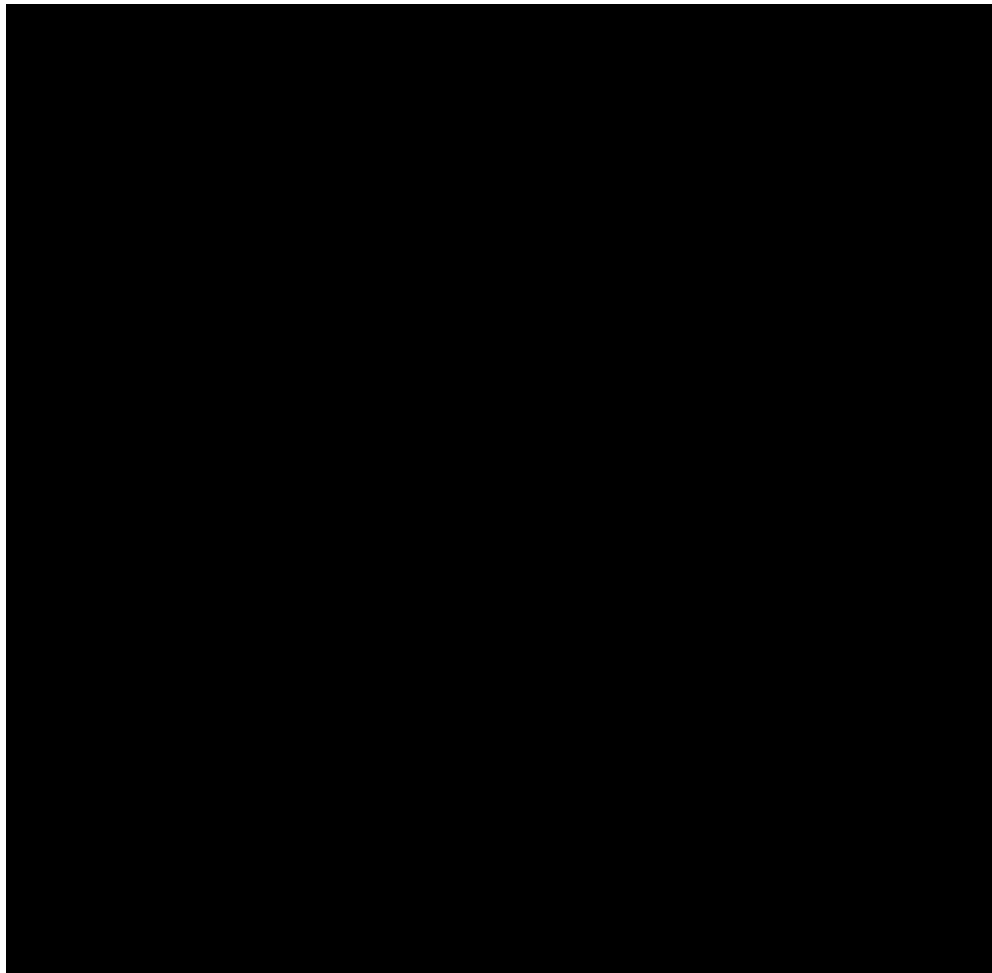


Spring 2019 Capstone Project

Progress Report 1

Algorithmic Comment Processing

March 25, 2019



* In alphabetical order

Table of Contents

1. Introduction

1.1. Project Scope and Goal

1.2. Concept Slide

2. PDF Ingestion

3. Feature Engineering

3.1. HTML Tag-based features

3.2. Text-based features

4. Modelling Approach

4.1. Target Variable: Manual Labelling

4.2. Train, Test splits and validation

4.3. Models used

4.4. Data Preparation

4.5 Putting it all together: Scikit-learn pipeline

4.6 Cross Validation and Parameter Tuning

4.7 Evaluation Metrics

4.8 Results

5. Conclusion and Next Steps

1 Introduction

1.1 Project Scope and Goal

This Capstone project is a collaboration between Columbia University at the Data Science Institute and KPMG. The objective of this Capstone Project is to build a model by leveraging Machine Learning and Natural Language Processing to automate comment processing. Comment processing involves manually reading documents and summarising the text in sections. Automating this time consuming and error prone process would make parsing through a large volume of documents extremely convenient and time-efficient.

This project has two modules, Module I comprises PDF ingestion and machine learning based document sectioning and Module II comprises comment summarization. Progress Report 1 focuses on Module I, which entails building a classification model to identify document sectioning using both structural and text features from HTML converted PDF files.

Our focus on this project is on regulatory comment data from Federal Government agencies and dataset are available as PDF files. First, we parse the PDF files to HTML format because markup languages such as HTML and XML may provide a more efficient and accurate way of extracting the metadata to a tabulated format.

1.2. Concept Slide

Conceptually,
what we do is...

Letters to
Fed. Gov.



119
PDFs

summarize comments for section 1

summarize comments for section 2

summarizing comments for section 3

summarizing comments for section 4

A. Definition of A "Closely Held" For-Profit Corporation Eligible for the Accommodation

1. None of the shares are publicly owned or offered

We support the Department's proposal that any corporation with shares that are publicly traded or offered for public trading, including publicly-traded companies in the definition of an eligible for-profit would likely include companies with shareholders who did not share a sincerely held religious belief. This definition would then not reflect the religious beliefs of all equity holders, as required by *Hobby Lobby*.

2. There is an expression of religious belief guiding the company's operation

Adherence to the company's religious belief is a necessary condition for the company to be eligible for the accommodation. The company must take the required corporate action asserting the accommodation. This step would follow whatever state law requires for a board to take an action for the corporation.

3. All equity holders must agree to the accommodation

For the unity of the company will be agreement in accordance with *Lobby*.

B. Valid Corporate Action

1. Equity holders must agree to the accommodation

We propose the order to assert the accommodation.

1. First, all resolution methods holders' manages governing corporate religious

C. Enforcement and Oversight

As with the entire ACA, the proposed rule for the accommodation must include oversight and enforcement. The Departments must ensure that employers, TPAs, and issuers meet the requirements of the accommodation so that women have full access to the coverage guaranteed to them under the women's preventive care benefit. We urge the Departments to be vigilant in their oversight and enforcement efforts for the proposed expanded accommodation.

D. Other Aspects of the July 2013 Final Regulations That Need to Be Modified, Enforced or Reiterated In Light of the Addition to the Definition of Eligible Organization

We applaud the Departments' efforts to ensure that the accommodation process works effectively, that women receive seamless coverage, and that the beneficiaries' legal protections remain in place.

Algorithmically,
what we do is...



Conversion
to 119 HTMLs

run Random Forest,
XGBoost, and etc.

Top: a distance from the top outer edge in pixel

regardless of their employers' religious objections. However, we agree that the Departments take whatever steps are possible to mitigate the potential harm of this decision, and if path is to follow the Court's suggestion, as these proposed rules would do. In doing so, Departments must maintain their focus on how to ensure that as few women as possible of the contraceptive coverage guarantee or face discriminatory burdens in making an coverage and accessing needed care.

LS: a line space in pixel between two consecutive lines.

Scope of the Accommodation

Leading Char Upper: 1 if line begins with an uppercase, 0 otherwise

In the proposed rules, the Departments lay out a framework for how to determine whether entities are eligible for an expanded accommodation. That framework first defines a "closely held for-profit entity" and then requires that the entity's religious objection to coverage be established by specific corporate action in accordance with applicable state corporate governance. Font-weight: 1 if bold, 0 if normal.

The Guttmacher Institute urges the Departments to tailor these definitions and require narrowly as possible to match the intent of the Court's decision. As the Departments

Document
sectioning

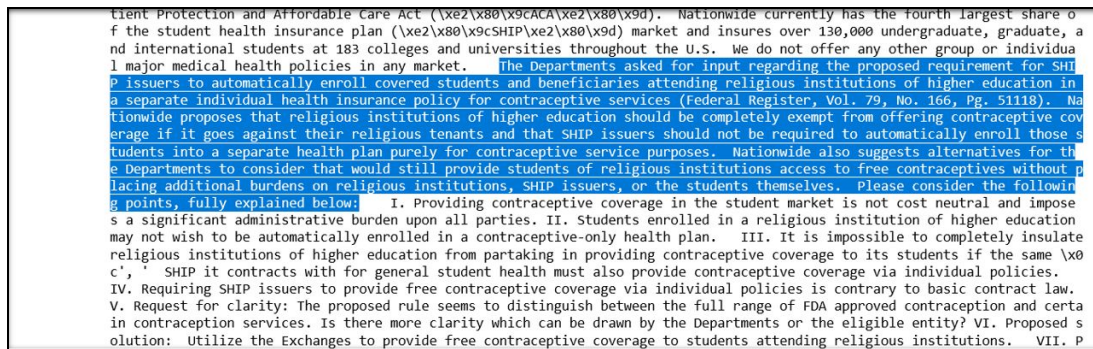
extract summary of
each section

- Section summary text
- Section raw text
- Document ID
- Section ID

Section
summarization

2 PDF Ingestion

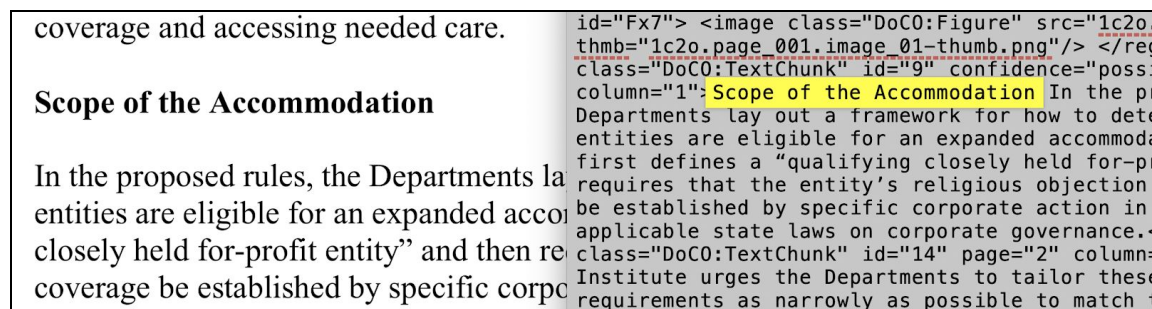
The first task of the project was PDF ingestion. In the beginning, we worked on the PDF ingestion using the pdftotree, pdfminer, and slate python packages and able to convert PDF to text using both pdfminer and slate packages. Fig.1 shows the sample output using the slate package and we can only see the different white spaces between the paragraphs and could not find a reliable way to identify the sections of the document.



lient Protection and Affordable Care Act (\xe2\x80\x9cACA\xe2\x80\x9d). Nationwide currently has the fourth largest share of the student health insurance plan (\xe2\x80\x9cSHIP\xe2\x80\x9d) market and insures over 130,000 undergraduate, graduate, and international students at 183 colleges and universities throughout the U.S. We do not offer any other group or individual major medical health policies in any market. The Departments asked for input regarding the proposed requirement for SHIP issuers to automatically enroll covered students and beneficiaries attending religious institutions of higher education in a separate individual health insurance policy for contraceptive services (Federal Register, Vol. 79, No. 166, Pg. 51118). Nationwide proposes that religious institutions of higher education should be completely exempt from offering contraceptive coverage if it goes against their religious tenants and that SHIP issuers should not be required to automatically enroll those students into a separate health plan purely for contraceptive service purposes. Nationwide also suggests alternatives for the Departments to consider that would still provide students of religious institutions access to free contraceptives without placing additional burdens on religious institutions, SHIP issuers, or the students themselves. Please consider the following points, fully explained below: I. Providing contraceptive coverage in the student market is not cost neutral and imposes a significant administrative burden upon all parties. II. Students enrolled in a religious institution of higher education may not wish to be automatically enrolled in a contraceptive-only health plan. III. It is impossible to completely insulate religious institutions of higher education from partaking in providing contraceptive coverage to its students if the same \xc3\x9c, SHIP it contracts with for general student health must also provide contraceptive coverage via individual policies. IV. Requiring SHIP issuers to provide free contraceptive coverage via individual policies is contrary to basic contract law. V. Request for clarity: The proposed rule seems to distinguish between the full range of FDA approved contraception and certain contraception services. Is there more clarity which can be drawn by the Departments or the eligible entity? VI. Proposed solution: Utilize the Exchanges to provide free contraceptive coverage to students attending religious institutions. VII. P

Figure 1: pdf to text output using slate packages.

Then we converted PDFs to XMLs using a pdf2xml package, called pdfx¹. Despite of its fully-automated functionality and reasonably-fast conversion time of approximately 3 documents per minute, a converted XML file lost important information such as font style (e.g. bold) and blank line. This is demonstrated in Fig.2 in that no clue exists for a blank line between section title, Scope of the Accommodation, and the first line of the following paragraph. We also found that a tag for bold was not included in the xml file.



coverage and accessing needed care.

Scope of the Accommodation

In the proposed rules, the Departments lay out a framework for how to determine if an entity is eligible for an expanded accommodation. The Departments first defines a “qualifying closely held for-profit entity” and then requires that the entity’s religious objection be established by specific corporate action in applicable state laws on corporate governance. The Institute urges the Departments to tailor these requirements as narrowly as possible to match

Figure 2 : pdf to xml output using the pdfx package. (left: a PDF file, right: an XML file)

¹ Provided by School of Computer Science, the University of Manchester via <http://pdfx.cs.man.ac.uk>.

This dramatically discouraged us to use XML files and we decided to move on the next exploration of converting tasks: from PDFs to HTMLs². For record, here is a snippet of the conversion code used. This was tested and used on MacOS. All Python scripts were interpreted by Python 3.7.

File name: convertf_to_xml.sh

```
find . -name "*.pdf" | while read file; do
curl --data-binary @$file -H "Content-Type: application/pdf" -L
"http://pdfx.cs.man.ac.uk" > "${file}.xml";
done
```

Execution: bash conver_pdf_to_xml.sh

Figure 3: a shell script to convert PDF files to XML files

Finally, all PDF files were converted into HTML files, using pdftohtml package, in an automated fashion by running a Python script. This script can be done through command line or python script. For simplicity and effectiveness we implemented this script using python language as an interface that would allow us to include it with the rest of the project. Also, this implies we can deploy this conversion as a part of one completed application without human intervention.

We selected pdftohtml³ package for its easy implementation and license terms' advantages. This software is under GNU General Public License version 2.0 (GPLv2).

```
import os
echo = "pdftohtml"
for filename in os.listdir("pdf_files"):
    if filename.endswith("pdf"):
        file_pdf = "pdf_files/" + filename
        file_html = "html_files/" + filename
```

Figure 4: Implementation code in Python. CPU times: user 9.64 ms, sys: 45.5 ms, total: 55.2 ms. Wall time: 2min 47s.

In order to read the information obtained from the files converted from pdf to html we used BeautifulSoup. This is a python package and the conversion successfully reserved

² We cannot conclude that every pdf2xml converter has the same limitation. Missing information could have remained intact with a more sophisticated converter. We chose not to explore another possibility of pdf2xml converters because we were unable able to find any that satisfied two criteria: a support for command-line execution for automation and free license

³ This package can be found <http://pdftohtml.sourceforge.net/>.

precise detailed layouts of the original PDFs in pixel-precision. In other words, HTMLs contain exact positions of each sentence in unit of pixel. This helped us extract structural features, which we believe value has been added in addition to textural features. For example, line spaces between two sentences should well complement first-three-token features to identify a breaking point of sections. In addition, it gives information about the type of font and its size. We extracted the information from the files using the Python script in figure 5.

```
features = []
for root, dirs, files in os.walk("html_files"):
    for file in files:
        if file.endswith(".html"):
            if file != "index.html":
                try:
                    file_path = os.path.join(root, file)
                    #print(file_path+ " found")
                    df = extract_features(file_path)
                    df = df.assign(page=file)
                    df = df.assign(document=root)
                    features.append(df)
                except:
                    print(file_path+" NOT found")
```

Figure 5: Implementation on how extract info from HTML files.

CPU times: user 3 μ s, sys: 1 μ s, total: 4 μ s. Wall time: 6.91 μ s.

The information that was extracted from the HTML files helped us to build the extra features we use in our models. You can observe in Figure 6 the way the document looks after converted to HTML.

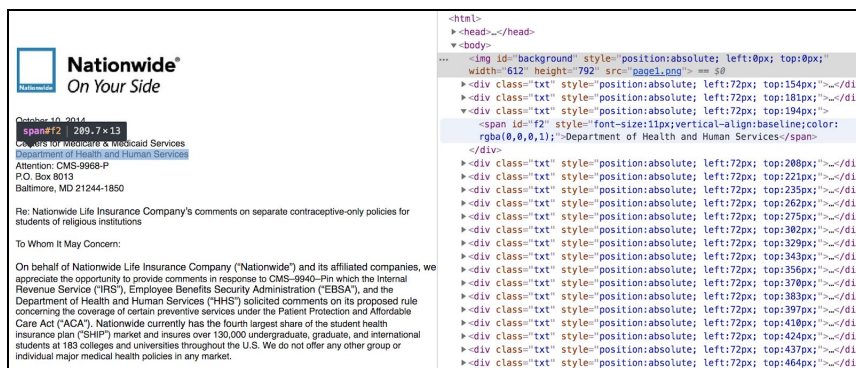


Figure 6: Left side, original document. Right side, HTML rendering code.

3 Feature Engineering

3.1 Features illustration

Some features that we used can be best illustrated by pictures than letters only. We have a full list of features in the subsequent sections, the following two exemplary documents from the original dataset used should help readers intuitively understand what features mean. Note that a word appearing before a colon(:) is a feature name in red color, followed by its description.

(a)

Top: a distance from the top outer edge in pixel

regardless of their employers' religious objections. However, we agree that the Departments take whatever steps are possible to mitigate the potential harm of this decision, and the path is to follow the Court's suggestion, as these proposed rules would do. In doing so, Departments must maintain their focus on how to ensure that as few women as possible are affected by the contraceptive coverage guarantee or face discriminatory burdens in making use of coverage and accessing needed care.

LS: a line space in pixel between two consecutive lines.

Scope of the Accommodation

Leading_Char_Upper: 1 if line begins with an uppercase, 0 otherwise

Left: a distance from the left outer edge in pixel

In the proposed rules, the Departments lay out a framework for how to determine whether entities are eligible for an expanded accommodation. That framework first defines a "closely held for-profit entity" and then requires that the entity's religious objection to coverage be established by specific corporate action in accordance with applicable state corporate governance. **Font-weight: 1 if bold, 0 if normal.**

The Guttmacher Institute urges the Departments to tailor these definitions and requirements narrowly as possible to match the intent of the Court's decision. As the Departments

* Document name: CMS_2014_0115_9981.pdf

(b)

We commend the agencies for moving to a plan-based exemption. However, we suggest that the Rules be revised in several respects to clarify the application of the Rules to multiple employer church plans.

Ends_In_Period: 1 if a line ends with a period(.), 0 otherwise.

I. Expand the Religious Exemption in Section 147.132

Roman_Numeral: 1 if a line begins with a Roman number followed by a period(.), 0 otherwise.

While the Rules provide for only one exemption, the exemption appears to have two parts. One part applies at the plan level and exempts the plan and any issuer providing coverage under the plan. The other part exempts certain employers that adopt an exempt plan. The two parts may

* Document name: CMS_2014_0115_51425.pdf

Figure 7: Feature examples (a) Top, LS, Leading_Char_Upper, Left and Font-weight
(b) Ends_In_Period, Roman_Numeral

After having converted PDFs to HTMLs, we extracted information from each HTML file using BeautifulSoup package⁴. On the one hand, we used the information directly as features. Font-size and font-weight (e.g. bold) may be good examples. On the other hand, more importantly, this HTML-tag-based information helped us engineer features: line spaces between two consecutive lines (**LS**), a distance from the left outer edge in pixel(**Left**) are representative use-cases of this indirect usage. Also, class *id* in HTML tags provided us a compressed information about not only font-relevant characteristics, but also human's intention in decorating a document to some extent. Those HTML-based features are described in the table 1 below. Whereas detailed descriptions for all features including HTML-based ones are stated in the subsequent sections, finally we note that *text* feature contains all the letters (, numbers and everything) appearing on each line. We would like to emphasize our decision process to choose to convert PDFs to machine-readable format by exploring pdf2text, pdf2xml and pdf2html.

Table 1: HTML-tag-based features

Feature category	Features
Continuous	<ul style="list-style-type: none"> • Font-size • Left • Top
Categorical	<ul style="list-style-type: none"> • Vertical-align • Font-family • Font-weight • font-style
Text-based	<ul style="list-style-type: none"> • Text

3.2 Text-based features

In our tabulated data, each row corresponds to one line of the pdf document. The text features were selected by visual inspection of section headings and perception. For example, *Leading_Char_Period*, *Leading_Roman_Period*, and *Leading_Number_Period* are important to identify sections because most of the sections are starting with particular letter or number such as "A.", "1.", or "I.". Font-size, font-weight, relative position within a document and linguistic markers are good features to model. Also going over sentences, a section title would be confusing an ML module, therefore few

⁴ <https://www.crummy.com/software/BeautifulSoup>

tokens such as a few first words were used as features instead of the whole text in a section title. Text-based features which we used in models are summarized in Table 2.

Table 2 : Text_based features

Category	ID	Feature name	Description
Binary	B1	Leading_Char_Upper	A line starts with uppercase character
	B2	Leading_Numeral	A line starts with Arabic and Roman numeral
	B3	Ends_In_Period	A line ends with a period
	B4	Leading_Number_Period	A line starts with any numerical combinations followed by period
	B5	Leading_Char_Period	A line starts with any uppercase or lowercase character followed by period
	B6	Leading_Roman Numeral	A line starts with any Roman numeral
	B7	Contains("Summary")	A line starts with Summary or summary
	B8	Contains("Re")	A line starts with Re
	B9	Contains("Attention")	A line starts with Attention
	BA	Contains("Conclusion")	A line starts with Conclusion or conclusion
	BB	Roman_Period	A line starts with Roman numeral followed by period
Numerical	N1	Num_of_Words	The number of words in the line
	N2	Font_Size	Font size of the text in the line
	N3	No_of_Special_Char	The number of special characters in the line
Categorical	C1	Color	Color of the text in the line
	C2	Font_Style	Font style of the text in the line
	C3	Font_Weight	Font weight of the text (1 bold or 0 normal)
Textural	T1	First_Word	First word of the text line
	T2	First_Three_Words	First three words of the text line

3.3. Structure-based and other features

In addition to text-based features, structural features are also important to explore. For example, we can think of a line space and pixel-distance from the leftmost on a paper. These two features may be used with other textural features such as Num_Words or Contains(). Then, we expect the probability that a section is identified would increase more than marginally. Structural features are summarized in Table 3.

Table 3 : Structure-based features

Category	ID	Feature name	Description
Structural	S1	LS	A line space between previous and current lines. Engineered from Top which is not directly used
	S2	Left_Px	A distance in pixel from the left outer edge of a document
	S3	Top	A distance in pexel from the top outer edge of a document
Other	O1	Document	File name
	O2	Id	Class id in HTML containing all the information on fonts

4 Modelling Approach

Our Target-Feature Matrix currently consists of 7744 rows. Here each row corresponds to one line (not a sentence) in a PDF document. Hence, any machine learning model we build makes predictions at the level of a single line. The task at hand is a binary classification, where a line containing a section header corresponds to 1, while a line not containing a section header corresponds to 0.

This section discusses our modelling approaches and discusses some initial results we achieved at the first stage of modelling. These are not the final results of the team,

since parameter tuning and optimization will continue after the completion of this report.

4.1 Target variable: Manual Labelling

Each line is assigned to two class labels {1,0} where 1 is a section start line and 0 is non-section start line. If a section header corresponds to two lines then we only label the first line as 1 and the second line as 0. Each line of the matrix was manually read by the team and labelled 1 or 0 by comparing it to the PDF document. 50 such documents were labelled corresponding to 7744 rows. This generated our entire training data.

4.2 Train, Test splits and validation

Since we were required to have an independent test size of 30%, we split the data such that we follow the guidelines. The split was as follows:

1. Train Data: 5420 rows corresponding to 37 documents
2. Test Data: 2324 rows corresponding to 13 documents

This split of test and train set was not necessarily random. The first 37 documents that had the target variable manually labelled were used in the training set to tune parameters and evaluate model performance. Different cross-validation strategies were used to tune the parameters and evaluate model performance. The 13 documents in the test set were labelled afterwards to create an independent test set. The test set was not touched until parameters were fully tuned.

4.3 Models used

Table 4 summarizes the modeling approaches that the team considered. The left-most column highlights the modelling algorithm. The two right columns highlight the advantages and disadvantages of using those models.

Table 4: Machine Learning Models under consideration⁵.

Algorithm	Advantages	Disadvantages
Random Forest (RF) Classification or regression model that improves the accuracy of a simple decision tree by generating multiple decision trees and taking a majority vote of them to predict the output, which is a continuous variable (e.g. age) for a regression problem and a discrete variable (e.g. either black, white, or red) for classification	<ul style="list-style-type: none"> - Generally give high prediction accuracy; - Good for baseline; - Does not require much data cleaning. 	<ul style="list-style-type: none"> - Not as explainable as Logistic Regression
Extreme Gradient Boosting (XGBoost)⁶	<ul style="list-style-type: none"> - Produces high prediction accuracy; - Generally handles missing by selecting the best imputation method. 	<ul style="list-style-type: none"> - Not as explainable as Logistic Regression - Requires more parameter tuning.

4.4 Data Preparation

For a given document the following pre processing is applied.

4.4.1: Categorical Features

⁵ "An executive's guide to AI | McKinsey & Company." output <https://www.mckinsey.com/business-functions/mckinsey-analytics/our-insights/an-executives-guide-to-ai> Accessed 27 Jun. 2018.

⁶ Library: XGBoost, <https://xgboost.readthedocs.io/en/latest/>
Tuning framework: Analytics Vidhya, <https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/>

First, we used categorical features which is extracted from HTML tags such as `font_style`, `font_weight`, and `color`. We [OneHotEncode](#) them, which is a representation of categorical variables as binary vectors.

4.4.2: Numerical Features

We extracted numerical features such as font size, number of words, and number of special characters in each text line and the line space between previous and current lines.

4.4.3: Text-based Features

Text _based binary features are extracted using the raw text lines and summarized in Table 2. Instead of using the whole line of text we extracted `first_word`, `first 3_words`, and `first_5_words`. Then, by using this representation, a document section heading is characterized by a set of words that appear at the start of the text line. We used various Natural Language Processing (NLP) techniques such as BagofWords (i.e. [Countvectorizer](#) and [Tfidfvectorizer](#) in *scikit-learn*) with stop words, without stop words and n-grams to create features.

4.5 Putting it all together: Scikit-learn Pipeline

We combine all our preprocessing and modelling steps using a [Pipeline](#) in *scikit-learn*, where the preprocessing takes place in 3 [ColumnTransformers](#), one for each type of feature i.e. one for numeric, one for text-based and one for categorical features. This ensures that there is absolutely no leakage in any of the preprocessing steps from the train to the test set since the preprocessing for the train set is done independently and for the test set is done automatically when `score` is called. The pipeline can be passed into the `GridSearchCV` object (discussed in the next section) to tune parameters that are robust using the train set.

4.6 Cross_validation and Parameter Tuning

We use grid-search with 5 fold cross validation to find the optimal *hyperparameters* of our models and evaluate their robustness using [GridSearchCV](#) in *scikit-learn*, which allows us to learn the best parameters on independent validation set using K Fold

Cross validation. We also use other Cross-validation techniques such as [KFold](#), [ShuffleSplit](#) and [StratifiedShuffleSplit](#) to ensure our optimal parameters are robust.

Table 5: Hyperparameters used in the models

Model	Best hyperparameter(s)
Random Forest	N_estimators(110), max_depth(20), bootstrap(false), class_weight = 'balanced'
XGBoost	learning_rate(0.1), n_estimators(175), max_depth(10), min_child_weight(1), gamma(0.04), colsample_bytree(0.9), subsample(0.75), objective('binary:logistic'), colsample_bylevel(1), reg_alpha(1.2e-05), reg_lambda(0.00014), scale_pos_weight(1)

4.7 Evaluation Metrics

The standard precision, recall, and [F1-score](#) metrics were used to tune the parameters and optimize the performance of the model⁷. In this project, it would be more important to have **high recall**, specifically they emphasized that it would be **critical to capture the first and the last section** correctly because those are the areas where the author put their main arguments. Hence, we decide on a model that gives us a **recall of greater than 95%** while we can apply some relaxed criteria i.e. **greater than 80% for precision**.

4.7.1 Confusion Matrix

Table 6: Confusion Matrix

Confusion Matrix	Ground Truth (Individual is LTU)	
	True	False

⁷ Find the descriptions of these metrics in table 7.

Prediction	True	True Positive (TP)	False Positive (FP)
	False	False Negative (FN)	True Negative (TN)

4.7.2 Metrics in consideration

Table 7: Metrics in consideration

Metric	Details
Accuracy	Measured when threshold is set to 0.5
Confusion matrix	Measured when threshold is set to 0.5
AUC	Area under the ROC curve. Measured when threshold is set to 0.5
F1 Score	Harmonic Average of Precision and Recall. Measured when threshold is set to 0.5
Precision at multiple percentages	The team started by measuring precision at 50% and experimented with precision at 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%. Finally, we measure precision at 50%.
Recall at multiple percentages	The team started by measuring recall at 50% and experimented with precision at 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%. Finally, we measure recall at 50%.

4.8 Results

This section discusses some preliminary results we achieved at the first stage of modelling.

Table 8: Performance metrics for different models using different features

Model	Feature ⁸ (used in models)	Preci- sion	Recall	F1 score	Accu- racy
1.RF with GridSearchCV*	B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB N1 N2 N3 C1 C2 C3 T1 T2 S1 S2 O1 O2	0.79	0.51	0.62	0.99
2.RF with GridSearchCV*	B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB N1 N2 N3 C1 C2 C3 T1 T2 S1 S2 O1 O2	0.81	0.49	0.61	0.99
3.RF with GridSearchCV*	B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB N1 N2 N3 C1 C2 C3 T1 T2 S1 S2 O1 O2	0.79	0.53	0.62	0.99
4.RF with GridSearchCV*	B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB N1 N2 N3 C1 C2 C3 T1 T2 S1 S2 O1 O2	0.95	0.42	0.58	0.99
5.RF with GridSearchCV*	B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB N1 N2 N3 C1 C2 C3 T1 T2 S1 S2 O1 O2	0.43	0.62	0.51	0.98
6.RF with GridSearchCV*	B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB N1 N2 N3 C1 C2 C3 T1 T2 S1 S2 O1 O2	1	0.53	0.70	0.99
7.XGBoost without parameter tune	B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB N1 N2 N3 C1 C2 C3 T1 T2 S1 S2 O1 O2	0.91	0.44	0.59	0.99
8.XGBoost with tuned parameters* (1st trial)	B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB N1 N2 N3 C1 C2 C3 T1 T2 S1 S2 O1 O2	0.97	0.47	0.63	0.99

⁸ See tables in 3. Feature Engineering for exact feature names with descriptions.

* 5 fold cross validation

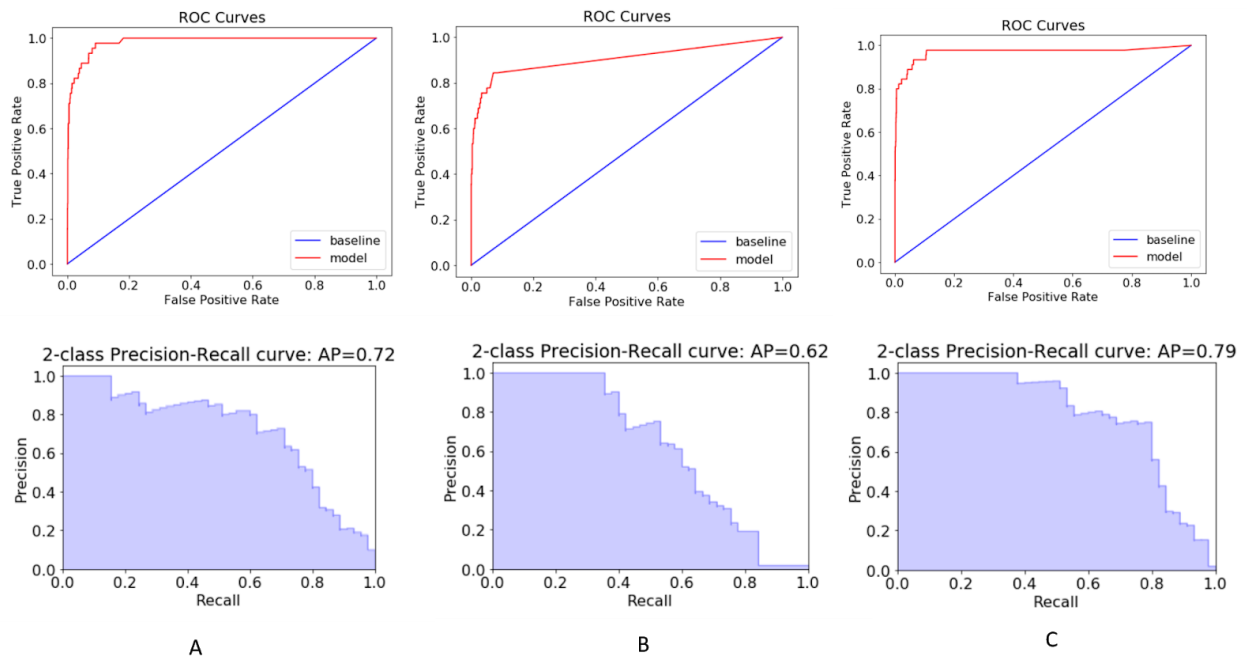


Figure 8: ROC and Precision-Recall plots for model 1, 3, and 4 in the table 8. (A) without using text features (B) using first_word (C) using first 3 words

Currently, at the 50% mark for this Capstone Project, we have created a fully functioning machine learning pipeline that allows us to tune various models and compare their results. Since we have not yet met our target i.e. achieving a recall of higher than 95% and a precision of higher than 80%, we will be continuing the tuning process. This comprises a clearly defined list of next steps which are discussed in the next section.

5 Conclusion and Next Steps

At the 50% mark for this Capstone Project, we have managed to take raw PDFs, convert them to HTML and extract raw features that were fed into various machine learning models. The target was manually labelled by us, separately for the train and test sets. The various models were tuned and cross-validated on the train dataset while the results were evaluated on the test set at the end of the training set. We tried various training strategies to improve our results and are still in the experimentation stage at this point, where we are trying to build different models and tuning them to optimize our target metrics: 95% recall and 80% precision on an independent test set. We expect to achieve our targets at the 70% mark of this Capstone Project, at which

point we will move to Module 2 of this Capstone Project: Comment Summarization. The next steps for module 1 i.e. until the 70% completion mark can be split into 3 sections as follows:

Preprocessing:

1. Try Target based encoding for categorical features to see if scores improve (use library called *dirtycat*⁹)
2. Scale numeric features (non-binary numeric features that can be treated as continuous) and feed them to a Linear Models

Modelling:

1. Tune XGBoost's learning rate and other hyperparameters to optimize performance
2. Experiment with bigger Grids for Random Forest and XGBoost
3. Try feature selection methods that come with *sklearn*: *SelectKBest*, *SelectPercentile*, *SelectFromModel* etc.

Evaluation:

1. Study the distribution of output scores from each model to decide an optimal threshold
2. Need to calculate precision and recall at multiple thresholds and understand where model performance is optimized.
3. Calibrate output scores for all models except Logistic Regression since they're already well calibrated

⁹ <https://dirty-cat.github.io/stable/>