Midterm Report

1. Description of Dataset

The dataset contains records of 200+ financial indicators (obtained from SEC 10-K filings) of listed US companies from 2014 to 2018. These financial indicators span the range of revenue, operating costs and inventory growth, thus offering a holistic picture of each company's financial performance over time.

In the original dataset, there are 22,077 observations that are linked to the financial performance of 4,980 companies over 5 years. The number of observations per year varies is slightly uneven, ranging from 3,808 observations in 2014 to 4,960 observations in 2017.

The dataset has 223 features excluding two possible labels. Of these features, 2 are non-financial descriptors (Company Ticker and Sector), while the remaining 221 are financial indicators.

The last two columns: Class and PRICE VAR [%] of the dataset can be the dependent variable in our dataset. PRICE VAR [%] is the perecentage increase or decrease in price of the stock in a year. Class is determined by PRICE VAR [%]. Class takes the value 1 for postive values of PRICE VAR [%] and 0 otherwise. Class value 1 indiactes the stock should be bought at the beginning of the year and sell by the end of the year for profit and Class value 0 indicates that the stock should not be bought as it's price will decrease by the end of the year.

Our data was preprocessed by concatenating the five years of data (originally split into 5 datasets), before cleaning the dataset to address missing data and outliers.

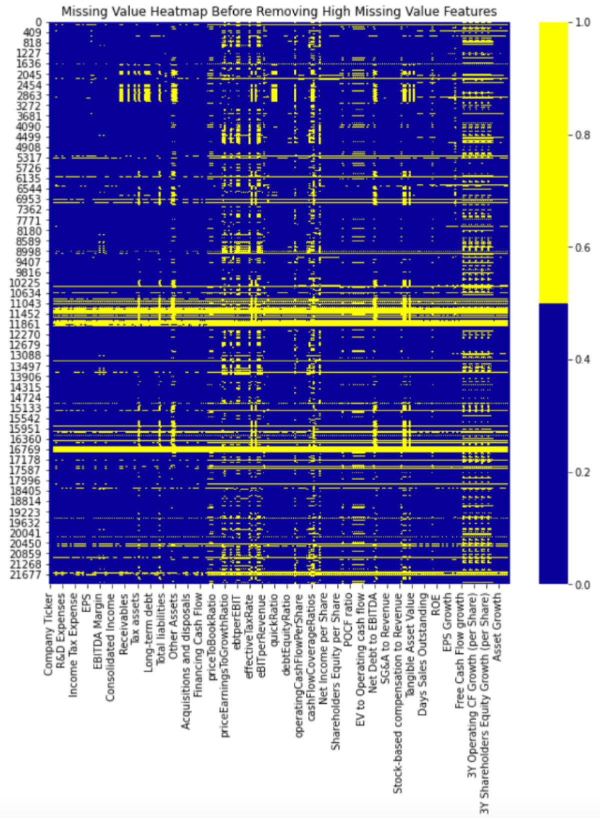
2. Problems in Dataset

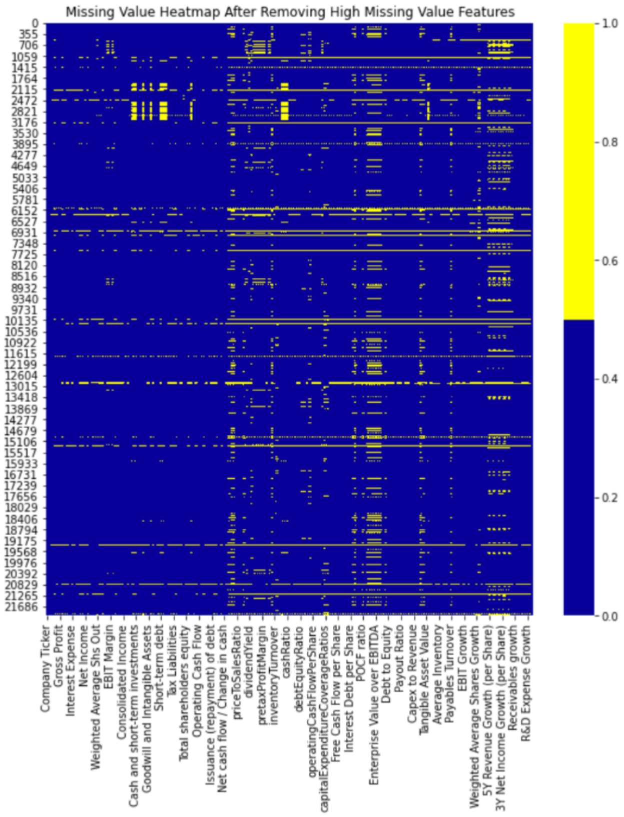
The two major problems in the original dataset are missing data and outliers.

2.1 Missing Data

Since we only have five years of financial data to work with, we decided to limit our analysis to companies with at least some data for each of the five years. Of the 4,980 companies, 3,726 were retained because they have data for all 5 years, while 1,254 companies were filtered out. This reduced the number of observations to 18,630.

Subsequently, we proceeded to examine the proportion of missing values within feature columns. 2 features (operatingCycle and cashConversionCycle) have >99% of their values missing, while another 29 features have more than 20% of their values missing—these are too high to be fixed by data imputation. Hence, we drop 31 features with >20% of values missing, leaving us with 193 features. The missing value heatmaps below show how the amount of missing values (denoted in





In the remaining feature columns, there were still some missing values. We proceeded to impute each missing value using the median value of the feature in the given sector (e.g. "Financial Services"). The reasoning for our sector-based approach is that financial indicators tend to be very sector-dependent. For instance, the median firm in "Healthcare" has a much higher R&D Expenses value than the median firm in "Basic Materials".

2.2 Outliers

We realized that there were large differences between the 75th percentile and maximum values for some features (e.g. 0.172 at 75th percentile and 42138.664 at maximum for Revenue Growth), which suggested possible mistakes in data entry.

Hence, lower and upper outliers were adjusted to take the values of the 5th or 95th percentile of values (respectively) for companies in the same sector.

After data cleaning was done, the dataset was left with 18,630 observations and 195 columns (including two output columns Class and PRICE VAR [%]). The distribution of Class in the cleaned dataset is slightly unbalanced, with 10,317 observations of 1 (increase in price) and 8,313 observations of 0 (decrease in price).

3. Feature selection

3.1 categorical data

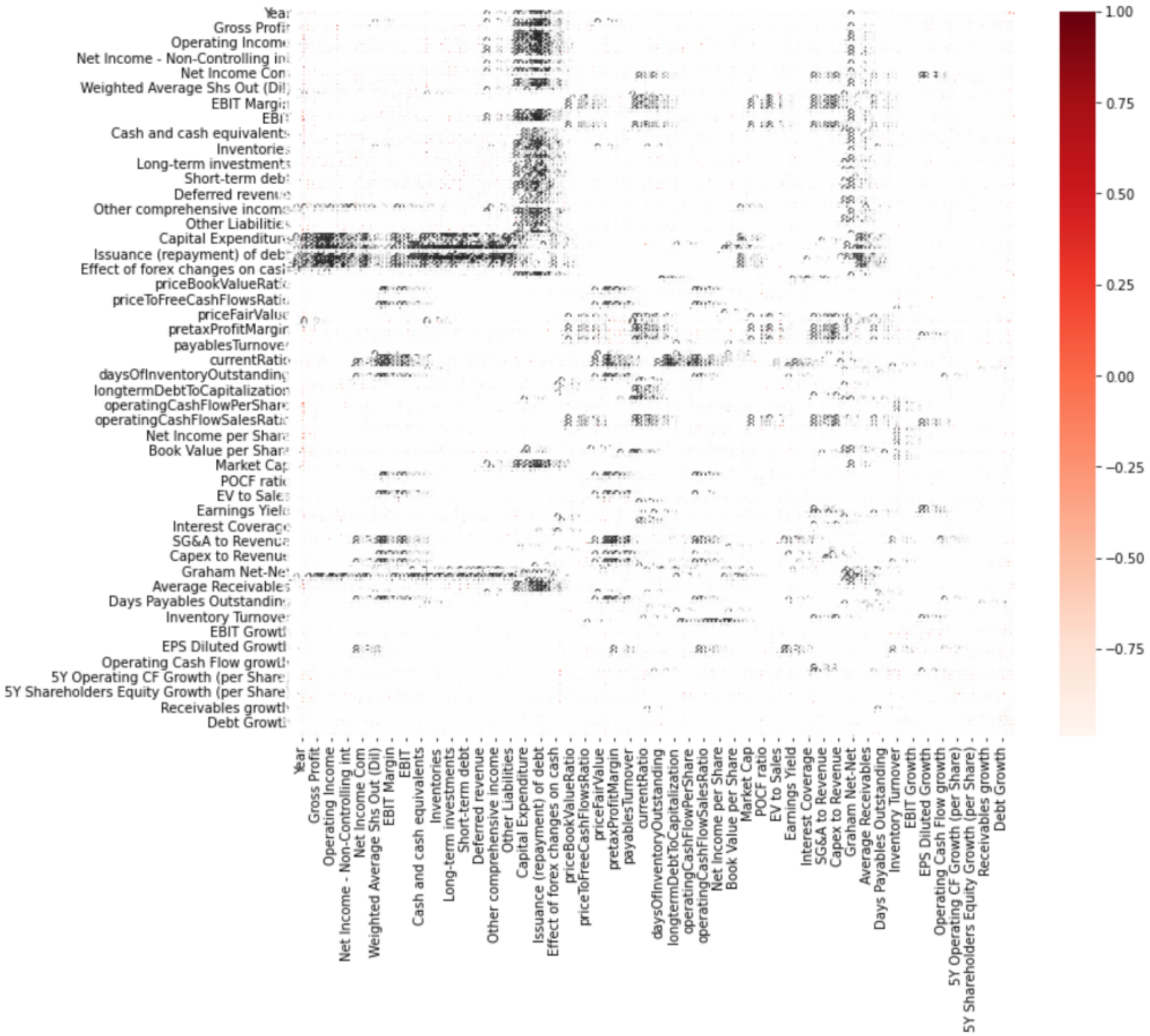
We have only two columns that contain categorical data: "Company Ticker" and "Sector".

We found out that if we create one-hot encoding of "Company Ticker", we would have each entry a size 3726 vector. It would be too sparse and the company ticker doesn't really give out information about the company. So we decide to drop "Company Ticker" and only use "Sector" by converting it to one-hot vectors

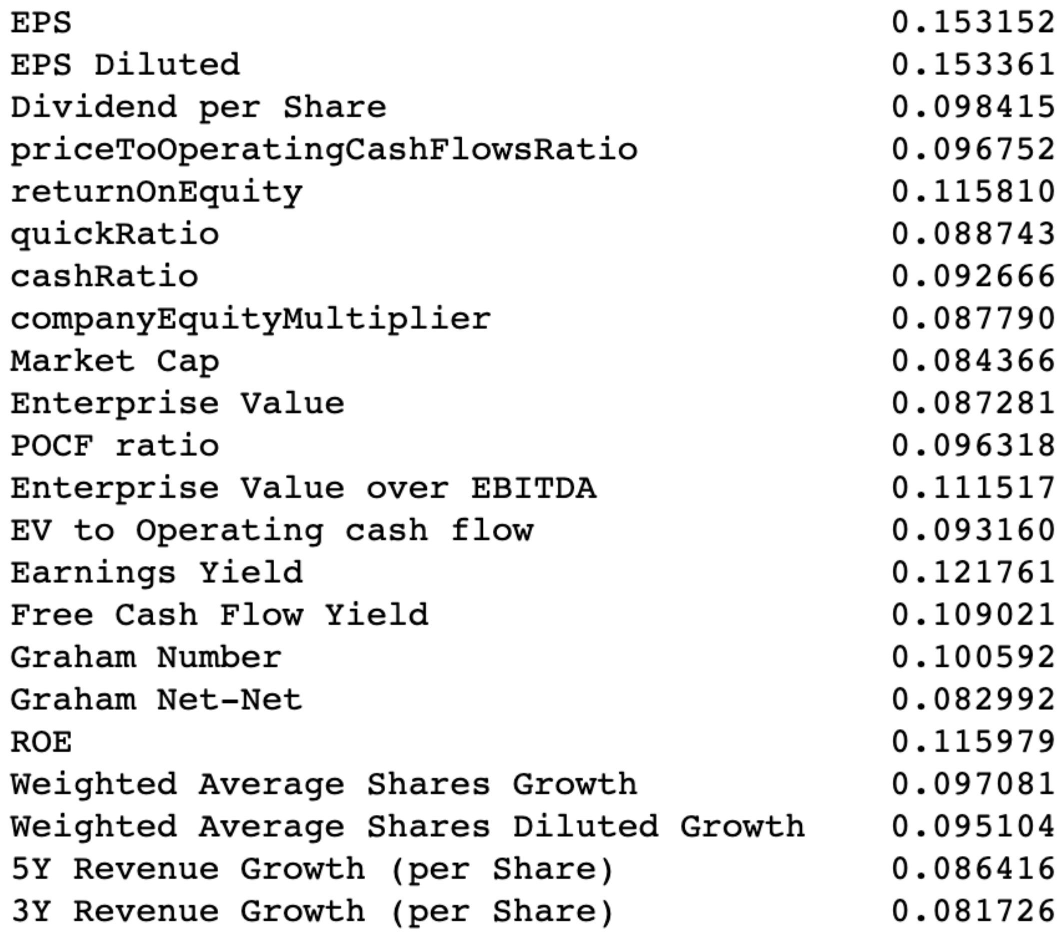
3.2 numerical data

We have almost 200 columns of numerical features. We have to select a few of them to prevent overfitting and unwanted noise.

We calcuted the pearson correlation of each feature along with the class label "Class".

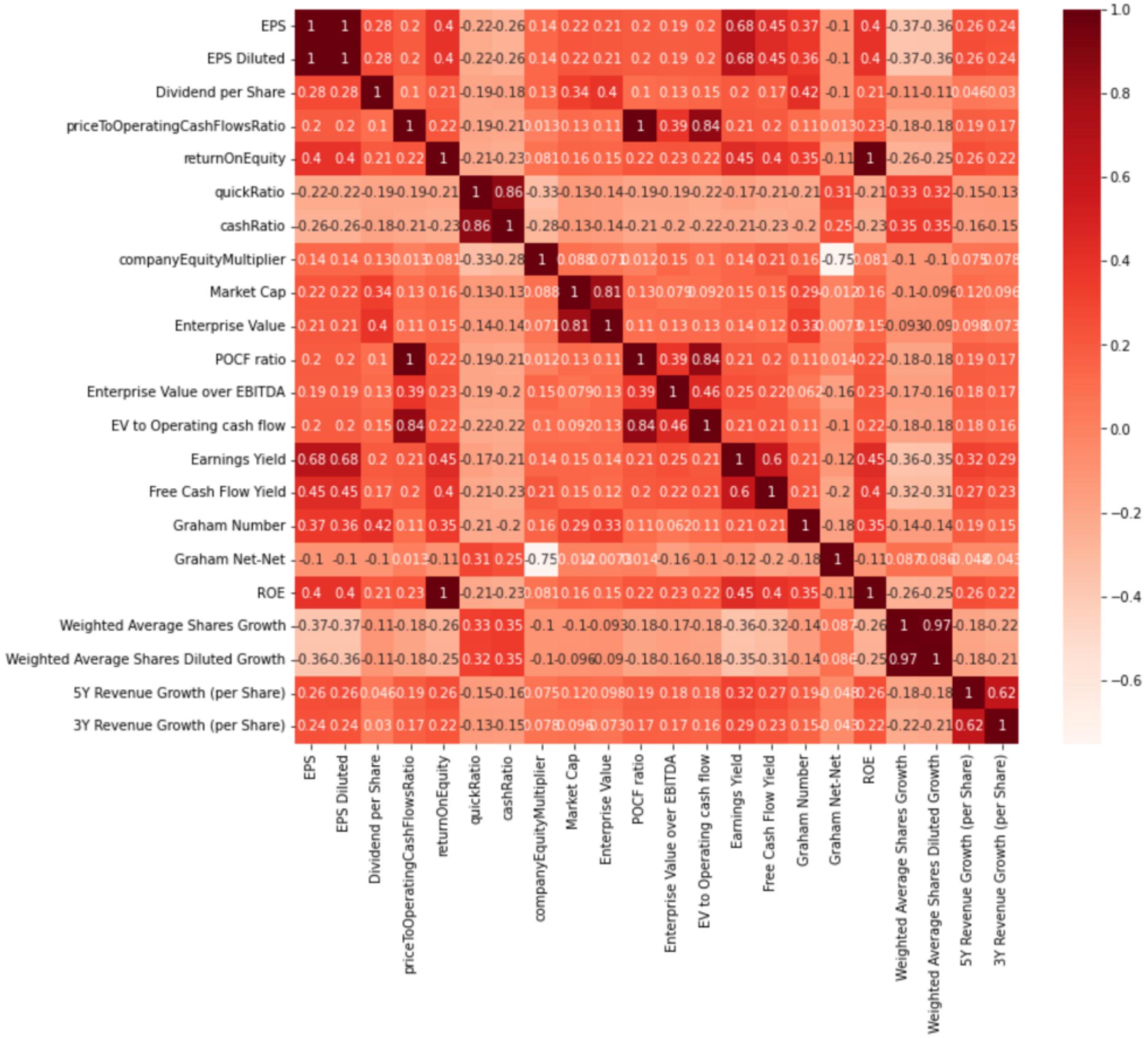


We then calcuted the correlation with respect to the label "Class" and filtered out features that are poorly correlated to the label by setting the threshold with value 0.08 of absolute value of correlation with respect to the label. And only 23 features are left



```
EPS                                        0.153152
EPS Diluted                                0.153361
Dividend per Share                         0.098415
priceToOperatingCashFlowsRatio             0.096752
returnOnEquity                             0.115810
quickRatio                                 0.088743
cashRatio                                  0.092666
companyEquityMultiplier                    0.087790
Market Cap                                 0.084366
Enterprise Value                           0.087281
POCF ratio                                 0.096318
Enterprise Value over EBITDA               0.111517
EV to Operating cash flow                  0.093160
Earnings Yield                             0.121761
Free Cash Flow Yield                       0.109021
Graham Number                              0.100592
Graham Net-Net                             0.082992
ROE                                        0.115979
Weighted Average Shares Growth             0.097081
Weighted Average Shares Diluted Growth     0.095104
5Y Revenue Growth (per Share)              0.086416
3Y Revenue Growth (per Share)              0.081726
```

- We then calcuted the correlation within each features also using pearson correlation.



We found out that there are a couple of features are highly correlated(either positively or negatively) with each other. Those are: EPS and EPS Diluted; cashRatio and quickRatio; POCF ratio and priceToOperatingCashFlowsRatio; EV to Operating cash flow and priceToOperatingCashFlowsRatio; ROE and returnOnEquity; Graham Net-Net and companyEquityMultiplier; Enterprise Value and Market Cap; EV to Operating cash flow and POCF ratio; Weighted Average Shares Diluted Growth and Weighted Average Shares Growth.

Thus, we drop one of each pair to avoid higher computational cost: EPS Diluted, quickRatio, POCF ratio, priceToOperatingCashFlowsRatio,returnOnEquity,Enterprise Value, companyEquityMultiplier, Weighted Average Shares Diluted Growth.

And now we only have 14 features left: 'EPS', 'Dividend per Share', 'cashRatio', 'Market Cap', 'Enterprise Value over EBITDA', 'EV to Operating cash flow', 'Earnings Yield', 'Free Cash Flow Yield', 'Graham Number', 'Graham Net-Net', 'ROE', 'Weighted Average Shares Growth', '5Y Revenue Growth (per Share)', '3Y Revenue Growth (per Share)'

3.3 Normalization

We then normalize the numerical data by subtracting the mean of each feature and divided by the standard deviation of each feature.

4. Classification

We employ three methods of classification.

1. Logistic Regression - Since our classification problem is a binary classification problem of whether to buy a stock or not, we employ Logistic Regression to classify as buyable or not.
2. Decision Trees - Decision Trees are another excellent machine learning model for classification problems. This can also be used if we decide to extend our problem statement to multinomial classification like low, medium, high risk stocks.
3. Random Forest - Since our dataset is large with large number of features, we also employ Randome Forest to reduce chances of overfitting.

5. Validation

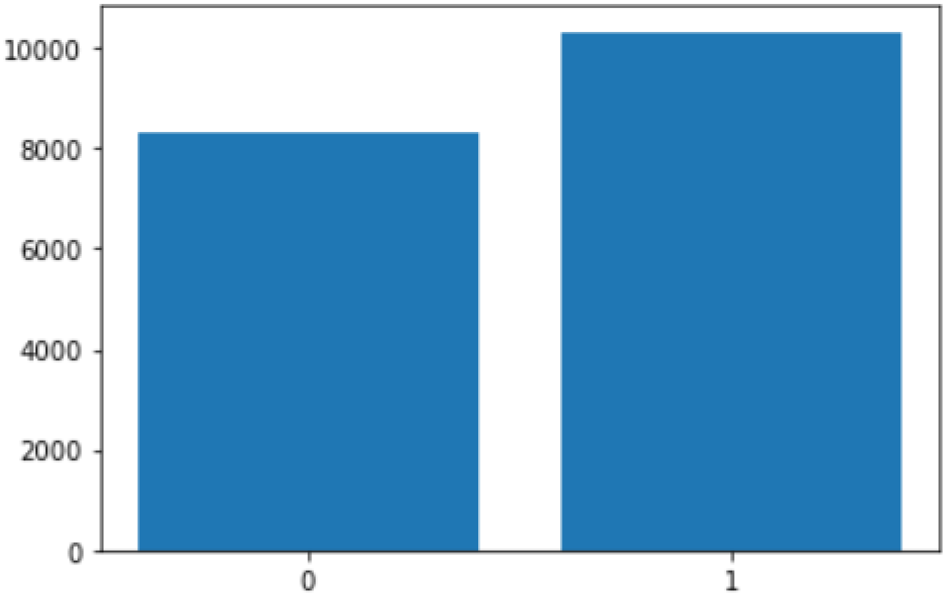
We use cross validation to assess the performance of each of these classification methods. We split the train and test data using three methods:

5.1 Random Shuffling

As mentioned in 1. of the report, we combine all the datasets of 5 years into one dataframe. After data cleaning, preprocessing, and feature selection, we do a random shuffle on the data and split the data into train and test with the ratio 4:1, i.e. 20% of the data is held for testing.

5.2 Stratified Sampling

The class distribution in the dataset is not evenly split. Following figure shows the class distribution for the entire dataset.



This uneven distribution can sometimes create a test set with uneven class distribution. Stratified sampling is used to ensure that the class distribution in the train and test set remain apprximately equal. To implement this, we compute the ratio of classes in the dataset and do random sampling while holding the same ratio on the Class column.

5.3 Temporal Sampling

Stock Performance Prediction has a temporal component associated with it and this can be explored by training the models on previous years and holding 2018 dataset as the test set. This method will approximately have a 80% and 20% distribution for train and test respectively as we are considering data from 2014-2018.

6. Result Analysis

6.1 Implementation

We use sklearn library to fit and compute classificatiom models - Logistic Regression, Decision Trees and Random Forest. Since ramdom sampling can create a bias in scores if only one model's score is computed. Hence to follow the law of large numbers, we compute the methods Ramdom Sampling (5.1) and Stratified Sampling (5.2) and fit all the models and average over 1000 models to get the best possible representation of classfication score for a particular sampling. Since temporal sampling does not have randomness associated, we only fit the models once for this.

6.2 Scores and Analysis

Classification scores is the mean accuracy on the given test data and labels. The scores for Ramdom Sampling and Stratified Sampling are averaged over 1000 variations of samples.

	Logistics Regression	**Decision Trees**	**Random Forest**
Random Sampling	0.606910	0.549506	0.600888
Stratified Sampling	0.606694	0.549630	0.600516
Temporal Sampling	0.674718	0.546967	0.605475

- The Random Sampling and Stratified sampling have somewhat similar scores with a variation of the degree 10^(-3). This is because the class distribution is 40% 60% and hence random sampling would approximately give similar distribution in train and test sets as the original data.
- Decision Tree gives lowest scores for all the cases of sampling. This is possibly because decision trees tend to overfit the data and we do have a large dataset, hence it is more prone to overfitting.
- Logistic Regression and Random Forest give similar results for Random and Stratified Sampling. Although, Logistic Regression outperforms Random Forest in Temporal Sampling.

7. Error Analysis

We use F1 scores for error analysis for comparison between classificatiom models.

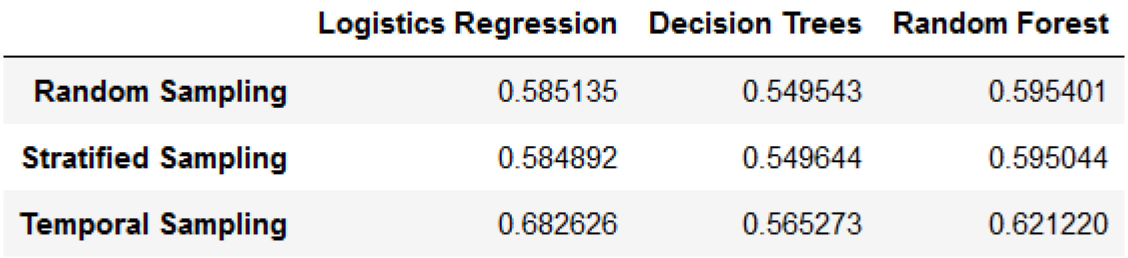
F1 Score

Precision = $\frac{|\mathrm{P} \cap \mathrm{C}|}{|\mathrm{C}|}$ and Recall = $\frac{|\mathrm{P} \cap \mathrm{C}|}{|\mathrm{P}|}$.

F1 = $\frac{2*\mathrm{Precision}*\mathrm{Recall}}{\mathrm{Precision}+\mathrm{Recall}}$.

$$\text{Mean F1} = \frac{\sum_{i=1}^{N} \mathrm{F1}_{label_i}}{N}.$$

We get following F1 scores for all the above sampling and models

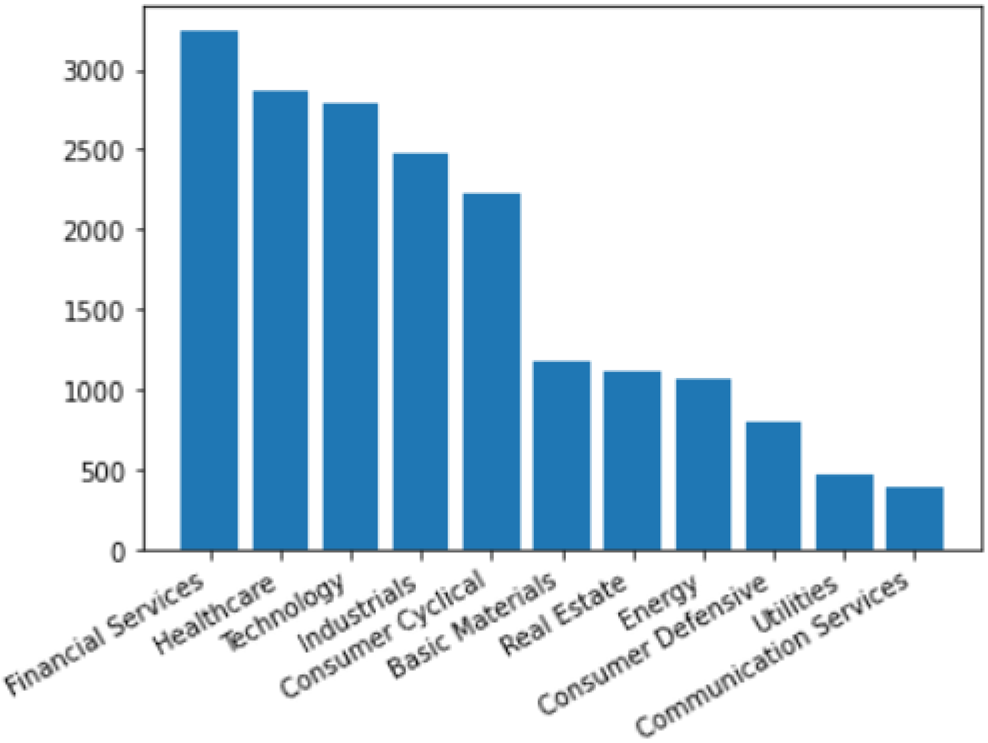


	Logistics Regression	Decision Trees	Random Forest
Random Sampling	0.585135	0.549543	0.595401
Stratified Sampling	0.584892	0.549644	0.595044
Temporal Sampling	0.682626	0.565273	0.621220

- The F1 score follows somewhat similar trend as classification scores. Decision Tree gives lowest scores for all the cases of sampling. Logistic Regression and Random Forest give similar results for Random and Stratified Sampling with Logistic Regression outperforming Random Forest in Temporal Sampling.
- Random Forest performs better than Logistic Regression with Random and Stratified Sampling according to F1 scores which is not the case with Classification scores, implying that it actually performs a bit better than Logistoc Regression as F1 is a better metric for accuracy.

8. Next Steps

The sector distribution for the dataset looks like this:



Stocks can be highly sector dependent as the prices and returns can vary a lot based on sector. For example, stocks in technology sector in recent years have grown by several 100s or 1000s percentage organically and this percetage increase might be organic for these but could be an outlier for other sectors.

We aim to do sector analysis and create a regression problem based on that. We also aim to explore ways to improve classification scores through sector based analysis.

How do we avoid overfitting?

We could use PCA to help reduce the dimension of features for linear regression models. However, we decided not to use PCA for our tree models because PCA produces linear combinations of features and the resulting features are not interpretable for tree models. We also could use different regulariziers(for example l1,l2,etc) to help mitigate the problem of overfitting.

Model Selection

We plan to use neural networks. And we plan to use Bayesian Optimization for hyperparameter tuning. We also plan to run a grid search for hyperparameter tuning and model selection.

● ✕