JSC "Kazakh-British Technical University"
Business School

Endterm Project

Custom credit scoring indicator for publicly traded companies in recreational products industry

Prepared by:
Malik Serikbolsyn - 20B060551

Almaty, 2023

KBTU Business School
Financial Statement Analysis
Spring 2022-2023

Table of contents

Introduction

In this work I have attempted to extract synthetic credit scoring by applying various machine learning and statistical frameworks on financial datasets consisting of financial ratios and metrics fetched from companies' statements. The idea of creating new variable is straightforward at first glance, yet I faced many challenges during the process of execution. General outline of idea looks as following:

- Train a machine learning model for a regression task. Use Altman-Z score as an output variable
- Extract weights for the synthetic credit scoring equation by applying permutation feature importance
- Apply the equation on selected dataframe columns to create a new vector named "Synthetic_credit_score"

Altman-Z score was chosen as proxy indicator, because it's one of most common indicators used by investors and analysts to assess the credit quality of public companies. It's quite flexible and provides insights into both long-term solvency and short-term liquidity of a company. Multiple variations of Altman-Z score were invented since nyu stern professor Edward Altman published "Financial ratios, discriminant analysis and the prediction of corporate bankruptcy" in 1968. I have employed the following variation in my model:

$$\text{Altman Z-Score} = 1.2A + 1.4B + 3.3C + 0.6D + 1.0E$$

Where:

- A = working capital / total assets
- B = retained earnings / total assets
- C = earnings before interest and tax / total assets
- D = market value of equity / total liabilities
- E = sales / total assets

Fetching relevant datasets from Refinitiv Eikon

In order to define and compile the list of samples I used purely a risk basis, so that companies in the list share identical operational, financial and business risks. For this purpose I have used refinitiv business classifications service. Moving further, recreational products industry contains 169 constituent companies globally and 44 in US. The model I used considered global coverage.

	Identifier (RIC)	Company Name	Market Cap\n(USD)\n(Σ=Avg)	1-day Price PCT Change\n(Σ=Avg)	P/E(Time Series Ratio)\n(Σ=Avg)	Return on Average Common Equity - %, TTM\n(FI0)\n(Σ=Avg)	Curr Div Yield Comm Stk Primary, LFI, Annualized\n(Σ=Avg)	Total Debt Percentage of Total Equity\n(FY0)\n(Σ=Avg)	Price V Sh Ratio)\u
0	ESCA.OQ	Escalade Inc	2.101670e+08	-0.008349	11.749308	0.117926	NaN	0.598713	
1	CHBE.PA	Beneteau SA	1.309270e+09	0.002805	11.417804	0.154390	NaN	0.583484	
2	TND.L	Tandem Group PLC	1.650003e+07	0.000000	20.263461	0.027217	NaN	0.180641	
3	1517.TW	Lee Chi Enterprises Co Ltd	1.417324e+08	0.002597	7.626981	0.147468	NaN	0.002561	
4	JOUT.OQ	Johnson Outdoors Inc	5.789371e+08	-0.009982	14.531050	0.082249	NaN	0.000000	
...	...	...	...	...	...	...	...	...	
164	SNOW.KL	Snowfit Group Berhad	9.462125e+06	0.000000	188.679245	NaN	NaN	1.368401	
165	BELLN.MI	Bellini Nautica SpA	1.650892e+07	0.017391	11.597363	NaN	NaN	0.377916	
166	RSEAS.PK	RSE Archive LLC	NaN	NaN	NaN	NaN	NaN	NaN	
167	WARRIXm.BK	Warrix Sport PCL	1.716549e+08	0.020942	27.904980	NaN	NaN	0.052736	
168	OLY1.WA	Olymp SA	1.493958e+06	0.047297	NaN	NaN	NaN	NaN	

169 rows × 12 columns

Figure 1. Companies from recreational products industry

Column named Identifier (RIC) is the most important column for the further parts and stands for Refinitiv Identification Code. These identification codes allow us to address prompts related to exact company we want. Then after obtaining the correct RICs list, I have started retrieving columns from the refinitiv server.

```
import eikon as ek
```

```
ek.set_app_key('0c7933c3333e4b15a8ce34d62c118df4377f4ace')
```

```
df1, err = ek.get_data(
    instruments = ric_col,
    fields = [
        'TR.F.WkgCap(AlignType=PeriodEndDate,Curn=USD,Scale=6,Frq=FQ,Period=FQ0,ReportingState=Rsdt,SDate=0,EDate=-27)',
        'TR.F.RetainedEarnTot(AlignType=PeriodEndDate,Curn=USD,Scale=6,Frq=FQ,Period=FQ0,SDate=0,EDate=-27)',
        'TR.F.TotAssets(AlignType=PeriodEndDate,Curn=USD,Scale=6,Frq=FQ,Period=FQ0,ReportingState=Rsdt,SDate=0,EDate=-27)',
        'TR.F.EBIT(AlignType=PeriodEndDate,Curn=USD,Scale=6,Frq=FQ,Period=FQ0,SDate=0,EDate=-27)',
        'TR.F.TotLiab(Curn=USD,Frq=FQ,Period=FQ0,Scale=6,SDate=0,EDate=-27,AlignType=PeriodEndDate)',
        'TR.F.TotRevenue(AlignType=PeriodEndDate,Curn=USD,Frq=FQ,Period=FQ0,SDate=0,EDate=-27,Scale=6)',
        'TR.CompanyMarketCap(Frq=FQ,Scale=6,Curn=USD,SDate=0,EDate=-27)',
        'TR.F.TotCurrAssets(Period=FQ0,Frq=FQ,AlignType=PeriodEndDate,Curn=USD,Scale=6,SDate=0,EDate=-27)',
        'TR.F.TotCurrLiab(Period=FQ0,Frq=FQ,AlignType=PeriodEndDate,Curn=USD,Scale=6,SDate=0,EDate=-27)',
        'TR.F.CashCashEquiv(Period=FQ0,Frq=FQ,AlignType=PeriodEndDate,Curn=USD,Scale=6,SDate=0,EDate=-27)',
        'TR.F.TradeAcctTradeNotesRcvblNet(Period=FQ0,Frq=FQ,AlignType=PeriodEndDate,Curn=USD,Scale=6,SDate=0,EDate=-27)',
        'TR.F.GrossProfIndPropTot(Period=FQ0,Frq=FQ,AlignType=PeriodEndDate,Curn=USD,Scale=6,SDate=0,EDate=-27)',
        'TR.F.IncBefTax(Period=FQ0,Frq=FQ,AlignType=PeriodEndDate,Curn=USD,Scale=6,SDate=0,EDate=-27)',
        'TR.F.NetIncAfterTax(Period=FQ0,Frq=FQ,AlignType=PeriodEndDate,Curn=USD,Scale=6,SDate=0,EDate=-27)',
        'TR.F.TotShHoldEq(Period=FQ0,Frq=FQ,AlignType=PeriodEndDate,Curn=USD,Scale=6,SDate=0,EDate=-27)',
        'TR.F.InvntTot(Period=FQ0,Frq=FQ,AlignType=PeriodEndDate,Curn=USD,Scale=6,SDate=0,EDate=-27)',
        'TR.F.DebtLTTot(Period=FQ0,Frq=FQ,AlignType=PeriodEndDate,Curn=USD,Scale=6,SDate=0,EDate=-27)',
        'TR.F.EBITDA(Period=FQ0,Frq=FQ,AlignType=PeriodEndDate,Curn=USD,Scale=6,SDate=0,EDate=-27)'
    ]
)
```

Figure 2. Data prompt parameters

```
display(df1.head(10))
```

	instrument	working_capital	retained_earnings_-_total	total_assets	earnings_before_interest_&_taxes_(ebit)	total_liabilities	revenue_from_business_activities_-_total	com
0	ESCA.OQ	149.257	144.881	298.718	4.883	140.243	72.136	
1	ESCA.OQ	159.73	143.699	320.324	4.22	163.035	74.904	
2	ESCA.OQ	151.384	142.403	316.852	8.189	160.859	94.337	
3	ESCA.OQ	144.927	138.034	307.061	9.023	155.442	72.38	
4	ESCA.OQ	122.862	133.122	251.798	6.409	105.183	73.444	
5	ESCA.OQ	122.195	131.162	254.156	7.672	109.436	81.298	
6	ESCA.OQ	115.847	131.354	231.617	10.686	86.484	99.679	
7	ESCA.OQ	118.155	127.975	231.96	7.129	90.06	59.191	
8	ESCA.OQ	99.326	125.237	220.705	6.921	81.549	74.767	
9	ESCA.OQ	83.707	126.86	198.716	12.815	57.687	78.069	

Figure 3. First 10 rows of dataframe1

Dataframe1 has following columns:

```
Column names:
- instrument
- working_capital
- retained_earnings_-_total
- total_assets
- earnings_before_interest_&_taxes_(ebit)
- total_liabilities
- revenue_from_business_activities_-_total
- company_market_cap
- total_current_assets
- total_current_liabilities
- cash_&_cash_equivalents
- trade_accounts_&_trade_notes_receivable_-_net
- gross_profit_-_industrials/property_-_total
- income_before_taxes
- net_income_after_tax
- total_shareholders'_equity_incl_minority_intr_&_hybrid_debt
- inventories_-_total
- debt_-_long-term_-_total
- earnings_before_interest_taxes_depreciation_&_amortization
```

Figure 4. Df1 Columns

Preprocessing dataframes

Regarding the parameters I've applied for retrieving data – columns above contain quarterly information for each of 169 companies for the last 7 years. Ideally, there should have been 7*28*169 of filled rows, but some of cells are not assigned due to reasons like relatively recent IPO date, inability to access the information about companies listed on small and unrecognized exchanges and e.t.c. Scale parameter is set to 6 which implies the scale in millions. Also, us dollars was used as default currency for all columns, fiscal quarter period end fx rates were used to convert values into US dollars. Parameter 'ReportingState=RSTD' means restated financial statements were used. Rows that contain NA cells were then dropped leaving us 2102 fully populated rows:

```
print(df1.shape)
display(df1)
```

```
(2102, 20)
```

Figure 5. Dataframe1 Shape

Indexes in the most left side represent periods in reversed manner. For example index 0 rep resents the most recent fiscal quarter while index 4 represents information that is 4 fiscal qu arters old. The information for the new company starts displaying every 28 index values.

Feature engineering

```
A = df1['working_capital'] / df1['total_assets']
B = df1['retained_earnings_-_total'] / df1['total_assets']
C = df1['earnings_before_interest_&_taxes_(ebit)'] / df1['total_assets']
D = df1['company_market_cap'] / df1['total_liabilities']
E = df1['revenue_from_business_activities_-_total'] / df1['total_assets']

df2 = pd.DataFrame({
    'altman_z_score': 1.2 * A + 1.4 * B + 3.3 * C + 0.6 * D + 1.0 * E
})
```

Figure 6. Calculation of Altman-Z score

Unlike in Bloomberg, there's no information on Altman-z score in Refinitiv Workspace, so I decided to calculate it by my own using previously fetched columns. The code snippet above creates new dataframe named df2 and calculates assigns altman-z-score values for all rows by adding column to newly created dataframe. After second dataframe is created we can easily assign additional new columns.

```
df2['current_ratio'] = df1['total_current_assets'] / df1['total_current_liabilities']
df2['quick_ratio'] = (df1['cash_&_cash_equivalents'] + df1['trade_accounts_&_trade_notes_receivable_-_net']) / df1['total_current_liabilities']
df2['cash_ratio'] = df1['cash_&_cash_equivalents'] / df1['total_current_liabilities']
df2['gross_profit_margin'] = (df1['gross_profit_-_industrials/property_-_total'] / df1['revenue_from_business_activities_-_total']) * 100
df2['operating_margin'] = (df1['earnings_before_interest_&_taxes_(ebit)'] / df1['revenue_from_business_activities_-_total']) * 100
df2['net_profit_margin'] = (df1['net_income_after_tax'] / df1['revenue_from_business_activities_-_total']) * 100
df2['pretax_profit_margin'] = (df1['income_before_taxes'] / df1['revenue_from_business_activities_-_total']) * 100
df2['ROA'] = (df1['net_income_after_tax'] / df1['total_assets']) * 100
df2['ROE'] = (df1['net_income_after_tax'] / df1['total_shareholders_equity']) * 100
df2['Inventory_turnover'] = (df1['revenue_from_business_activities_-_total'] - df1['gross_profit_-_industrials/property_-_total']) / df1['inventories_-_total']
df2['Asset_turnover'] = df1['revenue_from_business_activities_-_total'] / df1['total_assets']
df2['D/E'] = df1['debt_-_long-term_-_total'] / df1['total_shareholders_equity']
df2['D/Capital'] = df1['debt_-_long-term_-_total'] / (df1['debt_-_long-term_-_total'] + df1['total_shareholders_equity'])
df2['D/EBITDA'] = df1['debt_-_long-term_-_total'] / df1['earnings_before_interest_taxes_depreciation_&_amortization']
df2['Interest_coverage'] = df1['earnings_before_interest_&_taxes_(ebit)'] / (df1['earnings_before_interest_&_taxes_(ebit)'] - df1['income_before_taxes'])
```

Figure 7. Adding new columns to dataframe 2

As a result dataframe 2 looks as following

```
In [184]: display(df2)
          df2.shape
```

	altman_z_score	current_ratio	quick_ratio	cash_ratio	gross_profit_margin	operating_margin	net_profit_margin	pretax_profit_margin	ROA	ROE
0	2.424269	4.832901	1.576385	0.101872	21.727016	6.769158	3.748475	4.982256	0.905202	1.706263
1	2.012898	4.213689	1.393437	0.080478	17.340863	5.633878	3.949055	4.330877	0.92344	1.880615
2	2.089937	3.858891	1.250302	0.116993	24.241814	8.680581	6.013547	7.706414	1.790426	3.636702
3	2.209581	3.583968	1.313905	0.113966	27.008842	12.466151	9.193147	11.751865	2.166996	4.388632
4	2.717371	3.549374	1.460067	0.09076	21.632809	8.726377	6.632264	8.132727	1.934487	3.322307
...	...	...	...	...	...	...	...	...	...	...
4457	23.340918	18.451335	12.839057	0.000836	22.323123	19.826956	16.149833	20.352172	6.863667	7.36811
4458	5.589424	3.91314	2.935603	0.830425	16.773177	15.072381	10.927035	13.478012	3.329381	4.526365
4486	1.380466	1.067081	0.384703	0.365432	40.521668	17.920297	2.053755	-0.633206	0.509949	1.071033
4595	10.191806	3.697449	2.362343	1.517295	47.969587	19.878717	16.917905	18.6	3.204827	4.18014
4596	4.963643	1.092487	0.414684	0.074526	45.575085	18.681538	13.75359	17.273259	4.527704	14.983041

2102 rows × 16 columns

```
Out[184]: (2102, 16)
```

Figure 8. df2

And contains next columns:
Column names:
- altman_z_score
- current_ratio

- quick_ratio
- cash_ratio
- gross_profit_margin
- operating_margin
- net_profit_margin
- pretax_profit_margin
- ROA
- ROE
- Inventory_turnover
- Asset_turnover
- D/E
- D/Capital
- D/EBITDA
- Interest_coverage

Training GradientBoostingRegressor using preprocessed dataset

Gradient Boosting is a machine learning technique used for regression and classification tasks. It works by building an ensemble of weak models, typically decision trees, to create a strong model with improved performance. The term "gradient boosting" refers to how the algorithm uses the gradients of a loss function to boost the performance of the ensemble. Simple schema for gradient boosting model looks as following:

1. Initialization: The algorithm starts with an initial model, often a simple one like the average of the target values. This model will serve as the base for building the ensemble.
2. Calculate residuals: The residuals (differences between the predicted values and the actual values) are calculated for each observation in the dataset.
3. Train a weak learner: A weak model, usually a decision tree, is trained on the dataset using the residuals as the target values. This weak model tries to learn and predict the residuals, thereby capturing the errors made by the initial model.
4. Update the model: The weak learner's predictions are combined with the initial model's predictions using a learning rate (shrinkage) parameter. This updated model now makes improved predictions, as it has learned to correct some of the errors made by the initial model.
5. Iterate: Steps 2-4 are repeated for a predefined number of iterations or until a stopping criterion is met (e.g., no significant improvement in performance). Each iteration adds a new weak learner to the ensemble, which is trained to correct the errors made by the current ensemble.
6. Final model: The ensemble of weak learners is combined to form the final gradient boosting model. This model can be used to make predictions on new data by aggregating the predictions of all the weak learners in the ensemble.

Gradient boosting models are particularly effective because they can handle a wide variety of data types and automatically learn complex relationships between features and the target variable. The most important part of gradient boosting for which it stands out over

simple linear regression is that it can learn both linear and non-linear dependencies between regressor and regressand variables. The regressor variables in my particular case include all variables present in dataframe2 except for the Altman-z score, while the Altman-z score is a regressand variable.

```
X = df2.drop(["altman_z_score"], axis=1)
y = df2["altman_z_score"]
```

Figure 9. Assigning regressor and regressand variables

Next, I have printed descriptive statistics for altman-z score column in order to prevent the adverse effect of outliers, i.e there are values close to zero which makes Z-score extremely high or extremely low.

```
Statistics for 'altman_z_score':
Mean: 5.979325433676095
Min: -303.36310501591254
1%: -8.823337548414516
2%: -2.131881965847191
3%: -0.41146642277692114
4%: 0.17083330004582556
5%: 0.4454148507222065
15%: 1.08998014717955
25%: 1.464519419015567
50%: 2.491385581078278
75%: 3.9662588646921266
85%: 5.415700483666576
95%: 11.57410754119661
96%: 13.257894400254395
97%: 17.437104887063562
98%: 22.294491832045633
99%: 45.3872265413091
Max: 2012.4535581649482
Number of samples between 1.0% and 2.0%: 21
Number of samples between 2.0% and 3.0%: 21
Number of samples between 3.0% and 4.0%: 21
Number of samples between 4.0% and 5.0%: 21
Number of samples between 5.0% and 15.0%: 210
Number of samples between 15.0% and 25.0%: 210
Number of samples between 25.0% and 50.0%: 525
Number of samples between 50.0% and 75.0%: 525
Number of samples between 75.0% and 85.0%: 210
Number of samples between 85.0% and 95.0%: 210
Number of samples between 95.0% and 96.0%: 21
Number of samples between 96.0% and 97.0%: 21
Number of samples between 97.0% and 98.0%: 21
Number of samples between 98.0% and 99.0%: 21
```

Figure 10. Preliminary descriptive statistics for Altman-Z score

We can see that samples that correspond to min and max values heavily deviate from general cluster due to the small sample size and low statistical power, these data points cannot be relied upon and should be excluded from the analysis. Also, we can observe that main cluster of samples is located between 5^{th} 95^{th} percentiles. So, we simply cut the all rows that corresponds to z-scores located beyond these boundaries.

```
y_5th_percentile = y.quantile(0.05)
y_95th_percentile = y.quantile(0.95)

mask = (y >= y_5th_percentile) & (y <= y_95th_percentile)
X = X.loc[mask]
y = y.loc[mask]
```

Figure 12. Normalized Z-score

```
Statistics for 'altman_z_score':
Mean: 2.9763405579837934
Min: 0.4458046726050687
5%: 0.8208968258350551
15%: 1.2386834018690422
25%: 1.5466214623274706
50%: 2.491385581078278
75%: 3.759265989234293
85%: 4.724636918858335
95%: 7.179788149999908
Max: 11.481655713028786
```

Figure 11. normalized z-score descriptive statistics

Next, we scale each of input columns so that model won't have any size bias. This usually occurs when model understands inputs that have scale in millions as more powerful than those which have a scale in billions. Even though we initially prompted data in a scale of million us.dollars some values are still extremely large compared to other. Usually minmax or standardscaler are applied to datasets, but in our case we will apply RobustScaler which generally stands out for being robust to outliers. The RobustScaler works by first computing the median and interquartile range (IQR) of each feature. The median is the middle value in a dataset, while the IQR is a measure of the spread of the data that is less sensitive to outliers than the standard deviation. The RobustScaler then scales each feature by subtracting the median and dividing by the IQR. The RobustScaler is a good choice when dealing with datasets that contain outliers or when the data is not normally distributed.

```
from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()
X = scaler.fit_transform(X)
```

Figure 13.RobustScaler

Afterwards, I have handled the issue of gradient boosting model overfitting. Gradient Boosting is a powerful technique that can be prone to overfitting if not properly tuned. Overfitting occurs when model is too closely fitted to the training data and does not generalize well to new unseen data. To mitigate the issue of overfitting I will be using techniques such as cross-validation and gridsearch. Cross-validation is a technique used in machine learning to evaluate the performance of a model and estimate how well it will generalize to new, unseen data. The basic idea of cross-validation is to partition the available data into several subsets, called folds, and then use each fold as a validation set while the model is trained on the remaining data.

Gridsearch is a method for hyperparameter tuning in machine learning and deep learning models. It is an exhaustive search technique that systematically goes through all the possible combinations of hyperparameters within a specified range or a predefined set of values, in order to find the best configuration that maximizes the performance of the model. In a grid search, the model is trained and evaluated multiple times using different hyperparameter combinations, and the best performing combination is selected based on a specific evaluation metric, such as accuracy or mean squared error.

```
param_grid = {
    'n_estimators': [50, 100, 200, 300, 400, 500],
    'learning_rate': [0.001, 0.01, 0.1, 0.2],
    'max_depth': [3, 4, 5, 10, 15],
    'min_samples_split': [2, 3, 4],
    'min_samples_leaf': [1, 2, 3]
}
```

Figure 14. Parameters for gridsearch

The following step is to train the model:

```
from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.20, shuffle=True, random_state=1)
```

Figure 15. Train test split

In the snippet above, I have splitted dataset into training and validation sets, where 80% of samples (rows) go to training set and rest to the validation set. Shuffle=True means that samples will randomly shuffled, so model can learn patterns in the dataset without ordering interference. Random_state=1 accounts for returning same result over multiple execution of code.

```
gbr_cv = GridSearchCV(gbr, param_grid,scoring='neg_mean_squared_error', n_jobs=-1, cv=5, verbose=2)
gbr_cv.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 1080 candidates, totalling 5400 fits
```

Figure 16. Fitting the model

Parameter cv=5 stands for 5 fold cross validation. (6 x 4 x 5 x 3 x 3) = 1080 different combinations of hyperparameters to evaluate. With 5 fold cross validation that will be 5400 fits in total. Such model training took multiple hours of intensive computation and as a result will have RMSE of 1.28.

```
y_pred = best_gbr.predict(X_val)
rmse = mean_squared_error(y_val, y_pred, squared=False)
print("RMSE for validation set:", rmse)
```

```
RMSE for validation set: 1.2833748114614423
```

Figure 17. Evaluating the model on validation set

RMSE is the square root of the average of the squared differences between the predicted

values and the true values. It is expressed in the same units as the target variable being predicted. The formula for RMSE is:

RMSE = sqrt((1/n) * sum((y_pred - y_true)^2))

where:

- y_pred is the predicted value
- y_true is the true (observed) value
- n is the number of observations

Figure 18. RMSE

Considering the fact most of samples are located above (50%) the deviation of 1.28 is not the best but still satisfactory result.

Retrieving weights from Permutation feature importances

The main goal is not to train a model that can predict altman-z scores based on input variable, but rather fetch the weights to create the equation for our synthetic credit scoring model. So, basically what we want to do is to order the weights(coefficients) for each independent variable in terms of their contribution to overall model goodness of fit, given that altman z-score represents the reliable & relative measurement of credit quality. Another crucial idea is that we make the assumption that the independent variables can explain the variation in the output variable almost perfectly, and the achievability of that relationship is reliant on the researcher's knowledge and the technologies available to them.

Permutation importance is a technique used to evaluate the relative importance of features in a machine learning model by measuring the effect of randomly shuffling the values of a single feature on the model's performance. The idea is that if a feature is important for the model's predictions, then permuting its values should lead to a significant decrease in the model's performance (such as accuracy, F1 score, or mean squared error). Conversely, if a feature is not very important, permuting its values should have little impact on the model's performance.
The process to compute permutation importance is as follows:

1. Train a machine learning model using the original dataset with all features.
2. Measure the baseline performance of the model on a validation or test dataset.
3. For each feature, follow these steps:
   a. Create a copy of the validation or test dataset.
   b. Randomly shuffle the values of the selected feature in the copied dataset, keeping all other feature values intact.
   c. Measure the performance of the model on the shuffled dataset.

d. Calculate the difference between the baseline performance and the performance on the shuffled dataset.

4. Rank the features by their absolute performance differences. The larger the difference, the more important the feature is.

Why permutation importance instead of correlation coefficient?

- Permutation importance is more comprehensive than correlation coefficient as it can capture both linear and non-linear relationships between independent and dependent variables. When there is an exponential dependency between the regressor and regressand variables, correlation coefficient may underestimate the significance of the variables. This feature of permutation importance is vital for our model, because moderate dataleak is present in the dataframe.
- Multicollinearity. Permutation importance is an effective approach to handle multicollinearity by accounting for the correlation between independent variables. In contrast, using correlation coefficients to assess dependencies between the regressor and regressand variables can lead to biased and inaccurate insights.

```
import eli5
from eli5.sklearn import PermutationImportance
```

```
permImportance = PermutationImportance(best_gbr, random_state=1).fit(X_val,y_val)
eli5.show_weights(permImportance, feature_names = X.columns.tolist())
```

Figure 19, Snippet for applying permutation importance

Weight	Feature
0.3139 ± 0.0358	current_ratio
0.2784 ± 0.0634	D/Capital
0.1431 ± 0.0288	Inventory_turnover
0.0820 ± 0.0251	gross_profit_margin
0.0676 ± 0.0121	Asset_turnover
0.0534 ± 0.0066	cash_ratio
0.0470 ± 0.0420	ROA
0.0441 ± 0.0169	quick_ratio
0.0437 ± 0.0284	Interest_coverage
0.0416 ± 0.0215	ROE
0.0280 ± 0.0071	D/E
0.0243 ± 0.0083	net_profit_margin
0.0162 ± 0.0234	D/EBITDA
0.0073 ± 0.0048	operating_margin
0.0052 ± 0.0035	pretax_profit_margin

Figure 20. Permutation importance weights

We can interpret this weights as following:

1. For the last 7 years current ratio was the most important determinant that defined the credit quality of company in the recreational products industry.
2. Debt to capital ratio was the second most important ratio in the simillar fashion.
3. Same for remaining ratios

Looks pretty reasonable considering the business model that is inherent to recreational products industry, isn't it ?

Defining the equation of synthetic credit score

The goal of the synthetic credit score is to serve as a comparative metric, similar to the Altman-Z score. Therefore, instead of extensively experimenting to obtain an absolute measure, our focus is on observing the varying behavior of the synthetic credit score of company A in comparison to that of company B over time or across samples.
So for simplicity our synthetic credit score will be equal to weighted average of inputs:

Synthetic Credit Score = 0.3139 x Current Ratio + 0.2784 x Debt to Capital Ratio + 0.1431 x Inventory Turnover + 0.0820 x Gross Profit Margin + 0.0676 x Asset Turnover + 0.0534 x Cash Ratio + 0.0470 x Return on Assets + 0.0441 x Quick Ratio + 0.0437 x Interest Coverage + 0.0416 x Return on Equity + 0.0280 x Debt to Equity Ratio + 0.0243 x Net Profit Margin + 0.0162 x Debt to EBITDA Ratio + 0.0073 x Operating Margin + 0.0052 x Pre-Tax Profit Margin

After defining the equation we simply do the vectorization with this equation in order to create a new column. Now our dataframe 2 contains following columns:

```
Column names:
- altman_z_score
- current_ratio
- quick_ratio
- cash_ratio
- gross_profit_margin
- operating_margin
- net_profit_margin
- pretax_profit_margin
- ROA
- ROE
- Inventory_turnover
- Asset_turnover
- D/E
- D/Capital
- D/EBITDA
- Interest_coverage
- synthetic_credit_score
```

Figure 21. Dataframe2 with synthetic credit score

Exploratory data analysis

First we want to obtain descriptive statistics for the newly created synthetic_credit_score column.

```
z = df2['synthetic_credit_score']
```

```
z = z[np.isfinite(z)]
```

Figure 22. Snippet for synthetic credit score

Np.isfinite(z) is required to save only finite values in the new column. Infinite values could have occurred due to dividing to zero and e.t.c

Next we obtain descriptive statistics for the synthetic credit score

```
Mean: 75.90487642697033
Min: -4313.771650274892
1%: -35.129422576617365
2%: -11.626990698786553
3%: -5.250674374542088
4%: -3.8725606081054575
5%: -2.0297813588674347
10%: 0.778037584350063
15%: 1.8760945498631927
20%: 2.2936030759603985
25%: 2.5715416946365184
30%: 2.824194445214631
35%: 3.0752470021245064
40%: 3.314353250770683
45%: 3.5448738909230997
50%: 3.762185237531173
55%: 4.032452568393876
60%: 4.341273075088405
65%: 4.620022836434625
70%: 4.912225652760653
75%: 5.291725416738861
80%: 5.77619239535284
85%: 6.378934179291242
90%: 7.263538060717803
95%: 9.71230966042894
96%: 11.02772398379463
97%: 13.425534841951364
98%: 16.63434402413618
99%: 25.564291090521447
Max: 159487.85180210747
```

Figure 24.Descriptive statistics for synthetic credit score

```
Number of samples between 0% and 1.0%: 20
Number of samples between 1.0% and 2.0%: 21
Number of samples between 2.0% and 3.0%: 21
Number of samples between 3.0% and 4.0%: 21
Number of samples between 4.0% and 5.0%: 21
Number of samples between 5.0% and 10.0%: 104
Number of samples between 10.0% and 15.0%: 104
Number of samples between 15.0% and 25.0%: 208
Number of samples between 25.0% and 30.0%: 104
Number of samples between 30.0% and 35.0%: 104
Number of samples between 35.0% and 40.0%: 104
Number of samples between 40.0% and 45.0%: 104
Number of samples between 45.0% and 50.0%: 104
Number of samples between 50.0% and 55.00000000000001%: 104
Number of samples between 55.00000000000001% and 60.0%: 104
Number of samples between 60.0% and 65.0%: 104
Number of samples between 65.0% and 70.0%: 104
Number of samples between 70.0% and 75.0%: 104
Number of samples between 75.0% and 85.0%: 208
Number of samples between 85.0% and 95.0%: 208
Number of samples between 95.0% and 96.0%: 20
Number of samples between 96.0% and 97.0%: 21
Number of samples between 97.0% and 98.0%: 21
Number of samples between 98.0% and 99.0%: 21
Number of samples between 99.0% and 100%: 21
```

Figure 23. Descriptive statistics 2 for synthetic credit score

Here we can observe that again we have extreme values that distorts the mean and st.dev values. To derive more accurate mean and st.dev we just smooth the label (i.e save only values between 1% and 99% inclusively) as we did previously.

```
p1 = z.quantile(0.01)
p99 = z.quantile(0.99)

df22 = df2[(df2['synthetic_credit_score'] >= p1) & (df2['synthetic_credit_score'] <= p99)]
df22 = df22.round(2)
```

Figure 25.Snippet for label smoothing

Next we fetch mean and standard deviation figures in order to understand if certain company is doing good or bad. Also we plot histogram for the synthetic scores.

```
mean_synthetic_credit_score = df22['synthetic_credit_score'].mean()
std_synthetic_credit_score = df22['synthetic_credit_score'].std()

print(f"Mean of synthetic_credit_score: {mean_synthetic_credit_score}")
print(f"Standard deviation of synthetic_credit_score: {std_synthetic_credit_score}")
```

```
Mean of synthetic_credit_score: 3.831495831289848
Standard deviation of synthetic_credit_score: 4.261064321846587
```

Figure 26. Mean and St.dev

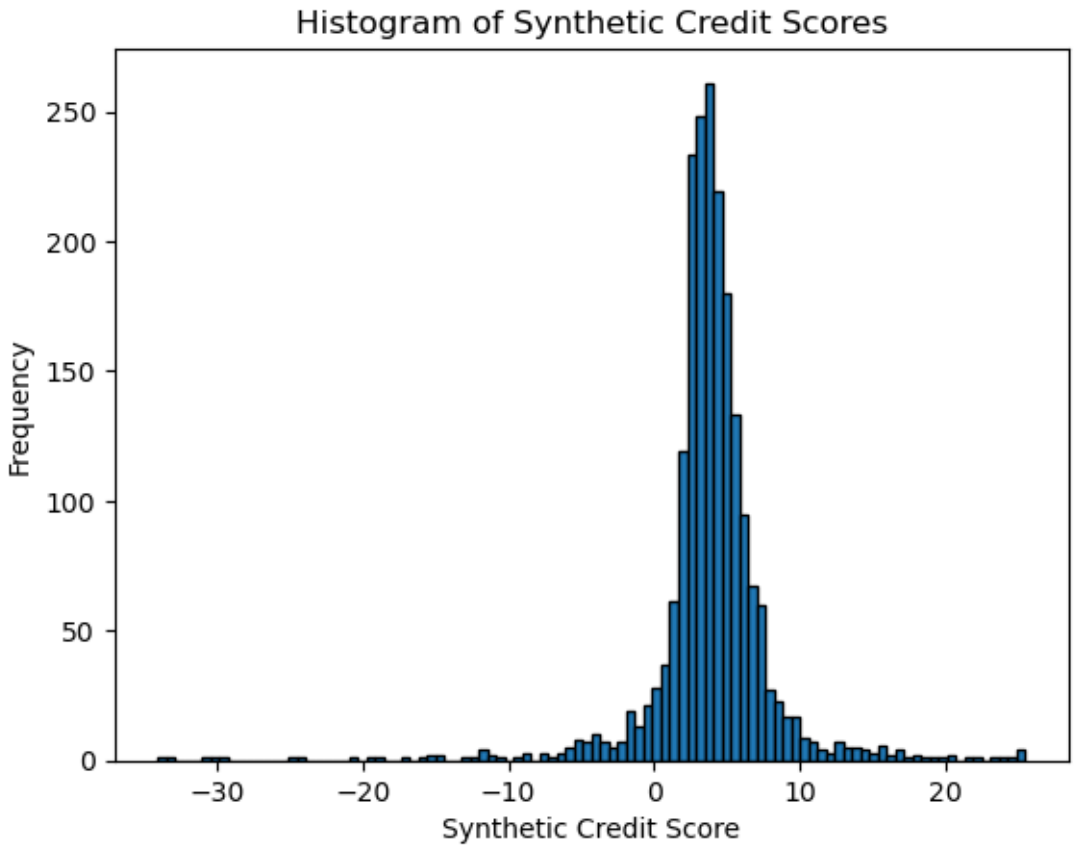


Figure 25. Burj khalifa in a nutshell

Despite mean is not equal to 0, we can observe that distribution has bell shape curve which is the sign of normal distribution. This tells that our synthetic credit score is not biased and makes reliable credit measurement. We can validate normality by following way

```
within_1_std = np.sum(((synthetic_credit_scores >= (mean_synthetic_credit_score - std_synthetic_credit_score)) &
                       (synthetic_credit_scores <= (mean_synthetic_credit_score + std_synthetic_credit_score))))

within_2_std = np.sum(((synthetic_credit_scores >= (mean_synthetic_credit_score - 2 * std_synthetic_credit_score)) &
                       (synthetic_credit_scores <= (mean_synthetic_credit_score + 2 * std_synthetic_credit_score))))

within_3_std = np.sum(((synthetic_credit_scores >= (mean_synthetic_credit_score - 3 * std_synthetic_credit_score)) &
                       (synthetic_credit_scores <= (mean_synthetic_credit_score + 3 * std_synthetic_credit_score))))

print(f"Number of values within 1 standard deviation: {within_1_std}")
print(f"Number of values within 2 standard deviations: {within_2_std}")
print(f"Number of values within 3 standard deviations: {within_3_std}")
```

```
Number of values within 1 standard deviation: 1773
Number of values within 2 standard deviations: 1937
Number of values within 3 standard deviations: 1991
```

Figure 26. Number of values

Next we plot correlation matrix for the industry

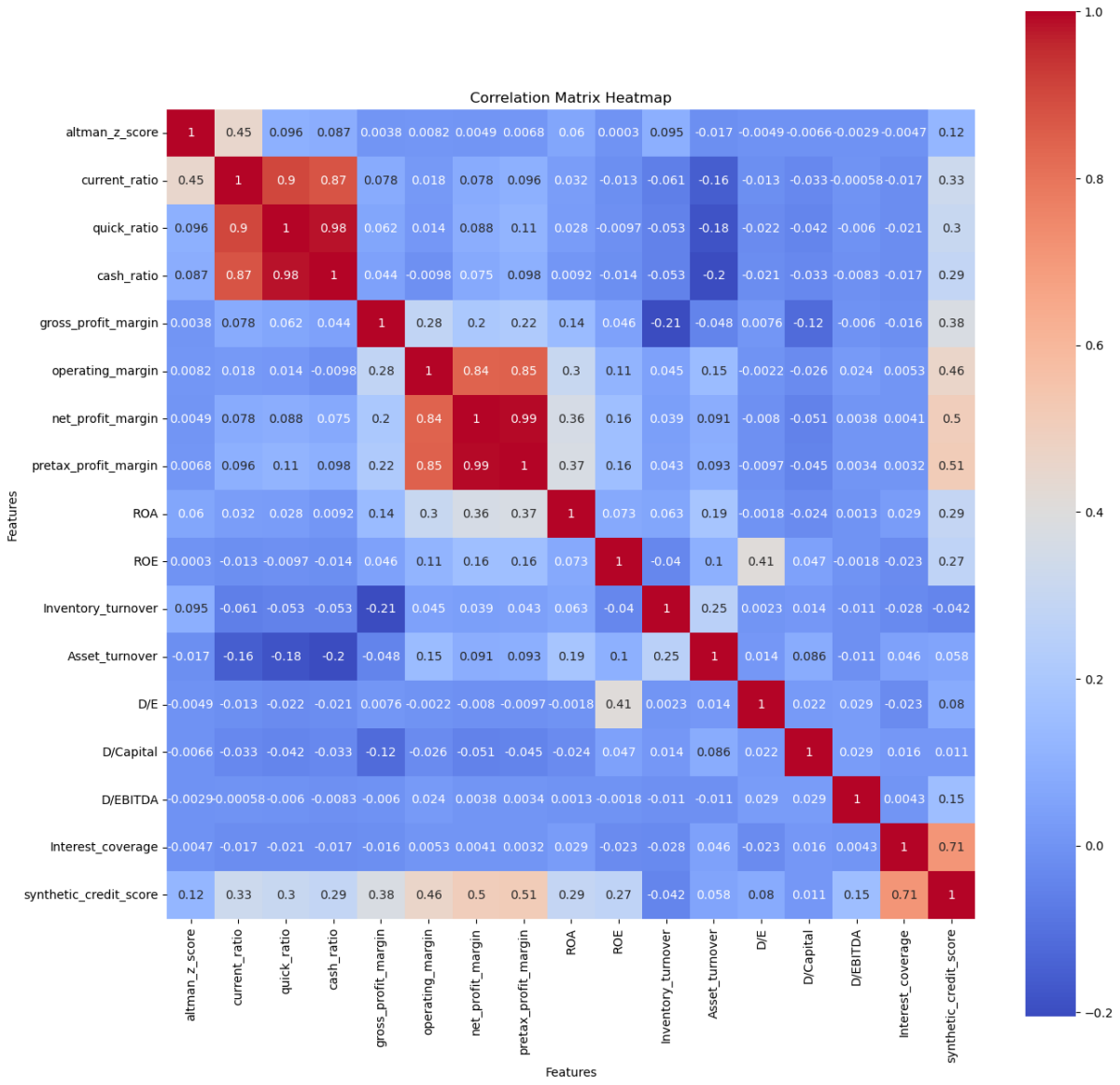
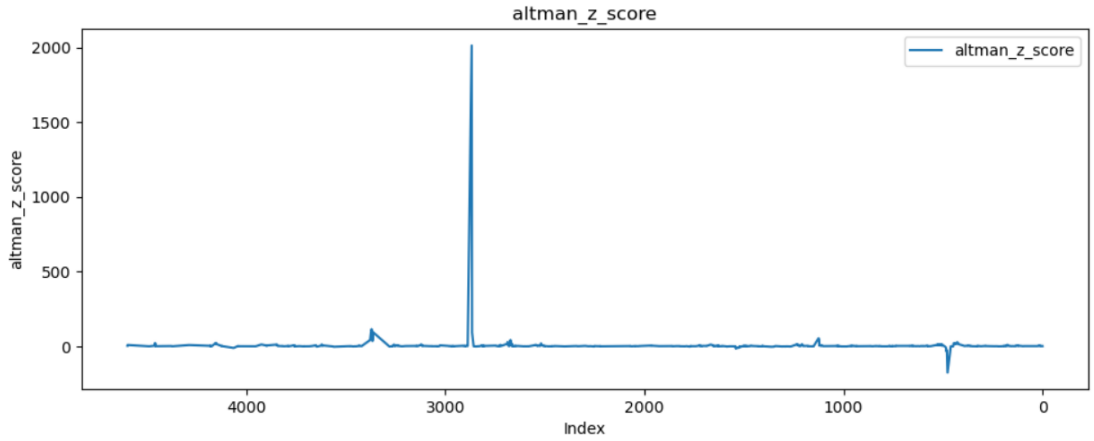
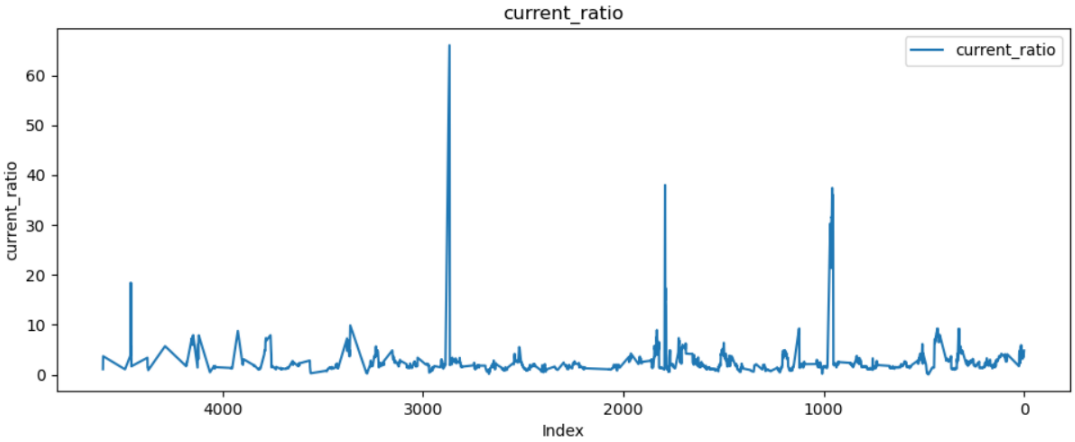


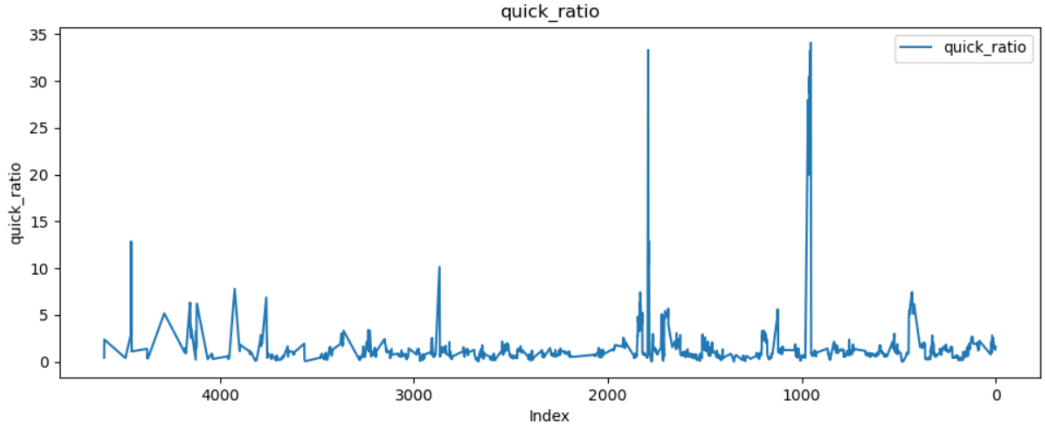
Figure 27. Correlation matrix for recreational products industry

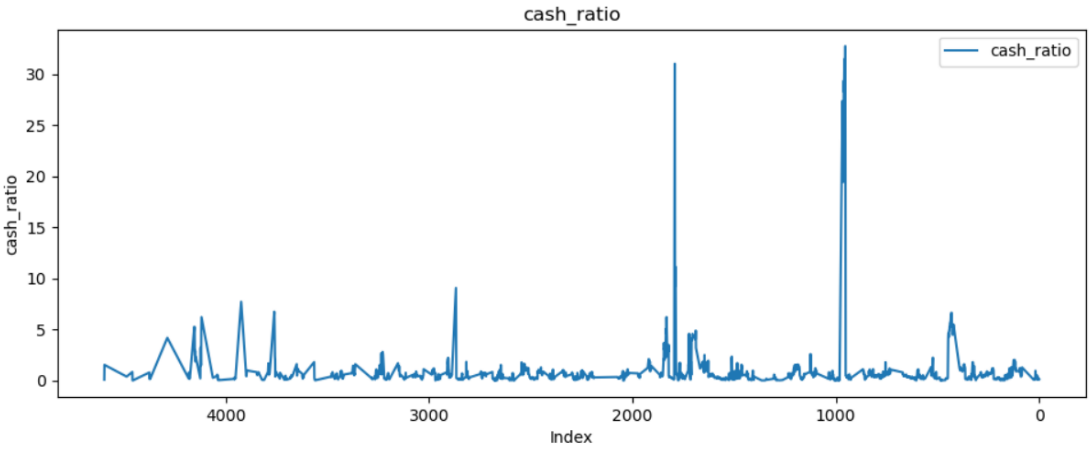
From the correlation matrix, it is evident that the majority of ratios exhibit a positive correlation with the synthetic credit score. This indicates that a higher synthetic credit score corresponds to better credit quality of the company. Moreover we can observe that interest coverage ratio has highest correlation with synthetic credit score which gives us additional insight on importance of managing operating expenses and debts if companies want to increase their credit quality. Furthermore, it is apparent that effective management of net income, EBIT (earnings before interest and taxes), and pre-tax profit margins are key factors in achieving higher credit quality.

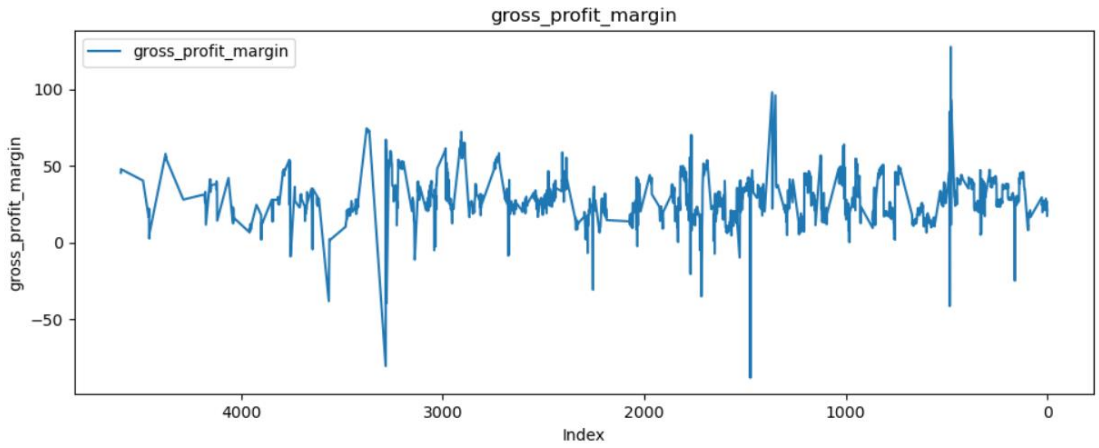
Proceeding further through the exploratory data analysis we can plot combined chart for the time series of all companies in the dataframe.
Click if you don't want to scroll through the charts

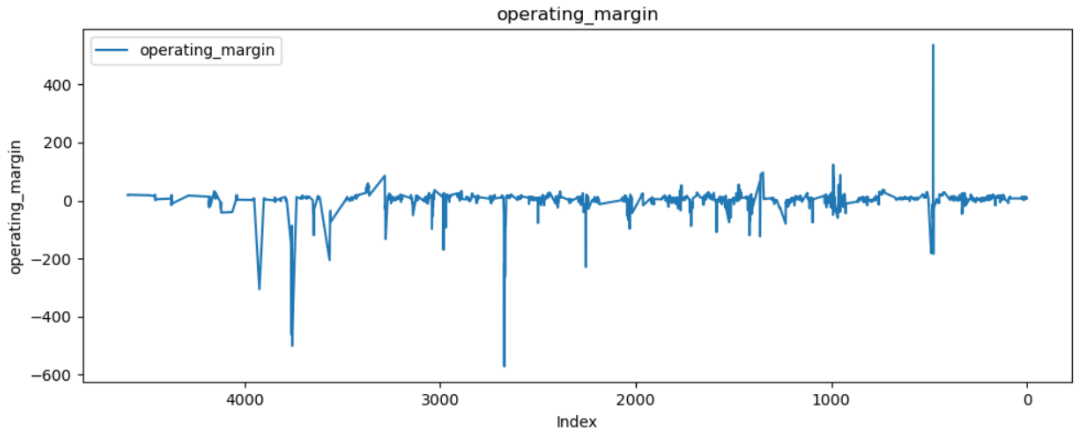

altman_z_score
altman_z_score
2000
1500
1000
500
0
4000
3000
2000
1000
0
Index

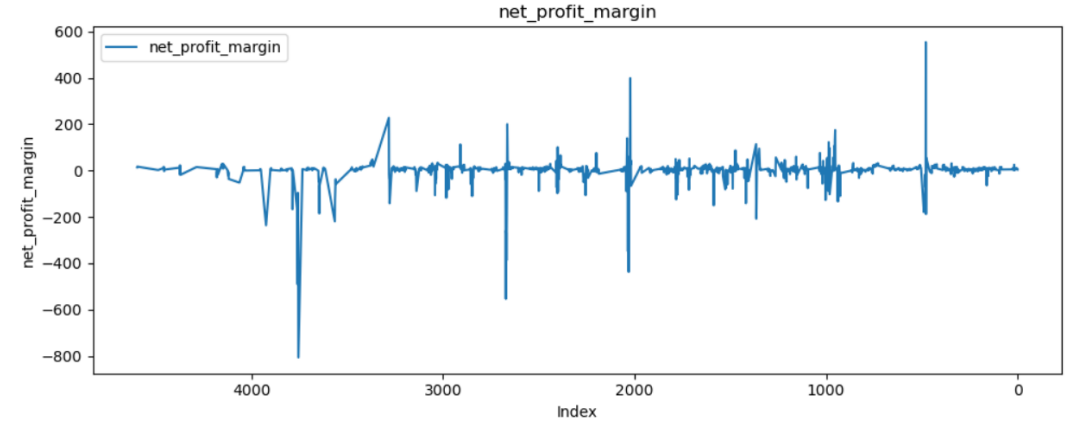

current_ratio
current_ratio
60
50
40
30
20
10
0
4000
3000
2000
1000
0
Index

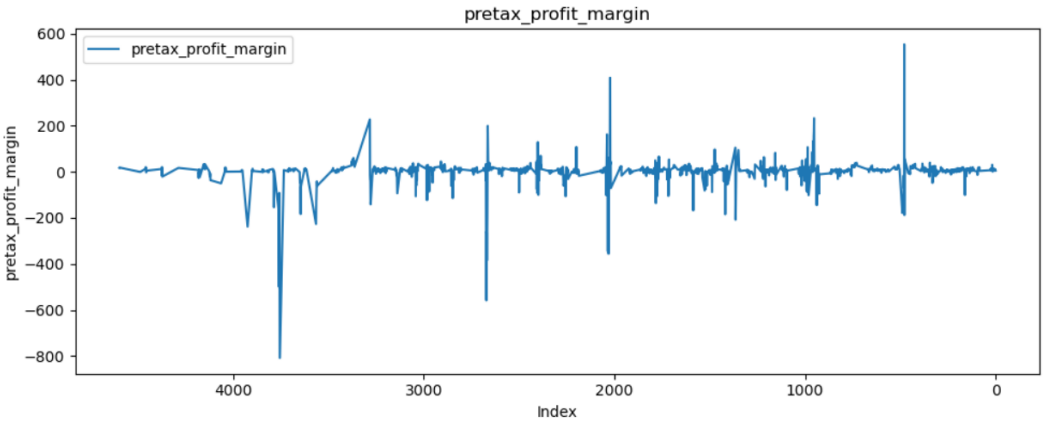

quick_ratio
quick_ratio
35
30
25
20
15
10
5
0
4000
3000
2000
1000
0
Index

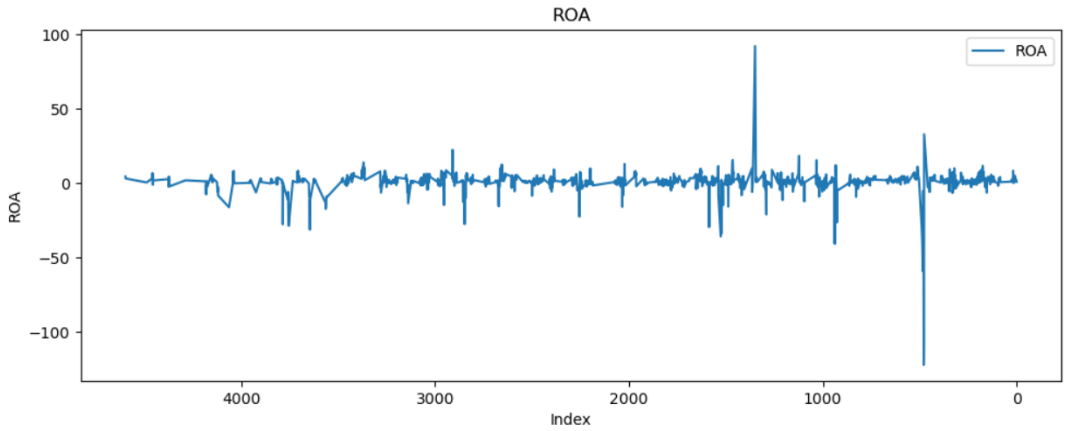

cash_ratio
cash_ratio
30
25
20
15
10
5
0
4000
3000
2000
1000
0
Index

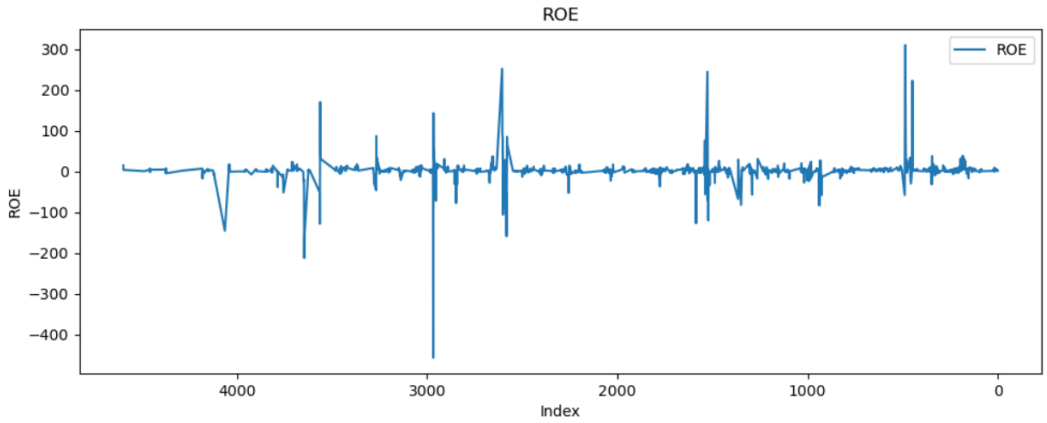

gross_profit_margin
gross_profit_margin
100
50
0
−50
gross_profit_margin
4000
3000
2000
1000
0
Index

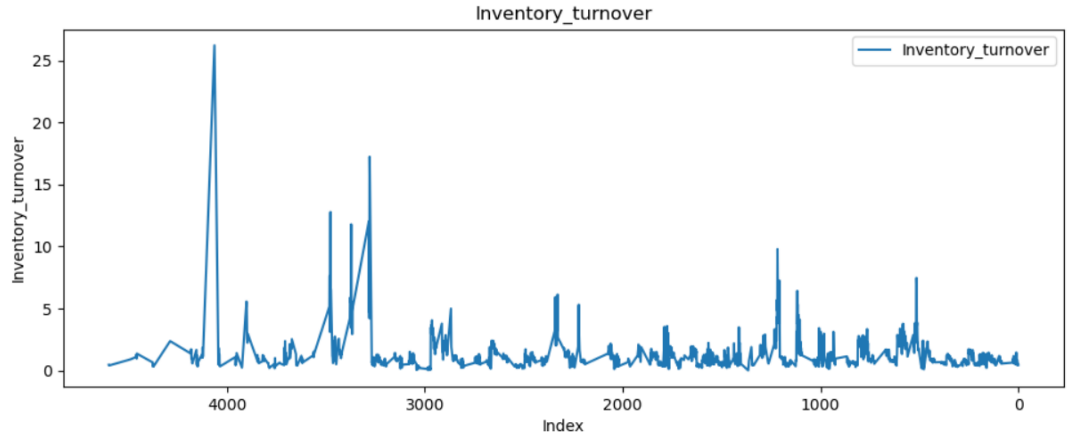

operating_margin
operating_margin
400
200
0
−200
−400
−600
operating_margin
4000
3000
2000
1000
0
Index

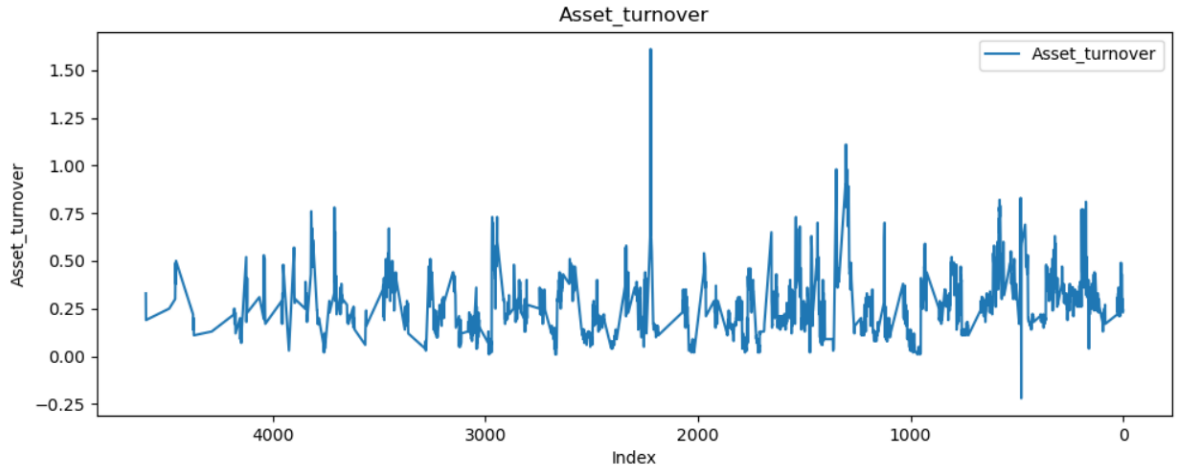

net_profit_margin
net_profit_margin
600
400
200
0
−200
−400
−600
−800
net_profit_margin
4000
3000
2000
1000
0
Index

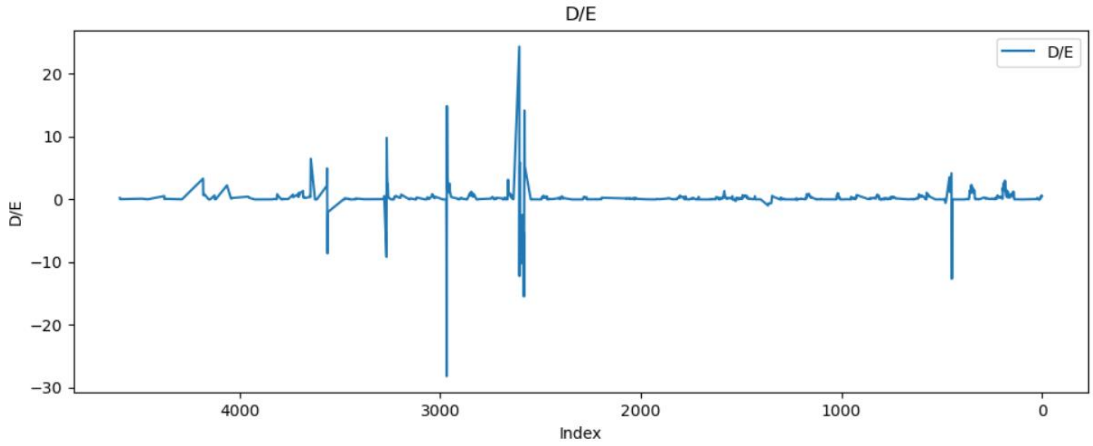

pretax_profit_margin
pretax_profit_margin
600
400
200
0
−200
−400
−600
−800
pretax_profit_margin
4000
3000
2000
1000
0
Index

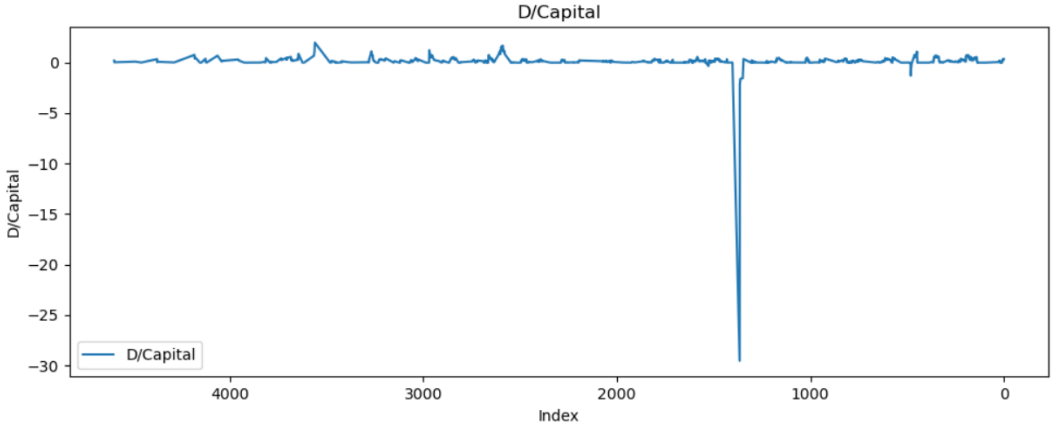

ROA
ROA
100
50
0
−50
−100
4000
3000
2000
1000
0
Index

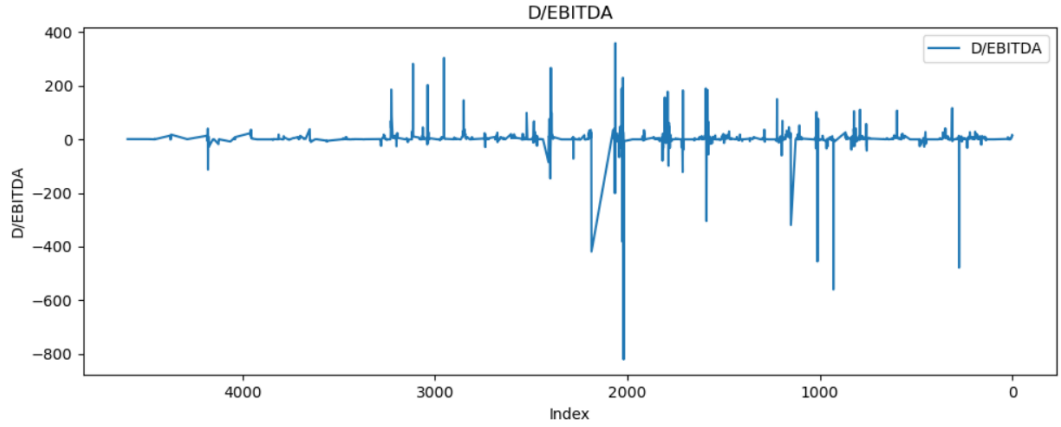

ROE
ROE
300
200
100
0
−100
−200
−300
−400
4000
3000
2000
1000
0
Index

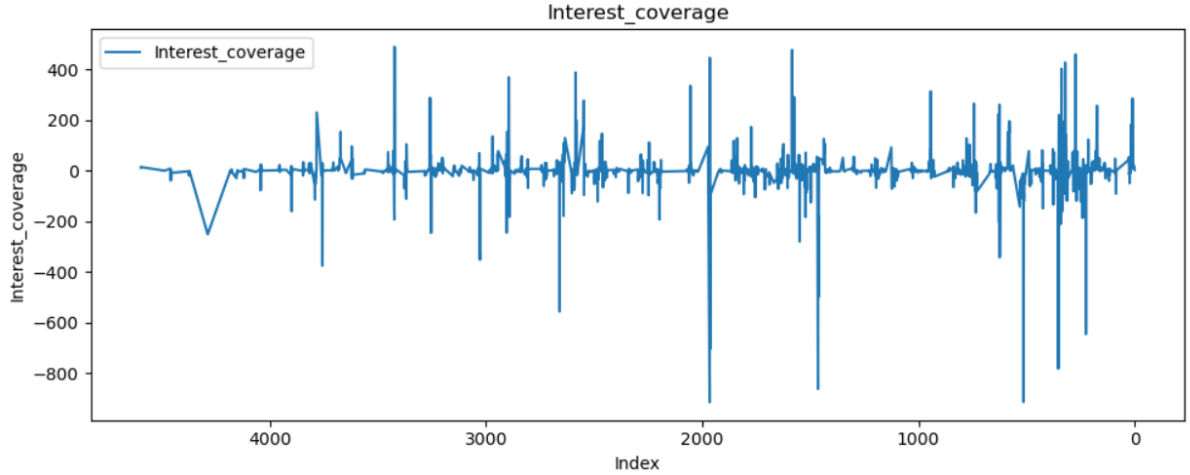

Inventory_turnover
Inventory_turnover
25
20
15
10
5
0
4000
3000
2000
1000
0
Index

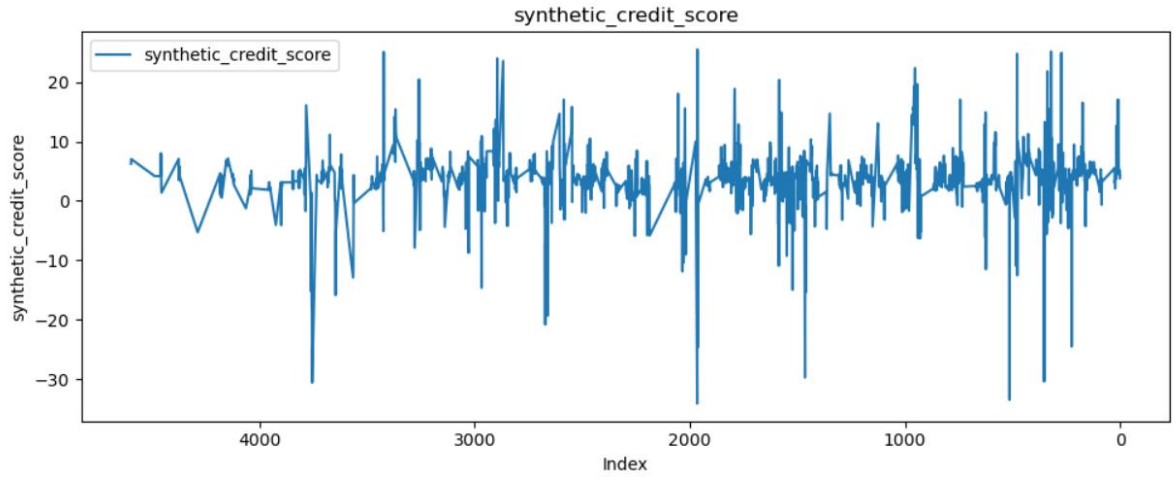

Asset_turnover
Asset_turnover
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
4000
3000
2000
1000
0
Index


D/E
D/E
20
10
0
−10
−20
−30
4000
3000
2000
1000
0
Index


D/Capital
D/Capital
0
−5
−10
−15
−20
−25
−30
D/Capital
4000
3000
2000
1000
0
Index


D/EBITDA
D/EBITDA
400
200
0
−200
−400
−600
−800
4000
3000
2000
1000
0
Index


Interest_coverage
Interest_coverage
400
200
0
−200
−400
−600
−800
4000
3000
2000
1000
0
Index



Basically, charts above represent corresponding values for 28 quarters for all companies in the dataframe. On the index axis, companies are arranged such that every 28 index points represent quarterly values for the last 7 years for a specific company. The timeline is ordered from left to right, with the left side representing earlier years(quarters) and the right side representing more recent years(quarters). Plotting all companies on a single plane in a chart can make the chart static, with no discernible trends. However, this approach enables us to easily identify outliers and visualize seasonal effects in the data.

Analysis of Peloton inc

Peloton, a fitness company established in 2012, offers at-home workout equipment and subscription-based fitness classes. In September 2019, Peloton went public, raising $1.6 billion on NASDAQ with 577,000 products sold and 500,000 paying subscribers. The COVID-19 pandemic in 2020 led to a surge in demand, with a 94% increase in subscribers and a 66% increase in sales. In response, Peloton introduced Bike+ and reduced the standard bike price from $2245 to $1895.

Despite the successes, delivery delays, broken pedals, and customer injuries resulted in recalls affecting 30,000 bikes. In 2021, Peloton reached its first billion-dollar quarter, driven by the pandemic and holiday sales. However, a tragic treadmill accident involving a child's death led to a decline in the company's trustworthiness. Peloton's founder, John Foley, invested heavily in manufacturing and delivery, but challenges persisted.

The year 2022 was difficult for Peloton, with its stock tanking and multiple setbacks, including the resignation of then-CEO John Foley, 2,800 employee layoffs, and product recalls. New products like the Peloton Guide and Peloton Row faced obstacles, and the company's stock price plunged 90% within 12 months.

In 2023, Peloton's recovery will depend on rebuilding trust and proving its viability. Despite the challenges, the company has a solid product lineup and a loyal customer base, which could help in its turnaround efforts.
Let’s take a look on how everything reflected on financial ratios of Peloton

Since the company is public from septermber 2019 we have only 15 10q reports. Long-term debt was reported by companies in only 9 out of the 15 quarterly reports. To incorporate solvency ratios into our analysis, we will examine only the 9 most recent quarterly reports.

```
dfpton1.rename(columns={
    'current_ratio': 'CR',
    'quick_ratio': 'QR',
    'cash_ratio': 'CashR',
    'gross_profit_margin': 'GPM',
    'operating_margin': 'OM',
    'net_profit_margin': 'NPM',
    'pretax_profit_margin': 'PPM',
    'inventory_turnover': 'IT',
    'asset_turnover': 'AT',
    'interest_coverage': 'IC',
    'synthetic_credit_score': 'synth.cr.score',
    'asset_turnover': 'AT'
}, inplace=True)
```

The column headers were changed for improved visual clarity.

	Date	CR	QR	CashR	GPM	OM	NPM	PPM	roa	roe	IT	AT	d/e	d/capital	d/ebitda	IC	synth.cr.score
0	2023 Q1	2.24	25.22	1.07	36.08	-28.66	-36.86	-36.75	-9.15	217.32	3.93	0.25	-13.21	1.08	-38.3	0.98	12.08
1	2022 Q4	2.28	20.04	0.96	29.66	-33.63	-42.31	-42.07	-10.16	-1099.67	4.13	0.24	54.95	0.98	-7.12	0.99	-41.29
2	2022 Q3	2.47	19.27	1.01	35.23	-32.07	-66.26	-66.13	-11.37	-158.03	3.07	0.17	6.48	0.87	-9.93	0.96	-3.97
3	2022 Q2	2.38	23.85	1.13	-4.39	-119.57	-184.94	-183.16	-31.16	-211.71	6.18	0.17	2.62	0.72	-2.02	0.99	-13.91
4	2022 Q1	2.49	19.8	0.85	19.1	-33.78	-78.51	-78.3	-17.15	-43.19	8.44	0.22	0.49	0.33	-2.97	0.98	-0.57
5	2021 Q4	2.33	33.09	1.08	24.71	-34.32	-38.75	-38.48	-8.01	-18.55	9.3	0.21	0.36	0.26	-2.4	0.98	3.17
6	2021 Q3	1.96	18.05	0.47	32.63	-40.92	-46.7	-46.4	-8.52	-24.95	6.26	0.18	0.56	0.36	-2.78	0.97	2.02
7	2021 Q2	2.27	32.02	0.91	27.07	-28.19	-33.43	-33.26	-6.98	-17.86	7.47	0.21	0.47	0.32	-3.47	0.97	3.3
8	2021 Q1	2.74	45.28	1.6	35.25	0.55	-0.68	-1.4	-0.18	-0.43	11.28	0.27	0.41	0.29	38.03	46.98	10.18

Figure 28. Financial ratios - quarterly

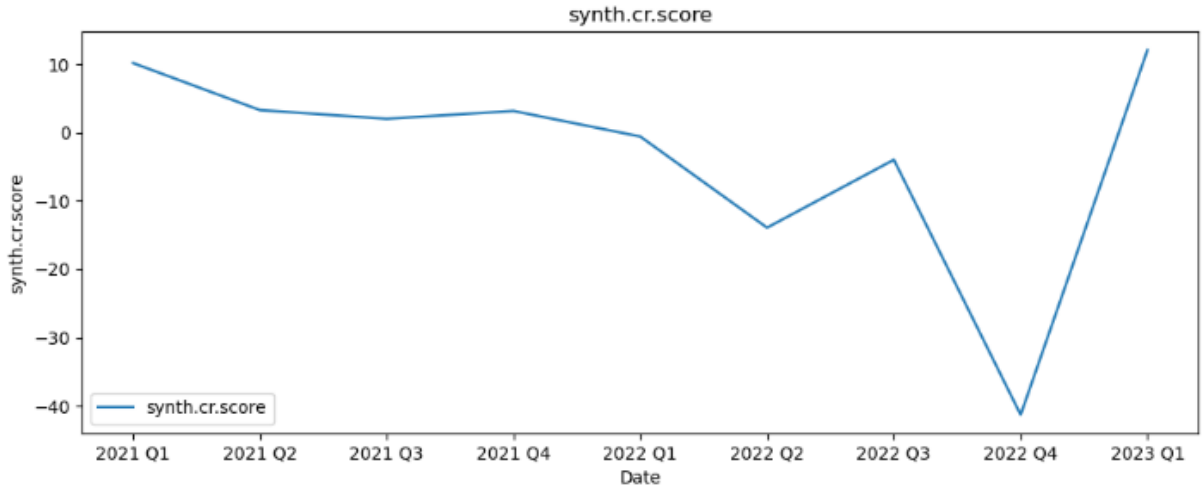


Figure 29. Synthetic Credit Score

In early 2021, Peloton's synthetic credit score was more than two standard deviations above the industry mean. Throughout the remainder of the year, the score remained within one standard deviation of the industry mean. However, in 2022, the score fell to more than two standard deviations below the industry mean and remained at that level until the first quarter of 2023, when it began to recover.

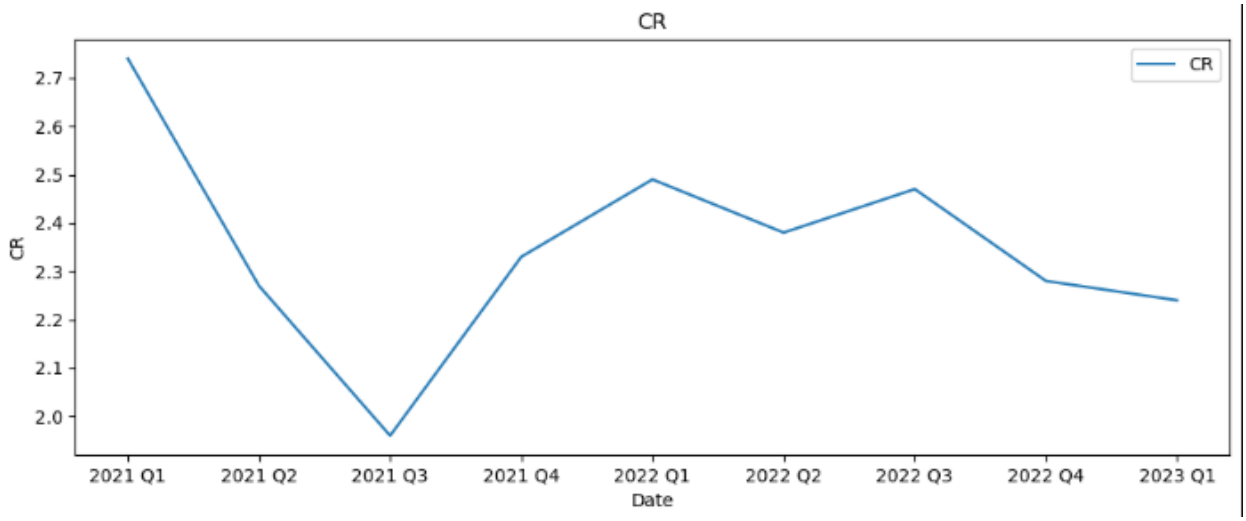


Figure 30. Current ratio

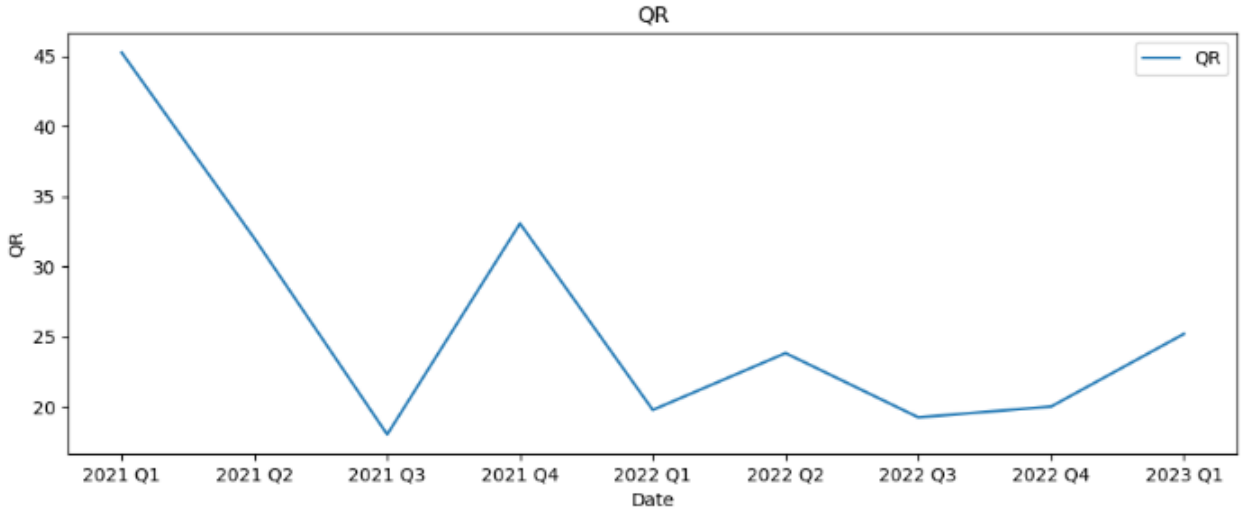


Figure 31.Quick ratio

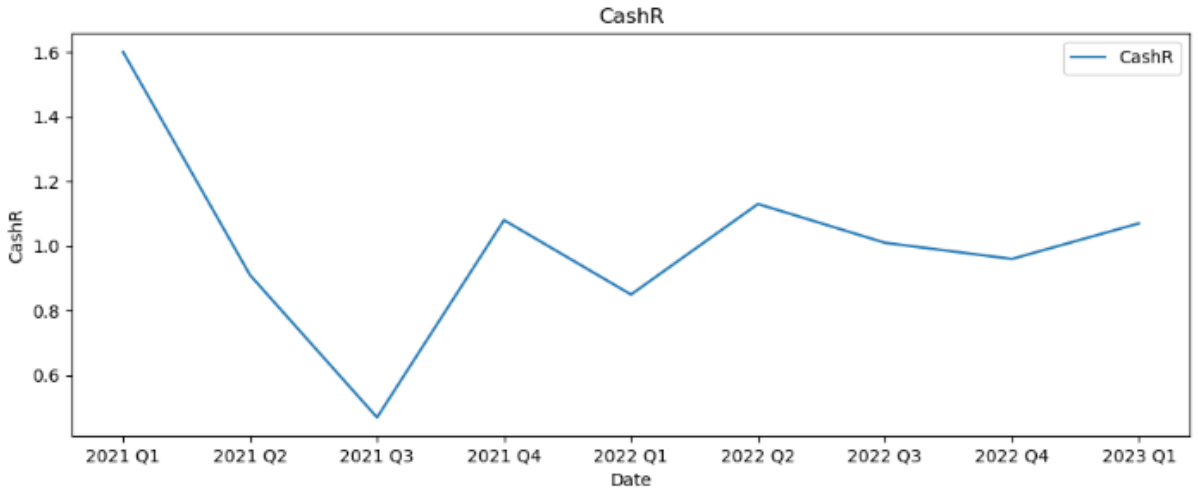


Figure 32.Cash ratio

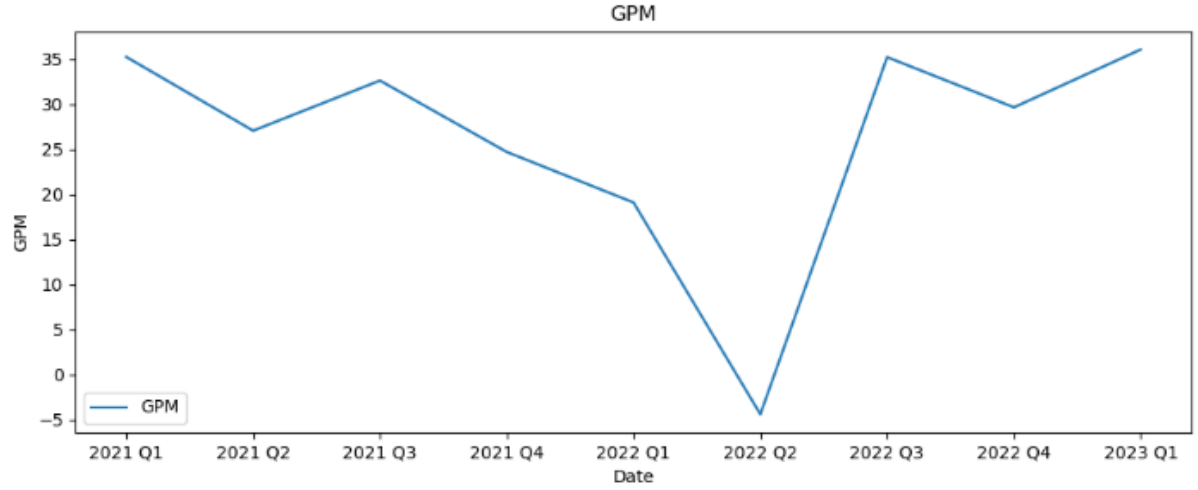


Figure 33. Gross Profit margin

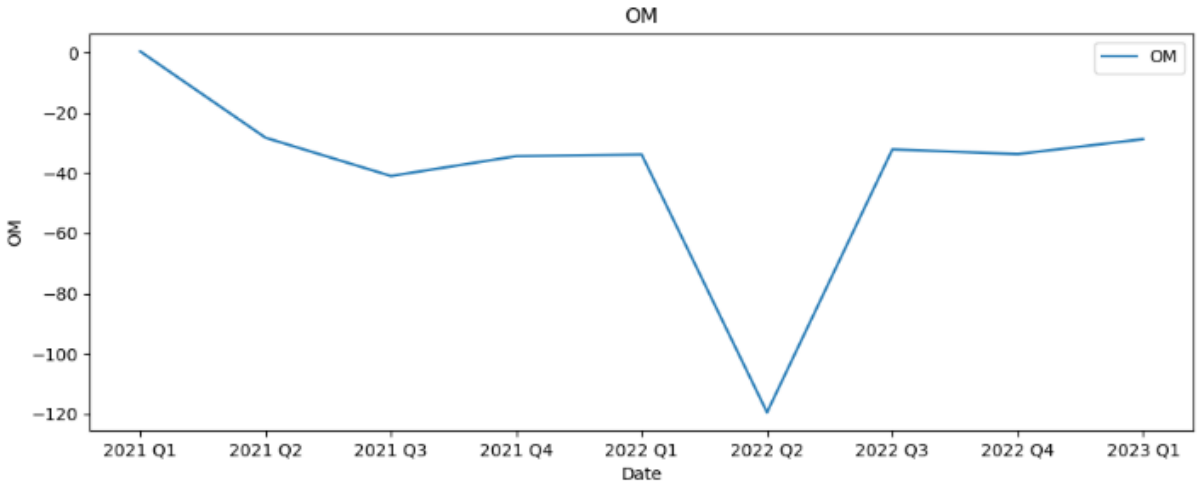


Figure 34. Operating margin

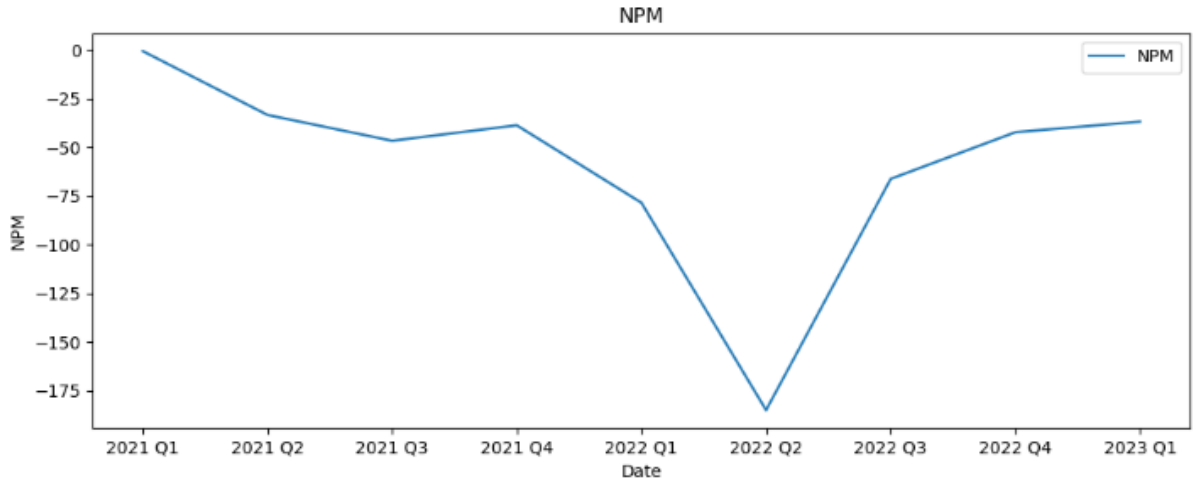


Figure 35. Net Profit margin

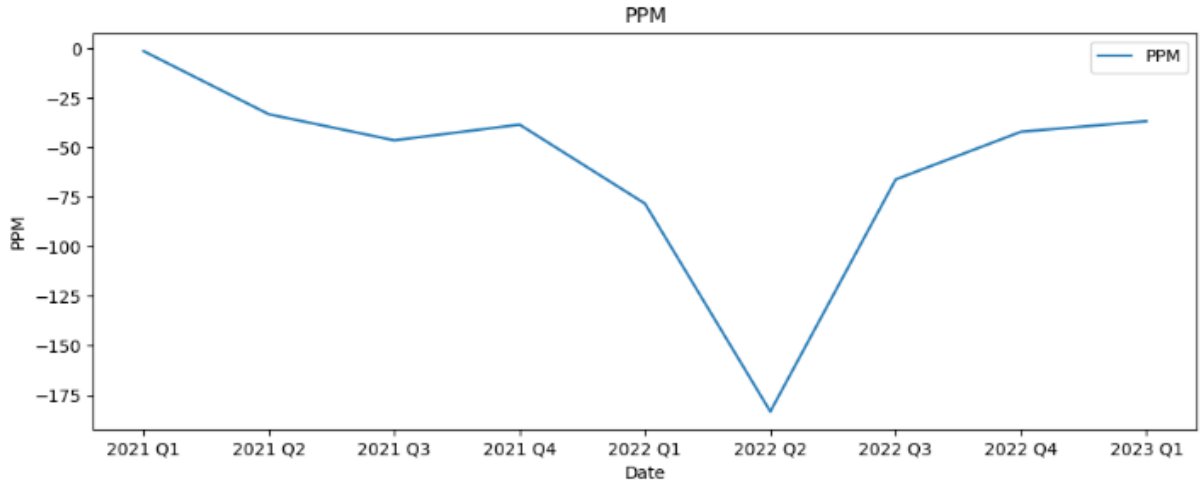


Figure 36. Pretax Profit margin

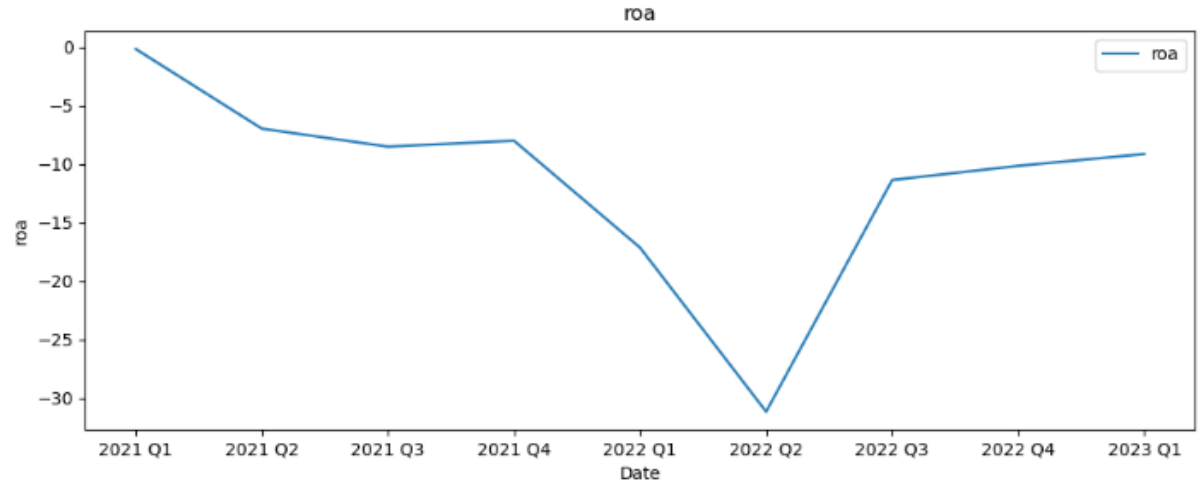


Figure 37. Return on assets

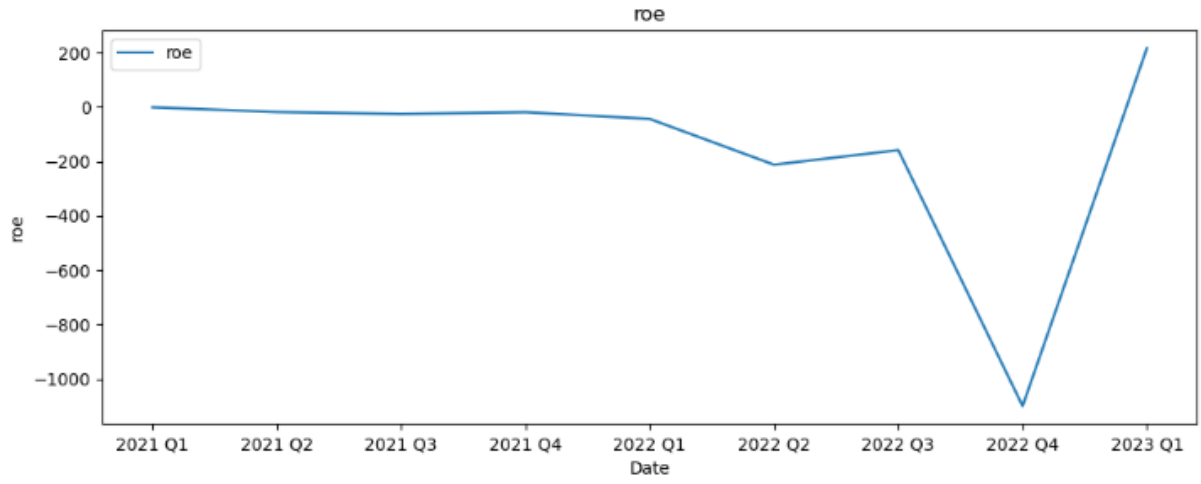


Figure 38.Return on equity

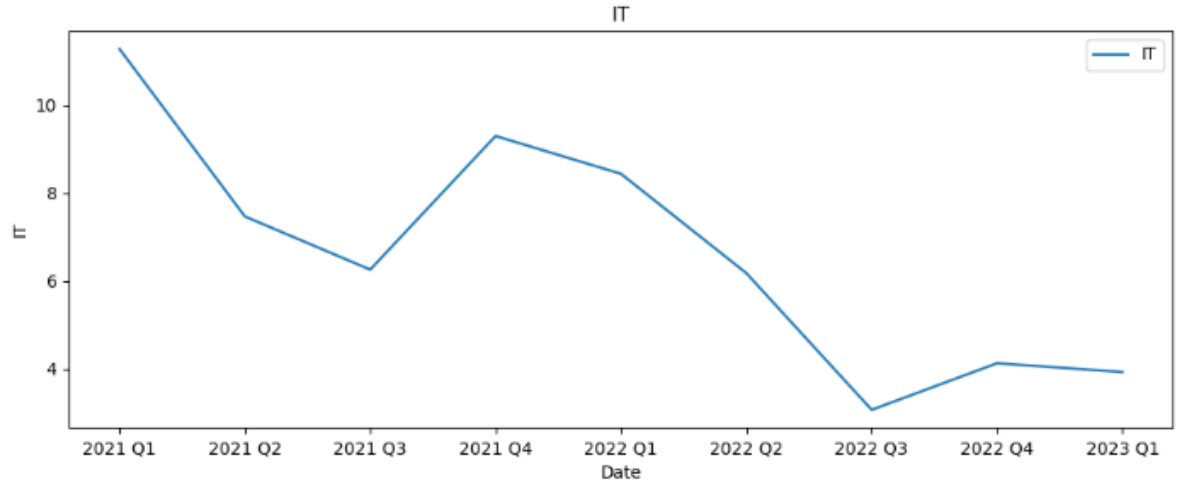


Figure 39.Inventory turnover

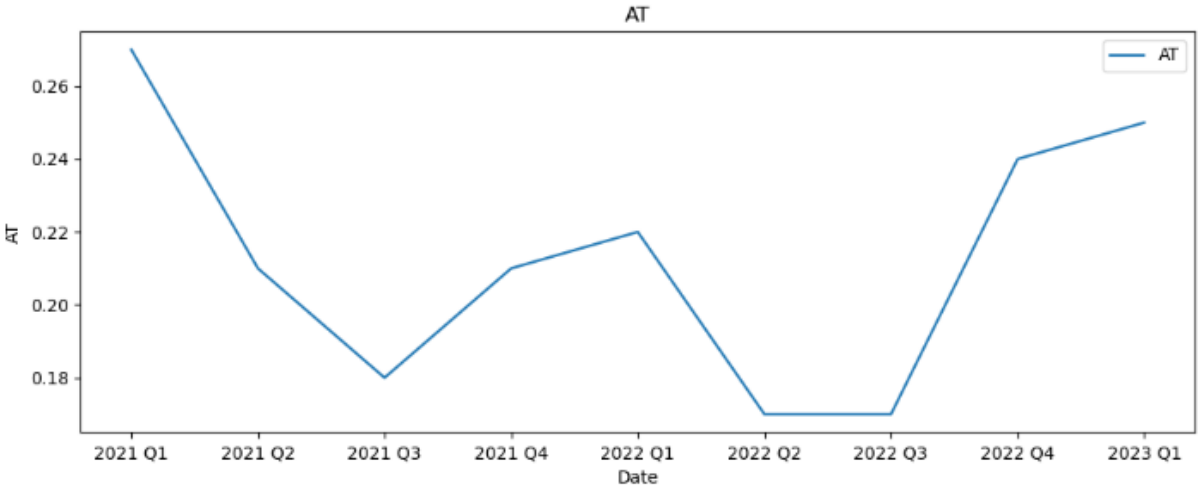


Figure 40.Asset turnover

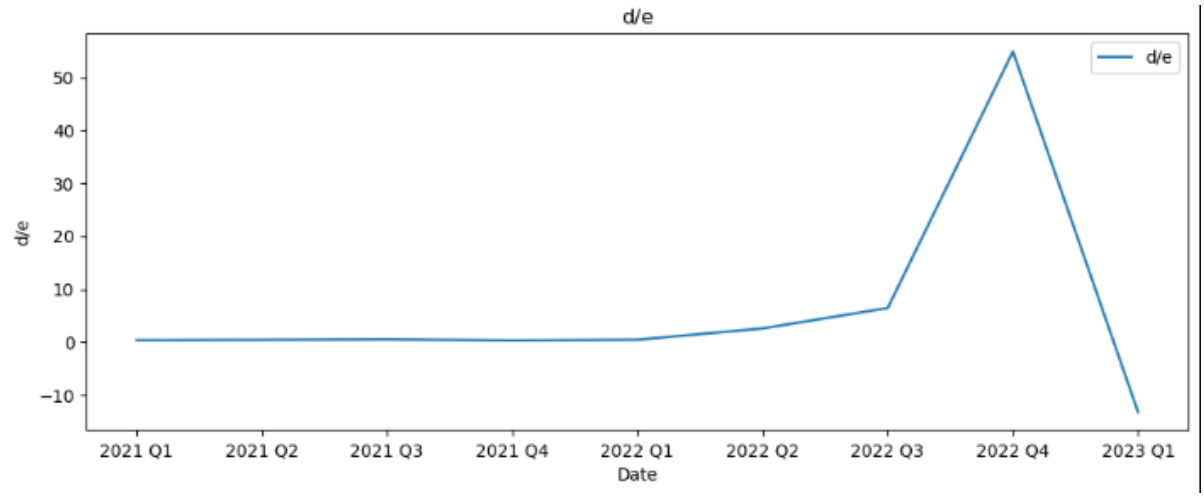


Figure 41. Debt to equity

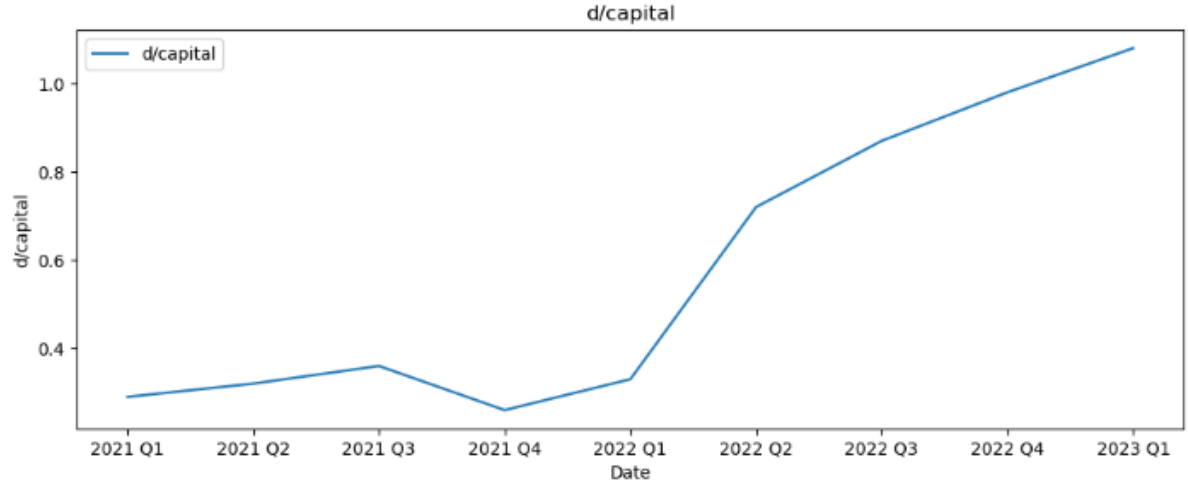


Figure 42.Debt to total capital

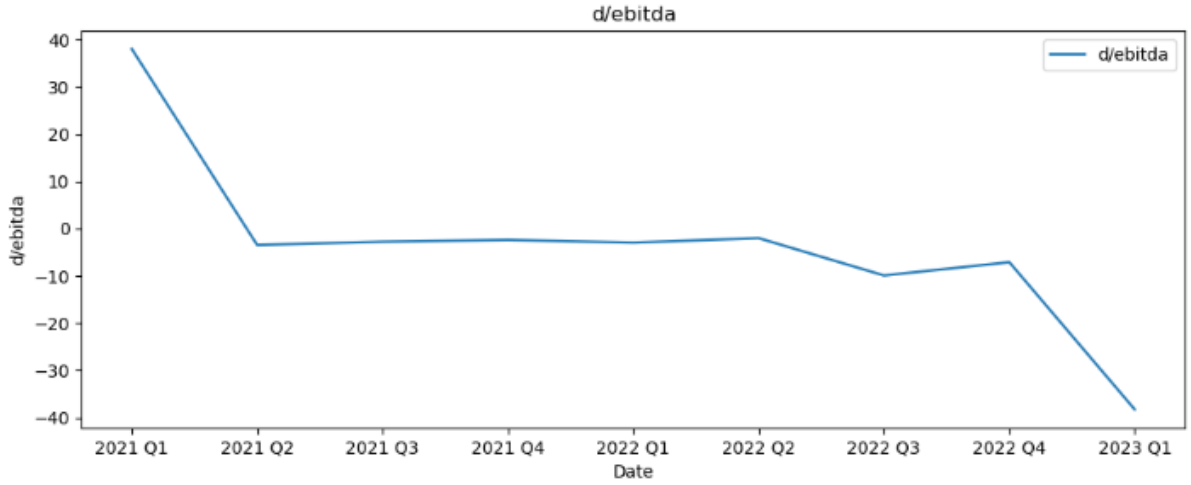


Figure 43. Debt to ebitda

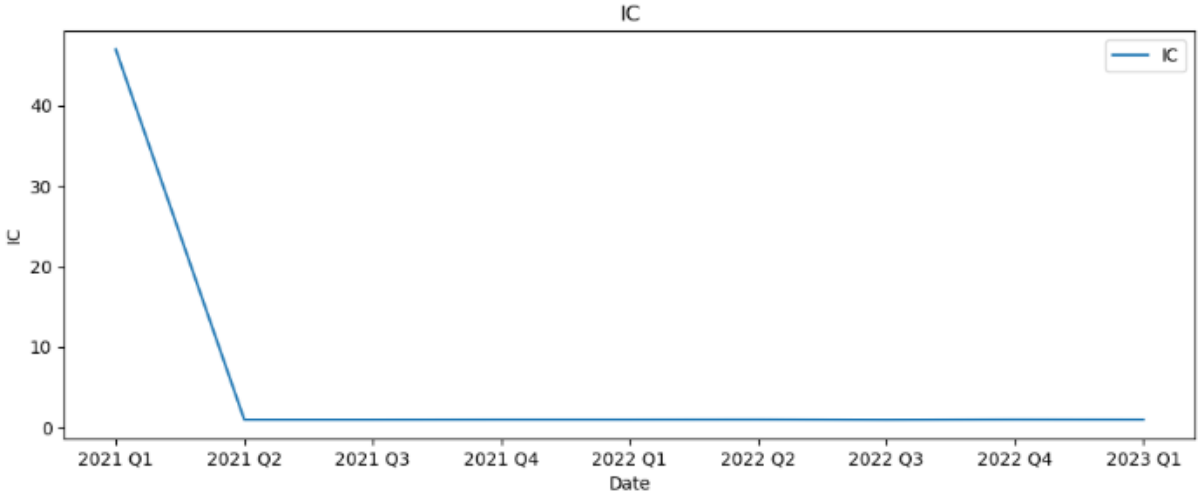


Figure 44. Interest coverage ratio

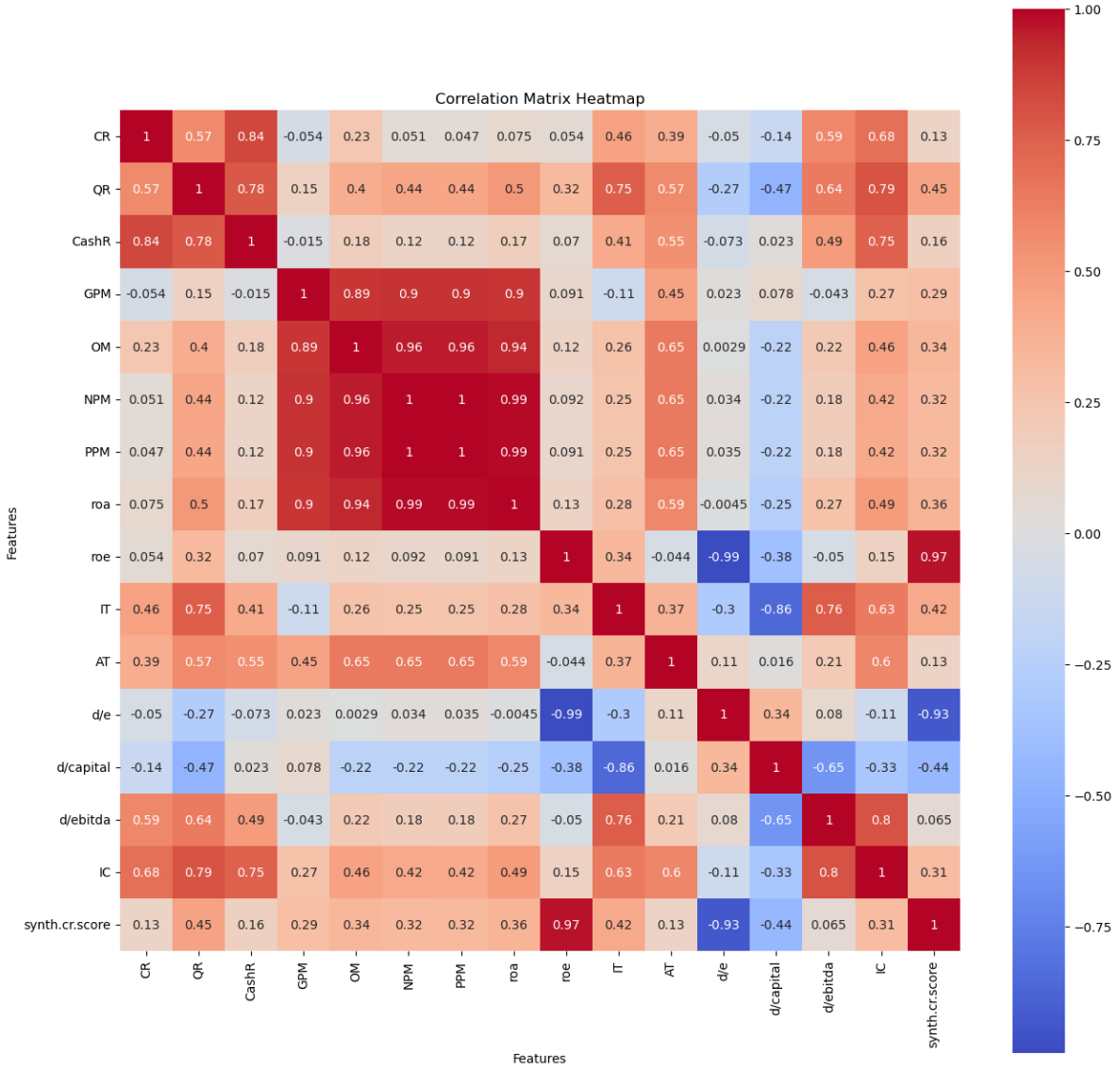


Figure 45. Correlation matrix peloton

The correlation matrix indicates a clear positive correlation between ROE and the synthetic credit score, and a negative correlation between debt-to-equity ratio and the synthetic credit score over the past 7 years. This suggests that Peloton has likely been successful in generating strong investment returns, which could positively impact its creditworthiness. However, it also indicates that Peloton has taken on more leverage than its industry peers, which may increase its financial risk.