles of best learners were used to improve on the accuracy of individual learners. All of this was tested via standard 10-fold cross validation on training data provided by the N-GRID challenge organizers after which the best three algorithms were selected for submission to N-GRID's blind testing, with the best of our submitted solutions garnering an overall final score of 0.863 according to the organizer's measure. All 3 of our submissions placed within the top 10 of all 65 submissions.

*Keywords:* clinical data analysis, natural language processing, N-GRID challenge

## 1. Introduction

When diagnosing patients seeking help for mental health-related issues, there are two critical factors: correctness and timeliness of the diagnosis. The correct diagnosis allows the clinical psychiatrist to devise and implement the appropriate treatment. The timeliness of the correct diagnosis means that the appropriate treatment can start as soon as possible. Correctly assessing the severity of a patient's psychological symptoms poses a challenge with substantial negative consequences if estimated incorrectly. If the severity of a patient's condition is underestimated, the patient will not receive proper treatment, and the condition may deteriorate; if the severity of the condition is overestimated, the patient may wind up receiving potentially harmful medications.

Initial psychiatric evaluations of patients therefore play a crucial role in both the timeliness and the correctness of the diagnosis. Such evaluations often contain a plethora of information, including the patient's mental health history, the family's history of mental conditions, and a detailed report of the patient's present symptoms. Some of this data is naturally generated in a well-structured form: e.g. as patient's answers to a series of self-assessment survey questions. Other parts of the evaluations come as unstructured text: doctor's notes, patient's verbatim comments, and so on.

Track 2 of the CEGS N-GRID 2016 Shared Task in Clinical Natural Language Processing challenged the participants to analyze the initial psychiatric evaluations of a group of patients for the purpose of predicting the severity of their symptoms. In this paper, we describe the methods used by our team to win this challenge, by leveraging known natural language processing (NLP) and machine learning methods into a single pipeline we called CREATE (Clinical REcords Analysis Technology Ensemble) shown in Figure 1.

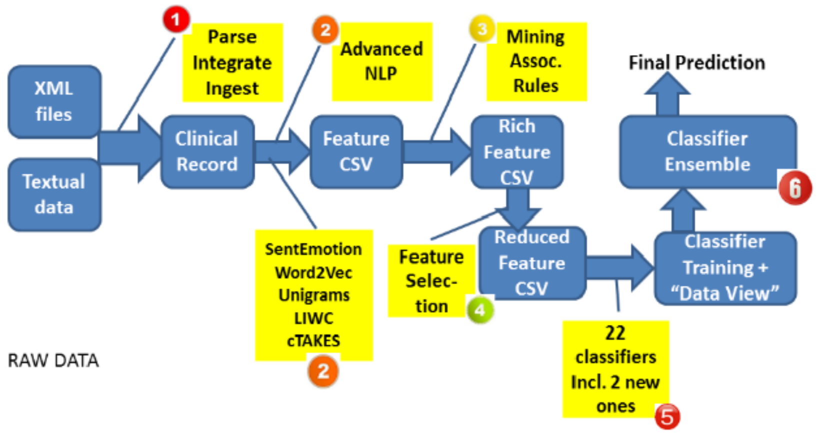

Figure 1: Architecture of the CREATE Framework

Our CREATE framework is a pipeline that includes several innovations. We start by taking the raw (noisy, cluttered) data provided to N-GRID challenge participants and use a data ingestion phase to ingest and clean it. Second, we apply a host of sophisticated methods to extract a "base" set of features for each clinical record from the ingested data. We expand this set of features via a variety of methods, adding 8198 new features to the original 86. Third, we apply association rule mining to learn approximately 345,000 association rules [18, 33], and then trim them to a set of 628 predictive rules. Though association rules have been widely used in the literature for classification, we use them to generate 628 new features, one for each association rule, which to the best of our knowledge, is a novel use. Fourth, with the new total of $9284 + 628 = 9912$ features we engage a set of feature selection operations in order to eliminate useless features. Fifth, we train a total of 22 classifiers, each on seven different subsets of features (data views). Of these, two are novel adaptations of existing Random Forest [19, 9] and AdaBoost [16] classifiers. All of these classifiers build on top of the association rule classifiers developed earlier, which is why we call them "stackable" in our framework. On our final step, we train different ensemble classifiers consisting of the subsets of best individual learners in order to improve the final accuracy of detection of patient condition severity. In both the fifth and sixth steps, we do extensive k-fold cross validation and move forward from there to make our final predictions. This is not the first time predictors have been stacked — in the past, we developed stacked regressors for predicting airline market share and demand [2] as well as for predicting the number of hosts in a given enterprise that will be attacked by a given malware [27].

The rest of the paper is organized as follows. In Section 2 we discuss prior work in this area. Section 3 explains the details of Track 2 of the CEGS N-GRID 2016 Shared Task in Clinical Natural Language Processing (which we, for brevity, refer to as "the N-GRID challenge" throughout the rest of the paper) and provides an overview of the data we have received from the challenge organizers. Section 4 provides an overview of CREATE's six-step approach to the N-GRID challenge. Because Step 1 (ingestion and cleaning) is straightforward, albeit tedious, we do not describe it. Section 5 describes the features we used, while Section 6 describes how we selected just 628 of 345,373 association rules generated by an off-the-shelf association rule mining engine called FP-Growth[18], and by adapting Quinlan's C5 decision tree induction algorithm [32, 33] for association rule mining. Section 7 shows how we determined which features (original and extended) were irrelevant. Section 8 describes how we engaged in an intensive classifier training and hyperparameter optimization procedure on seven different data views of our dataset, which included the adaptations of the well-known Random Forest and AdaBoost classifiers for our purpose. Finally, in Section 9 we describe how we put together a variety of ensembles made from the best-performing individual learners, to improve the final prediction accuracy. We conclude in Section 10.

## 2. Related Work

We limit discussion of related research to two categories: work on analysis of clinical records in the medical domain, primarily concentrating here on NLP approaches; and emerging machine learning and NLP technologies. In addition to these, our work on this project used a wide array of classification [45] and association rule mining [18] techniques, and traditional methods for text parsing and POS tagging [39]. We used the Python scikit-learn [29] and nltk [6] toolkits, the Stanford parser and Part of Speech Tagger [39], and the Snowball stemmer for English[7].

*SentEmotion.* The SentiMetrix team (consisting of the co-authors of this paper) became interested in this challenge due to prior work SentiMetrix has conducted in the domain of clinical diagnoses for mental health conditions [4, 20]. Founded in 2007, SentiMetrix is a data science and social analytics company offering a variety of solutions to address the "big data" needs of commercial companies and government agencies.

Through its work on past projects [15, 21, 40] SentiMetrix has built an array of technology-based solutions, focusing on the near real-time analysis of large quantities of complex data in multiple languages. Prior projects applied these solutions to data analytical challenges in the areas of national security, elections, marketing, and medicine. We built upon our prior work on detecting mental health disorders such as Depression, PTSD and TBI from patient notes[4, 20]. This system, known as COPTADs, is a text-based classification engine that was developed in cooperation with psychologists. Leading surveys had identified a set of signals that a victim of depression might have: for example, isloation from others. We previously built a classification engine on top of the Stanford Parser [39] to extract these signals, and then a second layer to extract emotions such as anger or fear. The third layer generates a confidence value for each of the disorders that COPTADs is configured to recognize.

*Feature Selection using Association Rules.* K. Rajeswari demonstrated the use of using Apriori rule mining in order to select features with high significance [35]. First, the authors mined a set of rules with a small $k$ value. The second step was removing all features that did not appear in at least one rule. The third step was to re-run the Apriori but on a much larger $k$. Finally, when training the final classifier, they included only the features that were an antecedent in at least one rule. The higher $k$ value omits some useful association rules, but should take an order of magnitude less time to complete, due to the large reduction of $n$ candidate rules. The author noticed that on a dataset attempting to classify the risk of heart disease, that computation time was cut by a full two orders of magnitude with this procedure.

*Word2Vec.* In Spring 2016, Google released its Convolutional Neural Network (CNN) deep learning package TensorFlow [1], and an information retrieval/natural language processing framework called Word2Vec [25] based on analysis of large quantities of text using TensorFlow.

The Word2Vec release contained a dataset of 3 million instances trained on a corpus of 100 billion words represented as vectors of 300 latent features computed by the TensorFlow-trained CNN using a large Google News text corpus as the training set. The information captured by the Word2Vec vectors relates to the co-occurrences of different words in the same contexts. Through empirical studies, Google showed that the Word2Vec vectors representing individual words contain enough information to represent synonymy (vectors of synonyms are similar), as well as a number of other semantic concepts.

*Med2Vec.* Building on top of Mikolov's Skip-Gram model, Choi et al. sought to create a deep learning document embedding strategy [12]. They had two datasets of 3 and 5 million documents with a combined total almost 30,000 medical codes that acted as a natural clustering. Med2Vec is constructed in a similar method as Skip-Gram Word2Vec, but treats sequential visits from the same patient as if they were sequence of words in a sentence. Compared to Skip-Gram Word2Vec and GloVe, Med2Vec achieves lower, better normalized Mutual Information Gain scores on Medication and Procedure, implying that it builds embeddings that are better clustered for those tasks [31]. While SVD of document text performed better than Med2Vec, SVD was demonstrated to have far lower interpretability [17].

*Ensemble Construction.* Constructing ensembles was a key step in winning both the N-GRID competition as well as others, such as Kaggle. However, constructing ensemble rules by hand can be time-consuming or miss optimal solutions. Cortes et al. describes an online machine-learning algorithm called ESPBoost that accepts hundreds of potential "experts" and the correct label [13]. ESPBoost uses coordinate descent to reduce the Hamming loss which finds a locally performant ensemble without enumerating all possibilities [5, 22]. ESPBoost did best on large problems with a large number of experts.

## 3. The Challenge

The specification of Track 2 of the CEGS N-GRID 2016 Shared Task in Clinical Natural Language Processing (also called the RDoC for Psychiatry Challenge) presented the goal of this particular track of the challenge as:

> *"Determine symptom severity in a domain for a patient, based on information included in their initial psychiatric evaluation. The domain has been rated on an ordinal scale of 0-3. There is one judgment per document, and one document per patient."*[43]

Research Domain Criteria (RDoC) is a framework for facilitating the study of human behavior, both normal and abnormal in various clinical domains. The RDoC provided the data, originally collected by Partners Healthcare Inc. and the Neuropsychiatric Genome-Scale and RDoC Individualized Domains (N-GRID) project at the Harvard Medical School [43]. The data was released to the challenge participants under a strict set of Rules of Conduct and the Data Use Agreement.

As shown in Table 1, a total of 649 records were released, broken into a training set of 433 files and a test set of 216 files with no ground truth — the latter was released only two days before the submissions were due. The initial release of the 433 patient records was broken into two categories: a suggested training set of 325 files, and 108 files called annotated_by_1. Since the contest organizers discouraged us from using the records from the annotated_by_1 set as training set data, we focused most of our training on the 325 record training set.

| | |
|---|---|
| Total number of records released | 649 |
| Number of records in suggested training set | 325 |
| Number of records in additional training set | 108 |
| Number of records in test set | 216 |

Table 1: Overview of released data.

| Value | Meaning |
|---|---|
| 0 | ABSENT |
| 1 | MILD |
| 2 | MODERATE |
| 3 | SEVERE |

Table 2: The scale of the target Valence variable in the N-GRID challenge training set data.

Each record, originally stored in a single XML file, represented the information from the initial psychiatric consultation of a single patient. For each record in the training set the challenge organizers supplied the ground truth about the severity of the patient's psychiatric condition. This information was stored in a single feature of the data files called Valence. Table 2 shows the scale on which the patients' conditions were evaluated. The judgment contained in the Valence field came from a clinical expert and was based solely on the symptoms and medical, social, mental health, and family history captured in the provided data.

The XML files provided by the challenge organizers contained both structured data, which documented demographic information, mental health history, education, employment, financial status, family history of mental health, medical history, prescription and recreational drug use and a few other categories of information; and unstructured, free-form textual data, which documented self-reported symptoms and attending psychiatrists' notes on the patient and their condition. The data was in its raw, originally recorded form, and contained numerous typos, a lot of missing attributes, inconsistent use of abbreviations, and freeform text. The data was also anonymized.

Figure 2 shows a *notional (not real) record* created by us to illustrate the nature of the data — it contains no information from the N-GRID dataset. However, this notional record can give the reader an idea of the type of information that the teams had access to while working on the challenge (as an aside, we note that the actual records contained significantly more data in them).

Table 3 contains a rough breakdown of the types of features found in the original data. Table 4 contains the list of features from the original data that were deemed by our team to be free-form text.

The results of the challenge were evaluated using the a variant of the Mean Absolute Error (MAE) metric. Given a vector $\mathbf{v} = (v_1, \ldots, v_n)$ of ground truth values and a prediction vector $\mathbf{p} = (p_1, \ldots, p_n)$, the MAE of the prediction is

| Type of feature | Number of features |
|---|---|
| All features | 102 |
| Demographic information | 3 |
| Harming Others Or Self | 4 |
| Symptoms of Mental Health Issues | 18 |
| Drug, Caffeine and Alcohol Use | 5 |
| Family History | 4 |
| Symptom Denial | 8 |
| Independence of Daily Activities | 9 |
| Marital Status and Abuse | 3 |
| Employment and Finances | 4 |
| Owns Firearms | 1 |
| Legal History | 1 |
| If Underage, Legal Guardian | 2 |
| Military History | 2 |
| Appearance | 14 |
| Mental Health | 18 |
| Physical Health | 6 |

Table 3: Breakdown of features in the original N-GRID 2016 challenge Track 2 dataset by category.

| Free-form Entries |
|---|
| Chief Complaint (Patient's Own Words) |
| History of Present Illness and Precipitating Events |
| Previous Treatments |
| Prior Medication Side-effects |
| Current Medication |
| Childhood History |
| Interpersonal Concerns |
| Education |
| Family Living Situation |
| Protective Factors |
| Risk Factors |
| Actions Taken |
| Formulation |
| Level of Care |

Table 4: List of free-form text fields found in the original data for Track 2 of the N-GRID 2016 challenge.

```
RAW DATA

 Name:  John Doe

 Age:  42

 Sex:  Male

Referred by Emergency Services
 Has difficulty remembering if he has taken prescription drugs.
Accidental overdose.
 Referral Notes:  Patient exhibits short-term memory loss - blood tests
reveals mixed alcohol with prescription.  Stayed overnight.
Found bruises on shoulders - possibly from falling.

 DEPRESSION: YES
OCD: No
 PANIC: Yes
 Prescriptions:
 Advil (3 times a day)

 Formulation:
 Patient has history of anxiety and bipolar.

 Recommendations:
 Change medication to Alpazolam.
 Require additional visit in 2 weeks.
```

Figure 2: A synthetic record illustrating the type of clinical medical records data contained in the dataset released for Track 2 of the N-GRID challenge.

computed as

$$MAE(\mathbf{p}, \mathbf{v}) = \frac{1}{n} \sum_{i=1}^{n} |p_i - v_i|$$

The MAE was altered by the challenge organizers so that the score was in the range of [0, 1], where 1 indicated a perfect score. The new MA-MAE, Macro-Averaged Mean Absolute Error, measure was computed by splitting the set of records into four categories (one per ground truth Valence value), computing the MAE for each of the four subsets independently, and combining the computed MAEs into a weighted sum. The normalizing factors for each of the component MAE values are the highest possible errors that can be achieved for a data point with the given Valence value (3 for Valence=0 and Valence=3, and 2 for Valence =1 and Valence = 2). To make the computed value correspond to the *higher is better* intuition, the computed weighted sum was subtracted from 1. The formula for computing the N-GRID Challenge version of MAE is:

$$MA\_MAE(\mathbf{p},\mathbf{v}) = 1 - \frac{(\frac{MAE(\mathbf{P0},\mathbf{V0})}{3} + \frac{MAE(\mathbf{P1},\mathbf{V1})}{2} + \frac{MAE(\mathbf{P2},\mathbf{V2})}{2} + \frac{MAE(\mathbf{P3},\mathbf{V3})}{3})}{4}$$

Every team participating in the challenge was allowed to submit up to three final guesses. Each guess was essentially a vector of predicted valences for each case history from the provided *test set*.

## 4. Overview Of CREATE

Figure 1 describes the 6 parts of CREATE. We provide brief overviews of the individual components of CREATE below.

1. Data Ingestion - a fairly straightforward, if tedious process of converting the XML data files provided to us into case $\times$ feature matrices. We limit our discussion of this step to what was presented in Section 3, when we discussed the provided dataset.
2. Feature Extraction - described in Section 5. We started our work on predicting the Valence variable by careful extraction of existing features from the raw XML data provided to us by the organizers. After starting with the features present verbatim (i.e., as unique elements) in the released dataset (see Table 3) we defined several other features to generate a single overarching dataset.
3. Development of Association Rule-based Features - described in Section 6. In this stage, we extracted a set of 345,373 Class Association Rules from the above dataset and then eliminated redundant ones to generate a total of 628 in all. For each retained Class Association Rule we included a binary feature into our dataset.
4. Feature Selection - described in Section 7. We next devised a set of tests to identify irrelevant features; a feature that failed all of the tests was eliminated.
5. Classifier Development & Training - described in Section 8. We devised seven different views of our data: each view containing a specific subset of the full set of features. We put together a battery of 22 machine learning algorithms, including two novel adaptations of Random Forests and AdaBoost that we developed. We trained the 22 classifiers on our seven data views and selected the best runs for the ensemble learning step.
6. Ensemble Learning - described in Section 9. On the last step, we evaluated ensembles of best-performing individual classifiers. We used both simple majority/plurality ensemble schemes, as well as more complicated voting techniques to see which, if any, provided the best solutions. At the end, a number of *simple* ensembles over subsets of our classifiers emerged with accuracies that were clear improvements over the best individual classifiers, and produced MA-MAE scores over 0.86. From those, we selected three predictors that we submitted to the N-GRID challenge organizers. We were glad and proud to discover that one of our submissions had the highest overall MA-MAE among the submitted solutions.

## 5. Step 2: Feature Engineering

To analyze the provided data, first we had to transform the original XML data into a tabular, textual format. Each XML file was structured so that it contained all the patient informaiton in a single CDATA block, along with a single tag describing the Valence. Manual examination of several XML files revealed the underlying structure of the patient records (see Figure 2). We have identified portions of the patient record that we elected to represent as free-form text features (see Table 4). These were primarily the restatements of symptoms experienced by the patients recorded from their own words, plus notes and observations of the psychiatrists conducting the evaluation of the patients. Most other content from XML files are represented as key-value pairs, with both keys and values relatively straightforward to determine and extract. During the extraction process, we were able to reduce the 102 features (identifiable in the XML files as individual prompts) to 86 features, which we term the "original" N-GRID dataset features.

The parser's output was a CSV file with each row containing information from a single case history (ie, XML file) and each column containing information about a single feature: textual or structured. The initial breakdown of features is described in Table 3. Not every XML document had values for all the extracted features; in fact, some features were present only in a handful of records, and other features were often omitted from records. Another data quality issue worth noting is the relative frequency of typos (which could have come either from the process of digitization of the records, or from the initial medical records themselves). Regular expressions were used to reduce the amount of error in boolean and categorical entries. Some examples include catching different ways to say No: N, Missing or Not. Other expressions simplified synonymous medical codes or shorthand in categorical features, such as ld for a learning disability. For free-form text, no typo detection or conjoined word detection was used.

We have then proceeded to enhance the original N-GRID dataset with a wide range of additional features. Below we discuss the nine different ways in which we augmented our feature set. Table 5 contains the summary of our feature enhancement efforts.

*Cumulative Scores.* Our initial investigation of the original features extracted from the raw data unveiled groups of related features, typically with "yes"/"no" values, where each individual feature was rarely set to "yes" and no relationship with Valence appeared to exist. Moreover, the overall number of such features set to "yes" in a single patient case history seemed to be in some relationship with Valence. In such cases, we added a new feature, a cumulative score of "yes" values in a group of features, to the dataset.

For example, the original features contained a relatively rich arsenal of substances that a patient could abuse or consume, from readily-available substances such as tobacco, caffeine, and alcohol to a wide range of recreational drugs. We identified all such features, and added a new feature Cumulative_Substance_Use which stored a count of substances which the patient admitted to using. Similar

| Approach | Explanation | # New Features |
|---|---|---|
| Original (Munged) Features | Features taken from original data | 86 |
| Cumulative Scores | Aggregations of like features | 62 |
| Medications | Individual medications taken by patients | 47 |
| Association Rules | ARs from features to Valence | 628 |
| Unigrams | Representations of textual data | 8033 |
| Word2Vec vectors | Representations of textual data | 300 |
| SentEmotion | Sentiment and emotion extraction from text | 49 |
| cTAKES | Medical symptom tagging | 658 |
| LIWC | Topic detection and POS counts | 93 |
| Commonality of Patient | A measure of how typical a patient is | 5 |
| TOTAL | | 9912 |

Table 5: A list of approaches to enhancing the feature set for the N-GRID Challenge (Track 2) dataset.

cumulative count features were created for a few more groups of variables: number of psychiatric review conditions deemed positive for the patient, number of "abnormal" items from the mental status exam, number of activities the patient does not perform independently, and more.

The reasoning behind adding such features to the dataset was straightforward: we saw features which appeared to carry important information, but which, due to relative lack of positive/abnormal/out-of-ordinary values could not solely by themselves contribute to the learning of Valence. By creating cumulative count features, we represented the quantitative effects: case histories with more positive/abnormal responses in those feature columns received higher counts. This removed some of the sparsity of the dataset.

*Extracting Medications.* To capitalize on the possibility of using medications in predicting Valence, we: (i) manually created a list of 47 medications deemed relevant for patient conditions, complete with alternate spellings and abbreviations where applicable, (ii) developed a Medication Extractor that analyzed the input data and produced the list of medications mentioned in it, and (iii) created a dataset of medication mentions with 47 columns corresponding to the medications our Medication Extractor tool was tracking.

*Emotion Features.* SentiMetrix's SentEmotion is a web service, developed as part of the COPTADS project [4, 20] (see also Section 2) that extracts the intensity of emotions such as anger, fear, depression, anxiety, stress, etc. from freeform text. In addition to labeling the overall sentiment of a text fragment [42] and individual emotions expressed in the text (anger, fear, depression, etc), the system outputs a confidence value, which expresses the level of confidence the system has in the presence of the emotion. We ran all textual information for each of the records through SentEmotion and added 49 new mental health-related features.

*Simple Representations of Textual Information.* At our initial examination of the provided data, we identified a number of features whose contents constituted free-form text. We considered using the free-form text from each of the features as a separate input into any text analysis procedures we were employing. However, at the end, we decided to concatenate the contents of all free-form features into a single free-form text feature, and conduct all text analysis on it. This resulted in the richest possible text being processed for each patient record.

We investigated a number of way to represent textual data in our dataset. The first, more straightforward approach we took resulted in the following steps, which are a part of our the SentiMetrix Common Pipeline framework for data processing and data ingestion.

- Removal of dates and integers from dataset into SMXDATE and SMXNUM features
- Stopword removal using the suggested english stopword lists in NLTK [6] and Scikit-Learn [29]
- Stemming using the Snowball Stemmer [7]
- Term-Frequency Inverse-Document Frequency [41, 3] of unigram features for each surviving word stem/term

*Word2Vec for Textual Information.* Our second approach used Word2Vec methodology[1] [25] introduced recently by Google. to represent each word found in each freeform text as a vector of 300 features. We used Google's own collection of Word2Vec vectors trained on the Google News corpus by Google. Despite N-GRID data containing a lot of specialized technical terms from psychiatric domain, and proper names such as names of medications, 96.8% of tokenized text contained in the N-GRID training set was also found in the Google's Word2Vec dataset with a coverage of 78.4% of unique words. Examples of words not covered are typos such as "weopons", "ibuprofin" or "bipolaar"; conjoined words such as "employment.He"; dates such as "8/17/86"; and medical jargon such as an exact dosage for a patient.

To represent the text from individual patient records, we took the vector representations of each term found in the free-form text in the patient's record, and computed the mean vector. This is the Word2Vec equivalent of the traditional Bag of Words model, and acknowledged as a naive baseline to construct a ParagraphVector by Mikolov and Le [26]. This procedure added 300 features to our dataset. We used gensim to load the binary Word2Vec word-to-vector file [34].

*cTAKES Features.* As mentioned in Section 2, Apache cTAKES is a framework for extracting a variety of information from medical records. cTAKES looks for terminology related to medical symptoms, mentions of medications, body parts,

[1]https://code.google.com/p/word2vec/

procedures, diseases, disorders, and a few other categories of information. For each patient record, we ran the concatenated free-form text extracted from the record through cTAKES to collect these signals.

*LIWC Features.* LIWC, Linguistic Inquiry and Word Count [30], is a linguistic computerized text analysis tool similar to SentEmotion. LIWC produces 93 signals, which include various low-level Parts-of-Speech analysis such as the number/frequency of pronouns, semantic features such as if the document has a positive or negative tone, and basic topic-analysis such as detecting if the document focuses on home, money, leisure, the past, or friends. We have run the free-form text extracted from each record, collected all LIWC features, and added them to our dataset.

*Common Value Features.* Common Value Features are another form of a cumulative feature, but rather than summarizing logically related features, they summarizes features that individually have little explanatory power. For example most individual observations of a variety of patient behaviors were labeled with the code "WNL" which is interpreted as "within normal limits". In fact, *most* patients had *all* their observations set to "WNL", so a group of seven-to-eight "WNL"-valued features formed a very well-defined, *but not very interesting* frequent itemset. These very frequent, but essentially benign itemsets give rise to a large number of useless association rules during the rule generation process. Since an exhaustive mining process on our dataset is extremely slow for any $k$ greater than 4, these very frequent itemsets tended both contribute to consume significant CPU resources while not producing any interesting results.

To reduce the size of our market baskets, we created the concept of a typical value. We set up five separate "typicality" thresholds: 51%, 62.5%, 75%, 87.5%, and 90%. Given a number $t$ from the list above, and given a feature from our feature set, a specific value of the feature was called *t-typical* if more than $t$ percent of all records in the training set contained this value.

We aggregated the notion of *t-typicality* by introducing five *common value features* into the dataset: one per typicality threshold. The common values feature for threshold $t$ was set to the total number of other features in the given record which contained *t-typical* values. These new features allowed us to quickly see whether a specific patient evaluation record yielded rare, atypical, unusual values for its features.

## 6. Step 3: Class Association Rules

We decided to see if we could discover some clear dependencies between the features present in, potentially small, subsets of patients, and the value of their Valence. To test this, we engaged in the process of mining our feature data for Class Association Rules.

We constructed a subset of binary and categorical features found in the data. These primarily included the original features, medication and cumulative features, and a small handful of features retreived from other sources. With

| Parameter | Value |
|---|---|
| Minimal Support | 20 records |
| Minimal Confidence | 0.6 |
| Maximal Inverse Confidence | 0.4 |
| Maximal Negative Confidence | 0.4 |

Table 6: Pruning conditions for Association Rule mining process.

these features, we concentrated on discovery of class association rules of the form:

$$F_1, F_2, \ldots, F_k \longrightarrow \mathsf{Valence},$$

where $F_1, \ldots, F_k$ are conditions on the binary/categorical features. Table 6 shows the parameters for our Class Association Rule search; we pruned away all rules that did not satisfy them.

We used an existing Python implementation [28] of the FP-Growth [2] [18] algorithm to perform an exhaustive search for Class Association Rules with $k \in \{1, 2, 3, 4, 5\}$. For larger values of $k$ ($k = 6 \ldots 9$) we used $C5$ [33, 32], which is non-exhaustive .

The discovered rules went through a rigorous pruning procedure. In addition to pruning away all discovered Class Association Rules (CARs) that did not pass the minimum standards shown in Table 6, we also conducted a $\chi^2$ test of significance for each discovered CAR (see Section 7 for a more detailed explanation of the $\chi^2$ tests conducted). All CARs that did not pass the $\chi^2$ test at the significance level of $p = 0.05$ were also eliminated from consideration. Essentially, failing the $\chi^2$ test meant that there are significant reasons to believe that the CAR is a by-product of individual frequencies of the features it contained, rather than an actual meaningful relationship between these features and the Valence variable.

Finally, we performed a *Coverage Test* as proposed by Li et al [23]. The purpose of the Coverage Test is to reduce the set of CARs to the ones that most accurately describe our data while avoiding excessive duplication. First, we sorted all of our generated CARs by confidence, support and $\chi^2$ score from best to worst. Starting with the first rule, all documents with features in the antecedent of the rule were marked. Then, we advanced onto the second round and marked all documents with features in the antecedent of that rule. The process was repeated until each input record was covered. After a single docu-

[2]Since the original Python implementation is not actively maintained, SentiMetrix has a private fork of the repository. SentiMetrix's has API tweaks that allow the emission of only Class Association Rules, rather than all Association Rules, cached metrics for aggressive filtering while mining, support Python 3, and utilizes Numpy arrays rather than Python lists for more compact memory allocation and faster cache coherence [44]. The overall improvements result in a modest reduction of memory, and a 33% reduction in run-time, while considering only Class Association Rules reduces the problem size by multiple orders of magnitude. This is significant, because even with these improvements mining higher $k \in \{4, 5\}$ took days to complete.

| Antecedent | Valence | Support | Confidence | Neg. Conf. |
|---|---|---|---|---|
| `patient is an inpatient` and `currently undergoing addiction treatment` | SEVERE | 22 | 21/22 (95.45%) | 18.25% |
| `patient NOT taking Aplenzin` and `has no history of drug abuse` and `suffers from OCD` | MODERATE | 25 | 20/25 (80%) | 22.06% |
| `patient is NOT inpatient` and `does not drink alcohol` and `is NOT taking Allernaze` and `is NOT taking Levothroid` and `is NOT taking Cultivate` and `is NOT taking Abilify` and `does not suffer from OCD` and `has no history of violence` and `suffers from depression` | MILD | 73 | 67/73 (92%) | 38.37% |

Table 7: Examples of discovered Association Rules.

ment has been marked five times, we removed it from future consideration. If a rule did not "cover"" any considered documents, we discarded the rule. Once all documents have been marked five times, we discarded all remaining rules.

Altogether, the pruning process reduced the total number of CARs extracted from the data from 345,373 to 628. *For each extracted Class Association Rule, we added one binary feature to the dataset, which was set to 1 on records where the antecedent of the Association Rule applied* [3]. Some examples of the Association Rules we mined during this process are presented in Table 7. The first two rules were found by the FP-Growth process, and the third by C5.

## 7. Step 4: Feature Selection

Because we now had thousands of features to consider, we developed a feature selection procedure that subjected each feature in our dataset to a battery of tests. *Features that failed every single test were eliminated from consideration.* The battery of tests is described below.

*Association Rule test.* This decision procedure can be boiled down to a simple statement: *keep a feature if it appears in the antecedent of at least one of the 628 Class Association Rules in our dataset.*

*$\chi^2$ test for categorical features.* We ran a $\chi^2$ test [46] for each categorical feature against the Valence variable. This test checks whether there are sufficient

[3]The conclusion of the CAR was not considered, as that would result in leaking the label information during training, nor could these features be constructed on a hidden test set

grounds to believe that a specific categorical feature is associated with another categorical feature purely by coincidence. We set our confidence estimate at 95% and rejected any categorical feature whose $\chi^2$ test yielded a p-value higher than 0.05. The $\chi^2$ test was implemented by using scipy's chisquare function to compute the p-value of each categorical feature [10].

*ANOVA F-test for continuous features.* ANOVA F-tests are used to test the significance of a regression model[8]. While we used the $\chi^2$ test to test for potential significance of our categorical features, we used the multi-way ANOVA F-test for our numeric features. For each feature tested, we separated the data into four subsets, based on the value of the target Valence attribute. We then randomly sampled from these four groups. We then tested the means and standard deviations in each of the four subsets to see if they represented similar or different distributions, and compared them across our multi-way samples to see if there is a statiscal bias. Similarly to the $\chi^2$ test, we set the confidence level at 95% and rejected any numeric attribute whose ANOVA F-test produced a p-value of more than 0.05. We uses scikit-learn's f_classif function to compute the multi-way ANOVA tests [29].

*Mutual Information Gain test (MIG).* Mutual Information Gain is typically used in measuring the robustness of clustering methods. In unsupervised problems, MIG is measured by calculating **P(X, Y)** - the probability that two variables **X** and **Y** occur in the same cluster - compared to the probability **P(X) * P(Y)** of their occurring in the same cluster by random chance. If there is a clear dependence between the two variables, then the probability of **P(X, Y)** will be higher than **P(X) * P(Y)**. Recent research shows that MIG provides an additional level of feature selection in the context of textual classification and clustering [48]. In the case of supervised feature selection, we compare the entropies and distributions of Valence vs. each feature using $K$ Nearest Neighbors. At the time of the N-GRID Challenge, scikit-learn [29] did not have a completed implmentation of mutual_info_classif, but it was in the process of being developed. We ported scikit-learn's partial implementation into our system.

*Linear SVM Recursive Feature Elimination.* Our final test involved running scikit-learn's version of the Support Vector Machine (SVM) classifier with a linear kernel [14] and observe whether the feature survived the *Recursive Feature Elimination* process implemented within it. An advantage of using a linear SVM to find support vectors is that it provided our system with multivariate feature selection. In addition, $\chi^2$ test and our Class Association Rules only worked on categorical features, while Mutual Information Gain used a heuristic [48] to operate on continuous features. The Linear SVM recursive feature elimination allowed us an additional test on the continuous features in our dataset.

Table 8 contains the overview of the features that survived this process: i.e., that passed successfully at least one of the tests from the list above. We

| Feature Category | Number of Features |
|---|---|
| TOTAL | 788 |
| Original | 30 |
| Cumulative scores | 34 |
| Medications | 10 |
| SentEmotion | 6 |
| cTakes | 40 |
| Common Value | 5 |
| Word2Vec | 34 |
| Unigrams | 1 |
| CAR | 628 |

Table 8: Description of the final set of features remaining in our operational dataset after the feature selection (pruning) step.

make a few observations here about the final shape of the dataset. Only LIWC did not contribute any features. All other means of enhancing non-textual features provided meaningful contributions, with cTakes, original dataset, and, interestingly enough, our cumulative scores accounting for the majority of non-textual features. All five Common Value features also made it. Our manual work on documenting medications resulted in 10 out of 47 medication features kept.

Another interesting outcome of this process was an essential depletion of direct natural language-related features from the dataset. Only 34 out of 300 features that came from Word2Vec were kept.

## 8. Step 5: Classifier Training and Adaptations

For our next step, we have constructed a battery of 22 different classifiers to train on the dataset we built on previous steps. Table 9 lists the classifiers we used on this project. 12 of the 22 classifiers came from scikit-learn. Another five classifiers came from the internal SentiMetrix implementations primarily developed prior to the N-GRID challenge, but modified where needed to work with the data from this challenge. Additionally, we used two neural network learners from Google's TensorFlow: their deep neural network implementation; and their so called *deep and wide* classifier, which combines neural nets (deep learning) with Support Vector Machines (wide learning). Finally, two extra classifiers — XGBoost, the boosted gradient classifier [11], and Quinlan's implementation of C5.0 decision tree classifier [33] — were used as well.

Of the 22 classifiers we used two, the Random Forest Regression with Classification Inference (RF-reg-clf in Table 9, and the SVM-initialized Naïve Bayes AdaBoost were novel adaptations of the well-known Random Forest [9, 19] and AdaBoost [16] machine learning techniques. They are described below.

### *8.1. Classifier Adaptations*

Our two novel adaptations of existing classifiers are discussed below.

| No. | Abbreviation | Classifier | Source |
|---|---|---|---|
| 1 | MB NB | Multinomial/Bernoulli Naïve Bayes | scikit-learn |
| 2 | Lin-SVM | Linear Kernel SVM | scikit-learn |
| 3 | RBF SVM | Radial Basis Function Kernel SVM | scikit-learn |
| 4 | LogReg | Logistic Regression | scikit-learn |
| 5 | RF | Random Forests | scikit-learn |
| 6 | Adaboost NB | AdaBoosted NaiveBayes | scikit-learn |
| 7 | KNN | K-Nearest Neighbors | scikit-learn |
| 8 | SGD | Stochastic Gradient Descent | scikit-learn |
| 9 | BRBM | Bernoulli Restricted Boltzmann Machine | scikit-learn |
| 10 | RF-reg | Regression version of Random Forest | scikit-learn |
| 11 | RF-reg-clf | Train: Regression RF; inference: Classification RF | scikit-learn |
| 12 | Lin-SVM-reg | Regression version of Lin-SVM | scikit-learn |
| 13 | RBF SVM-reg | Regression version of RBF SVM | scikit-learn |
| 14 | DNN | Deep Neural Network | TensorFlow |
| 15 | Deep & Wide | Deep-and-Wide classifier | TensorFlow |
| 16 | SVM-Init AdaBoost | AdaBoost: 1 ropund Linear-SVM, 49 rounds of NB | SentiMetrix |
| 17 | MI SVR | Mutual-Info Feature Boosted SVM | SentiMetrix |
| 18 | CMAR | CMAR (Classifier on Multiple Assoc. Rules) | SentiMetrix |
| 19 | CMAR SVM | CMAR-Boosted SVM | SentiMetrix |
| 20 | CBA | CBA (Classification Based on Associations) | SentiMetrix |
| 21 | XGBoost | XGBoost (scalable gradient boosting) | XGBoost |
| 22 | C5 | C5.0 Decision Tree Classifier | RuleQuest |

Table 9: All the classifiers that were tried as part of the N-GRID Shared Task Challenge

*Random Forest Regression with Classification Inference (RF-reg-clf).* Random Forest is a powerful yet forgiving algorithm that can perform a modest amount of feature selection due to its subsampling [9, 19]. In scikit-learn, there are both regression and classification modes of Random Forest [29]. As Valence can be treated as both a class or an ordinal value, we tried both methods. Since regression provides additional insight for the classifier, it often had slightly higher scores. However, regression biases the kernel into averaging around the mean of all the labels. The result of this is MILD Valences might be moved to be slightly more MODERATE and vice-versa. When it comes to building the ensemble, this small amount of drift can result in large classification errors if it causes the ensemble's vote to cross a rounding threshold. Our adaptation was to train the Random Forest on the regression version of the problem. Then, during inference, round the inferred value to the nearest Valence.

*SVM Initialized AdaBoost (SVM-Init-ada).* The novel technique we used on this project is the initialization of AdaBoost learning process with SVM (SVM-Init-AdaBoost classifier in our parlance). AdaBoost trains a sequence of estimators one after another [16]. After each iteration, the training set is be reweighed; documents that were just misclassified will have their weight increased, while documents that were just classified correctly will have their weight decreased. This forces the next classifier to correct the mistakes its predecessor made. While AdaBoost is traditionally done a fast and weaker classifier such as Naive Bayes, any kernel can be used.

In other project, SentiMetrix has had success by introducing a single round of a slow and strong classifier as a seed for AdaBoost [40]. As Support Vector Machines was one of our better classifiers and provides a meaningful decision function, it works well as the bootstrapping classifier and provides an anchor for the weak classifiers. We modified the AdaBoost process as follows:

1. Step 1: Run Linear kernel SVM classifier on input data.
2. Step 2: Analyze kernel decision function to reweigh document weights
3. Step 3...52: Run Naïve Bayes classification 49 times, reweighing document weights after every iteration

On the input N-GRID challenge data, linear kernel SVM produced better accuracy results than a single Naïve Bayes run. This allowed our modification of AdaBoost to start with a sufficiently accurate bootstrap. This additional accuracy gained on the first step has proven to be a core factor in the overall accuracy of this classifier, as one of its runs wound up being the best individual classifier in our battery.

## *8.2. Data Views*

Each of or 22 classifiers was separately trained on nine different *data views* described in Table 10. A data view is a collection of features onto which the data is projected prior to being supplied to the classifier. Different subsets of features were selected due to their distinct origins, and the desire of our team to see if certain minimalistic sets of features contain enough information for training the classifiers.

Two of the nine data views listed, ANOVA-wordvector34 (the 34 Word2Vec features that passed our ANOVA significance tests) and WordVector (all 300 Word2Vec features) yielded abysmal accuracy for all classifiers, and were eliminated from further consideration.

*Of the remaining seven data views* one, Full, represents the entire collection of features selected during the process described in Section 7, five are its subsets, and one, TF-IDF is the complete set of tf-idf vectors representing the textual portion of each record. The subsets of the Full data view were selected to represent different categories of features (CARs, Numeric[4]) as well as the best features that passed a specific test: $\chi^2$, ANOVA or Multiple Information Gain. We experimented briefly with top 100 and top 75 best features for each of the tests, but settled on top 50, as this provided better accuracy.

[4]The name of this view is a bit of a misnomer, and is kept for historical reasons. This view includes both numeric and categorical features that were present in the original dataset, as well as constructed using cTAKES, SentEmotion, and LIWC toolkits.

| No. | Label | Explanation | Size |
|---|---|---|---|
| 1 | Full | All features that passed filtering | 788 |
| 2 | Numeric | All numeric (and categorical) filtered | 125 |
| 3 | CARs | All CAR features | 628 |
| 4 | TF-IDF | All tf-idf unigram features | 8033 |
| 5 | WordVector | All Word2Vec features | 300 |
| 6 | Chi-square-best50 | 50 features with highest $\chi^2$ value | 50 |
| 7 | ANOVA-best50 | 50 features with highest ANOVA F-scores | 50 |
| 8 | MIG-best50 | 50 features with best MIG values | 50 |
| 9 | ANOVA-wordvector34 | Word2Vec features that passed ANOVA | 34 |

Table 10: Different data views used for classifier training.

| No. | Name | Algorithm | View |
|---|---|---|---|
| 1 | MNB-CARs | MNB | CARs |
| 2 | RF-full | RF | Full |
| 3 | RF-reg-full | RF-reg | Full |
| 4 | RF-reg-clf-full | RF-reg-clf | Full |
| 5 | Lin-SVM-chi2-best | Lin-SVM | chi2-square-best50 |
| 6 | Lin-SVM-anova-best | Lin-SVM | ANOVA-best50 |
| 7 | RBF-SVR-mig-best | RBF-SVR | MIG-best50 |
| 8 | SVM-Init-Ada-CARS | SVM-Init AdaBoost | Rules |
| 9 | D&W-num | Deep & Wide | Numeric |
| 10 | DNN-full | DNN | Full |

Table 11: Abbreviated names for classifiers for the next sections

As a result, at the end of the process we had a total of $22 \times 7 = 154$ trained (classifier,data view) pairs to go through. As a final preprocessing step, we normalized the data view as appropriate for each classifier. For most classifiers, the normalization centered each feature and scaled it to have unit standard deviation using the interquartile range. For classifiers that cannot use negative numbers, such as Multinomial NB, we did the above normalization and then rescaled the data in the range of $[0, 1]$. This was accomplished with scikit-learn's RobustScaler and MinMaxScaler, respectively [29].

## *8.3. Classifier Training and Evaluation*

For each classifier – data view pair we used 10-fold Cross Validation across the entire training set of 433 data points from both the official training set and the additional annotated_by_1 set, using a seeded stratified method provided by scikit-learn. For evaluating the pipelines, we performed a 10-fold Train & Predict using the best hyper-parameters from the previous step across the organizer-recommended 325 training files. These predictions were eventually fed into the Ensemble Creation procedures.

Of the 154 total runs, we eventually chose 20 best learners to use in the next step: ensemble training. Of these 20 runs, 10 runs shown in Table 11

| RF-reg-full | Pred NONE | Pred MILD | Pred MODERATE | Pred SEVERE |
|---|---|---|---|---|
| True NONE | 13 | 28 | 4 | 0 |
| True MILD | 3 | 94 | 33 | 0 |
| True MODERATE | 0 | 16 | 60 | 6 |
| True SEVERE | 0 | 4 | 49 | 15 |
| Lin-SVM-chi2-best | Pred NONE | Pred MILD | Pred MODERATE | Pred SEVERE |
| True NONE | 28 | 13 | 4 | 0 |
| True MILD | 11 | 103 | 13 | 3 |
| True MODERATE | 1 | 18 | 41 | 22 |
| True SEVERE | 1 | 2 | 21 | 44 |
| RBF-SVR-mig-best | Pred NONE | Pred MILD | Pred MODERATE | Pred SEVERE |
| True NONE | 17 | 25 | 3 | 0 |
| True MILD | 10 | 90 | 30 | 0 |
| True MODERATE | 1 | 26 | 43 | 12 |
| True SEVERE | 1 | 1 | 47 | 19 |
| D&W-num | Pred NONE | Pred MILD | Pred MODERATE | Pred SEVERE |
| True NONE | 27 | 17 | 1 | 0 |
| True MILD | 13 | 84 | 25 | 8 |
| True MODERATE | 3 | 17 | 41 | 21 |
| True SEVERE | 1 | 4 | 20 | 43 |
| SVM-Init-Ada-CARS | Pred NONE | Pred MILD | Pred MODERATE | Pred SEVERE |
| True NONE | 30 | 2 | 13 | 0 |
| True MILD | 9 | 96 | 21 | 4 |
| True MODERATE | 1 | 16 | 58 | 7 |
| True SEVERE | 0 | 2 | 14 | 52 |
| MNB-CARs | Pred NONE | Pred MILD | Pred MODERATE | Pred SEVERE |
| True NONE | 31 | 14 | 0 | 0 |
| True MILD | 16 | 103 | 3 | 8 |
| True MODERATE | 5 | 33 | 34 | 10 |
| True SEVERE | 1 | 6 | 5 | 56 |

Table 12: Individual Confusion Matrices on the 325 document training set for the 6 Classifiers in the Competition-Winning Ensemble.

| RF-reg-full | Precision | Recall | MAE |
|---|---|---|---|
| NONE | 0.812 | 0.289 | 0.733 |
| MILD | 0.662 | 0.723 | 0.824 |
| MODERATE | 0.411 | 0.732 | 0.809 |
| SEVERE | 0.714 | 0.221 | 0.738 |
| **Lin-SVM-chi2-best** | **Precision** | **Recall** | **MAE** |
| NONE | 0.683 | 0.622 | 0.844 |
| MILD | 0.757 | 0.792 | 0.885 |
| MODERATE | 0.519 | 0.500 | 0.744 |
| SEVERE | 0.638 | 0.647 | 0.863 |
| **RBF-SVR-mig-best** | **Precision** | **Recall** | **MAE** |
| NONE | 0.586 | 0.378 | 0.757 |
| MILD | 0.634 | 0.692 | 0.787 |
| MODERATE | 0.350 | 0.524 | 0.745 |
| SEVERE | 0.613 | 0.279 | 0.739 |
| **D&W-num** | **Precision** | **Recall** | **MAE** |
| NONE | 0.614 | 0.600 | 0.859 |
| MILD | 0.689 | 0.646 | 0.793 |
| MODERATE | 0.471 | 0.500 | 0.732 |
| SEVERE | 0.597 | 0.632 | 0.848 |
| **SVM-Init-Ada-CARS** | **Precision** | **Recall** | **MAE** |
| NONE | 0.750 | 0.667 | 0.793 |
| MILD | 0.828 | 0.738 | 0.854 |
| MODERATE | 0.547 | 0.707 | 0.848 |
| SEVERE | 0.825 | 0.765 | 0.912 |
| **MNB-CARs** | **Precision** | **Recall** | **MAE** |
| NONE | 0.585 | 0.689 | 0.896 |
| MILD | 0.660 | 0.792 | 0.866 |
| MODERATE | 0.810 | 0.415 | 0.677 |
| SEVERE | 0.757 | 0.824 | 0.902 |

Table 13: Multi-class Precision, Recall and MAE on the 325 document training set for the 6 Classifiers in the Competition-Winning Ensemble. Per-class MAE is normalized with the assumption that all predictions are maximally incorrect for each class.

| Classifier | MA-MAE | ROC-AUC | $R^2$ |
|---|---|---|---|
| RF-reg-full | 0.776 | 0.683 | 0.554 |
| Lin-SVM-chi2-best | 0.834 | 0.765 | 0.521 |
| RBF-SVR-mig-best | 0.757 | 0.657 | 0.476 |
| D&W-num | 0.808 | 0.721 | 0.394 |
| SVM-Init-Ada-CARS | 0.851 | 0.811 | 0.515 |
| MNB-CARs | 0.835 | 0.773 | 0.459 |

Table 14: Summary metrics on the 325 document training set for each of the 6 Classifiers in the Competition-Winning Ensemble. All applicable metrics are macro-averaged when necessary. Higher is better.

| Classifier | Precision | Recall | F1-Score | Accuracy |
|---|---|---|---|---|
| RF-reg-full | 0.650 | 0.491 | 0.495 | 0.560 |
| Lin-SVM-chi2-best | 0.649 | 0.640 | 0.644 | 0.665 |
| RBF-SVR-mig-best | 0.546 | 0.468 | 0.481 | 0.520 |
| D&W-num | 0.593 | 0.595 | 0.593 | 0.600 |
| SVM-Init-Ada-CARS | 0.738 | 0.719 | 0.724 | 0.726 |
| MNB-CARs | 0.703 | 0.680 | 0.673 | 0.689 |

Table 15: Summary metrics on the 325 document training set for each of the 6 Classifiers in the Competition-Winning Ensemble. All applicable metrics are macro-averaged when necessary. Higher is better.

participated in the five *best* ensembles (see Section 9.) For each run, we show the classifier used, the data view on which it ran and the abbreviation we use in this section and in Section 9 to refer to it.

For the sake of space, we limit the demonstration and discussion of the results of the individual classifiers to the six classifiers from Table 11 which constituted our top performing classification ensemble. These classifiers are:

1. RF-ref-full: the Random Forest regression run on the Full data view.
2. Lin-SVM-chi2-best: Linear kernel SVM classifier run on the 50 best features selected by the $\chi^2$ test.
3. RBF-SVR-mig-best: Radial basis function kernel SVM regressor run on the top 50 best features selected by the mutual information gain test.
4. D&W-num: the TensorFlow's Deep and Wide classifier run on the numeric and categorical features.
5. SVM-Init-Ada-CARs: SVM-initialized Adaboost running on our CAR features.
6. MNB-CARs: Multinomial Naïve Bayes running on our CAR features.

Table 12 contains the confusion matrices for these six runs, Table 13 shows precision, recall and MAE for each class, while Tables 14 and 15 document the overall accuracy metrics: MA-MAE, RoC-AUC and the $R^2$ metrics (Table 14), and precision, recall, f-score and accuracy (Table 15). We discuss the work of individual classifiers below.

Our RandomForest regression run on the full data view (RF-reg-full) tended to over-predict MILD and MODERATE valences at the expense of NONE and SEVERE, however, it contained excellent separation between the NONE/MILD, and MODERATE/SEVERE pairs of valences, with only MILD$\Rightarrow$MODERATE false positives being of concern. While this run had the second lowest MA-MAE value out of our six runs, it should be noted (see Section 9) that this is *the only* run that participated in all final ensembles.

The linear kernel SVM classifier running on our top 50 $\chi^2$ features (Lin-SVM-chi2-best) has the third highest MA-MAE and has produced an excellent confusion matrix, with majority of NONE and SEVERE conditions being classified correctly, and with very few "costly" misses.

The SVM regressor with RBF kernel running on our top 50 Mutual Information Gain features (RBF-SVR-mig-best) had the lowest performance of these six runs (although was still among the better classifiers overall). It over-predicted the MODERATE class, and had some trouble distinguishing MODERATE and MILD valences. It also was very strict at predicting NONE and SEVERE valences.

TensorFlow's Deep and Wide classifier, run on all our numeric and categorical attributes, excluding CAR and Word2Vec attributes ((D&W-num) had no significant distinctive features as compared to other runs. It did the worst on properly capturing MILD valences (MILD recall), and tended to admit more "big" mistakes (misclassifications two or more classes apart) than some other methods. But it kept the overall number of misclassified cases reasonable, and hence earned a MA-MAE in excess of 0.8.

Our overall best single run came from our own Adaboost classifier trained on a single round of SVM followed by 49 rounds of Naïve Bayes applied to the dataset consisting solely of CAR attributes (SVM-Init-Ada-CARs, see Section 8.1). This classifier excelled almost everywhere, giving by far the most accurate predictions of SEVERE valence and minimizing false positives. The only "weak spot" for this method came from improperly classifying 13 cases with valence of NONE as MODERATE. However, as this was a clear outlier prediction among our six runs (the other runs predicted anywhere from 0 to 4 cases this way), this miss was effectively eliminated in the followup ensembles.

The final classifier run, Multinomial Naïve Bayes run on the same data view of CAR attributes (MNB-CARs), edged the Lin-SVM-chi2-best run by a hair to give us our second best single run MA-MAE of 0.835. It got the largest number of both NONE and SEVERE true positives, as well as tying the Lin-SVM-chi2-best for the largest number of MILD true positives, only stumbling a bit on the MODERATE valence, where it had a very high precision, but low recall.

## 9. Step 6: Ensemble Learning

From our prior research [27, 2], we know that ensembles frequently beat vanilla classifiers. As a consequence, we decided to try out ensembles on our data. From our set of 154 classifier data view runs, we selected the 20 best runs (six of which were presented in detail in Section 8). We constructed a variety of ensembles of size 2 to 9 classifiers in each from these runs, and via attrition zeroed in on the best performing ones. Our measure of performance of an ensemble was straightforward:

> *the MA-MAE of the ensemble must be higher than* 0.851*, the MA-MAE of our best standalone method* (SVM-Init-Ada-CARS).

Our classifier ensembles were constructed in a straightforward way. Each ensemble consisted of a subset of classifiers from our list of 20 best runs. Each classifier in the ensemble received an equal vote share (i.e., we did not attempt to weigh classifiers differently). We devised six different voting schemes to determine the ensemble prediction of the Valence based on the predictions of the constituent classifiers. These voting schemes are defined below.

*Majority voting.* A value of the Valence is selected if it was predicted by the majority (at least half) of classifiers in the ensemble. If such value does not exist, this method picks the *most common* Valence in the training set[5].

*Plurality voting.* Given a parameter *min_votes*, a specific value of the Valence is selected if it is the most common value and more than *min_votes* classifiers in the ensemble predict it. If such value does not exist, this method picks the *most common* Valence in the training set.

*Majority_favor_MODERATE voting.* This scheme selects the majority value of predicted Valence if one exists, the same way the *majority* scheme works. However, if a *majority* value does not exist, this voting scheme favors Valence = MODERATE: it selects this value if *at least one* classifier predicts it. If no classifier predicts Valence = MODERATE, this scheme defaults to the most common Valence in the training set.

*Plurality_favor_MODERATE voting.* This scheme selects the plurality value of Valence if it is predicted by more than *min_votes* votes. If such value does not exist, but at least one classifier predicts Valence = MODERATE, this scheme selects this value. If no Valence = MODERATE prediction exists in the ensemble, the voting scheme defaults to the most common Valence in the training set.

*Simple Round voting.* This voting scheme simply finds the average prediction value among the ensemble classifiers, and rounds it to the nearest Valence value.

*Tuned Round voting.* Our most complicated voting scheme acts like another hyper-parametrization search.

The general idea behind this method is fairly straightforward, although the implementation does require some explanation. The *simple round voting* scheme assumes that the average valence of 2.6 (out of 3) points to the class Valence = SEVERE, as 2.6 is greater than the midpoint between the numeric Valence scores for MODERATE and SEVERE classes. However, Valence (despite how we choose to treat it on occasion) is **not** a continuous variable, but an ordinal one. Therefore, 0.5, 1.5 and 2.5 *do not have to be* the threshold values separating the neighboring valence classes. What should these values be? Well, we can treat this as yet another hyperparameter tuning problem, and find such values of the three threshold parameters that optimize the MA-MAE score.

For our experiments we used the following threshold sets, which give rise to a search space of 343 possibilities.

- NONE/MILD threshold: 0.25, 0.3, 0.4, 0.5, 0.6, 0.7, 0.75
- MILD/MODERATE threshold: 1.25, 1.3, 1.4, 1.5, 1.6, 1.7, 1.75
- MODERATE/SEVERE threshold: 2.25, 2.3, 2.4, 2.5, 2.6, 2.7, 2.75

[5]In our training set, this was Valence=MILD.

To run all our ensembles through this voting mechanism we would have to generate in excess of 13.3 million combinations. To achieve this, the *tuned round voting* ensemble algorithm was parallelized onto a c4.4xlarge Cloud Instance on Amazon AWS. In addition, the mean votes of each of the 38,760 candidate ensembles were pre-computed using OpenBLAS [47]. Nonetheless, the entire computation took 8 hours with classifier ensembles of no larger than 6, whereas *all* Majority and Plurality schemes up to 9 completed on a single core in less time.

## *9.1. Submitted Ensembles*

Among the multitude of voting ensembles, we selected the five top performers (all providing us with 1.5 – 3% of lift over the best individual classifier) shown in Table 16. Notably, all these ensembles used the *tuned voting* ensemble voting, confirming to us that the tuning of the thresholds separating the neighboring Valence classes was a useful procedure. The MA-MAE scores reported in Table 16 were computed over the 325-record training set.

As we could only submit three guesses, we had to make our final choices from these five ensembles. *We selected ensembles **A**, **B** and **C** for official submission.*

Ensemble B consisted only of the Random Forest regressor run on the full data view, and our most accurate standalone classifier, SVM-initialized Naïve Bayes AdaBoost on the Class Association Rules data view. It also was the best performer on the training set.

Ensembles A and C were selected to diversify our pool of guesses. Ensemble A was selected as the most accurate ensemble *that did not feature classifiers trained on Class Association Rules alone.* We chose Ensemble C over Ensemble E because in a secondary run on the 108 annotated_by_1 records Ensemble E had a drop in accuracy that worried us. Additionally, Ensemble C was far better than the other ensembles in properly recognizing the Valence = SEVERE class.

Table 17 shows the confusion matrices of the three submitted ensembles on the 325-record training set. Table 19 shows the precision, recall and MAE for each Valence class for each ensemble. Table 19 shows the MA-MAE as well as the ROC-AUC and the $R^2$ metrics. Table 20 shows overall precision, recall, f-measure and accuracy of the ensembles.

## *9.2. Test Results*

The results of running our three submitted ensembles on the 216-record test set are shown in Tables 21—24. Table 21 shows the confusion matrices of the three ensembles on the test set.

It is worth immediately noting, by comparing confusion matrices in Table 21 to those in Table 17 (for the training set), that all three ensembles overall performed as expected and did not overfit the training set by much. We note that all three ensembles excelled at *not making huge mistakes*: they had 10, 11, and 7 data points (respectively) with classification error of 2 or more. This compares with 7, 5, and 8 such data points misclassified on (somewhat larger) training set.

| Name | Classifiers | Voting | MA-MAE |
|---|---|---|---|
| A | RF-full<br>RF-reg-clf-full<br>RF-Reg-full<br>Lin-SVM-chi2-best<br>Lin-SVM-anova-best<br>DNN-full | Tuned round (0.7,1.6,2.25) | 0.865 |
| B | RF-reg-full<br>SVM-Init-Ada-CARS | Tuned round (0.7,1.7,2.3) | 0.882 |
| C | RF-Reg-full<br>Lin-SVM-chi2-best<br>RBF-SVR-mig-best<br>D&W-num<br>SVM-Init-Ada-CARS<br>MNB-CARs | Tuned round (0.75,1.5,2.25) | 0.865 |
| D | RF-reg-full<br>Lin-SVM-chi2-best<br>Lin-SVM-ANOVA<br>DNN-full | Tuned round (0.5,1.3,2.3) | 0.850 |
| E | RF-Reg-full<br>SVM-Init-Ada-CARS<br>DNN-full<br>Lin-SVM-chi2-best | Tuned round (0.7,1.4,2.4) | 0.866 |

Table 16: Top five voting ensembles. The MA-MAE value is computed on the 325-record training set.

| Ensemble A | Pred NONE | Pred MILD | Pred MODERATE | Pred SEVERE |
|---|---|---|---|---|
| NONE | 34 | 10 | 1 | 0 |
| MILD | 9 | 102 | 17 | 2 |
| MODERATE | 1 | 15 | 48 | 18 |
| SEVERE | 0 | 3 | 19 | 46 |
| ENSEMBLE B | Pred NONE | Pred MILD | Pred MODERATE | Pred SEVERE |
| NONE | 29 | 15 | 1 | 0 |
| MILD | 8 | 116 | 4 | 2 |
| MODERATE | 1 | 20 | 54 | 7 |
| SEVERE | 0 | 1 | 20 | 47 |
| ENSEMBLE C | Pred NONE | Pred MILD | Pred MODERATE | Pred SEVERE |
| NONE | 31 | 14 | 0 | 0 |
| MILD | 10 | 107 | 10 | 3 |
| MODERATE | 2 | 21 | 42 | 17 |
| SEVERE | 0 | 3 | 10 | 55 |

Table 17: Confusion Matrices on the 325 document training set for the 3 submitted Ensembles

| ENSEMBLE A | Precision | Recall | MAE |
|---|---|---|---|
| NONE | 0.773 | 0.756 | 0.911 |
| MILD | 0.785 | 0.785 | 0.885 |
| MODERATE | 0.565 | 0.585 | 0.787 |
| SEVERE | 0.697 | 0.676 | 0.877 |
| **ENSEMBLE B** | **Precision** | **Recall** | **MAE** |
| NONE | 0.763 | 0.644 | 0.874 |
| MILD | 0.763 | 0.892 | 0.939 |
| MODERATE | 0.684 | 0.659 | 0.823 |
| SEVERE | 0.839 | 0.691 | 0.892 |
| **ENSEMBLE C** | **Precision** | **Recall** | **MAE** |
| NONE | 0.721 | 0.689 | 0.896 |
| MILD | 0.738 | 0.823 | 0.900 |
| MODERATE | 0.677 | 0.512 | 0.744 |
| SEVERE | 0.733 | 0.809 | 0.922 |

Table 18: Multi-class Precision, Recall and MAE on the 325 document training set for the 3 submitted Ensembles. Per-class MAE is normalized with the assumption that all predictions are maximally incorrect for each class.

| Ensemble | MA-MAE | ROC-AUC | $R^2$ |
|---|---|---|---|
| A | 0.865 | 0.795 | 0.622 |
| B | 0.882 | 0.823 | 0.694 |
| C | 0.865 | 0.801 | 0.629 |

Table 19: Summary metrics on the 325 document training set for each of the 3 submitted Ensembles. All applicable metrics are macro-averaged when necessary. Higher is better.

| Ensemble | Precision | Recall | F1-Score | Accuracy |
|---|---|---|---|---|
| A | 0.705 | 0.701 | 0.703 | 0.708 |
| B | 0.762 | 0.722 | 0.738 | 0.757 |
| C | 0.717 | 0.708 | 0.709 | 0.723 |

Table 20: Summary metrics on the 325 document training set for each of the 3 submitted Ensembles. All applicable metrics are macro-averaged when necessary. Higher is better.

| ENSEMBLE A | Pred NONE | Pred MILD | Pred MODERATE | Pred SEVERE |
|---|---|---|---|---|
| NONE | 20 | 10 | 1 | 0 |
| MILD | 12 | 66 | 5 | 3 |
| MODERATE | 2 | 16 | 23 | 5 |
| SEVERE | 1 | 3 | 10 | 39 |
| ENSEMBLE B | Pred NONE | Pred MILD | Pred MODERATE | Pred SEVERE |
| NONE | 20 | 11 | 0 | 0 |
| MILD | 4 | 68 | 13 | 1 |
| MODERATE | 2 | 19 | 21 | 4 |
| SEVERE | 0 | 8 | 7 | 38 |
| ENSEMBLE C | Pred NONE | Pred MILD | Pred MODERATE | Pred SEVERE |
| NONE | 21 | 10 | 0 | 0 |
| MILD | 7 | 64 | 13 | 2 |
| MODERATE | 2 | 12 | 29 | 3 |
| SEVERE | 0 | 3 | 9 | 41 |

Table 21: Confusion Matrices on the 216 document hidden test set for the 3 submitted Ensembles

| ENSEMBLE A | Precision | Recall | MAE |
|---|---|---|---|
| NONE | 0.571 | 0.645 | 0.871 |
| MILD | 0.695 | 0.767 | 0.867 |
| MODERATE | 0.590 | 0.500 | 0.739 |
| SEVERE | 0.830 | 0.736 | 0.881 |
| ENSEMBLE B | Precision | Recall | MAE |
| NONE | 0.769 | 0.645 | 0.882 |
| MILD | 0.642 | 0.791 | 0.890 |
| MODERATE | 0.512 | 0.457 | 0.707 |
| SEVERE | 0.884 | 0.717 | 0.855 |
| ENSEMBLE C | Precision | Recall | MAE |
| NONE | 0.700 | 0.677 | 0.892 |
| MILD | 0.719 | 0.744 | 0.861 |
| MODERATE | 0.569 | 0.630 | 0.794 |
| SEVERE | 0.891 | 0.774 | 0.906 |

Table 22: Multi-class Precision, Recall and MAE on the 216 document hidden test set for the 3 submitted Ensembles. Per-class MAE is normalized with the assumption that all predictions are maximally incorrect for each class.

| Ensemble | MA-MAE | ROC-AUC | $R^2$ |
|---|---|---|---|
| A | 0.837 | 0.776 | 0.534 |
| B | 0.833 | 0.763 | 0.539 |
| C | 0.863 | 0.799 | 0.629 |

Table 23: Summary metrics on the 216 document hidden test set for each of the three submitted Ensembles. All applicable metrics are macro-averaged when necessary. Higher is better.

| Ensemble | Precision | Recall | F1-Score | Accuracy |
|---|---|---|---|---|
| A | 0.671 | 0.662 | 0.664 | 0.685 |
| B | 0.702 | 0.652 | 0.671 | 0.681 |
| C | 0.720 | 0.706 | 0.712 | 0.718 |

Table 24: Summary metrics on the 216 document hidden test set for each of the three submitted Ensembles. All applicable metrics are macro-averaged when necessary. Higher is better.

All three ensembles exhibited very similar performance on the 31 records with Valence = NONE. Ensemble A tended to underrate MILD cases, preferring to predict Valence = NONE when it made a mistake. Ensembles B and C went in the other direction, overrating the majority of mistakes on cases with Valence=MILD.

It is on cases with Valence=MODERATE and Valence=SEVERE Ensemble C showed a clearly better performance, both in terms of recall (correctly classifying 29 out of 46 MODERATE cases and 41 out of 53 SEVERE cases), and in terms of precision (keeping it above 50% for Valence=MODERATE, and allowing for only 5 false positives for Valence=SEVERE). These numbers, especially the precision for the Valence=SEVERE class wound up actually being better than the training set results (where Ensemble C had 20 false positives and 55 true positives in this)!

Table 22 shows precision, recall and MAE for each valence class for each ensemble. Table 23 shows the overall MA-MAE, ROC-AUC and $R^2$ metrics for each ensemble, while Table 24 summarizes precision, recall, f1-score and accuracy.

As seen from Table 22, Ensemble C wound up being the top scorer among our submissions. As we learned shortly after the submission period closed, Ensemble C *wound up being the best overall predictor of patient condition severity* in the entire CEGS N-GRID 2016 Shared Task in Clinical Natural Language Processing (Track 2).

## 10. Conclusion and Future Work

The CREATEframework we built for Track 2 of the CEGS N-GRID 2016 Shared Task in Clinical Natural Language Processing introduces a number of novel features in the field of automated analysis of medical records. The core novel features of CREATE that proved to be crucial to our success included:

- **Enhanced features.** An aggressive approach to enhancing the initial patient evaluation records provided to us with a multitude of features from diverse sources. Almost all of our feature enhancement efforts contributed non-trivial amounts of features to the final feature set. In addition to traditional features used for medical data analysis, such as diagnosis signals and sentiment, we have added novel categories of features: cumulative scores, commonality features, and medication use features, that proved important.

- **Use of Class Association Rules as features.** Class Association Rules are often used by themselves to classify underlying data. In CREATE we "stacked" the learning processes by using a set of CARs with complete five-fold[6] coverage of our training set as *additional features* in our dataset, and using both the CARs-only and combined feature sets in subsequent classification and regression tasks.

- **Feature Pruning and Data View construction.** Our aggressive battery of feature pruning tests eliminated redundant or useless features. In addition, rather than using the full set of features for each classification tasks, we attempted to zero in on useful subsets of the features, either by feature type (all CAR features, all non-CAR features) or by the scores assigned to them by some of our pruning tests (features with highest $\chi^2$, mutual information gain, ANOVA $F-value$). Separation of our data into these data views allowed us to better train our classifiers: the winning ensemble of six classifiers used four out of seven data views. The fact that some of the classifiers in the ensemble were trained on disjoint sets of features helped prevent overfit in the ensemble.

- **Adaptations of classifiers.** We adapted two classifiers to better work with the data. The Random Forest regressor with classification inference adaptation was made specifically to account for the nature of the target Valence class attribute and resulted in improved performance of the Random Forest classifier. This regressor was featured in one of the three of our final submissions. The SVM-Initialized Adaboost by far outperformed all individual classifiers, and featured prominently in our winning ensemble.

- **Tuned Round Voting scheme for ensembles of classifiers.** While our classifier ensembles were formed in a simple way by giving each classifier an equal vote in each outcome, the *tuned round* voting scheme for deciding the results of the vote, *which was featured in all five best classifier ensembles* was the third "stacked" learner in CREATE: it performed the hyper-parameter tuning to determine the best way to separate averaged (and therefore no longer integer) values between neighboring Valence classes. As seen from Table 16, the class thresholds learned by this method were different than the default values in almost all cases, which, by virtue of the method, improved the final accuracy of the ensembles.

*Future Work.* Word2Vec and other emerging text embedding NLP strategies have gained a large amount of notariety since the release of TensorFlow. Although Google's GoogleNews vectors worked well, utilizing PubMed's massive database of medical text would be a more domain-aware embedding strategy and training our own PubMedWordVectors would likely increase the amount of topic coherency. A second area of improvement is using of deep learning algorithms such as LSTMs from TensorFlow in an attempt to find convoluted, non-linear feature interactions.

[6]Meaning that each record in the training set was covered by at least five discovered Class Association Rules.

Finally, the CREATE framework currently exists purely offline and is driven by a command line interface. Developing a more user-friendly and automated pipeline would allow SentiMetrix to more easily extend the applicability of the framework to a larger domain of medical records analysis, as well as to any data analytical tasks that involve large combined structured data and textual data feature sets.

## Acknowledgements

Our team's work was supported by US Army Medcom/TATRC grant W81XWH-13-C-0030. The CEGS N-GRID 2016 Shared Task in Clinical Natural Language Processing was sponsored in part by two NIH awards: NIH P50 MH106933 (PI: Isaac Kohane) and NIH 4R13LM011411 (PI: Ozlem Uzuner).

## Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D.G. Murray, B. Steiner, P.A. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, X. Zheng (2016) TensorFlow: A System for Large-Scale Machine Learning. in *Proc. OSDI 2016*, pp. 265–283

[2] B. An, H. Chen, N. Park, and VS Subrahmanian. (2016) MAP: Frequency-Based Maximization of Airline Profits based on an Ensemble Forecasting Approach, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 421–430,

[3] R.A. Baeza-Yates, B. Ribeiro-Neto. (1999) *Modern Information Retrieval*, Addison Wesley.

[4] S. Banaszak, V. Kagan, A. Stevens, and V.S. Subrahmanian. (2013). COP-TADS:Clinical Online PTSD and TBI Analysis and Detection System, in *Proc. Workshop on Visual Analytics in Healthcare'2013*, pp. 86-89.

[5] J. C Bezdek, R. J. Hathaway, R.E. Howard, C.A Wilson, M.P Windham. (1987). Local convergence analysis of a grouped variable version of coordinate descent, in *Journal of Optimization Theory and Applications*, pp. 471–477.

[6] S. Bird, E. Klein, E. Loper. (2009) Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit, *O'Reilly Media*, June 2009.

[7] R. Boulton, M. Porter. (2001) Snowball. http://snowballstem.org.

[8] B.E.P. Box. (1953). Non-Normality and Tests on Variances. *Biometrika*, Biometrika Trust, Vol. 40 (3/4), pp. 318-–33

[9] L. Breiman. (2001). Random Forests. *Machine Learning*, Vol. 45 (1): pp. 5-–32.

[10] Eli Bressert. (2012 )SciPy and NumPy: Examples to Jumpstart your Scientific Python Programming, *O'Reilly Media*.

[11] T. Chen, C. Guestrin (2016) XGBoost: A Scalable Tree Boosting System, in *Proc. KDD 2016*, pp. 785–794

[12] E. Choi, M.T. Bahadori, E. Searles, C. Coffey, M. Thompson, J. Bost, J. Tejedor-Sojo, J. Sun. (2016). Multi-layer representation learning for medical concepts in *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (ACM 2016)*. pp. 1495–1504.

[13] C. Cortes, V. Kuznetsov, M. Mohri. Learning Ensembles of Structured Predictions Rules in *Proceedings of the 52nd Annual Meeting of Association for Computational Linguistics (ACL 2014)*. pp. 1–12.

[14] C. Cortes, V. Vapnik (1995) Support-Vector Networks, *Machine Learning* Vol. 20(3), pp. 273–297

[15] J. Dickerson, V. Kagan and V.S. Subrahmanian. (2014) Using Sentiment to Detect Bots on Twitter: Are Humans more Opinionated than Bots?, in *Proc. ACM/IEEE Intl. Conf. on Advances in Social Network Analysis and Mining (ASONAM'2014)*, Beijing, pp. 620-627, Aug. 2014 (industrial papers session).

[16] Y. Freund, and R. E. Schapire. (1999) A Short Introduction to Boosting, *Journal of Japanese Society for Artificial Intelligence*, Vol. 14(5):771-780, September, 1999.

[17] G. H. Golub, R. Christian. (1970). Singular value decomposition and least squares solutions in *Numerische mathematik 14.5*. pp. 403-420.

[18] J. Han, H. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation. In *Proc. Conf. on the Management of Data* (SIGMOD'00, Dallas, TX). ACM Press, New York, NY, USA 2000.

[19] T.K. Ho (1995). Random Decision Forests. In *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, Montreal, QC, 14–16 August 1995. pp. 278–282.

[20] V. Kagan, E. Rossini, and D. Sapounas. (2013) *Sentiment analysis for PTSD signals.* New York: Springer Science + Business Media.

[21] V. Kagan, A. Stevens, and V. S. Subrahmanian (2015) Using Twitter Sentiment to Forecast the 2013 Pakistani Election and the 2014 Indian Election. *IEEE Intelligent Systems* Vol. 30(1): pp. 2—5

[22] R. W. Hamming. (1950). Error detecting and error correcting codes in *Bell System Technical Journal*. Vol 29 (2): 147–160.

[23] W. Li, J. Han, J. Pei (2001) CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules. *IEEE Computer Society* ICDM '01: pp. 369–376

[24] B. Liu, W. Hsu, Y. Ma (1998) Integrating Classification and Association Rule Mining. *Association for the Advancement of Artifical Intelligence* KDD: 98 Proceedings: pp. 80–86

[25] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean (2013) Distributed Representations of Words and Phrases and their Compositionality. in *Proc. NIPS 2013* pp. 3111–3119.

[26] T. Mikolav, Q. Le (2014) Distributed Representations of Sentences and Documents. https://cs.stanford.edu/ quocle/paragraph_vector.pdf

[27] C.Kang, N. Park,B.A. Prakash, E. Serra, and VS Subrahmanian. (2016) Ensemble Models for Data-driven Prediction of Malware Infections, in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pp. 583–592.

[28] E. Naeseth (2009) Python FP-Growth *Github*. MIT Licence. https://github.com/enaeseth/python-fp-growth

[29] F.V. Pedregosa. (2011) Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825-2830.

[30] J.W. Pennebaker. (2001) Linguistic Inquiry and Word Count: LIWC 2001. *Lawrence Erlbaum Associates.*

[31] J. Pennington, R. Socher, C. D. Manning. (2014). Glove: Global Vectors for Word Representation in *EMNLP*. Vol: 14. pp. 1532–1543.

[32] J.R. Quinlan. (1992) *C4.5 Programs for Machine Learning*, Morgan Kaufmann.

[33] J.R. Quinlan (2010) Rulequest Research Data Mining Tools. http://www.rulequest.com

[34] R. Rehurekm, P. Sojka. (2010). Software Framework for Topic Modelling with Large Corpora in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks.* pp. 45–50.

[35] K. Rajeswari. Feature Selection by Mining Optimized Association Rules based on Apriori. (2015) in *International Journal of Computer Applications.* 119(20).

[36] B.C. Ross. (2014) Mutual Information between Discrete and Continuous Data Sets. *PLoS ONE*, Volume 9(2): e87357.

[37] D.L. Roter. (2011) The Expression of Emotion Through Nonverbal Behavior in Medical Visits: Mechanisms and Outcomes. *Journal of General Internal Medicine*, 21(Suppl 1), S28–S34.

[38] G.K. Savova, J.J. Masanz, Ph.V. Ogren, J. Zheng, S. Sohn, K.C. Kipper-Schuler, C.G. Chute. (2010) Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association* , 17(5), pp. 507-513.

[39] R. Socher, J. Bauer, C.D. Manning and A.Y. Ng. (2013) Parsing With Compositional Vector Grammars. in *Proceedings of ACL'2013.*

[40] V. S. Subrahmanian, A. Azaria, S. Durst, V. Kagan, A. Galstyan, K. Lerman, L. Zhu, E. Ferrara, A. Flammini, F. Menczer, R. Waltzman, A. Stevens, A. Dekhtyar, S. Gao, T. Hogg, F. Kooti, Y. Liu, O. Varol, P. Shiralkar, V. G. Vinod Vydiswaran, Q. Mei, T. Huang. (2016) The DARPA Twitter Bot Challenge. *IEEE Computer*, Issue No. 06, June 2016, vol. 49, pp. 38–46.

[41] K. Sparck Jones. (1972). A Statistical Interpretation of Term Specificity and Its Application in Retrieval. *Journal of Documentation*, Vol 28, pp. 11–21

[42] V.S. Subrahmanian, S. Venkatramana, D. Reforgiato. (2008). AVA: Adjective-verb-adverb combinations for sentiment analysis in *IEEE Intelligent Systems*. Vol 23(4). pp. 43–50.

[43] O. Uzuner, A. Stubbs, M. Filannino, T. Cai, S. Churchill, I. Kohane, Th.H. McCoy, R.H. Perlis, P. Szolovits, U. Vaidyanathan, Ph. Wang. (2016) Announcement of Data Release and Call for Participation 2016 CEGS N-GRID Shared-Tasks and Workshop on Challenges in Natural Language Processing for Clinical Data, https://www.i2b2.org/NLP/RDoCforPsychiatry/.

[44] S. Walt, S. Colbert, G. Varoquaux The NumPy Array: A Structure for Efficient Numerical Computation *IEEE Computer Society* in *Computing in Science & Engineering* Vol 13(2), pp 22–30

[45] X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A.F.M. Ng, B. Liu, Ph.S. Yu, Z-H. Zhou, M. Steinbach, D.J. Hand, D. Steinberg. (2008) Top 10 algorithms in data mining. *Knowl. Inf. Syst.* Vol 14(1), pp 1–37.

[46] F. Yates. (1934) Contingency table involving small numbers and the $\chi^2$ test. Supplement to *Journal of the Royal Statistical Society* Vol 1(2): 217–235.

[47] Z. Xianyi, W. Qian, W. Saar OpenBLAS: An optimized BLAS library http://www.openblas.net/

[48] Y. Xu, G.J.F. Jones, JT. Li, B. Wang, and CM. Sun. (2007) A study on mutual information-based feature selection for text categorization *Journal of Computational Information Systems*, Vol. 3 (3). pp. 1007–1012.