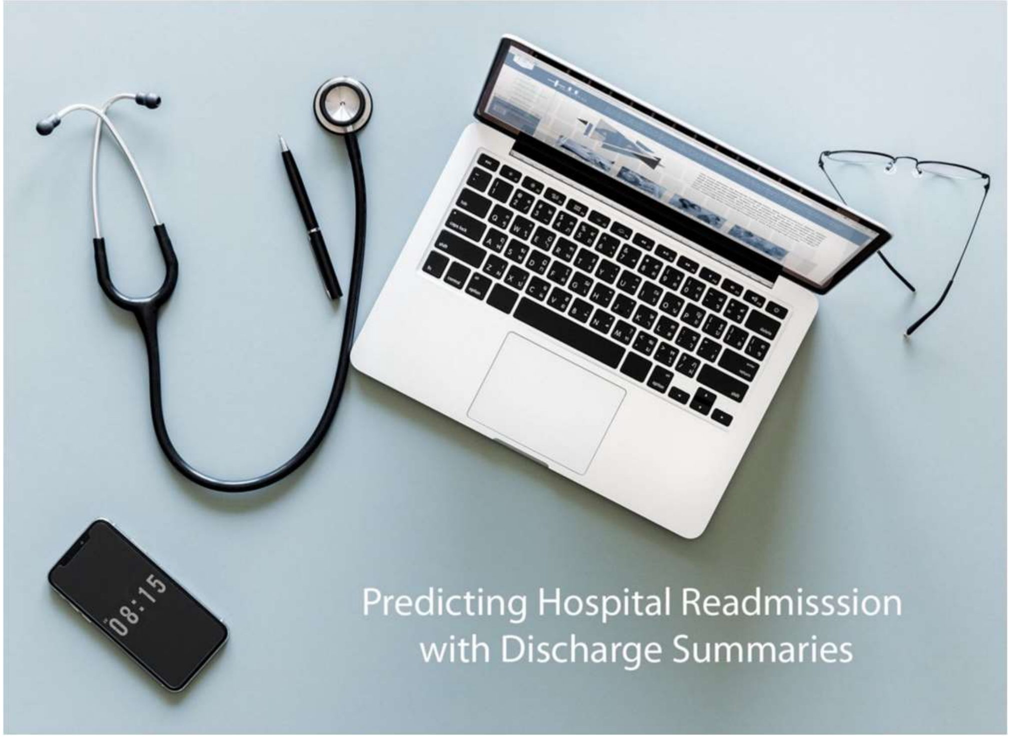
OPEN DATA SCIENCE CONFERENCE

#ODSC

@ODSC

San Francisco | October 31 - Nov. 3 2018

Introduction to Clinical Natural Language Processing

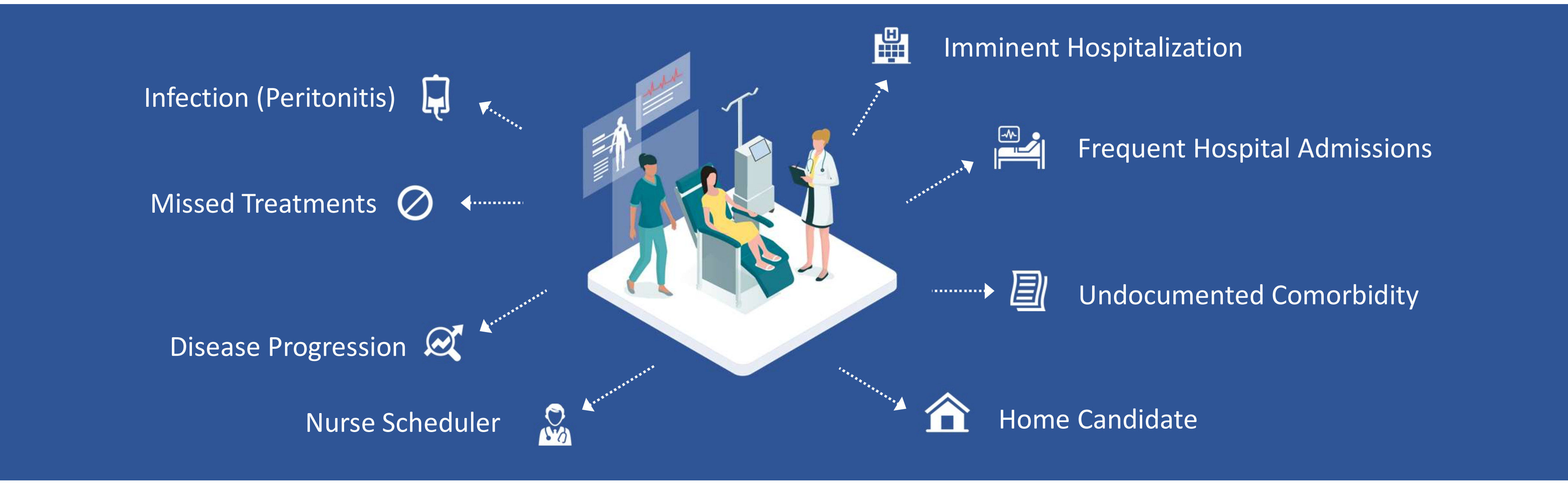


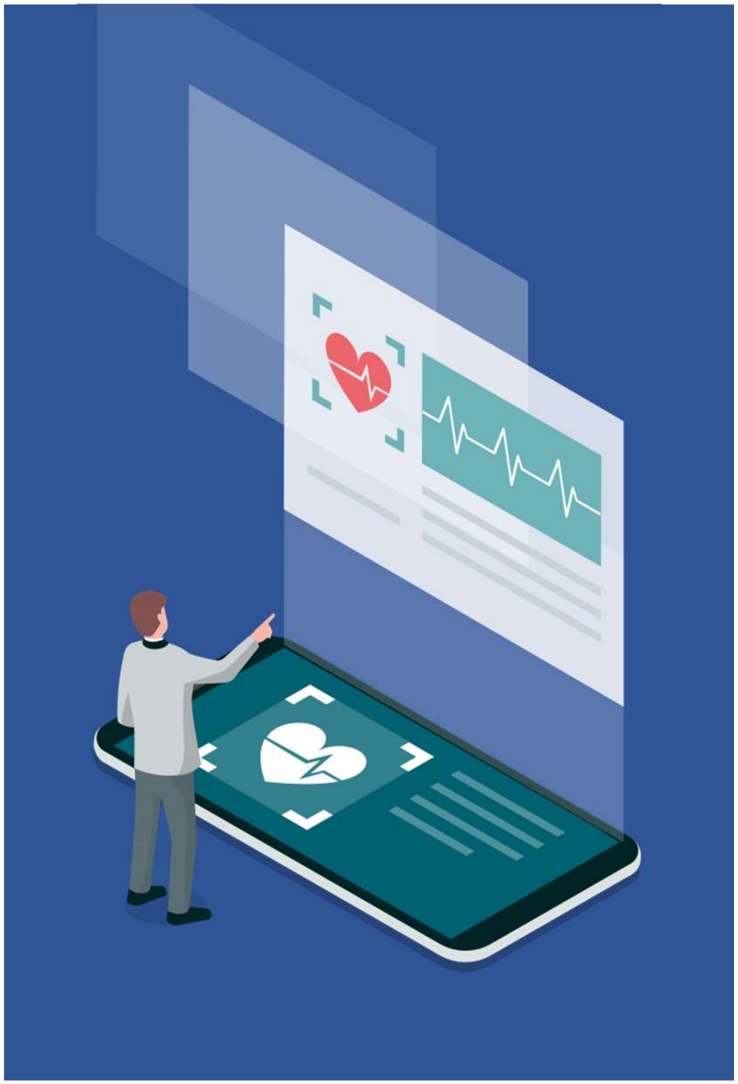
Andrew Long

https://towardsdatascience.com/introduction-to-clinical-natural-language-processing-predicting-hospital-readmission-with-1736d52bc709

FRESENIUS MEDICAL CARE

180,000+	26M	50+	2,200+	60,000+
U.S. PATIENTS SERVED	ANNUAL HEMODIALYSIS TREATMENT	STATES AND TERRITORIES IN OUR NETWORK	U.S. DIALYSIS CLINICS	U.S. EMPLOYEES





Clinical Notes

- Chest pain
- Shortness of breathe
- Nausea, vomiting, diarrhea
- Weakness
- Sick

Build predictive models that incorporate free-text clinical notes

Workshop Overview

- Brief overview of clinical dataset (MIMIC III)
- How to prepare data for a machine learning project
- How to preprocess the unstructured notes
- How to build a simple predictive model using a bag-of-words approach
- How to assess the quality of your model
- How to decide the next step for improving the model

- Note: I created an artificial dataset based on Stanford's IMDB which you can use if you don't have MIMIC access for the workshop

Workshop Project Question

Scalable and accurate deep learning for electronic health records
Rajkomar et al. (paper at https://arxiv.org/abs/1801.07860)

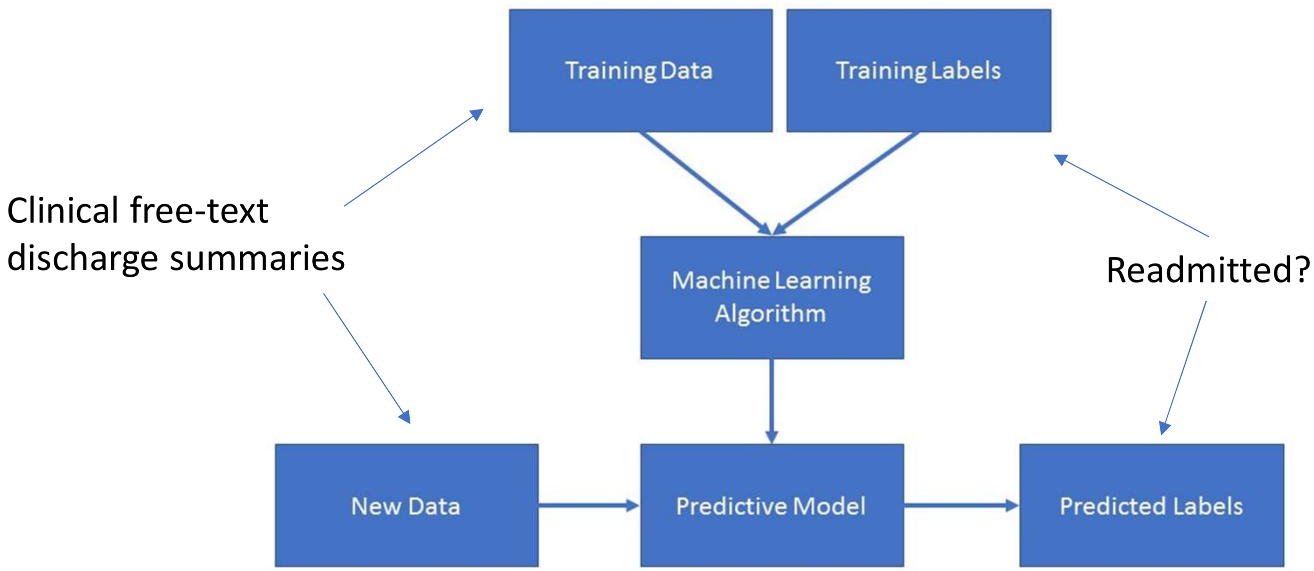
- in-hospital mortality (AUC = 0.93–0.94)
- 30-day unplanned readmission (AUC = 0.75–76)
- prolonged length of stay (AUC = 0.85–0.86)
- discharge diagnoses (AUC = 0.90)

AUC is a data science performance metric where closer to 1 is better

How good of a model can we get if use the discharge free-text summaries with a simple predictive model to predict readmission?

Classification Model Definition

- Predict which patients are at risk for 30-day unplanned readmission utilizing free-text hospital discharge summaries.



Part 0: MIMIC III dataset

Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

#ODSC

MIMIC III dataset

- This database contains de-identified data from over 40,000 patients who were admitted to Beth Israel Deaconess Medical Center in Boston, Massachusetts from 2001 to 2012

Access:

https://mimic.physionet.org/gettingstarted/access/

https://towardsdatascience.com/getting-access-to-mimic-iii-hospital-database-for-data-science-projects-791813feb735

- Since dataset has restricted access, any single subject data shown in this workshop is artificially created.

MIMIC III datasets for tutorial

1. checksum_md5_zipped.txt (MD5 checksum for zipped files)
2. checksum_md5_unzipped.txt (MD5 checksum for unzipped files)
3. ADMISSIONS.csv.gz (2.5M compressed, 12M decompressed)
4. CALLOUT.csv.gz (1.2M compressed, 6.1M decompressed)
5. CAREGIVERS.csv.gz (49K compressed, 199K decompressed)
6. CHARTEVENTS.csv.gz (4.0G compressed, 33G decompressed)
7. CPTEVENTS.csv.gz (4.8M compressed, 56M decompressed)
8. DATETIMEEVENTS.csv.gz (53M compressed, 502M decompressed)
9. DIAGNOSES_ICD.csv.gz (4.5M compressed, 19M decompressed)
10. DRGCODES.csv.gz (1.7M compressed, 11M decompressed)
11. D_CPT.csv.gz (3.9K compressed, 14K decompressed)
12. D_ICD_DIAGNOSES.csv.gz (279K compressed, 1.4M decompressed)
13. D_ICD_PROCEDURES.csv.gz (75K compressed, 305K decompressed)
14. D_ITEMS.csv.gz (184K compressed, 933K decompressed)
15. D_LABITEMS.csv.gz (12K compressed, 43K decompressed)
16. ICUSTAYS.csv.gz (1.9M compressed, 6.1M decompressed)
17. INPUTEVENTS_CV.csv.gz (403M compressed, 2.3G decompressed)
18. INPUTEVENTS_MV.csv.gz (144M compressed, 931M decompressed)
19. LABEVENTS.csv.gz (321M compressed, 1.8G decompressed)
20. MICROBIOLOGYEVENTS.csv.gz (7.3M compressed, 70M decompressed)
21. NOTEEVENTS.csv.gz (1.1G compressed, 3.8G decompressed)
22. OUTPUTEVENTS.csv.gz (56M compressed, 379M decompressed)
23. PATIENTS.csv.gz (559K compressed, 2.6M decompressed)
24. PRESCRIPTIONS.csv.gz (99M compressed, 735M decompressed)
25. PROCEDUREEVENTS_MV.csv.gz (7.5M compressed, 47M decompressed
26. PROCEDURES_ICD.csv.gz (1.8M compressed, 6.5M decompressed)
27. SERVICES.csv.gz (1.2M compressed, 3.4M decompressed)
28. TRANSFERS.csv.gz (5.3M compressed, 24M decompressed)

https://physionet.org/works/MIMICIIIClinicalDatabase

- ADMISSIONS.csv.gz
- NOTEEVENTS.csv.gz
- placed in a 'data' folder in same folder as this workshop's notebook

```
import gzip

for filename in ["data/ADMISSIONS.csv.gz", "data/NOTEEVENTS.csv.gz"]:

    with gzip.open(filename, 'rt') as f:
        data = f.read()
    with open(filename[:-3], 'wt') as f:
        f.write(data)
```

Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

Part 1: How to prepare data for a machine learning project

Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

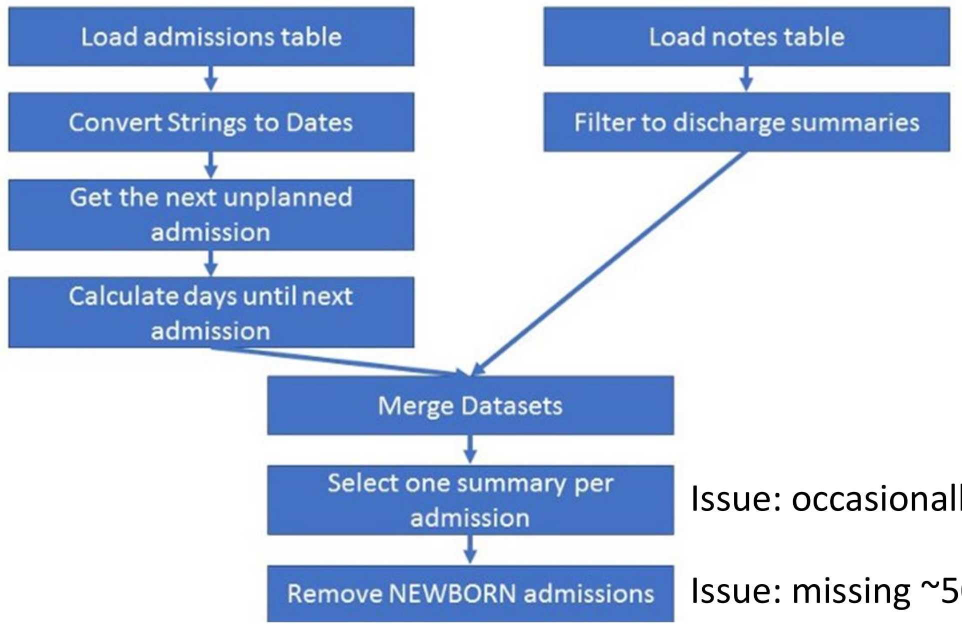
#ODSC

Load, clean, merge dataset

```
In [2]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt

        import odsc2018_utils

        df_adm_notes_clean = odsc2018_utils.load_clean_merge_dataset('data/ADMISSIONS.csv','data/NOTEEVENTS.csv')

C:\Users\3236283\AppData\Local\Continuum\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:2910:
ing: Columns (4,5) have mixed types. Specify dtype option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

We skip this process to save some time In workshop.
See additional Jupyter Notebook (odsc_2018_mimic_pre) for tutorial on these steps.

Prepare data for ML Project



Remove next ELECTIVE admissions

Issue: occasionally multiple

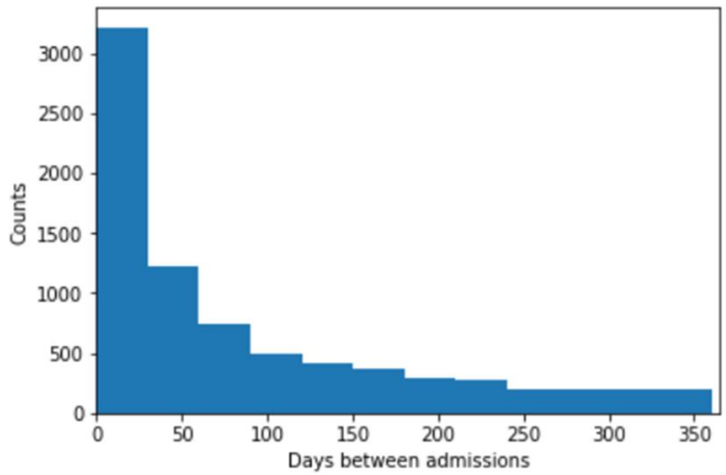
Issue: missing ~50% discharge summaries

Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

df_adm_notes_clean

- SUBJECT_ID – unique patient identifier
- HADM_ID- unique admission identifier
- ADMITTIME – admission date
- DISCHTIME – discharge date
- DEATHTIME – death date
- DAYS_NEXT_ADMIT – days until next admission if it exists
- TEXT – discharge summary for this admission

Add OUTPUT_LABEL




```
In [3]: df_adm_notes_clean['OUTPUT_LABEL'] = (df_adm_notes_clean.DAYS_NEXT_ADMIT < 30).astype('int')
```

```
In [4]: print('Number of positive samples:', (df_adm_notes_clean.OUTPUT_LABEL == 1).sum())
        print('Number of negative samples:',  (df_adm_notes_clean.OUTPUT_LABEL == 0).sum())
        print('Total samples:', len(df_adm_notes_clean))

        Number of positive samples: 3004
        Number of negative samples: 48109
        Total samples: 51113
```

Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

#ODSC

Make training/validation/test sets

70/15/15 is a design choice

- Training samples: these samples are used to train the model
- Validation samples: these samples are held out from the training data and are used to make decisions on how to improve the model
- Test samples: these samples are held out from all decisions and are used to measure the generalized performance of the model

Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

Make training/validation/test sets

70% Training | 15% Validation | 15% Test

70/15/15 is a design choice

```
In [5]: # shuffle the samples
        df_adm_notes_clean = df_adm_notes_clean.sample(n = len(df_adm_notes_clean), random_state = 42)
        df_adm_notes_clean = df_adm_notes_clean.reset_index(drop = True)

        # Save 30% of the data as validation and test data
        df_valid_test=df_adm_notes_clean.sample(frac=0.30,random_state=42)

        df_test = df_valid_test.sample(frac = 0.5, random_state = 42)
        df_valid = df_valid_test.drop(df_test.index)

        # use the rest of the data as training data
        df_train_all=df_adm_notes_clean.drop(df_valid_test.index)
```

Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

Make training/validation/test sets

70% Training	15% Validation	15% Test

70/15/15 is a design choice

Verify that positive prevalence is approximately the same in the 3 groups

```
In [6]: print('Test prevalence(n = %d):%.3f'%(len(df_test),df_test.OUTPUT_LABEL.sum()/ len(df_test)))
        print('Valid prevalence(n = %d):%.3f'%(len(df_valid),df_valid.OUTPUT_LABEL.sum()/ len(df_valid)))
        print('Train all prevalence(n = %d):%.3f'%(len(df_train_all), df_train_all.OUTPUT_LABEL.sum()/ len(df_train_all)))
        print('all samples (n = %d)'%len(df_adm_notes_clean))
        assert len(df_adm_notes_clean) == (len(df_test)+len(df_valid)+len(df_train_all)),'math didnt work'

        Test prevalence(n = 7667):0.062
        Valid prevalence(n = 7667):0.057
        Train all prevalence(n = 35779):0.058
        all samples (n = 51113)
```

```
In [7]: df_train_all.to_csv('data/df_train_all.csv',index=False)
        df_valid.to_csv('data/df_valid.csv',index=False)
        df_test.to_csv('data/df_test.csv',index=False)
```

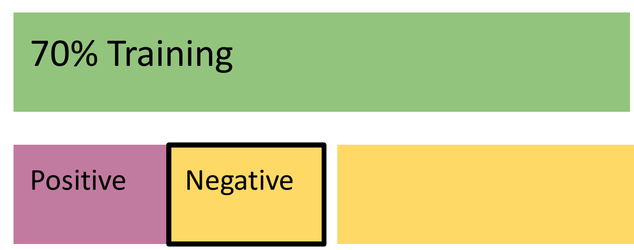
Artificial Dataset

- For those without MIMIC data, I created an artificial dataset based on IMDB dataset.
- data_artificial/df_train_all_imdb.csv
- data_artificial/df_valid_imdb.csv
- data_artificial/df_test_imdb.csv

- validation and test sets were created to have approximately same number and prevalence as the MIMIC sets

Imbalanced Classification

- Model that always guesses ‘Not readmitted’ → 94% accuracy, but never catches any readmissions (0% recall)
- To prevent this from happening, we need to balance the training set
  - sub-sample the more dominant class: use a random subset of the negatives
  - over-sample the imbalanced class: use the same positive samples multiple times
  - create synthetic positive data



Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

#ODSC

Subsample Training Dataset

In [9]:

```
# split the training data into positive and negative
rows_pos = df_train_all.OUTPUT_LABEL == 1
df_train_pos = df_train_all.loc[rows_pos]
df_train_neg = df_train_all.loc[~rows_pos]

n = np.min([len(df_train_pos),len(df_train_neg)])

# merge the balanced data
df_train = pd.concat([df_train_pos.sample(n = n, random_state = 42), \
                      df_train_neg.sample(n = n, random_state = 42)],axis = 0)

# shuffle the order of training samples
df_train = df_train.sample(n = len(df_train), random_state = 42).reset_index(drop = True)

print('Train prevalence (n = %d):'%len(df_train), df_train.OUTPUT_LABEL.sum()/ len(df_train))
```

```
Train prevalence (n = 4184): 0.5
```

70% Training

Positive | Negative

Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

Part 2: How to preprocess the unstructured notes

Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

#ODSC

Pre-process Text Data

- Occasionally, need to modify the text to make useable (for example drop newlines, carriage returns, numbers, etc)

- Two Methods:
  - Modify the original dataframe TEXT column
  - Pre-process as part of the pipeline

Modify Original Text

Drop newline, carriage returns
Replace missing notes with a space

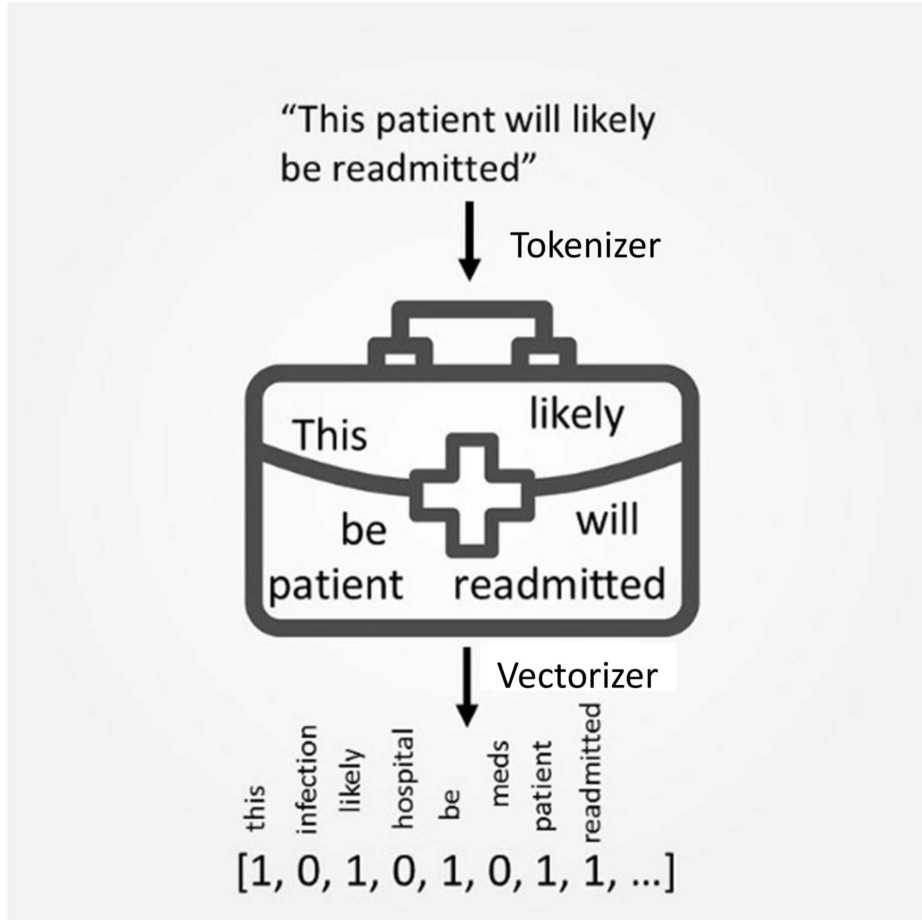
```
In [12]: def preprocess_text(df):
             # This function preprocesses the text by filling
             df.TEXT = df.TEXT.fillna(' ')
             df.TEXT =df.TEXT.str.replace('\n',' ')
             df.TEXT =df.TEXT.str.replace('\r',' ')
             return df
```

```
In [13]: # preprocess the text to deal with known issues
         df_train = preprocess_text(df_train)
         df_valid = preprocess_text(df_valid)
         df_test = preprocess_text(df_test)
```

Bag-of-words

- Split a note into tokens (i.e. words) then 'count' the number of each token
- Use these 'counts' as feature columns

- Note: different techniques for 'counts'



Bag-of-words design choices

- How to preprocess the words into tokens
- How to count the tokens
- Which tokens to use

Build a tokenizer

```
In [14]: import nltk
         from nltk import word_tokenize
         word_tokenize('This should be tokenized. 11/01/2018 sentence has stars**')

Out[14]: ['This',
          'should',
          'be',
          'tokenized',
          '.',
          '11/01/2018',
          'sentence',
          'has',
          'stars**']
```

- Sentence is tokenized by spaces and some punctuation but not all.
- Numbers are also still included
- ‘This’ would be considered a different token than ‘this’
- ‘stars**’ would be different than ‘stars’

Build a custom tokenizer

- Replace punctuation with spaces
- Replace numbers with spaces
- Lowercase all words

```
In [15]: import string
         print(string.punctuation)

!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~

In [16]: def tokenizer_better(text):
             # tokenize the text by replacing punctuation and numbers with spaces and lowercase all words

             punc_list = string.punctuation+'0123456789'
             t = str.maketrans(dict.fromkeys(punc_list, " "))
             text = text.lower().translate(t)
             tokens = word_tokenize(text)
             return tokens

In [17]: tokenizer_better('This should be tokenized. 11/01/2018 sentence has stars**')

Out[17]: ['this', 'should', 'be', 'tokenized', 'sentence', 'has', 'stars']
```

Fast way to replace characters with spaces

Build a simple vectorizer

```
In [18]: sample_text = ['Open Data Science Conference is about learning',
                        'Data data DATA',
                        'Learning is part of data science']
```

```
In [19]: from sklearn.feature_extraction.text import CountVectorizer
         vect = CountVectorizer(tokenizer = tokenizer_better)
         vect.fit(sample_text)

         # matrix is stored as a sparse matrix (since you have a lot of zeros)
         X = vect.transform(sample_text)
```

Specify custom tokenizer

- CountVectorizer is the simplest method for bag-of-words
- Counts the number of occurrences of each word

- Other common method is TfidfVectorizer which takes into account frequency of word usage across notes

Build a simple vectorizer

```
In [18]: sample_text = ['Open Data Science Conference is about learning',
                        'Data data DATA',
                        'Learning is part of data science']
```

```
In [19]: from sklearn.feature_extraction.text import CountVectorizer
         vect = CountVectorizer(tokenizer = tokenizer_better)
         vect.fit(sample_text)

         # matrix is stored as a sparse matrix (since you have a lot of zeros)
         X = vect.transform(sample_text)
```

```
In [20]: X
Out[20]: <3x9 sparse matrix of type '<class 'numpy.int64'>'
                 with 14 stored elements in Compressed Sparse Row format>
```

```
In [21]: # we can visualize this small example if we convert it to an array
         X.toarray()
Out[21]: array([[1, 1, 1, 1, 1, 0, 1, 0, 1],
                [0, 0, 3, 0, 0, 0, 0, 0, 0],
                [0, 0, 1, 1, 1, 1, 0, 1, 1]], dtype=int64)
```

```
: # get the column names
  vect.get_feature_names()
: ['about',
   'conference',
   'data',
   'is',
   'learning',
   'of',
   'open',
   'part',
   'science']
```

Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

#ODSC

Train clinical vectorizer

```
In [23]: from sklearn.feature_extraction.text import CountVectorizer
         vect = CountVectorizer(max_features = 3000, tokenizer = tokenizer_better)

         # this could take a while
         vect.fit(df_train.TEXT.values)

Out[23]: CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                 dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                 lowercase=True, max_df=1.0, max_features=3000, min_df=1,
                 ngram_range=(1, 1), preprocessor=None, stop_words=None,
                 strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                 tokenizer=<function tokenizer_better at 0x000001CA1DA7A400>,
                 vocabulary=None)
```

Good practice to specify the max_features (otherwise it could take a long time with big data set)
Size of max_features is then a hyperparameter to tune

Stop words

“the”, “is”, “are”, “and”

- Stop word – commonly used words with little value to ML model
- Frequency of word use depends on domain (clinical, twitter, Wikipedia)

Stop words

https://www.linkedin.com/pulse/another-twitter-sentiment-analysis-python-part-2-ricky-kim/

Fast technique for finding term frequency

In [24]:
```
neg_doc_matrix = vect.transform(df_train[df_train.OUTPUT_LABEL == 0].TEXT)
pos_doc_matrix = vect.transform(df_train[df_train.OUTPUT_LABEL == 1].TEXT)
neg_tf = np.sum(neg_doc_matrix,axis=0)
pos_tf = np.sum(pos_doc_matrix,axis=0)
neg = np.squeeze(np.asarray(neg_tf))
pos = np.squeeze(np.asarray(pos_tf))

term_freq_df = pd.DataFrame([neg,pos],columns=vect.get_feature_names()).transpose()
term_freq_df.columns = ['negative', 'positive']
term_freq_df['total'] = term_freq_df['negative'] + term_freq_df['positive']
term_freq_df.sort_values(by='total', ascending=False).iloc[:10]
```

Out[24]:

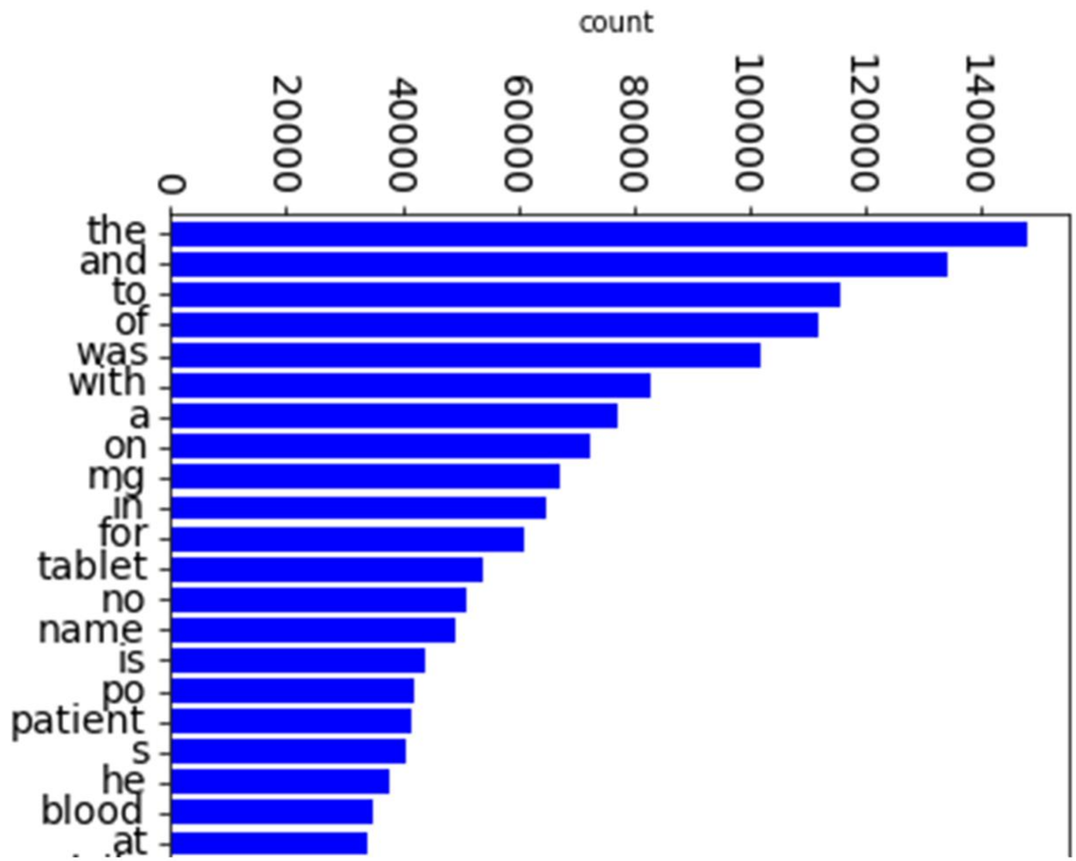
	negative	positive	total
the	71054	76756	147810
and	62455	71658	134113
to	53226	62085	115311
of	51303	60491	111794
was	48074	53521	101595
with	38036	44583	82619
a	35428	41629	77057
on	32290	39765	72055
mg	27718	39045	66763
in	29567	34755	64322

More medications predictive of readmission?

Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

#ODSC

Stop words




```
In [26]: my_stop_words = ['the','and','to','of','was','with','a','on','in','for','name',
                           'is','patient','s','he','at','as','or','one','she','his','her','am',
                           'were','you','pt','pm','by','be','had','your','this','date',
                          'from','there','an','that','p','are','have','has','h','but','o',
                          'namepattern','which','every','also']
```

Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

#ODSC

Build a vectorizer removing stop words

```
In [27]: from sklearn.feature_extraction.text import CountVectorizer
         vect = CountVectorizer(max_features = 3000,
                                tokenizer = tokenizer_better,
                                stop_words = my_stop_words)
         # this could take a while
         vect.fit(df_train.TEXT.values)
```

```
Out[27]: CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                 dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                 lowercase=True, max_df=1.0, max_features=3000, min_df=1,
                 ngram_range=(1, 1), preprocessor=None,
                 stop_words=['the', 'and', 'to', 'of', 'was', 'with', 'a', 'on', 'in', 'for', 'name', 'is', 'patient', 's', 'h
         e', 'at', 'as', 'or', 'one', 'she', 'his', 'her', 'am', 'were', 'you', 'pt', 'pm', 'by', 'be', 'had', 'your', 'this',
         'date', 'from', 'there', 'an', 'that', 'p', 'are', 'have', 'has', 'h', 'but', 'o', 'namepattern', 'which', 'every', '
         also'],
                 strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                 tokenizer=<function tokenizer_better at 0x000001CA1DA7A400>,
                 vocabulary=None)
```

Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

Create X, y

```
In [28]: X_train_tf = vect.transform(df_train.TEXT.values)
         X_valid_tf = vect.transform(df_valid.TEXT.values)
```

Get labels

```
In [29]: y_train = df_train.OUTPUT_LABEL
         y_valid = df_valid.OUTPUT_LABEL
```

Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

Part 3: How to build a simple predictive model using a bag-of-words approach

Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

#ODSC

Logistic Regression

- Traditional Machine Learning algorithm
- Works well with sparse matrices
- Fast to train
- Interpretable

Logistic Regression

```
In [30]: # logistic regression
         from sklearn.linear_model import LogisticRegression
         clf=LogisticRegression(C = 0.0001, penalty = 'l2', random_state = 42)
         clf.fit(X_train_tf, y_train)

         C:\Users\Andy\AppData\Local\conda\conda\envs\odsc_west_2018\lib\site-packag
         utureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
           FutureWarning)

Out[30]: LogisticRegression(C=0.0001, class_weight=None, dual=False,
                   fit_intercept=True, intercept_scaling=1, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2', random_state=42,
                   solver='warn', tol=0.0001, verbose=0, warm_start=False)
```

Hyperparameter C is helps control the effect of regularization
We will discuss how to optimize C

Tip: the same C usually doesn’t work as well for both CountVectorizer and TfidfVectorizer

Predictions

```
In [31]: model = clf
         y_train_preds = model.predict_proba(X_train_tf)[:,1]
         y_valid_preds = model.predict_proba(X_valid_tf)[:,1]
```

```
In [32]: print(y_train[:10].values)
         print(y_train_preds[:10])

[1 1 0 1 1 1 0 0 1 1]
[0.76307111 0.63114288 0.29772094 0.77926068 0.59694889 0.55643044
 0.36292154 0.90984735 0.47806099 0.67622763]
```

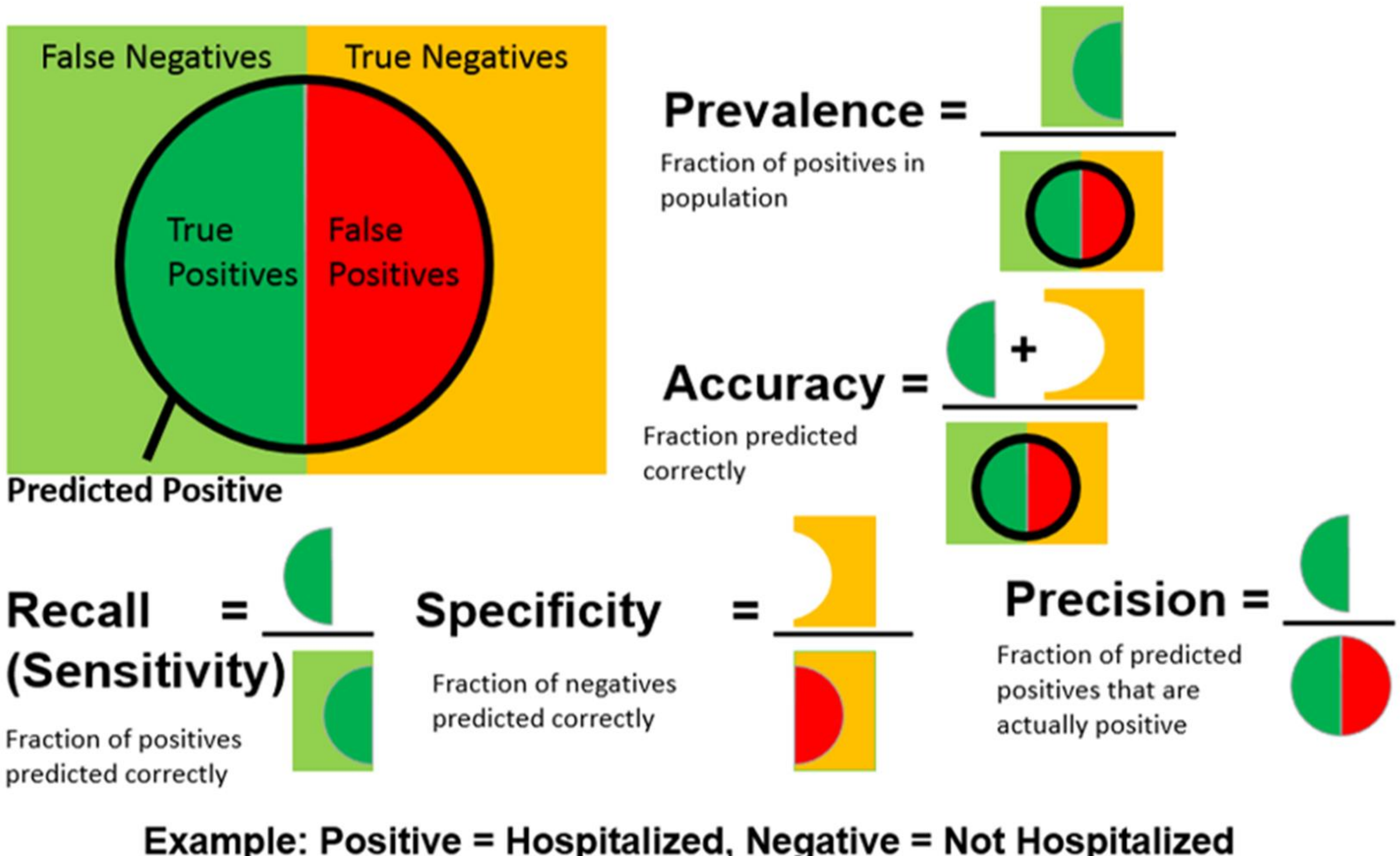
Part 4: How to assess the quality of your model

Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

#ODSC

https://towardsdatascience.com/understanding-data-science-classification-metrics-in-scikit-learn-in-python-3bc336865019

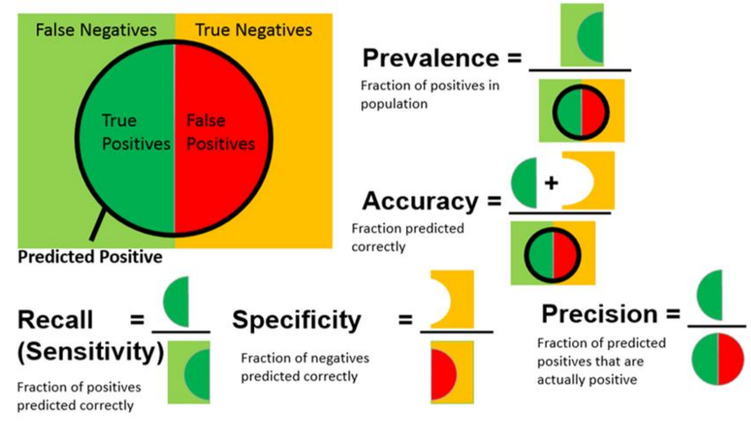
Performance Metrics

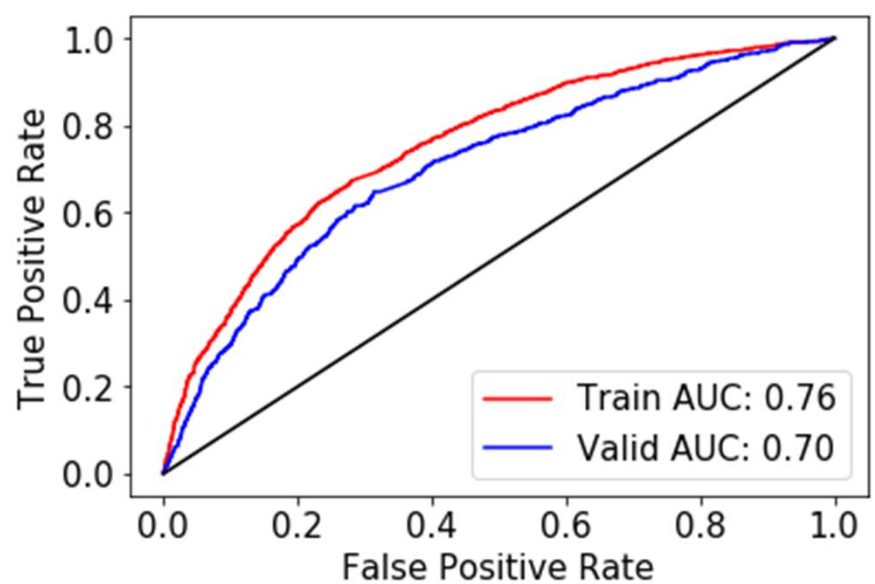


Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

Performance

Metric	Training	Validation
Prevalence	**50%**	**5.7 %**
Accuracy	69.5%	68.2 %
Recall	66.6 %	64.8 %
Precision	**70.6 %**	**11.0 %**
Specificity	72.3 %	68.4 %
Area Under ROC Curve (AUC)	0.757	0.704





Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

#ODSC

Part 5: How to decide the next step for improving the model

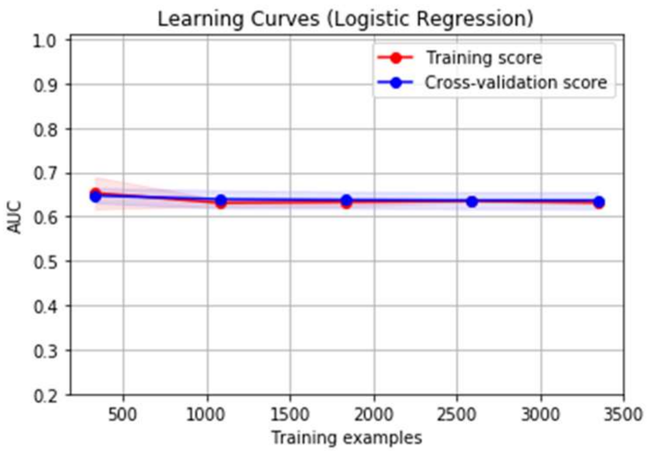
Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

#ODSC

Design decisions

- Which and how much data to use? Should we spend time collecting more data?
- How to tokenize?
  - Should we use stemming? (“stemming” → “stem”)
- How to vectorizer?
  - Change number of words?
  - Switch to tfidfvectorizer?
- How to select hyperparameters in Logistic regression?
- Should we switch to a different ML model?

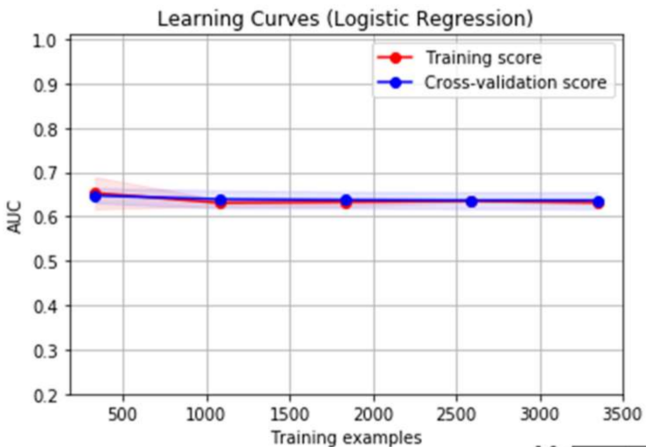
Learning Curve (diagnose bias/variance)



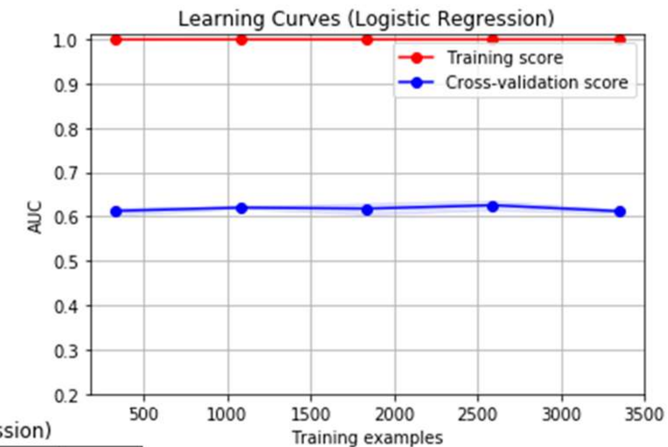
High Bias

High Variance

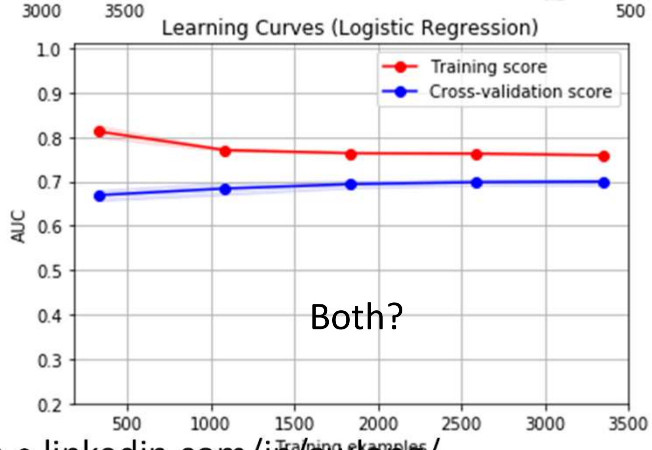
Learning Curve (diagnose bias/variance)



High Bias



High Variance



Both?

<u>Adding more samples</u>
<u>Probably won't help majorly</u>
<u>at this point</u>

Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

#ODSC

Helpful techniques

- Techniques for reducing bias (underfitting)
  - Add new features
  - Increase model complexity
  - Reduce regularization
  - Change model architecture

- Techniques for reducing variance (overfitting)
  - Add more samples
  - Add regularization
  - Reduce number of features
  - Decrease model complexity

  - Add better features
  - Change model architecture

Helpful techniques

- Techniques for reducing bias (underfitting)
  - Add new features
  - Increase model complexity
  - Reduce regularization
  - Change model architecture

- Techniques for reducing variance (overfitting)
  - Add more samples
  - Add regularization
  - Reduce number of features
  - Decrease model complexity

  - Add better features
  - Change model architecture

Feature Importance

More Negative

Logistic regression coefficient

More positive

Helpful techniques

- Techniques for reducing bias (underfitting)
  - Add new features
  - Increase model complexity
  - Reduce regularization
  - Change model architecture

- Techniques for reducing variance (overfitting)
  - Add more samples
  - Add regularization
  - Reduce number of features
  - Decrease model complexity

  - Add better features
  - Change model architecture

Hyper parameter tuning

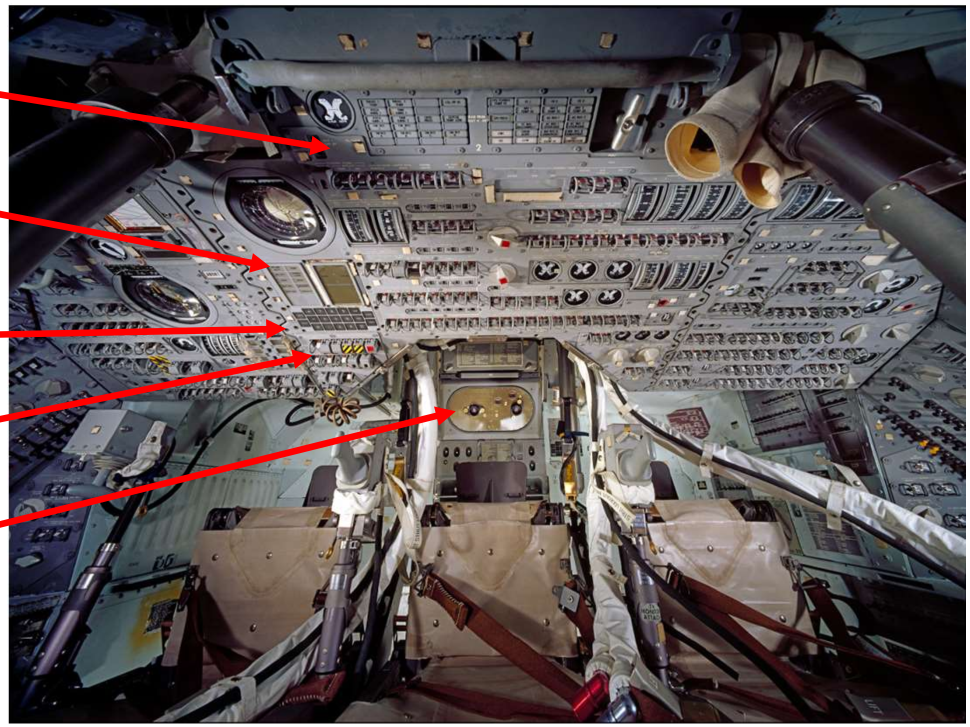
Regularization C

Stemming
(hospitalizations → hospital)

CountVectorizer or
Tfidfvectorizer

max_features

N grams

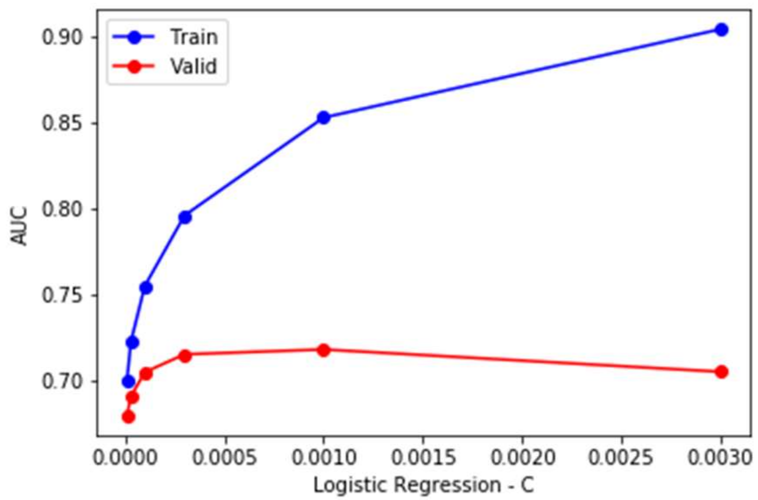


Source: https://airandspace.si.edu/multimedia-gallery/5128hjpg

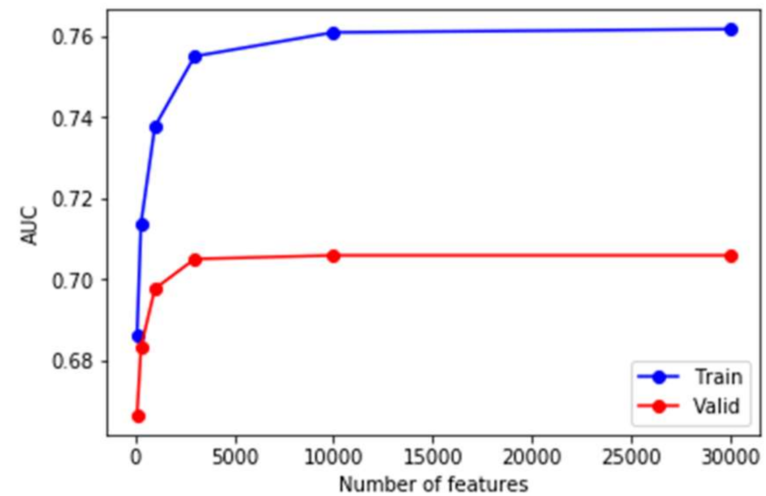
Source: https://en.wikipedia.org/wiki/
Apollo_(spacecraft)

Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

Hyper parameter tuning



Higher C = more overfitting



Higher max_features = more overfitting

Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

#ODSC

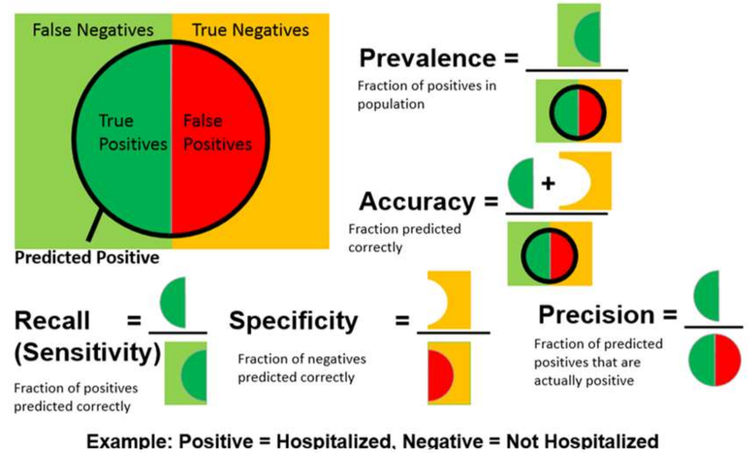
Model Architecture

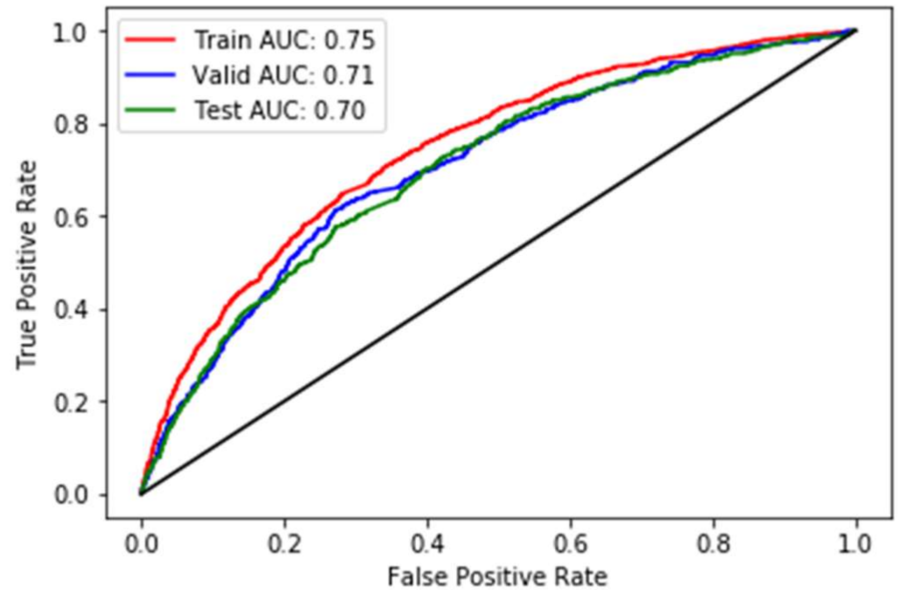
- Naïve Bayes
- Neural Networks (CNN, RNN) with Word2Vec

Final Model (excluding deaths)

Metric	Training	Validation	Test
Prevalence	50%	6.9%	6.6%
Accuracy	0.683	0.672	0.681
Recall	0.644	0.651	0.607
Precision	0.698	0.129	0.120
Specificity	0.722	0.674	0.686
Area Under ROC Curve (AUC)	0.745	0.709	0.704

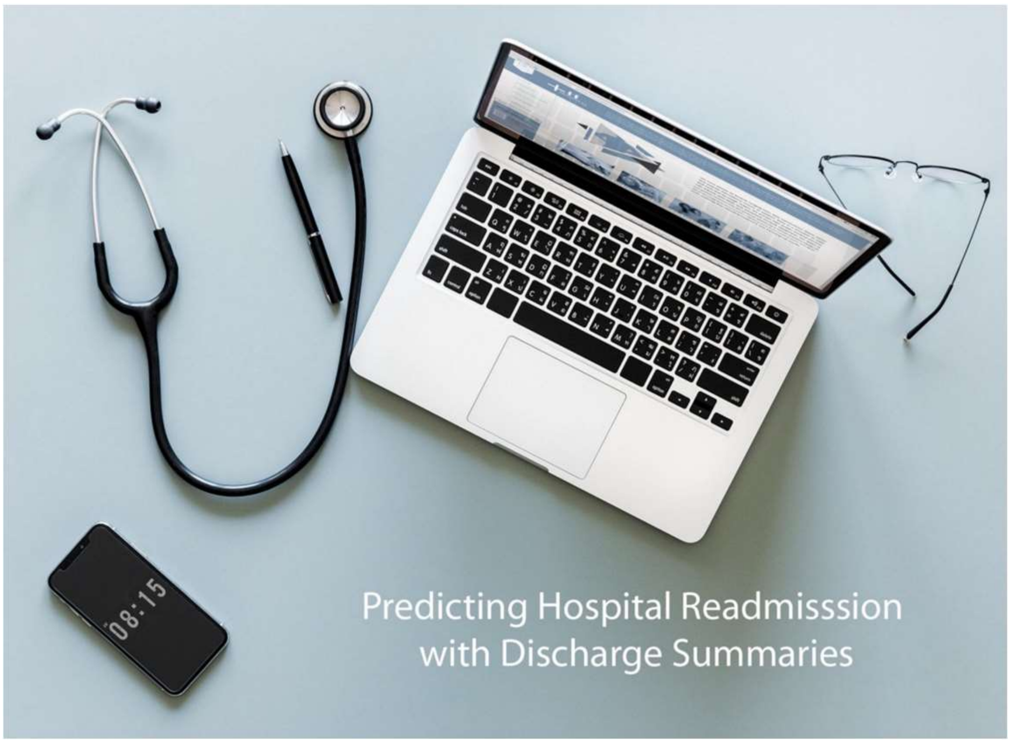
30-day unplanned readmission (AUC = 0.75–76)
(Rajkomar et al 2017)





Andrew Long • awlong20@gmail.com • linkedin.com/in/awlong/
https://github.com/andrewwlong/odsc_west_2018

#ODSC



Introduction to Clinical Natural Language Processing

Andrew Long

https://towardsdatascience.com/introduction-to-clinical-natural-language-processing-predicting-hospital-readmission-with-1736d52bc709