**Document Classification using NLP and Machine Learning**

**Contents**

1. **Introduction**

1.1 Objective

1.2 Requirements

1. **Data Insights**

2.1 Data Understanding

2.2 Data Size

2.3. Exploratory Data Analysis (EDA)

1. **Dataset Creation**

3.1 Extracting text from PDF files

3.2 Dataframe Creation

1. **Model Architecture**
2. **Model Compilation and Training**

5.1 Experimentation Details and Results

1. **Final Results**

6.1 Confusion Matrix

6.2 Accuracy Score

1. Introduction :

**1.1 Objective**

The objective of this assignment is to build an NLP solution for the provided dataset. The dataset consists of scanned documents from an archive.

**1.2 Requirements**

1. The dataset can be downloaded from [here](https://www.sec.gov/Archives/edgar/vprr/index.html). Choose all files in directories starting with 00 or 01. These contain scanned documents - mostly different kinds of regulatory forms and other documents.
2. Create a model for classifying a document into one of multiple form types or if the document isn’t a form, then a category called “Other”.

For example Form D, Form CB, Form X-17 ……. Other

1. Check the accuracy of your model.

2. Data Insights

**2.1 Data Understanding**

* The dataset consists of scanned pdfs from the above mentioned source. These pdfs are regulatory forms and consist of different types.
* The different types of forms are Form D, Form 13F, Form X-17A-5, Form TA-2, Form 6-K, Form 11-K, Form 19B-4 and some documents which do not fall under these categories of forms. Such documents are to be referred to as Other.

**2.2 Data Size**

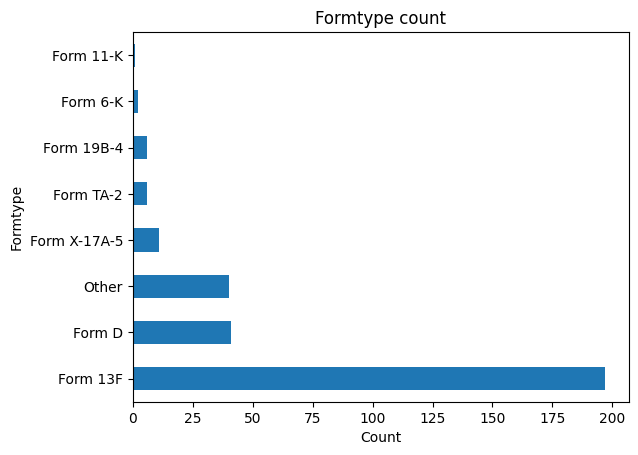
* Only the pdf files from folders starting with 00 and 01 were supposed to be taken into consideration.
* There were 14 such folders and the total numbers of pdf files present in them were 304.

**2.3 Exploratory Data Analysis (EDA)**

Count of different types of pdf files downloaded from the required folders

| Form 13F | 197 |
| --- | --- |
| Form D | 41 |
| Other | 40 |
| Form X-17A-5 | 11 |
| Form TA-2 | 6 |
| Form 19B-4 | 6 |
| Form 6-K | 2 |
| Form 11-K | 1 |

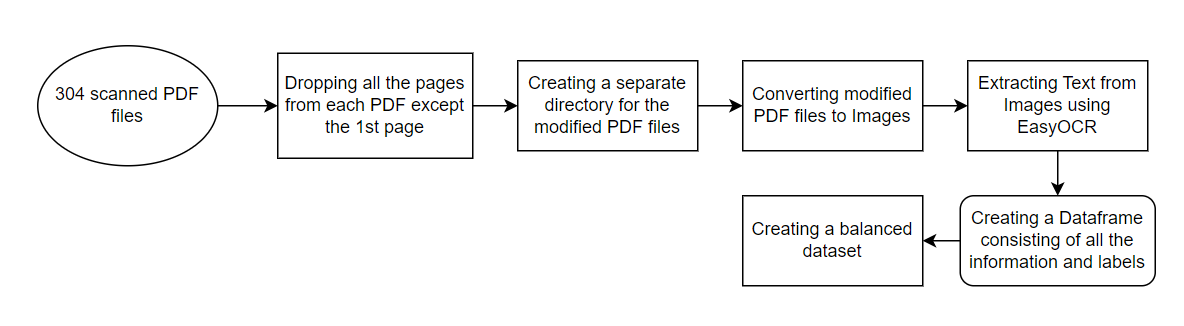
Count bar plot for each type of Form:



3. Dataset Creation

The pipeline for Dataset creation is as follows:

* The pdf files consist of multiple pages ranging from as low as 1 page to as high as more than 1200 pages.
* As the main concept of the pipeline is to extract the text content from all the pdf files, the time taken to do this, for all the 304 pdf files and that too for all the pages in the individual pdf files, was very high.
* Also, considering that the code was developed in a Google Colab notebook, while converting for just 5 pdf files with less than 10 pages in each of them, it was taking more than 1 minute for each file to pass through the functions used.
* Hence, taking these observations into consideration, I decided to drop all the pages from the pdf files except the first page since the first page consisted of the name of the type of form.
* Also it was observed that the first page of different types of forms differed from each other in a significant manner.



Flowchart for Dataset Creation Pipeline

**3.1 Extracting Text from PDF files**

* First a separate directory is created of all the modified PDF files after keeping just the 1st page of all the files.
* These modified files are then passed to a function using pypdfium2 library to convert the PDF files to Images.
* These converted Images are then passed to another function using the EasyOCR library to extract the text from images.

**3.2 Dataframe Creation**

* All the extracted texts from images are then populated inside a pandas dataframe against their respective pdf names.
* The dataframe also consists of the name of the type of form and the numeric labels for each type.
* As we can see from the EDA, that the dataset is highly skewed towards the Form 13F which has 197 samples and Form D being the second highest with 41 samples.
* Hence, it is necessary to create a balance between samples to avoid Overfitting.
* The Form 13F samples were downsampled to the size of Form D and some form samples which were having count less than 10 were dropped.
* Finally, we were left with **41** samples of Form 13F and Form D each, **40** samples of Other category and **11** samples of Form X-17A-5.

4. Model Architecture

In this step, we will define the architecture of our single-input single-output multiclass classification model. In our case, input is the extracted text content of 1 pdf and the output is the type of Form it is extracted from.

* Our model is composed of a swifter package which makes it easy to apply any function to your pandas series or dataframe in the fastestavailable manner.
* First, swifter tries to run your operation in a vectorized fashion. Failing that, it automatically decides whether it is faster to perform parallel processing or use a simple pandas apply.
* There are functions included which cleans our extracted text off unnecessary information. This is done so that only the required information is retained for the model to learn things better.
* The cleaned data is then split into Training and Testing sets in an 80-20 split.
* Our model pipeline consists of the following components:

1. CountVectorizer: A feature extraction step which converts a collection of text documents into a matrix of token counts.
2. TfidfTransformer: This transforms a count matrix to a normalized tf or tf-idf representation.

Tf means term-frequency while tf-idf means term-frequency times inverse document-frequency. This is a common term weighting scheme in information retrieval that has also found good use in document classification.

The goal of using tf-idf instead of the raw frequencies of occurrence of a token in a given document is to scale down the impact of tokens that occur very frequently in a given corpus and that are hence empirically less informative than features that occur in a small fraction of the training corpus.

1. MultinomialNB: The multinomial naïve Bayes is widely used for assigning documents to classes based on the statistical analysis of their contents. It provides an alternative to the "heavy" AI-based semantic analysis and drastically simplifies textual data classification.

The classification aims to assign fragments of text (i.e. documents) to classes by determining the probability that a document belongs to the class of other documents, having the same subject.

The labeling of documents with one of the existing classes is done by performing the statistical analysis, testing the hypothesis that a document’s terms already occurred in other documents from a particular class. This increases the probability that a document is from the same class as the documents, already classified.

5. Model Compilation and Training

Once we have the dataset ready to use and the model architecture defined, we can train the model.

**5.1 Experimentation Details and Results**

TEST 1: Here, we have consumed all the 304 pdf files and passed it through the model without dropping anything.

Result:

* All the predictions were giving the result as **‘Form 13F’** because the original sample was skewed towards Form 13F. This was an effect of the Overfitting problem.

To avoid this Overfitting issue, we downsampled and dropped the pdf files to make an equivalent sample size for all the classes.

TEST 2: Here, we have consumed the downsampled version of the dataframe consisting of 133 samples and passed it through the model.

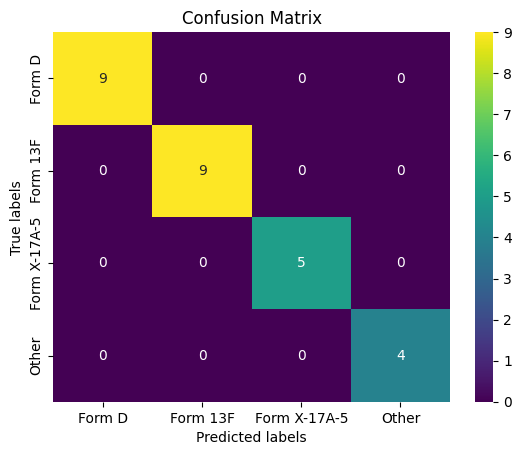
Result:

* The predictions were correct this time and the model was performing well on the trained dataset. The Overfitting issue was also resolved.

6. Final Results

We got our best results with TEST 2, details of which are highlighted in the above section. Confusion Matrix and Accuracy Score are presented in this section below:

Confusion Matrix:



Accuracy Score: The accuracy score of the model is 1.0