Problem Statement.

Develope a model to predict the impact of a book, a composite score achieved post-publication.

In the interest of time and scope ofthe interview processm, I am focusing on the core pipeline. However, I will be adding quick findings, potential feature engineering and model building ideas.

The behaviour, to code+pipeline is closely governed by the *config.toml* file

Dataset:

```
Index(['Unnamed: 0', 'Title', 'description', 'authors', 'publisher',
       'publishedDate', 'categories', 'Impact'],
```

Here “Unname: 0” can be dropped since this is not useful for us.

Description:

```
data["description"], print(f"missing descriptiton {len(data['description'][data['description'].isnull()])/len(data)}")
```

```
missing descriptiton 0.09190190594273522

(0                                                       NaN
 1         Philip Nel takes a fascinating look into the k...
 2         This resource includes twelve principles in un...
 3         Julia Thomas finds her life spinning out of co...
 4         In The Church of Christ: A Biblical Ecclesiolo...
                                ...
 138719    "The Magic of the Soul, Applying Spiritual Pow...
 138720    Autodesk Inventor 2017 Essentials Plus provide...
 138721    During a school trip to Ellis Island, Dominick...
 138722    Everyone in the village of Friedensdorf is hap...
 138723    Alex-Li Tandem sells autographs. His business ...
 Name: description, Length: 138724, dtype: object,
 None)
```

Here approximately 9% of values are missing, there are several ways to go about it.

- **Missing Values**
    - Fetch the description from the public websites like “GoodReads”, “Wikipedia”.
    - Impute themissing description is a text label like “missing description”, “information not found”. This missing information can *potentially* be caught by contextual embedders or language models. **USING THIS.**
    - If the data is abundant, we can drop these rows.

- **Potential Investigation:**
    - These missing values/descriptions could have a common pattern between themselves:
        - Common publisher/pulishedDate etc.

Authors:

```
data["authors"], len(data["authors"][data["authors"].isnull()])/len(data)
```

```
(0                          ['Julie Strain']
 1                            ['Philip Nel']
 2                          ['David R. Ray']
 3                       ['Veronica Haddon']
 4                      ['Everett Ferguson']
                            ...
 138719               ['Patrick J. Harbula']
 138720     ['Daniel Banach', 'Travis Jones']
 138721                  ['Elvira Woodruff']
 138722                                  NaN
 138723                      ['Zadie Smith']
 Name: authors, Length: 138724, dtype: object,
 0.019628903434157033)
```

- **Missing values**
  - Same as Description
- **Preprocessing**:
  - Quotes and brackets can be removed. USING THIS
  - Commas can be retained since language models or contextual embedders can capture that.
- **Potential Investigations:**
  - What is the effect of multiple authors on the Impact?
  - Are there authors in milti-author books that do not exist in isolation?
- **Potential FeatureEngineering:**
  - Is_multi_author : True
    - SUM(avg_individual_author_impact)
    - Sum(author_impact_marginal_contribution)
  - Is_multi_author: False
    - Author Avg. historial impact
    - Author permuted marginal contribution

Publisher:

```
data["publisher"], print(f'missing {(data["publisher"].isna().sum() / len(data) * 100)}%')
```

```
missing 0.0%

(0              Smithsonian Institution
 1                            A&C Black
 2                              OUP USA
 3                            iUniverse
 4          Wm. B. Eerdmans Publishing
                           ...
 138719                Love & Logic Press
 138720                  SDC Publications
 138721             Scholastic Paperbacks
 138722        Wm. B. Eerdmans Publishing
 138723                           Vintage
 Name: publisher, Length: 138724, dtype: object,
 None)
```

Nothing missing, no particular preprocessing needed.

```
data["publisher"].value_counts()
✓ 0.0s

publisher
Tan Books & Pub               3635
Simon and Schuster            3600
Smithsonian Institution       3216
Penguin                       2788
Wm. B. Eerdmans Publishing    2563
                              ...
Rider                            1
Robert Davies Pub                1
Astrology Sight                  1
McQueen Enterprises              1
Torah Aura Prod                  1
Name: count, Length: 12855, dtype: int64
```

We can potentially bucket the very rare publishers.

Published Date

```
data["publishedDate"].value_counts()
✓ 0.0s

publishedDate
2000          3362
2004          3218
1999          3159
2002          3110
2003          3070
              ...
1981-12-18       1
1969-11-15       1
2000-03-05       1
2004-07-24       1
2007-08-26       1
Name: count, Length: 10819, dtype: int64
```

```
data["publishedDate"] = data["publishedDate"].fillna("10")

def extract_year(date_str: str) -> str:
    return date_str.split("-")[0]

data["publishedDate"] = data["publishedDate"].map(extract_year)
data["publishedDate"].value_counts()
✓ 0.0s

publishedDate
2004     7196
2005     6976
2003     6794
2002     6400
2000     6083
         ...
1797        1
1712        1
1947*       1
1814        1
2014*       1
Name: count, Length: 257, dtype: int64
```

- Missing dd-mm can potentially mean the unknown month/date of publication date of publication
    - Given everything else is same, do missing dd-mm exhibit a abnormal behaviour?
- Some years are appended by `*`. Which can potentially mean disputed years or some similar behaviour as missing mm-dd
- Some years are only 3 digits with `?` at the end. This potentially a book with with disputed years, where year of publishing could only traced back to a decade.
- Potential investigation:
    - All the 3 cases above can have a potential effect on the “impact” of the book.

- Disputed/missing dates can also be related to missing descriptions.

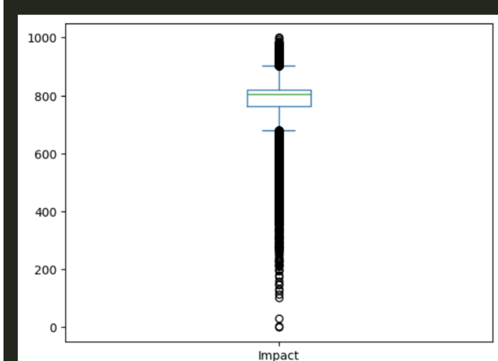
Category:

```
1 data["categories"].value_counts(), len(data["categories"][data["categories"].isnull()])
✓ 0.0s

(categories
 ['Fiction']                        23419
 ['Religion']                        9459
 ['History']                         9330
 ['Juvenile Fiction']                6643
 ['Biography & Autobiography']       6324
                                    ...
 ['Christianity']                      79
 ['Young Adult Nonfiction']            79
 ['Railroads']                         78
 ['Brothers and sisters']              76
 ['Automobiles']                       74
 Name: count, Length: 100, dtype: int64,
 0)
```

- No Missing values
- Preprocessing, brackets and quotes can be removed.
- There is a potential for bucketing rarer categories, EG: “Christianity” and “Judaism” can be moved under “Relegion”.

Impact:

```
1 data["Impact"].plot.box()
✓ 0.3s

<Axes: >
```



- Left Skewed! And root/log transform is not helping this one.
- The tail must be explained via feature engineering and finding explanations for the missing values and the outliers.

Unexplored Methods:

- **Word Analysys:**
  - What at the words or group of words associated with high-Impact/low-impact.
  - What happens when when the contextual meaning of the Title and Description contrast with each other or comply with each other. What would be the effect on Impact? (Reverse psychology found in Lireature).
- The word analysis can potentially turn into features
  - Is_contrasting_description?
  - Is_aligning_description?
  - Many more that have escaped my head..

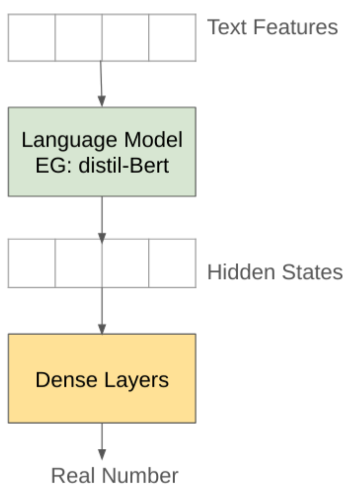
Model:

This is a language based regression problem, Never seen that before!

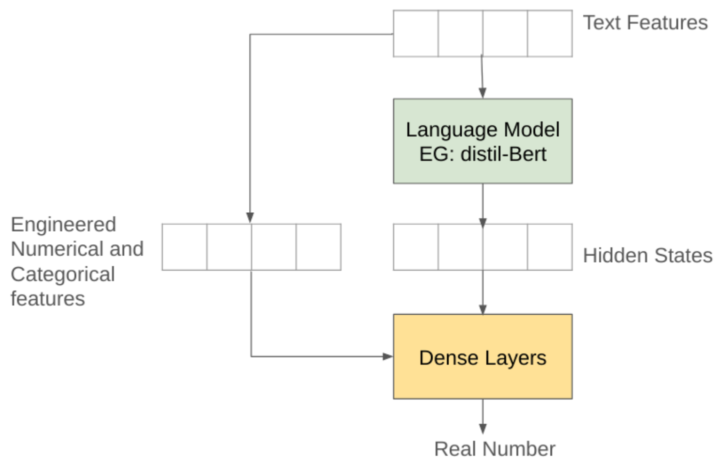
There are many different ways to approach this, I will explain the path of least resistance I took while I will also look into other promising possibilities.

In the dataset we basically have “almost” all “text” variables.

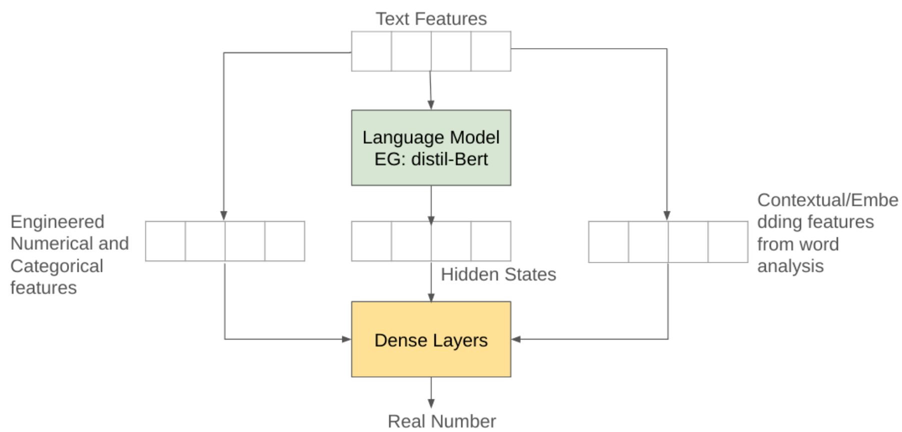
1. Exclusive Language Model, The path of least resistance I followed: Make everything a single string profile.



2. Language model with Feature Engineering added. Feature Engineering ideas captured earlier can be integrated during the dense layer stage.



3. Language models + Engineered features + Contextual Embedding features



Obviously there are going to methods that I completely overlooked.

Loss Function:
Since it was stated that we will be looking at the MAPE error in this problem, I tried directly adding it in as a loss function itself. Not sure if it was the best plan, only experimentation would reveal the best was possible.

Experiment Tracking:
Since this was a fairly simple Pipeline, I went ahead with a simple JSON experiment tracker. In more complicated projects we can go for a `MLFlow` or “ClearML” solutions.

Workers/Clusters:
Since the data was only 100MB, Single threaded performance was so fast that multi-processing was not exactly visible in the preprocessing phase. I could have potentially used Dask.LocalCluster()

But in tokenization phase, having multiple workers do add up but not in a linear way. So it is likely to have the diminishing returns effect.