2020

Predicting Drug Misuse Status Using Machine Learning on Electronic Health Records

Robert Arnold Kania

LOYOLA UNIVERSITY CHICAGO

PREDICTING DRUG MISUSE STATUS USING MACHINE LEARNING ON ELECTRONIC HEALTH RECORDS

A THESIS SUBMITTED TO

THE FACULTY OF THE GRADUATE SCHOOL

IN CANDIDACY FOR THE DEGREE OF

MASTER OF SCIENCE

PROGRAM IN COMPUTER SCIENCE

BY

ROBERT KANIA

CHICAGO, IL

MAY 2020

If I have seen further, it is by standing upon the shoulders of giants

– Sir Isaac Newton

ACKNOWLEDGEMENTS

Thank you to Dr. Dmitriy Dligach for giving me the opportunity to learn about natural language processing, and for his guidance and mentorship throughout this project. Thank you to Dr. Majid Afshar for his support throughout the project, helping me better understand the problem domain and verifying the assumptions. Thank you to Dr. George K. Thiruvathukal for his encouragement and valuable remarks. Thank you to my parents and my family for their support, encouragement, and inspiring me to pursue opportunities I did not know existed. Finally, I would like to thank all my friends and colleagues for their support throughout this journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AUC	Area under curve
CUI	Concept Unique Identifiers
CNN	Convolutional Neural Network
EHR	Electronic Health Records
LR	Learning rate
ML	Machine Learning
NLP	Natural Language Processing
PR	Precision recall
ROC	Receiver operating characteristic
SVM	Support-vector machine
TF	TensorFlow
UMLS	Unified Medical Language System

ABSTRACT

Substance misuse is a major problem in the world. In 2014, as many as 52,404 deaths in the US were caused by drug overdoses. In 2001, the monetary cost of drug misuse has been estimated to be 414 billion dollars. In this work, we explore the use of different machine learning algorithms in the prediction of cocaine misuse using structured and unstructured data found in electronic health records. These records contain various attributes that can help with this prediction, including but not limited to chart text data, previous diagnoses of certain diseases and information about the area the patient lives in. We compare models which are trained on only one kind of data, ensembles of models trained on different kinds of data, and models which are trained on different representations of both kinds of data. Finally, the models are evaluated using the area under precision recall curve (PR AUC) metric, which is a suitable metric for imbalanced data sets. Early results show that the addition of structured data to processed chart notes can in some cases improve performance by up to three points.

CHAPTER ONE: INTRODUCTION

Substance misuse is a major problem in the world. In 2014, as many as 52,404 deaths in the US were caused by drug overdoses (Rudd et al. 2016). In 2001, the monetary cost of drug misuse has been estimated to be 414 billion dollars (Horgan et al. 2001). In this work, we explore the use of different machine learning algorithms in the prediction of cocaine misuse using structured and unstructured data found in electronic health records.

In recent years, there have been many advancements in the field of Natural Language Processing (NLP). NLP is a field of study which concerns itself with making computers understand human language, and building models which make decisions based on text. There are standardized benchmarks such as the GLUE ("GLUE Benchmark" n.d.) which among other tasks ask models to answer questions based on a paragraph of text that the model has had time to ingest. Recently, the top score has been improved upon several times. Although these models are trained to understand the English language in general, such models can also be applied to specific problem domains.

NLP has been proven to be a valuable tool in a plethora of tasks in the clinical domain. In particular, clinical NLP is useful when applied to drug misuse prediction and understanding. Examples include detecting opioid misuse in electronic health records (Carrell et al. 2015), or pinpointing drug misuse hotspots by running a classifier on social media posts stemming from specified regions (Sarker et al. 2019).

While there is concentrated effort to better understand text data, structured data pertaining the same subject is often overlooked. In the general domain, not every piece of text has corresponding numerical data to describe it. However, in the medical domain, clinicians collect all kinds of structured data pertaining a patient, from age and weight to specialized test results. In

this study, we analyze different methods of bringing together unstructured textual data with structured numeric data in conjunction to create a cocaine misuse classifier. The approach is not specific to cocaine misuse, but instead can be used to either predict presence of other drugs or an entirely distinct feature.

The preprocessing of both structured and unstructured data is discussed. Two different methods of unstructured text processing are employed, as neural networks require inputs different from traditional models. The data is split into a train and test set using the holdout method. The models are then tuned using either cross-validation or a validation set extracted from the train data. At each stage, the model with the highest PR AUC is chosen.

In the course of the study we establish separate baseline models for structured and unstructured data, that is models with the best performance on one of the two types of data. Following this, several different approaches for using both kinds of data are testes. Neural networks with multiple inputs, a CNN and a structured component are evaluated. Models which work on one kind of data are merged into one classifier in a technique called ensemble learning. Some traditional models which are suitable for concatenations of very different data are tested as well. Finally, all models are compared to the structured and unstructured baselines. The models are compared on basis of their performance in different metrics, as well as features of the respective models, such as required hardware, time to train, predict and the complexity of the model itself. The results show that using two different kinds of data does improve the performance of the classifiers, for some models by up to four PR AUC points.

CHAPTER TWO: METHODOLOGY

This section discusses the data used to train and evaluate the models, the preprocessing that has been performed, and the metrics and the method used to evaluate the models.

Data

Patient information is available in two different forms, unstructured text data and structured information about the patient. The text data contains information such as chart notes, consultations or test results. The structured data contains numerical data about a patient, such as their age, gender, admission date, diagnoses of certain diseases such as diabetes or depression, as well as as drug test results.

Data split

The data is first split into a train and test set. The proportion used is 80% train and 20% test. The method used is stratified splitting, resulting in a train and test dataset which have a similar proportion of positive and negative samples.

Table 1. Data Split Overview

Data	Positive	Negative	Total
Train	1555	14814	16369
Test	415	3678	4093

Data preprocessing

The domain data available is often not suitable for direct use in a machine learning algorithm. Many models require data to be entirely numerical, therefore categorical or boolean data must be converted first. Additionally, the data provided will often have features missing or have entries that fail sanity checks, such as a negative or implausibly high patient age. This section describes

the data processing applied in order to be able to use the data in a modern ML algorithm. Metrics about the dataset that are used within the preprocessing methods (e.g. means, standard deviations, categories) are calculated using the train set only, and later applied on the test set.

Structured

In the first step, columns with missing data are split into two types:

- Those that will be replaced by the value zero.
- Those that will be replaced with the median value of the available data points.

All boolean features pertaining diagnoses or symptoms are considered to be of the first type. The second category contains features pertaining statistical data, such as median earnings in the area where the patient lives, proportion of the population that is unemployed and similar.

In the second step, all categorical data is split up into separate boolean columns in a process known as one-hot encoding (Raschka and Mirjalili 2018). This type of processing is applied to the following features: insurance type, gender, ethnic, race and age. Age is split into a first group from 18 to 24, and into 10 year increments subsequently. If a new value is encountered during the processing of the test set, all columns assume the value zero and the new value is discarded, as the model would not be able to infer any information from the new value without having encountered it during training.

Figure 1. Example of one-hot conversion of categorical data.

color		**color_red**	**color_blue**	**color_green**
red		1	0	0
blue	⇨	0	1	0
green		0	0	1
green		0	0	1
red		1	0	0

Finally, all features of the dataset are scaled. This additional step is performed because not all models that are being evaluated are scale-invariant. This means that some of the models might ascribe a higher weight to features that due to their inherent nature have a higher magnitude. In such a scenario, a prediction might become overpowered by a feature such as total population in an area, typically counted in thousands to millions, as opposed to using a more predictive feature with smaller absolute values such as age, which typically ranges from 0 to 110.

Unstructured

The text data is provided in the form of a text file filled with Unified Medical Language System Concept Unique Identifiers (UMLS CUIs) gathered from all types of notes pertaining a patient.

Figure 2. Example text to CUI conversion

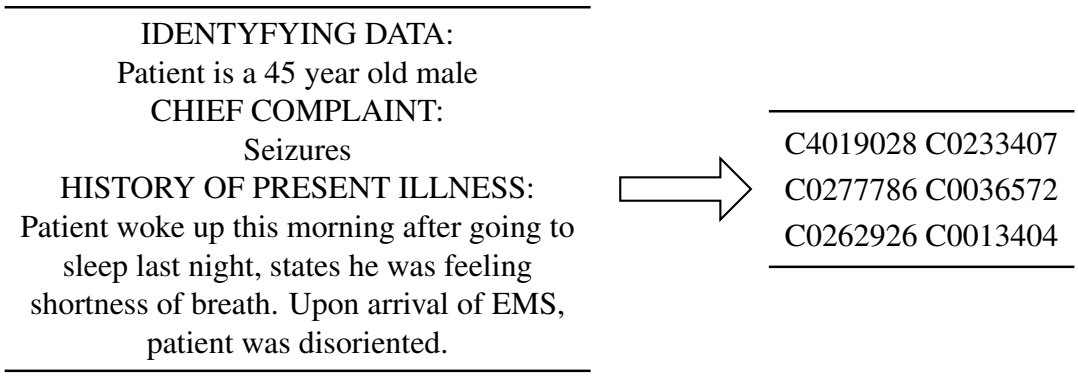

Table 2. CUI Examples

CUI	Meaning
C4019028	Morning after
C0233407	Disorientation
C0277786	Chief complaint (finding)
C0036572	Seizures
C0262926	Medical History
C0013404	Dyspnea

For traditional models, the text is then converted into a Bag Of Words sparse matrix, which

represents the number of times a term has appeared within a sample. In this approach, all information about the ordering within the text is lost.

For certain neural network models, the text is converted into a sequence of indexes, using the TensorFlow hashing trick ("tf.keras.preprocessing.text.hashing_trick : TensorFlow Core r2.1" 2020).

Hyperparameter tuning and metrics

In order to obtain more data in our hyperparameter tuning process, 10-fold cross validation is used to evaluate performance on train data when evaluating traditional models. During the evaluation of CNN based models, cross-validation has been found unfeasible due to the required time and problematic early-stopping mechanism. Instead, a 12.5% (10% of total) evaluation set is split from the train set. As a result, train scores are not directly comparable between linear models and some neural networks. Nevertheless, the train set is not affected by this change and has been kept separate throughout the entire model selection process. As a result, the test set constitutes a valid comparison between all models.

Area under precision recall curve (PR AUC) is used as the main metric to evaluate models. That is, in every grid search, the model with the highest PR AUC score is chosen. In a PR AUC curve, for a specific model, every point corresponds to a certain decision threshold and the respective precision and recall achieved at that threshold. In a clinical context, using this metric has an added benefit of being easily interpretable: how many patients with a condition might be missed, and how many patients might be subjected to a needless procedure or confirmation test when the model returns a false positive. Nevertheless, PR AUC and its curve are only one of many metrics used to evaluate a model, as such metrics such as Receiver Operating Characteristic (ROC AUC), positive F1, positive and negative predictive values, sensitivity and specificity are reported for the best model among the different model types.

Environment

All linear models are run on a machine with a Intel Xeon Gold 5122 CPU and 512 GB RAM and sklearn version 0.22.1. Whenever possible, grid searches are run in parallel. Particularly for some models using both structured and unstructured data, the memory requirement to utilize all sixteen cores surpasses 128GB RAM. The remaining neural network models are run on a machine with a Intel Core i7-8700K CPU, 64 GB RAM and two GeForce GTX 1080 Ti GPUs. The versions of keras and TensorFlow were respectively 2.3.1 and 1.13.2. In order to use both GPUs at the same time, the *multi_gpu_model* decorator was used, which splits each batch into sub-batches, calculates them on separate GPUs and concatenates the results back into a larger batch.

CHAPTER THREE: MODELS

In this chapter, the different models developed on the data will be described. Notably, some models will only work on a subsection of the data (e.g. only on structured, unstructured or a combination of both), while other models will have dependences on models previously described (e.g. ensembles and later iterations of neural networks).

SVM

Basic information

SVM is a model which tries to find a hyperplane separating samples from two classes. It does this by maximizing margins, that is the distance from the hyperplane to the nearest samples on each side. For the SVM Model, the SVC implementation of sklearn was used (Pedregosa et al. 2011). Because SVM is not well suited for large datasets due to very long training times, the model is trained to work with the structured data only.

Hyperparameters

Parameters as seen in Table 3 were used in a grid search. Following 10-fold cross-validation, the best model with an average PR AUC of *0.42* was chosen. The following hyperparameters were used:

'C': 100000.0, 'gamma': 0.0001, 'kernel': 'rbf', 'class_weight': 'balanced'

XGBoost

Introduction

XGBoost is a decision tree based model, which creates ensembles of decision trees using the gradient boosting framework. For the XGBoost model, the xgboost Python library was used (Chen and Guestrin 2016). The model is trained to work with the unstructured data only because

Table 3. SVM hyperparameter space

kernel: rbf	
Parameter	Search space
C	[1e-05, 1e-04, ..., 1e4, 1e5]
gamma	[1e-4, 1e-5, 1e-6, 1e-7, 1e-8]
kernel: poly	
Parameter	Search space
degree	[2, 3, 4, 5]
gamma	scale

XGBoost is known to work very well on structured/tabular data. An added benefit on XGBoost is it's short training time, compared to previous models. This model acts as a baseline for performance that can be achieved with strucutred data only, without the addition of text data.

Hyperparameters

Parameters as seen in Table 4 were evaluated using a grid search. Following 10-fold cross-validation, the best model with an average PR AUC of *0.47* was chosen. The following hyperparameters were used:

'colsample_bytree': 0.8, 'gamma': 1.5, 'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5, 'scale_pos_weight': 2, 'subsample': 0.8

Table 4. XGBoost hyperparameter space

Parameter	Search space
min_child_weight	[1, 3, 5]
gamma	[1, 1.5, 2]
subsample	[0.6, 0.8, 1.0]
colsample_bytree	[0.6, 0.8, 1.0]
max_depth	[3, 5, 7, 9]
scale_pos_weight	[2, 3, 5, 10]
learning_rate	[0.1, 0.2]

Logistic regression

Introduction

Logistic regression is a model which projects the data encountered on a euclidean space and tries to find a decision boundary between two classes using statistical methods. One of it's advantages is the small amount of hyperparameters to tune, making a grid-search feasible. It acts as a baseline for performance that can be achieved without aiding the text data with structural facts about a patient. The Logistic Regression model is trained on unstructured data only. The scikit-learn implementation was used.

Hyperparameters

Parameters as seen in Table 5 were evaluated in a grid search. The best results were achieved with the parameters *C: 10, class_weight: None*. The model achieved a PR AUC of *0.684* on average on a 10-fold cross-validation.

Table 5. Logistic regression hyperparameter space

Parameter	Search space
C	[.01, .1, 1, 10]
class_weight	[balanced, None]

Elasticnet

Introduction

Elastic net is a logistic regression model with linearly combined l1 and l2 regularizations used in the lasso and ridge methods. For this model, sklearn's LogisticRegression classifier with a *elasticnet* penalty is used. The model is trained to work with a concatenated dataset of structured and unstructured data.

Hyperparameters

Table 6. Elasticnet hyperparameter space

Parameter	Search space
C	[.01, .1, 1, 10]
l1_ratio	[.4, .5, .6]
class_weight	[None, 'balanced']

The hyperparameters used in a grid search are shown in Table 6. Performing a grid search on this model is noticably slower than all previous models, requiring around 40 minutes on a multi-core machine. The best results were achieved with the following hyperparameters: *penalty='elasticnet', solver='saga', l1_ratio=0.5, class_weight='balanced', C=10*

The model achieved a PR AUC of *0.697* on average on a 10-fold cross-validation.

Ensembles

Introduction

In many problems, superior results can be obtained by using more than one model. In a famous competition held by Netflix, the best score was tied by an ensemble of two contenders (“Leaderboard” n.d.). In this model, we will be using the best performing models on structured and unstructured data, that is XGBoost and Logistic regression respectively.

For the implementation, we use the VotingClassifier from scikitlearn (“1.11. Ensemble methods” n.d.). Because we used only two models at a time, we use the VotingClassifier in soft mode, which means it averages the probabilities of its children. The alternative mode is hard voting, which only works with an uneven amount of classifiers.

By default scikitlearn does not support ensembles working on separate pieces of data. Instead, the unstructured and structured datasets were merged, and each classifier was encapsulated in a pipeline with a transformer selecting features applicable to the model.

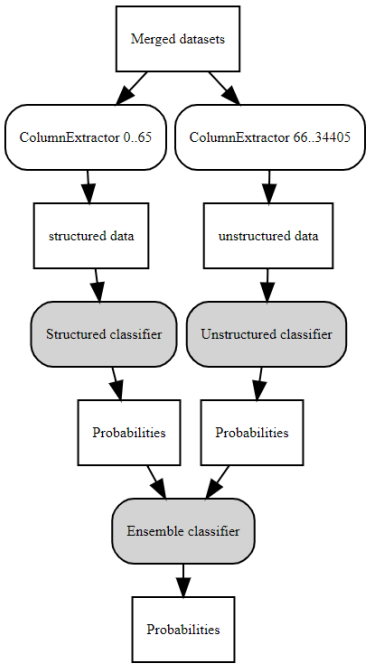

Figure 3. Overview of the ensemble architecture

Hyperparameters

Due to the number of parameters to optimize, we estimated a full grid search of an ensemble of two models would take approximately 317 years on our machines. Instead, we decided to use the best candidate of each individual model, and only tune the weights within the ensemble itself.

The hyperparameters used in a grid search are shown in Table 7. The best results were achieved with the following hyperparameters: *unstructured_weight: 1, structured_weight: 0.375*

The model achieved a PR AUC of *0.706* on average on a 10-fold cross-validation.

Table 7. Ensemble hyperparameter space

Parameter	Search space
unstructured_weight	[.25, .375, .5, .625, .75, 1]
structured_weight	[.25, .375, .5, .625, .75, 1]

Deep - Dense only

Introduction

The Dense Neural Network is trained used both, the structured and unstructured data. A simple model with only one dense layer and a simple concatenation of structured and unstructured data was chosen in order to work as a baseline for future, more complex neural networks. The model was implemented using the keras library(Chollet et al. 2015) and its functional API.

Hyperparameters

Hyperparameters as seen in Table 8 were considered. Following a 200 sample random search, the following hyperparameters performed best:

'epochs' : 10, 'batches' : 32, 'lr' : 5e-4, 'layer_size' : 48, 'drop_rate' : 0.45, 'optimizer' : optimizers.adam, 'activation_hidden' : 'custom_gelu'

Resulting in a PR AUC of *0.669* in a 10-fold cross-validation.

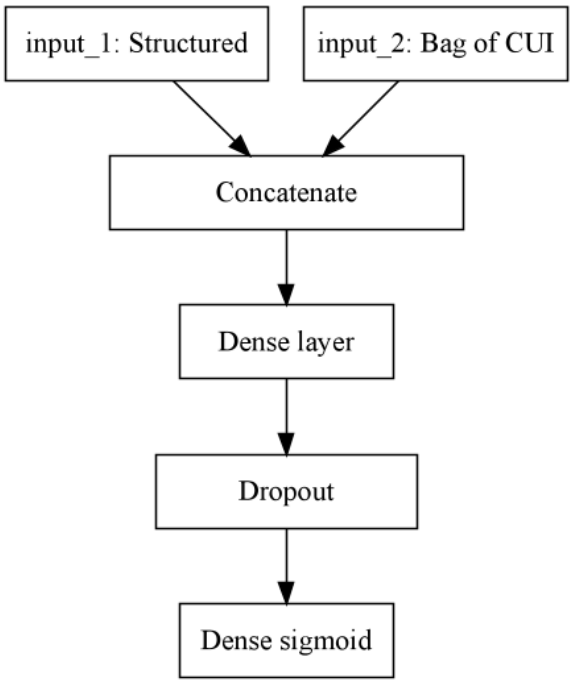

Figure 4. Overview of the dense only neural network architecture

Table 8. Dense only hyperparameter space

Parameter	Search space
epochs	[3, 5, 7, 10]
batches	[32, 64, 128]
lr	[2e-3, 1e-3, 5e-4, 2e-4]
class_weight	[None]
bias	[None]
layer_size	[32, 48, 64]
drop_rate	[0.35, 0.4, 0.45]
optimizer_candidates	[optimizers.adam]
activation_hidden	['linear', 'custom_gelu']

Deep - CNN

Introduction

The CNN is trained used both, the structured and unstructured data. Unlike previous models, the unstructured data is processed using the hashing trick("tf.keras.preprocessing.text.hashing_trick : TensorFlow Core r2.1" 2020) and subsequently

fed into an embedding layer.

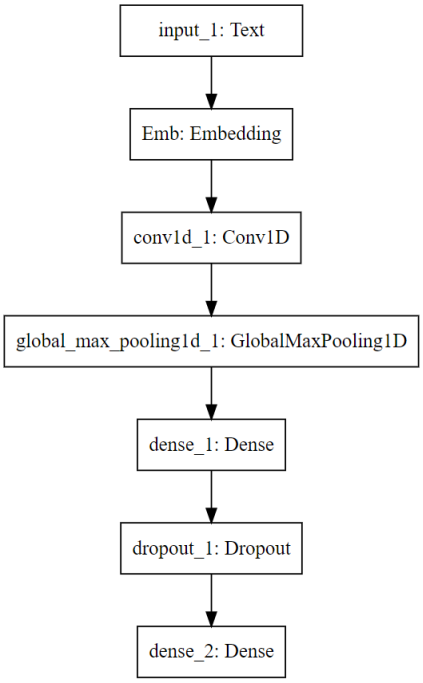

Figure 5. Overview of the CNN architecture

Hyperparameters

In the first step, a random search with 200 samples with parameters as seen in Table 9. Following 10-fold cross-validation, the best model with an average PR AUC of *0.64* was found. The following hyperparameters were used:

'optimizer':'Adam', 'lr': 0.0001, 'layer_size': 48, 'epochs': 20, 'drop_rate': 0.4, 'class_weight': None, 'bias': array([-2.25409713]), 'batch_size': 32, 'activation_hidden': 'custom_gelu', 'CNN_out': 300, 'CNN_kern': 8, 'CNN_inp': 1000, 'CNN_filt': 1024

class_weight represents a dictionary where each class label is a key with the value of half of the total number of samples divided by the number of the samples in the respective class. initial_class represents the logarithm of the number of positive samples divided by the number of negative samples (*Classification on imbalanced data : TensorFlow Core*). In the next step, distributions of PR AUC scores vs hyperparameter were analyzed. As seen in Figure 7, the key insights gained were the following:

- CNN with the maximum kernel size of 8 performed best (7a).
- Models with a lower learning rate performed best (7b).
- Models with the maximum epoch limit performed best (7c).
- Models with smaller dense layer sizes (<100) performed best (7d).

With the above insights, a second hyperparameter search was performed as seen in Table 10. Due to the increasing time complexity of the models, instead of cross-validation, a validation set was separated from the train set (10% of the entire dataset) and used to train the model with early stopping enabled and a maximum of 40 epochs and a patience of 5. Additionally, the *custom_gelu* activation function as implemented by Boris Banushev (Banushev 2019), as well as the Nadam optimizer were added to the random search. Following a 200 sample random search, the following hyperparameters performed best:

batches = [64], lr = [1e-4], class_weight = [None], bias = [None], layer_size = [56], drop_rate = [0.45], optimizer = [optimizers.adam], activation_hidden = ['linear'], CNN_inp = [1000], CNN_out = [500], CNN_filt = [1024], CNN_kern = [8]

The above model stopped training after 12 epochs and achieved a PR AUC of *0.680* on the validation set.

Deep - CNN with structured

Basic information

This model combines approaches from the previous Deep - Dense only and Deep - CNN models as seen in Figure 6. The unstructured data is presented in word embedding form, processed by a CNN followed by GlobalMaxPooling as opposed to a bag of CUIs concatenated directly to the structured data. For the hyperparameter search, the same space was used as in Table 10. Following a 200 sample random search, the following hyperparameters performed best, with a PR AUC of *0.731*:

'optimizer': nadam, 'lr': 5e-05, 'layer_size': 32, 'epochs': 40, 'drop_rate': 0.35,

Table 9. Initial CNN hyperparameter space

Parameter	Search space
epochs	[2, 3, 5, 10, 20]
batches	[32, 64, 128]
lr	[2e-3, 1e-3, 1e-4]
class_weight	[None, class_weight]
bias	[None, initial_bias]
layer_size	[32, 48, 64, 128, 256]
drop_rate	[0.3, 0.4, 0.5]
CNN_inp	[1000, 300, 500]
CNN_out	[100, 200, 300, 1000]
CNN_filt	[128, 256, 512, 1024]
CNN_kern	[1, 2, 4, 8]
vocab_size	[300]
max_length	[1000]

Table 10. Second CNN hyperparameter space

Parameter	Search space
batches	[32, 64, 128, 256]
lr	[1e-4, 5e-4, 5e-5]
class_weight	[None]
bias	[None, initial_bias]
layer_size	[32, 48, 56, 64]
drop_rate	[0.35, 0.4, 0.45]
CNN_inp	[1000]
CNN_out	[300, 400, 500]
CNN_filt	[128, 256, 512, 1024]
CNN_kern	[8, 10, 12, 16]
optimizer	[optimizers.adam, optimizers.nadam]
activation	['linear', 'custom_gelu']

'class_weight': None, 'bias': array([-2.25409713]), 'batch_size': 32, 'activation_hidden': 'custom_gelu', 'CNN_out': 500, 'CNN_kern': 8, 'CNN_inp': 1000, 'CNN_filt': 128

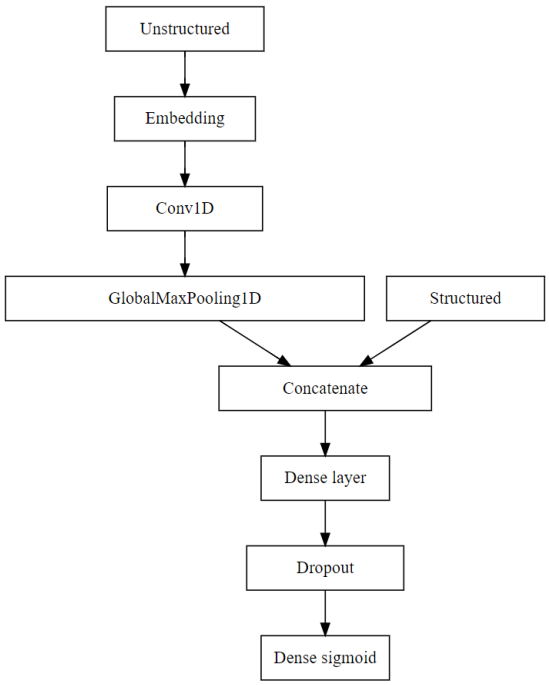

Figure 6. Overview of the CNN with structured architecture

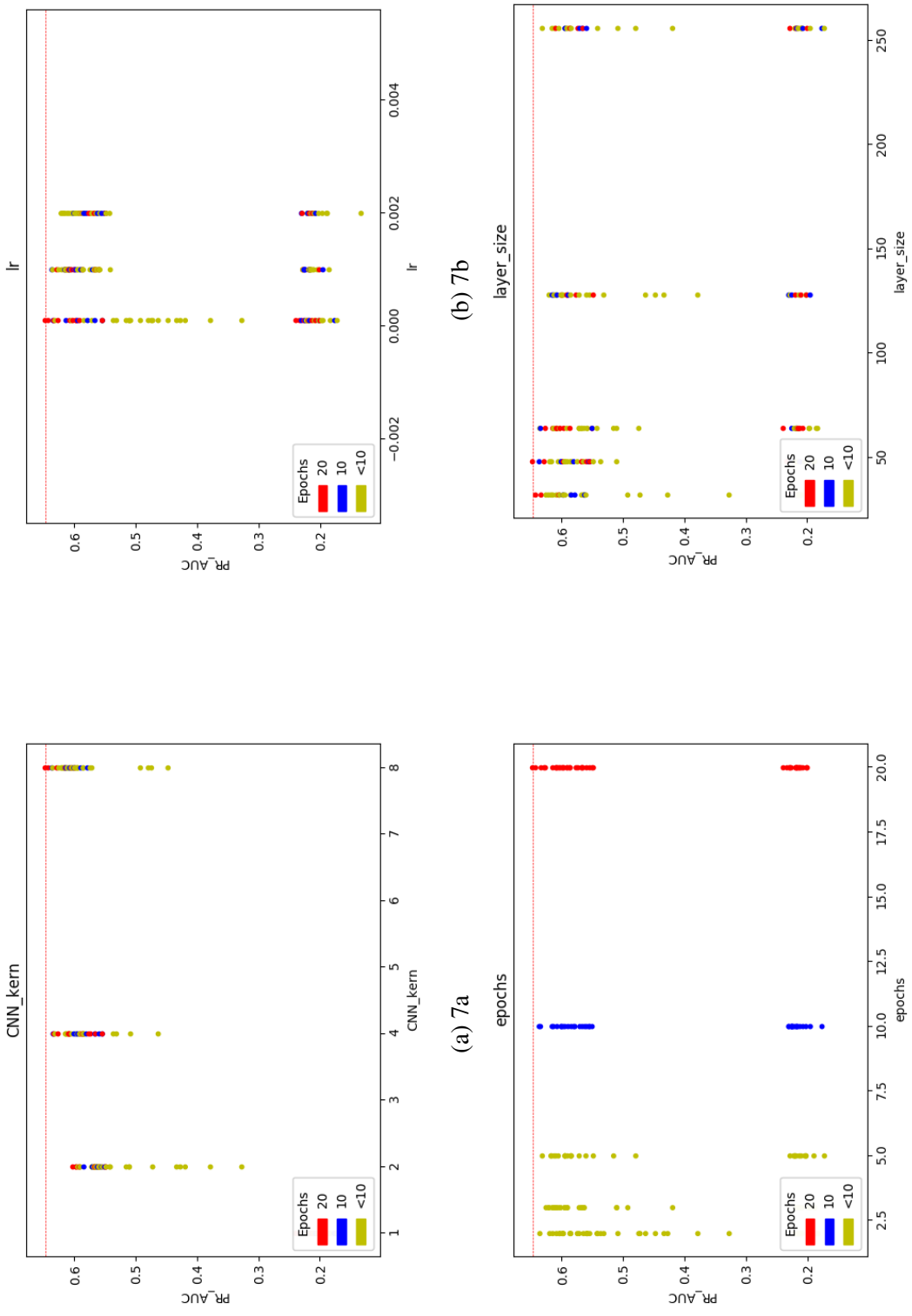

Figure 7. Plots of achieved PR AUC vs hyperparameter in first CNN iteration

CHAPTER FOUR: RESULTS AND DISCUSSION

Table 11 shows all metrics collected for all models. Figures 8 and 9 show the ROC and PR curves calculated on test data of the best performing models in each respective category: SVM on structured data, Logistic Regression on unstructured data, a neural network working on both kinds of data, and a linear model (ElasticNet) working on both kinds of data.

Performance

On both ROC and PR curves, as well as in most other metrics, the models using structured data only are clearly trailing behind the models which have access to the unstructured data. Nevertheless, on the ROC curve we can see XGBoost is visibly outperforming a no skill classifier, it can therefore be reasoned that the structured data has some predictive value.

The remaining three models have very similar PR AUC scores. When compared on the ROC curve, the logistic regression model is trailing behind slightly. When compared on the PR curve, it is hard to distinguish one clearly superior model. The structured data is clearly having an impact on the higher recall region, as this is the point where the model limited to unstructured data is clearly falling behind.

Even though the models were specifically selected using the PR AUC metric, there are other metrics one might find more important. When the remaining metrics are considered, there is no clearly superior model to all the others. Elasticnet has the greatest score in positive F1, positive recall and negative precision, but has a much worse positive precision than other models. If one would prefer a model with a high positive precision, the ensemble on XGB and Logistic regression is the clear winner, with no significant compromises in the remaining metrics other than positive recall, which is only at 0.504. Finally, the greatest positive F1 score is achieved by

the Neural CNN network concatinated with the structured data.

Time, cost and prerequisites

When considering the top three models by PR_AUC, Logistic regression is the clear winner in terms of time, cost and prerequisites. The model can be trained on any modern CPU, does not require a GPU and training in our environment only took about 5 minutes. The elasticnet can also be trained on a machine with only a CPU, but training was substantially slower. In our training run, the model failed to converge and stopped training after the sklearn's default cut off. The process took around two hours, the longest of all models listed, but this might be shortened by a careful choice of hyperparameters. The use of linear models can however lead to a more sparse, parsimonious model, as the L1 regularization component can effectively eliminate a subset of the features. The final neural model technically does not require a GPU to be operated, but when run on a CPU is slower by orders of magnitude. The model reached peak performance after about 13 epochs, which corresponds to a train time of about 5 minutes when run concurrently on two GPUs in our environment.

Model complexity

In terms of amount of hyperparameters to be tuned there are significant differences in the models. The logistic regression and elasticnet models manageable hyperparameter amount, insofar that almost all options can be explored with a grid search within a reasonable time frame. The flexibility and expressiveness of neural networks, while both desirable qualities, make it practically impossible to explore all possible hyperparameters and to conclude that a certain architecture is ideal or superior to all others. Additionally, they are not easily interpretable, which could serve as a reason to prefer traditional models. Similarly, ensemble models, while having only few parameters to tune, can be constructed of a infinite set of combinations of other models.

Conclusion

Throughout this study we discovered that the best models based on structured and unstructured data individually have very different predictive value - 0.48 and 0.69 PR AUC respectively. When the two types of data are used together, two models with very different architectures were able to improve this score by at least one point. When comparing the same model with and without the addition of structured data, the CNN model improved by 4 points. This effect might become even more pronounced with either more data, or an architecture which is better suited to put the two types of data together. More work is needed in the area of neural network architecture fine tuning. Additionally, ensembles of carefully selected calibrated models might be able to further improve the best score achieved in this study.

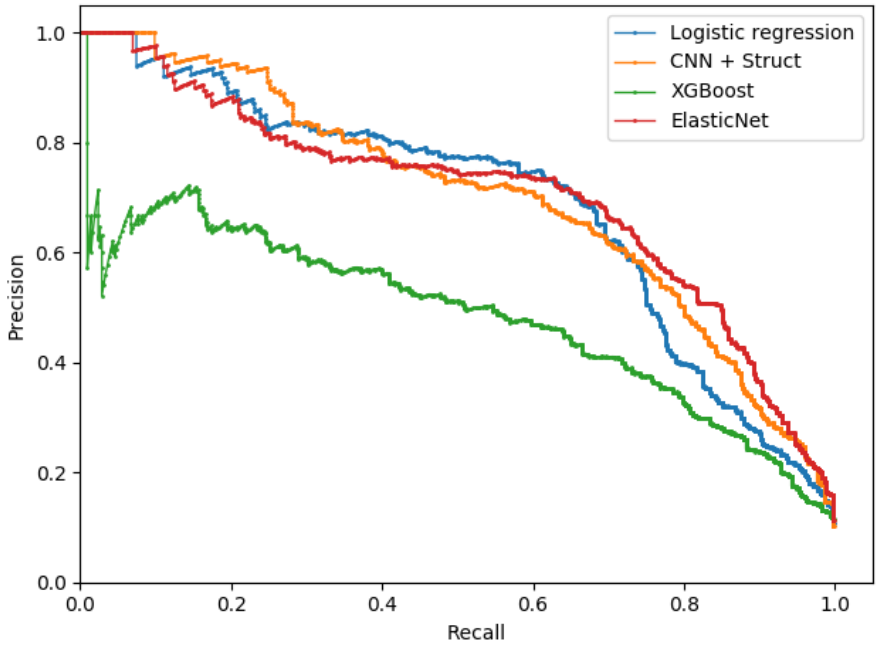

Figure 8. PR curves of selected models

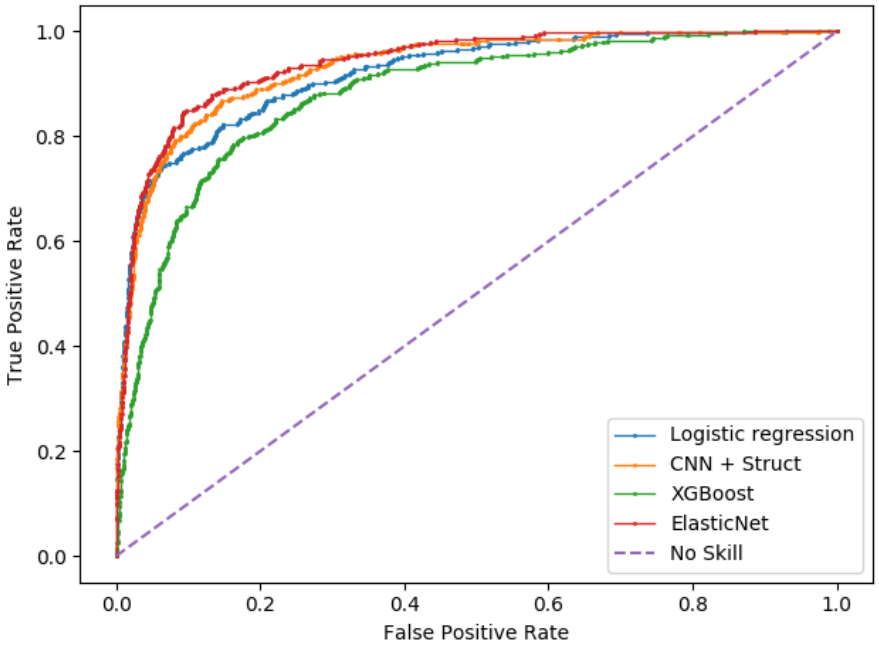

Figure 9. ROC curves of selected models

Table 11. Model results

Model	Data	PR AUC		ROC AUC		F1_P		Precision_P (PPV)		Recall_P (Sensitivity)		Precision_N (NPV)		Recall_N (Specificity)	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
SVM[a]	structured	0.421	0.425	0.882	0.880	0.484	0.488	0.344	0.349	0.815	0.812	0.977	0.975	0.837	0.829
XGB[a]	structured	0.473	0.484	0.883	0.881	0.467	0.479	0.533	0.532	0.415	0.436	0.940	0.938	0.962	0.957
Neural CNN[b]	unstructured	0.676	0.674	0.914	0.918	0.620	0.616	0.730	0.722	0.540	0.537	0.950	0.949	0.977	0.977
Logistic regression[a]	unstructured	0.684	0.690	0.930	0.919	0.606	0.600	0.779	0.775	0.497	0.489	0.949	0.945	0.985	0.984
XGB + LogReg[a]	both	0.706	0.674	0.937	0.744	0.609	0.617	0.802	0.795	0.491	0.504	0.949	0.946	0.987	0.985
Elasticnet[a]	both	0.697	0.704	0.941	0.940	0.597	0.624	0.468	0.492	0.825	0.851	0.980	0.982	0.901	0.901
Neural Dense[b]	both	0.668	0.669	0.925	0.924	0.585	0.553	0.740	0.728	0.487	0.446	0.948	0.940	0.982	0.981
Neural CNN + Dense[b]	both	0.731	0.703	0.936	0.933	0.682	0.647	0.688	0.710	0.677	0.595	0.966	0.955	0.968	0.973

a. Train scores are calculated as average of 10-fold cross validation
b. Train scores are calculated on 10% validation set

REFERENCE LIST

Banushev, Boris. 2019. "Using the latest advancements in AI to predict stock market movements." February. https://github.com/borisbanushev/stockpredictionai.

Carrell, David S., David Cronkite, Roy E. Palmer, Kathleen Saunders, David E. Gross, Elizabeth T. Masters, Timothy R. Hylan, and Michael Von Korff. 2015. "Using natural language processing to identify problem usage of prescription opioids." *International Journal of Medical Informatics* 84 (12): 1057–1064. ISSN: 1386-5056. doi:https://doi.org/10.1016/j.ijmedinf.2015.09.002. http://www.sciencedirect.com/science/article/pii/S138650561530037X.

Chen, Tianqi, and Carlos Guestrin. 2016. "XGBoost: A Scalable Tree Boosting System." In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 785–794. KDD '16. San Francisco, California, USA: ACM. ISBN: 978-1-4503-4232-2. doi:10.1145/2939672.2939785. http://doi.acm.org/10.1145/2939672.2939785.

Chollet, François, et al. 2015. "Keras."

"GLUE Benchmark." n.d. https://gluebenchmark.com/leaderboard.

Horgan, Constance M., Brandeis University., Schneider Institute for Health Policy., and Robert Wood Johnson Foundation. 2001. *Substance abuse : the nation's number one health problem : key indicators for policy update.* Princeton, NJ: The Foundation.

"Leaderboard." n.d. https://www.netflixprize.com/leaderboard.html.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research* 12:2825–2830.

Raschka, Sebastian, and Vahid Mirjalili. 2018. *Python machine learning machine learning and deep learning with Python, scikit-learn, and TensorFlow.* Packt Publishing.

Rudd, Rose A., Puja Seth, Felicita David, and Lawrence Scholl. 2016. "Increases in Drug and Opioid-Involved Overdose Deaths — United States, 2010–2015." *Morbidity and Mortality Weekly Report* 65 (50 & 51): 1445–1452. ISSN: 01492195, 1545861X. https://www.jstor.org/stable/24876516.

Sarker, Abeed, Graciela Gonzalez-Hernandez, Yucheng Ruan, and Jeanmarie Perrone. 2019. "Machine Learning and Natural Language Processing for Geolocation-Centric Monitoring and Characterization of Opioid-Related Social Media Chatter." *JAMA Network Open* 2, no. 11 (November): e1914672–e1914672. ISSN: 2574-3805. doi:10.1001/jamanetworkopen.2019.14672. eprint: https://jamanetwork.com/journals/jamanetworkopen/articlepdf/2753983/sarker_2019_oi_190564.pdf. https://doi.org/10.1001/jamanetworkopen.2019.14672.

"1.11. Ensemble methods." n.d. https://scikit-learn.org/stable/modules/ensemble.html#voting-classifier.

Classification on imbalanced data : TensorFlow Core. https://www.tensorflow.org/tutorials/structured_data/imbalanced_data.

"tf.keras.preprocessing.text.hashing_trick : TensorFlow Core r2.1." 2020. Accessed February 8, 2020. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/hashing_trick.

VITA

Robert Kania was born to Dr. Rudolf and Mrs. Aldona Kania on December 6th, 1993 and raised in Opole, Poland. He earned his Bachelor degree in Computer Engineering in July 2015 at Opole University of Technology, where he was awarded a Erasmus scholarship to study at Lucerne University of Applied Sciences and Arts in Switzerland, as well as a Confucius Institude scholarship to study chinese at Beijing University of Technology in China. Following his graduation, he worked in the automotive industry in Germany, developing modern infotainment systems as Software Engineer. As part of a Fulbright scholarship award, he continued his education at Loyola University Chicago, where he pursued a master's degree in Computer Science.