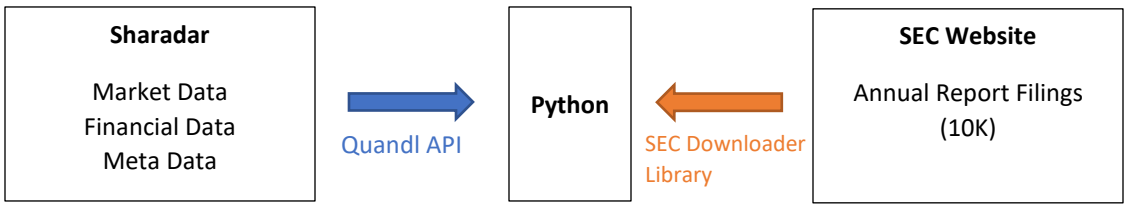
Data Wrangling

The project seeks to investigate if sharp and extreme equity drawdown, as a proxy for bankruptcy / credit risk, can be predicted from the language of a company's annual report. This is set-up as a binary classification problem with the target being the maximum rolling 20-day drawdown of the equity price in the year following the release of the annual report. The target registers a positive event when the drawdown is greater than 80% and is otherwise negative. The feature set is derived from preprocessing the annual reports and representing these in a td-idf matrix. The performance of the model is compared against a similar model using various company financial metrics as features.

Step 1: Pulling Data from Databases

The structured data comes from Sharadar's paid subscription and is pulled via the Quandl API. Sharadar offers data over 20 years of history across 14,000+ US companies. Importantly for our application, these include bankrupt and other delisted companies. Three databases are utilized: (i) share price data, (ii) financial fundamentals and (iii) metadata. The unstructured data consists of company annual reports (10K) filed with the SEC and is automatically downloaded from the SEC website using an existing python library. The unique company identifier for the SEC website is its CIK number and this is linked to the structured data via the metadata which provides the CIK number for each company ticker.



Step 2: Processing Data Separately

The market data and 10Ks are preprocessed before merging while the financial data is processed in the step concurrent with merging.

2.1: Preprocessing Target Data from Market Data

The closing daily share price is used to compute the 20-day rolling drawdown across the entire Sharadar dataset since 1997. This yields a data frame with 16,973 columns representing company stock exchange tickers and 5,649 rows representing consecutive business days from Dec 1997 to June 2020. No corrections are made for missing data at this point aside from dropping rows where all entries are nan.

While it will be unrealistic to download all 10Ks of all companies, the main purpose of starting with this wide approach is to ensure that all positive events are identified and their 10ks are downloaded. This will be crucial for later modelling in this highly imbalanced classification problem.

2.2 Preprocessing Text from 10Ks

While CIK numbers are unique, these may map to multiple tickers if companies have had name changes, acquisitions or other changes on their stock exchange listing. The preprocessing ensures that 10Ks are

mapped to all relevant tickers so that these can be matched with a company's share price across its history. Document metadata such as Filing Date is also pulled directly from the text and stored separately. A custom error log is created storing a reference to the documents that failed to process. In the event, only 13 of 43,536 documents failed.

Given the long period covered in the SEC database, the filed 10ks are inconsistent in format with some in text, some in html and others in XBRL. Resultantly, these documents were processed using Regex to remove special characters instead of an html parser which failed on many occasions. It is worth remembering that one consequence of this is that most of the corpus will be in lower case. Further processing, such as lemmatization and stemming, were decided against on the advice of my mentor.

The end of this process yields a data frame with 44,958 rows and 15 columns. The rows represent unique 10K documents while the columns contain the ticker, filing date and processed text among other document metadata.

Step 3: Merging Data

3.1 Merging Target and Feature Data

Data is lost across the merge in two ways. Firstly, there are companies with market data but no 10K filings at all and those with incomplete filing databases. For example, companies with some 10Ks but not those that correspond to the year of positive events. From an initial list of 16,973 company tickers, the downloaded 10K data base has 4,456 of these companies. This reduces to 4,365 after the final merge providing a final accessible database of 38,807 documents. These documents are stored in a column of the final database as a string. Other columns include tickers, Filing Date and other metadata such as current company sector. This step includes a function that takes the maximum of the drawdown in the year following the annual report filing date and stores this target variable as a column. Missing values are removed via an inner join merge between the drawdown and 10k data frames.

The output of this step is a final data frame with 38,807 rows being the samples and x columns representing the preliminary target variable, f

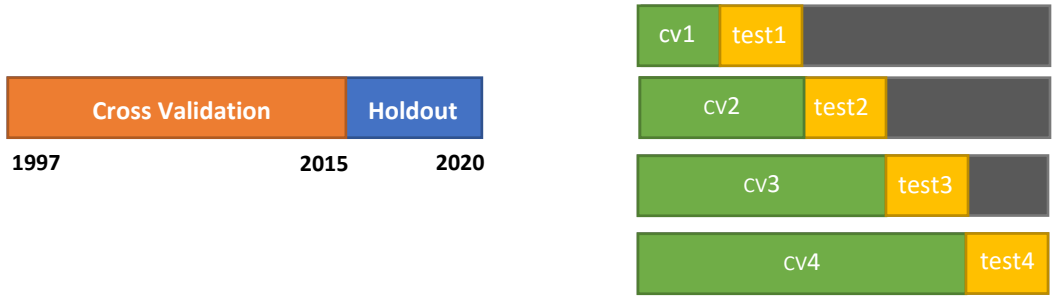
3.2 Converting 10K text to TD-IDF Vector

The next step is to convert the columns containing the 10K strings into a TD-IDF matrix. This is done using SK-Learn's inbuilt vectorizer function. Again, we opt for minimal processing and do not remove stop words. We add functionality that deletes vector columns containing nan. We convert the resultant sparse scipy matrix into a spares data frame and explicitly label the columns with the vocabulary words.

3.3 Processing Cross Validation and Holdout sets

In order to expedite model validation and testing, we build various merged datasets for the cross validation and holdout sets. The holdout set consists of all data from 2015 filing dates onward and the balance is used for cross validation.

The time-series cross-validation set is separated into 5 equal parts and training and testing sets are defined as below



The td-idf vectorizer is formed on the cv training data and *transform* is then used to convert the testing documents into the model vector matrix.

3.4 Merging Target Data and Financial Data

The purpose of the financial ratio analysis is to create a more traditional baseline against which to compare the NLP model. In this case, we use the financial data reported in the annual report and match these, via an inner join on ticker and filing date columns, with the samples obtained in 3.1. Any rows with a nan or infinity are dropped from the resultant data frame.

Appendix: Data Schematic with Codebase Dependencies