Identifying Determinants of Business Success to Aide in Making Profitable Investment Choices

1 Introduction

Investment decisions can be costly, but if made correctly, can yield immense profit. However, making an investment decision comes with great risk and so being able to identify a potentially profitable avenue for investment can be of great importance for investors [BG96]. It has been shown in a previous study that investing in a fortune 500 company is likely to be profitable [AS06], however, investment in one of these companies is not always possible. It stands to reason that companies with the potential to reach a similar level of success can be a possible alternative. It might therefore be beneficial to identify potential markers of a successful business to help in gauging investment potential.

In this study, we have looked at Forbes's fortune 500 dataset, and have utilised two regression models (lasso and ridge regression), in order to identify features that are most closely associated with business success to make decision making easier for investors.

Due to the limits of our dataset, our analysis cannot account for various relevant factors such as company network, initial capital, legal systems, and social and political factors, as well as intangible factors that are difficult to pin point but can be impactful [GMY04, BG96, AS06]. As the mentioned factors can have a large impact on the return on investment, we have chosen to use the USA specific dataset in order to lessen the impact of several of these factors such as politics when comparing internally. This does however, have the drawback of overlooking the impact of these factor externally.

2 Methodology

2.1 Data description

The Forbes fortune 500 dataset ranks companies by total revenues for their respective fiscal years and reports features of the 500 top ranked companies. The companies included are incorporated in the U.S., operate in the U.S., and file financial statements with a government agency [DeC22].

Revenues are the total amount of income generated by companies [Wik22b]. The amount is as reported, including revenues from discontinued operations when published. Revenues figures include consolidated subsidiaries and exclude excise taxes. Percent change calculations are not restated and are based on originally reported data [DeC22].

Profits is income distributed to the owner in a profitable market production process [Wik22a]. They are after taxes, extraordinary credits or charges, cumulative effects of accounting changes, and non-controlling interests (including subsidiary preferred dividends), and before preferred dividends of the company [DeC22].

Assets are the company's year-end total. Total stockholders' equity is the sum of all capital stock that is paid-in capital, and earnings retained by the company's year-end. Employees is a fiscal year-end number as published by the company in its annual report. When a distinction is made, a part-time employee is counted as one-half of a full-time employee [DeC22].

The calculated feature, revenue per employee is used to determine the revenue generated per individual working at a company to gauge efficiency [CFI22].

The feature rank is ordinal, features title, website, sector, industry, HQ zip, HQ location,

HQ address, HQ city, HQ state, HQ telephone, CEO, CEO title, address, ticker and full name are nominal, while employees, revenue, revenue change, profits, assets and total stake-holder equity are continuous. Profit change was initially a string and was changed to a float data type. The data contained no null values.
The feature ”Revenue per employee” was calculated by dividing the revenue by the total number of employees for each company; The categorical feature ”Ranking group” was determined based on the distribution of rank by revenue (Figure 1), and the nominal feature ”Growth direction” was calculated based on the change in profit, with ”positive” showing a growth in profit from previous year, ”negative” showing a decrease in profit, and ”same” signifying no change.

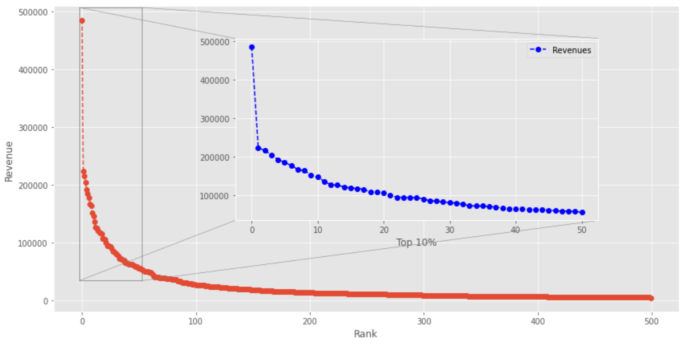


Fig. 1: Distribution of rank by revenue

The features used in our analysis were Rank, Sector, Industry, HQ state, CEO, CEO title, Employee, Revenue, Revenue change, Profits, Profit change, Assets and Total stake-holder equity with the rest being discarded.

2.2 Data exploration and preprocessing

In the data exploration phase, summary statistics and distribution of potentially relevant variables were looked at. To begin the analysis, I generated scatter plots to visualise the relationship between all possibly relevant numerical variables. The scatter plots are shown in Figure 2. From our plots, we can see that a relationship between revenue (the dependent variable) and total shareholder equity, profit and assets exists. There is almost no relationship to revenue per employee and changes in profit and revenue. It can also be seen that the revenue outlier, as well as the revenue per employee outlier have the potential to skew our prediction. As our model assumes independence between our independent variables, features ”Profits” and ”Assets” were removed due to high multicollinearity. Dealing with multicollinearity helps in improving the interpretability of the models. Figure 3 shows the heat map for the correlation between numerical variables with the exception of revenue, which is our dependent variable and is the basis for the calculation of ranking [DeC22], and rank which was also excluded as it would provide no additional insight for our purpose.
Furthermore, the results of Chi-square testing between categorical variables, with a significance level of 0.10, showed a relationship between growth direction and ranking group, sector and industry, growth direction and HQ state, and industry and ranking group. No notable relationship was found between other categorical variables. Based on this, industry, sector and HQ state were included in the regression model data.
Extreme outliers that greatly skewed our data were also removed in order to increase prediction accuracy.
The data was then prepared for regression by changing non-numerical data to a binary format suitable for regression and then the resulting dataset was split to train (70%) and test (30%) data; The numerical data was normalised before Fitting.
Lasso and Ridge regression were used to observe useful features for revenue prediction. Root mean square error (RMSE) and R^2 were used as the measure of model accuracy, while the model coefficients helped in explaining feature importance. The detailed steps can be found in the linked Jupyter notebook at the end of the document.
Lastly, modelling results were compared with graphs and descriptive statistics to derive an answer to the research question.

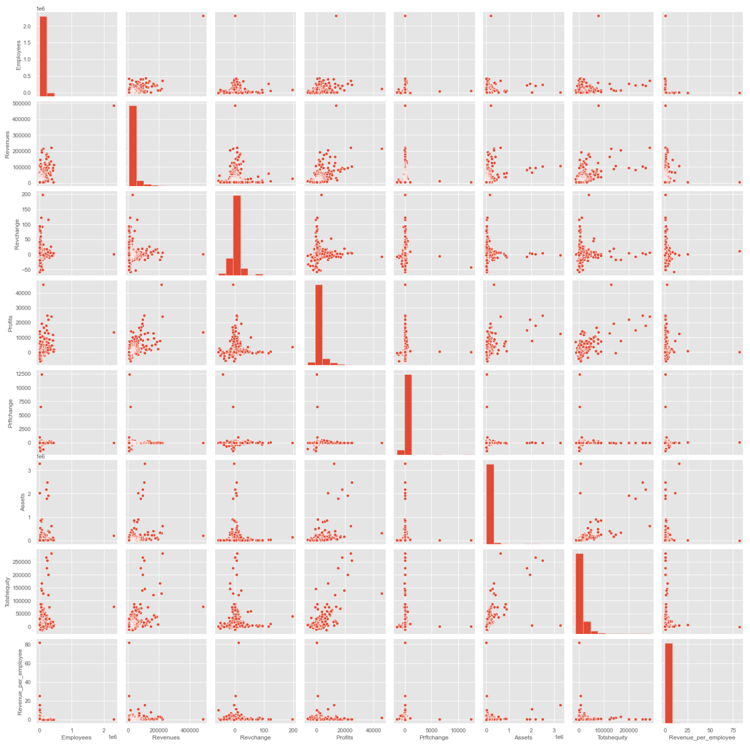


Fig. 2: Quantitative feature distributions

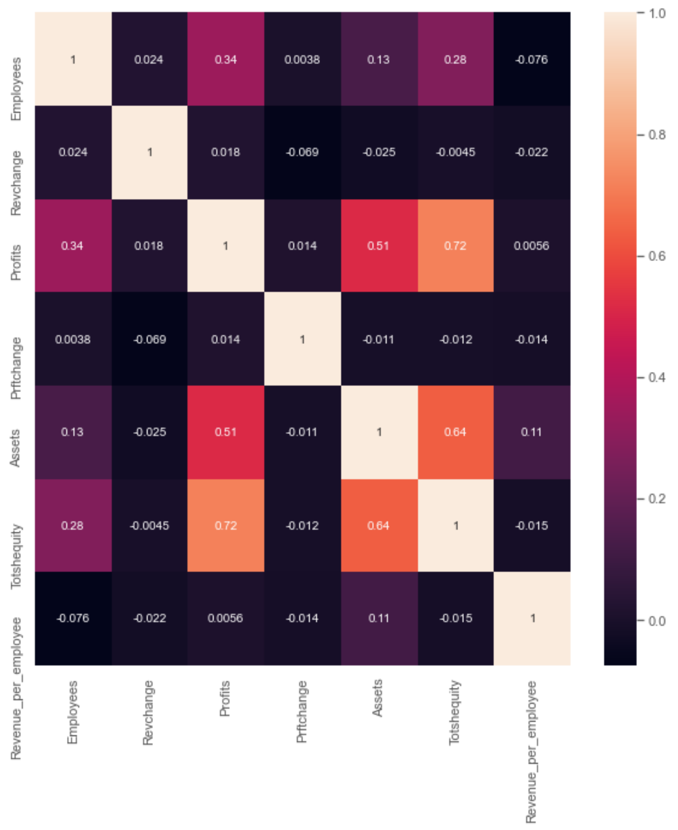


Fig. 3: Heat-map of correlations between quantitative variables

3 Results

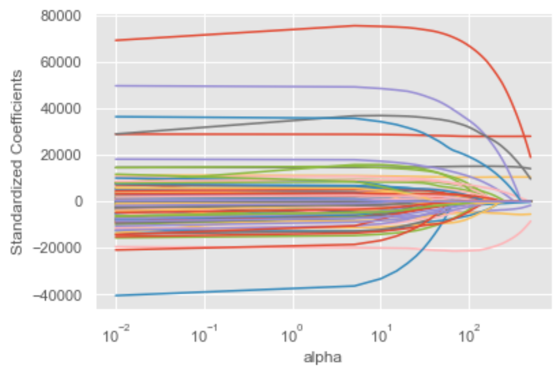
3.1 Modelling

My objective for the use of regression models was to see which of the chosen variables can be used in order to better predict company revenue.
First I have fitted a ridge regression model to the data with the dependant variable being revenues and the independent variables being employees, revenue change, profit change, total shareholder equity, revenue per employee as well as HQ state, industry and sector. While the train data reached a high R2 value, the test data had a low R2 value. In order to achieve more accurate results, In the second phase of my analysis, I have fitted a lasso regression model to the same data, which can set the impact of unimportant variables to zero. The models were then compared for best prediction results. The optimal alpha value was selected using five-fold cross validation in order to tackle the issue of overfitting. Prediction accuracy was checked both with and without extreme outliers. The change in variable relevance based on alpha can be seen in Figure 4 and Figure 5.

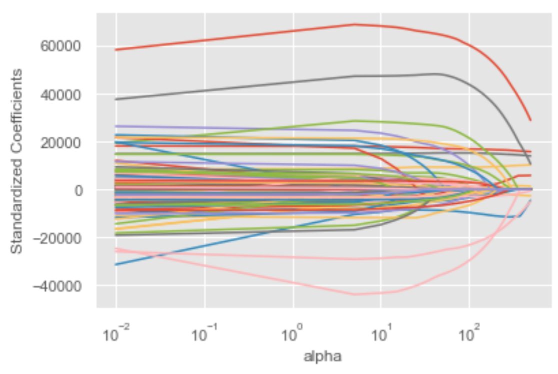
With outliers:
For the ridge model, the R^2 for the train set was 86.44% and the R^2 for the test set was 35.74% for alpha = 1. After cross validation, alpha became 10 and R^2 became 82.45% and 39.61%. For the lasso model the R^2 before the most efficient alpha was chosen (alpha = 1) was 86.93% for training data and 29.63% for testing data. This figure changed to 74.16% for training set and 42.86% for test set after adjusting alpha to the optimal value.

Without outliers:
After the removal of the two outliers, the R^2 for the train and test sets of the ridge model became 78.82% and 66.2% respectively for alpha = 1 (alpha remained 1 after cross validation), while the lasso model had the R^2 values of 79.41% and 63.78% for training and test sets at alpha = 1, and 77.23% and 64.47% at alpha = 96 (optimal value).

As can be seen from Table 1, the Root mean

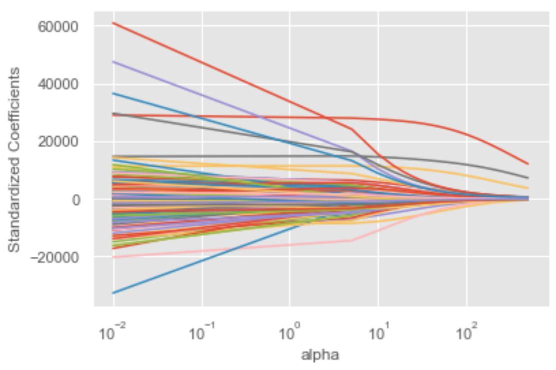


(a) With outliers

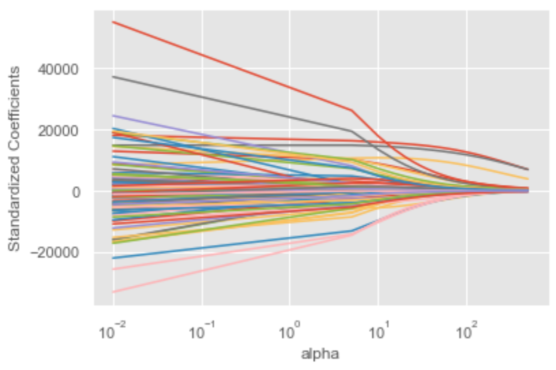


(b) Without outliers

Fig. 4: Lasso coefficients as a function of alpha



(a) With outliers



(b) Without outliers

Fig. 5: Ridge coefficients as a function of alpha

square error (RMSE) reflects the same pattern as the R^2 values.

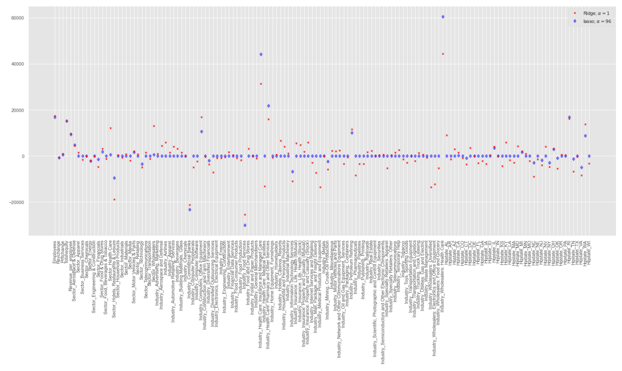
Both models see a drastic improvement in accuracy after the two outliers were removed, with the lasso model slightly outperforming the ridge model with outliers, and the Ridge model slightly outperforming the Lasso model without outliers. Out of all models, the Ridge regression model with an optimised alpha and without extreme outliers preformed best. Feature relevance of our models can be seen in Figure 6.

3.2 Model output

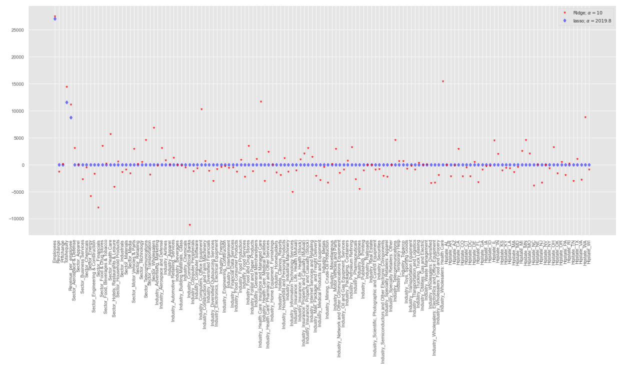
Lasso regression completely eliminates any unnecessary variables while ridge regression keeps all variables but the general extent of variable relevance for quantitative variables was shown to be close between the two models in all iterations. In the lasso model before the removal of outliers, qualitative variables were all set to zero and had no relevance to our output. After outlier removal, a number of qualitative variables gained relevance while the extent of the relevance of quantitative variables greatly lessened.
For the ridge model, the impact of almost all variables was smaller without outliers and the rest did not have significant change.
After outlier removal, the importance of qualitative variables was generally less pronounced in the ridge model compared to the lasso model as the coefficients were much smaller but the type of effect was the same; This is likely because the lasso model increases the importance of certain parameters to compensate for lost features.
From Figure 6 we can see that number of employees, total share-holder equity and revenue per employee can be used to predict our outcome. Among sectors, aerospace and

TABLE 1: Regression evaluation metrics

Lasso regression									
With outlier					Without outlier				
Alpha	R^2 train	R^2 test	RMSE train	RMSE test	Alpha	R^2 train	R^2 test	RMSE train	RMSE test
1	86.93%	29.63%	14722.22	26845.77	1	79.41%	63.78%	14963.62	18483.53
2019.8	74.16%	42.86%	20698.22	24189.51	96	77.23%	64.47%	15738.82	18306.84
Ridge regression									
With outlier					Without outlier				
Alpha	R^2 train	R^2 test	RMSE train	RMSE test	Alpha	R^2 train	R^2 test	RMSE train	RMSE test
1	86.44%	35.74%	14995.76	25653.65	1	78.82%	66.2%	15179.8	17855.47
10	82.45%	39.61%	17059.72	24868.25	1	78.82%	66.2%	15179.8	17855.47



(a) With outliers



(b) Without outliers

Fig. 6: Lasso and Ridge coefficients comparison

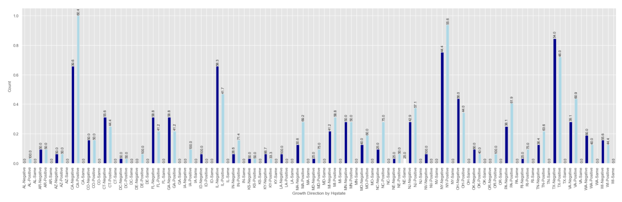
defence, food and drug stores, healthcare, wholesalers and motor-vehicles and parts were shown to have positive a relationship to our outcome, while energy, financials, hotels, restaurants and leisure as well as technology had a negative effect. For industry, computers and office equipment, health insurance and managed care and pharmacy as well as petroleum refining and healthcare wholesalers were shown to have a positive impact, while mining and crude oil production, IT service, commercial banks, food services and diversified financials has a negative impact. As for headquarter state, California, Illinois, Michigan, Ohio, Rhode Island and Washington were also shown to have a positive effect, while Connecticut, North Carolina, New Jersey, New York, Oklahoma, Tennessee and Virginia had a negative effect.

The model may be further improved by using clustering to identify noise, and then removing noise from the data to improve clarity and better find strong patterns. Additional optimisation techniques for coefficients can also be used.

4 Discussion

As per the results of our regression models, considering the cross-section of industry with sector, we can conclude that drug and health care related wholesales are quite profitable while Aerospace and defence, and motor vehicles and parts are profitable in general without a subsection standing out. Petroleum refining and office computers were also quite profitable while crude oils production and technology were less so. When looking at Figure 7b, which shows the direction of profit growth by sector, we can see that health care had noticeably more profit increase than decrease, therefore this could be a reasonably safe and profitable avenue for investment.

While both aerospace and defence, and motor vehicles and parts saw a positive profit increase, they made up a small portion of the data and the data size was not sufficient to reach a conclusion regarding potential growth and risk of investment. Technology


(a) Based on headquarter state

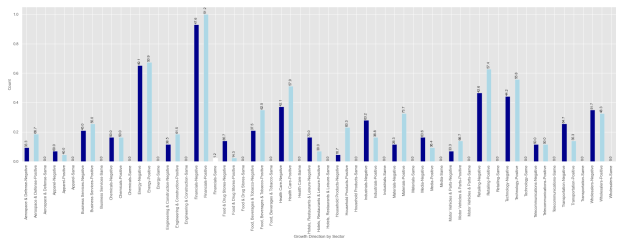

(b) Based on sector

Fig. 7: Direction of profit growth

is a growing industry but is not as big as healthcare. Office computers are a reasonably safe avenue for investment in technology.

Four CEOs (Gregory J. Goff, Miles D. White, Gregory B. Maffei and John J. Christmann IV) appeared on the list twice while every other CEO only appeared once. This indicates that a second company run by the CEO of a fortune 500 company is unlikely to reach the same level of success, although it does not show the likelihood of success relative to a non-fortune 500 company.
Regarding the CEO title, the vast majority of fortune 500 CEOs fell into one of four categories: 1- President, Chief Executive Officer and Director (189), 2- Chairman and Chief Executive Officer (127), 3- Chairman, President and Chief Executive Officer (100), 4- Chief Executive Officer and Director (75). Choosing a company where the CEO takes on similar responsibilities as the ones listed above may be a more reliable decision.
Out of the headquarter states that have been preforming well, California and Michigan have had growth in profit and have a big enough sample size in our dataset. Ohio, Washington and Illinois were performing well, but based on Figure 7a, they had seen a decrease in profit. New York, Virginia, North Carolina and New Jersey have not been preforming as well but they have been seeing an increase in profit. The headquarter states that had produced the best results and had also seen growth were California and Michigan, therefore, these two states could be a safer option to look into. Other states either had less favourable results or had a relatively low density of fortune 500 companies and so it is difficult to derive a meaningful conclusion for them.

The limitations of the modelling approach were that the lasso regression model preforms erratically for correlated variables which means that we had to remove variables that could have provided additional insight. Additionally, as the model coefficients are our main point of interest, the bias in coefficients is another issue that both models have. Using advanced coefficient optimisation techniques might be beneficial. Moreover, based on Figure 2 it is possible that using a non-linear model may improve results. An alternative approach could be to use a feature selection model rather than a regression model as the main intent of the modelling is finding feature relevance.
As there are many factors that influence a company's success, adding more possibly relevant features to the dataset such as marketing methods, characteristics of CEO and characteristics of targeted market, can help in better gauging potential for business success and investment risk [BG96]. A bigger dataset such as fortune 1000 can also help in clearing uncertainty caused by dataset size limits. Additionally, a dataset with global data can help in discovering features that cannot be captured with a nation specific dataset such as the effect of international relations, national policies, country reputation and legal system[GMY04]. Noise reduction of data can also help in reaching better result clarity.

References

[AS06] Jeff Anderson and Gary Smith. A great company can be a great investment. *Financial Analysts Journal*, 62(4):86–93, 2006.

[BG96] J Bachher and Paul Guild. ”financing early stage technology based companies: investment criteria used by investors.” frontiers of entrepreneurship research. *Frontiers of Entrepreneurship Research*, 01 1996.

[CFI22] CFIteam. Revenue per employee, 2022. accessed on 30 November 2022.

[DeC22] Scott DeCarlo. Methodology for fortune 500, 2022. accessed on 30 November 2022.

[GMY04] Klaus Gugler, Dennis C Mueller, and B Burcin Yurtoglu. Corporate governance and the returns on investment. *The Journal of Law and Economics*, 47(2):589–633, 2004.

[Wik22a] Wikipedia. Profit (accounting), 2022. accessed on 30 November 2022.

[Wik22b] Wikipedia. Revenue, 2022. accessed on 30 November 2022.

Appendix

:

Python_Code

December 1, 2022

[ ]:
```python
# imports

import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from matplotlib import style
import seaborn as sns
from scipy import stats
import networkx as nx
from mpl_toolkits.axes_grid1.inset_locator import mark_inset
from sklearn.cluster import OPTICS, cluster_optics_dbscan
import matplotlib.gridspec as gridspec
from sklearn.cluster import DBSCAN
from mpl_toolkits.mplot3d import Axes3D
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge, RidgeCV, Lasso
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LassoCV
import math
```

[ ]:
```python
# reading and looking at the dataset

df = pd.read_csv('Fortune_500_2017.csv')
df.info()
df.describe()
```

[ ]:
```python
# changing profit change from string to float

df['Prftchange'] = df['Prftchange'].apply(lambda x: float(x.replace(',','')))
```

[ ]:
```python
# chacking to see how many ceos only appeared once
df['Ceo'].nunique()

# seeing who appeared more than once
a = df[df['Ceo'].duplicated(keep=False)]
```

```
a[['Rank','Ceo', 'Title']]
```

[ ]:
```
# chacking to see if there are any null values

df.isnull().any().any()
```

[ ]:
```
# graph of revenue by rank

plt.figure(figsize=(14,7))
ax1 = plt.axes() # standard axes
ax2 = plt.axes([0.35, 0.35, 0.45, 0.45])
ax1.set_xlabel("Rank")
ax1.set_ylabel("Revenue")
ax2.set_xlabel("Top 10%")
ax1.plot(df['Revenues'], '--o')
ax2.plot(df['Revenues'].iloc[0:51], '--bo', label='Revenues')
mark_inset(ax1, ax2, loc1=1, loc2=4, fc="none", ec="0.5")
mark_inset(ax1, ax2, loc1=2, loc2=3, fc="none", ec="0.5")
ax2.legend()
plt.savefig('Revenue.png',bbox_inches='tight')
```

[ ]:
```
# feature creation:

# Growth direction: if profit has increased or decreased

def growth_direction (row):
   num = row['Prftchange']
   if num > 0:
      return 'Positive'
   elif num == 0:
      return 'Same'
   else:
       return 'Negative'

# Ranking group: ranking ranges based on distribution of revenue
def ranking_group (row):
   rank = int(row['Rank'])
   if rank <= 15:
      return '0-3'
   elif rank <= 30:
      return '3-6'
   elif rank <= 50:
      return '6-10'
   elif rank <= 100:
      return '10-20'
   elif rank <= 200:
      return '20-40'
```

```
    elif rank <= 300:
        return '40-60'
    elif rank <= 400:
        return '60-80'
    else:
        return '80-100'

# Revenue per employee
df['Revenue_per_employee'] = df.apply(lambda row: row.Revenues / row.Employees,␣
 ↪axis=1)

df['Ranking_group'] = df.apply(lambda row: ranking_group(row), axis=1)
df['Growth_direction'] = df.apply(lambda row: growth_direction(row), axis=1)
```

[ ]:
```
# show dataset

df.head()
```

[ ]:
```
# graph of distribution of assets

df['Assets'].plot(kind="box")
```

[ ]:
```
# creating normalised dataset for graphs

normal_df = df.copy()
x, y = df.Revenues.mean(), df.Revenues.std()
normal_df['Revenues'] = (df.Revenues - x) / y
x, y = df.Assets.mean(), df.Assets.std()
normal_df['Assets'] = (df.Assets - x) / y
x, y = df.Totshequity.mean(), df.Totshequity.std()
normal_df['Totshequity'] = (df.Totshequity - x) / y
x, y = df.Profits.mean(), df.Profits.std()
normal_df['Profits'] = (df.Profits - x) / y
x, y = df.Prftchange.mean(), df.Prftchange.std()
normal_df['Prftchange'] = (df.Prftchange - x) / y
x, y = df.Revchange.mean(), df.Revchange.std()
normal_df['Revchange'] = (df.Revchange - x) / y
x, y = df.Revenue_per_employee.mean(), df.Revenue_per_employee.std()
normal_df['Revenue_per_employee'] = (df.Revenue_per_employee - x) / y
x, y = df.Employees.mean(), df.Employees.std()
normal_df['Employees'] = (df.Employees - x) / y
normal_df.head()
```

[ ]:
```
# how many people are in each sector

ind = df[['Sector','Rank']].groupby('Sector').count()
ind['Rank'].sort_values().head(10)
```

```
[ ]: # how many ceos have each title

     ind = df[['Ceo-title','Rank']].groupby('Ceo-title').count()
     ind['Rank'].sort_values().head(20)
```

```
[ ]: # graph of distribution by hq state

     state = df['Hqstate'].value_counts(ascending=False)
     plt.style.use('ggplot')
     plt.figure(figsize=(15,5))
     plt.ylabel("Count")
     plt.xlabel("Hqstate")
     state.plot(kind='bar')
     plt.savefig('state.png',bbox_inches='tight')
```

```
[ ]: # graph of distribution by industry

     prob = df['Industry'].value_counts(ascending=False)
     plt.style.use('ggplot')
     plt.figure(figsize=(15,5))
     plt.ylabel("Count")
     prob.plot(kind='bar')
     plt.savefig('foo.png',bbox_inches='tight')
```

```
[ ]: # graph of profit (not revenue, to take into account expenses) for each industry

     prof = df.groupby('Industry')['Profits'].mean()
     plt.style.use('ggplot')
     plt.figure(figsize=(20,7))
     plt.ylabel("Profits")

     prof.plot(kind='bar', color="darkgreen")
     plt.savefig('Profits_by_Industry.png',bbox_inches='tight')
```

```
[ ]: # graph of distribution of revenue per employee based on rank

     plt.figure(figsize=(25,12))
     ax1 = plt.axes()
     ax1.set_xlabel("Rank")
     ax1.set_ylabel("Revenue per Employee")
     ax1.plot(df['Revenue_per_employee'], color="darkblue")
     plt.savefig('Revenue_per_Employee.png',bbox_inches='tight')
```

```
[ ]: # graph of revenue by industry

     rev = df.groupby('Industry')['Revenue_per_employee'].mean()
     plt.style.use('ggplot')
     plt.figure(figsize=(15,5))
     plt.ylabel("Revenue")
     rev.plot(kind='bar', color="violet")
     plt.savefig('Revenue_per_Employee_for_Industry.png',bbox_inches='tight')
```

[ ]:
```
# graph of revenue by industry without outlier

df['Industry'].loc[df['Industry'] == 'Miscellaneous']
df.iloc[394]
a = df.copy()
a = a.drop(394)
rev = a.groupby('Industry')['Revenue_per_employee'].mean()
plt.style.use('ggplot')
plt.figure(figsize=(15,5))
plt.ylabel("Revenue")
rev.plot(kind='bar', color="purple")
plt.savefig('Revenue_per_Employee_per_industry_without_misc.
 ↪png',bbox_inches='tight')
```

[ ]:
```
# graph of revenue by hq state

rev = df.groupby('Hqstate')['Revenues'].mean()
plt.style.use('ggplot')
plt.figure(figsize=(15,5))
plt.ylabel("Revenue")
rev.plot(kind='bar', color="orange")
plt.savefig('rev_per_state.png',bbox_inches='tight')
```

[ ]:
```
# graph of revenue by hq state without outliers

b = df.iloc[5:]
rev = b.groupby('Hqstate')['Revenues'].mean()
plt.style.use('ggplot')
plt.figure(figsize=(15,5))
plt.ylabel("Revenue")
rev.plot(kind='bar', color="darkorange")
plt.savefig('rev_per_state_without_top_one_precent.png',bbox_inches='tight')
```

[ ]:
```
# distribution graph of numerical data without rank and hqzip

a = df.drop(['Rank','Hqzip'], axis =1)
sns.pairplot(a)
plt.savefig('pairplot.png',bbox_inches='tight')
```

[ ]:
```
# distribution graph of numerical data without rank, hqzip and profit change
# and without ouliers
```

```
a = df.drop(['Rank','Hqzip','Prftchange',], axis =1)
a = a.drop(a['Revenues'].idxmax())
a = a.drop(a['Revenue_per_employee'].idxmax())
sns.pairplot(a)
plt.savefig('pairplot2.png',bbox_inches='tight')
```

[ ]:
```
# heatmap of correlations
plt.figure(figsize = (10, 10))
sns.heatmap(df.drop(['Hqzip','Revenues','Rank'], axis =1).corr(), annot = True)
plt.savefig('correlation.png',bbox_inches='tight')
```

[ ]:
```
# graph of distribution of growth direction by sector

direct = df.groupby('Sector')['Growth_direction'].value_counts(ascending=False).
 ↪unstack(fill_value=0).stack()
direct = direct.reset_index()
direct = direct.rename(columns= {0: 'Count'})
direct.index.name = 'Index'
direct2 = direct.copy()
direct2['Growth_direction'] = direct2[['Sector', 'Growth_direction']].agg('-'.
 ↪join, axis=1)
x, y = direct2.Count.min(), direct2.Count.max()
direct2['Count'] = (direct2.Count - x) / y - x
direct2['Precentage'] = 0

i = 0
while (i < len(direct2)):
    sum = direct2['Count'].iloc[i] + direct2['Count'].iloc[i+1] +␣
 ↪direct2['Count'].iloc[i+2]
    direct2['Precentage'].iloc[i] = (direct2['Count'].iloc[i] / sum) * 100
    direct2['Precentage'].iloc[i+1] = (direct2['Count'].iloc[i+1] / sum) * 100
    direct2['Precentage'].iloc[i+2] = (direct2['Count'].iloc[i+2] / sum) * 100
    i=i+3

direct2['Precentage'] = round(direct2['Precentage'],1)

colors_list = ['darkblue','lightblue', 'white']
plt.style.use('ggplot')
plt.figure(figsize=(30,8))
ax = direct2.plot(x="Growth_direction", y="Count", kind="bar", figsize=(30, 8),␣
 ↪color = colors_list, legend=False)
ax.set_xlabel("Growth Direction by Sector")
ax.set_ylabel("Count")
i=0
for bar in ax.patches:
    plt.text(bar.get_x() - 0.2 + bar.get_width() / 2,
             bar.get_height()+0.01, direct2['Precentage'].iloc[i],rotation=90)
    i=i+1

plt.savefig('Growth_direction_by_Sector.png',bbox_inches='tight')
```

[ ]:
```
# graph of distribution of growth direction by hq state

direct = df.groupby('Hqstate')['Growth_direction'].
↪value_counts(ascending=False).unstack(fill_value=0).stack()
direct = direct.reset_index()
direct = direct.rename(columns= {0: 'Count'})
direct.index.name = 'Index'
direct2 = direct.copy()
direct2['Growth_direction'] = direct2[['Hqstate', 'Growth_direction']].agg('-'.
↪join, axis=1)
x, y = direct2.Count.min(), direct2.Count.max()
direct2['Count'] = (direct2.Count - x) / y - x
direct2['Precentage'] = 0
i = 0
while (i < len(direct2)):
    sum = direct2['Count'].iloc[i] + direct2['Count'].iloc[i+1] +␣
↪direct2['Count'].iloc[i+2]
    direct2['Precentage'].iloc[i] = (direct2['Count'].iloc[i] / sum) * 100
    direct2['Precentage'].iloc[i+1] = (direct2['Count'].iloc[i+1] / sum) * 100
    direct2['Precentage'].iloc[i+2] = (direct2['Count'].iloc[i+2] / sum) * 100
    i=i+3

direct2['Precentage'] = round(direct2['Precentage'],1)

colors_list = ['darkblue','lightblue', 'white']
plt.style.use('ggplot')
plt.figure(figsize=(30,8))
ax = direct2.plot(x="Growth_direction", y="Count", kind="bar", figsize=(30, 8),␣
↪color = colors_list, legend=False)
ax.set_xlabel("Growth Direction by Hqstate")
ax.set_ylabel("Count")

i=0
for bar in ax.patches:
    plt.text(bar.get_x() - 0.2 + bar.get_width() / 2,
             bar.get_height()+0.01, direct2['Precentage'].iloc[i],rotation=90)
    i=i+1

plt.savefig('Growth_direction_by_Hqstate.png',bbox_inches='tight')
```

[ ]:
```
#  preparing data for clustering

X = normal_df.copy()
```

```
X = X[['Employees', 'Prftchange', 'Revchange']]

# outlier removal
X=X.drop(X['Employees'].idxmax())
X=X.drop(X['Revchange'].idxmax())
X=X.drop(X['Prftchange'].idxmax())
```

[ ]:
```
# OPTICS clustering (3 features)

clustering = OPTICS(min_samples=5).fit(X)

# plot of principle components
plt.style.use('ggplot')
pca = PCA(n_components=2)
pca.fit(X)
X['PC1'] = pca.fit_transform(X)[:,0]
X['PC2'] = pca.fit_transform(X)[:,1]
X['OPTICS'] = clustering.labels_
print(clustering.labels_)
sns.scatterplot(data=X,x="PC1",y="PC2",hue=X['OPTICS'])
plt.savefig('components_optics2.png',bbox_inches='tight')
```

[ ]:
```
# optics 3d plot

fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection='3d')
ax.set_xlabel('Employees')
ax.set_ylabel('Prftchange')
ax.set_zlabel('Revchange')

for s in X.OPTICS.unique():
    ax.scatter(X.Employees[X.OPTICS==s], X.Prftchange[X.OPTICS==s],
               X.Revchange[X.OPTICS==s],label=s)

ax.legend(loc='upper left')
print(X.OPTICS.unique())
plt.savefig('optics2.png',bbox_inches='tight')
```

[ ]:
```
# DBSCAN clustering with 3d plot

clusters = DBSCAN(eps=0.5, min_samples=5).fit(X[['Employees', 'Prftchange',␣
 ↪'Revchange']])
X['DBSCAN'] = clusters.labels_

sns.set(style="whitegrid")
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection='3d')
```

```
ax.set_xlabel('Employees')
ax.set_ylabel('Prftchange')
ax.set_zlabel('Revchange')

for s in X.DBSCAN.unique():
    ax.scatter(X.Employees[X.DBSCAN==s], X.Prftchange[X.DBSCAN==s],
               X.Revchange[X.DBSCAN==s],label=s)
ax.legend(loc='upper left')
print(X.DBSCAN.unique())
plt.savefig('dbscan2.png',bbox_inches='tight')
```

[ ]:
```
# preparing data for clustering

X = normal_df.copy()
X = X[['Prftchange', 'Revchange']]

# outlier removal
X=X.drop(X['Revchange'].idxmax())
X=X.drop(X['Prftchange'].idxmax())
X=X.drop(X['Prftchange'].idxmax())
```

[ ]:
```
# OPTICS clustering (2 features) with plot
clustering = OPTICS(min_samples=5).fit(X)
X['OPTICS'] = clustering.labels_

plt.figure(figsize=(14,7))
sns.scatterplot(data=X,x="Prftchange",y="Revchange",hue=X['OPTICS'])
plt.savefig('optics_revchange_profitchange.png',bbox_inches='tight')
```

[ ]:
```
# DBSCAN clustering (2 features) with plot

clusters = DBSCAN(eps=0.5, min_samples=3).fit(X[['Prftchange', 'Revchange']])
X['DBSCAN'] = clusters.labels_

plt.figure(figsize=(14,7))
sns.scatterplot(data=X,x="Prftchange",y="Revchange",hue=X['DBSCAN'])
plt.savefig('dbscan_revchange_profitchange.png',bbox_inches='tight')
```

[ ]:
```
# chi square test

contigency= pd.crosstab(df['Growth_direction'], df['Sector'])
c, p, dof, expected = stats.chi2_contingency(contigency)
print(round(p,3))
contigency= pd.crosstab(df['Growth_direction'], df['Industry'])
c, p, dof, expected = stats.chi2_contingency(contigency)
print(round(p,3))
contigency= pd.crosstab(df['Growth_direction'], df['Hqstate'])
```

```
c, p, dof, expected = stats.chi2_contingency(contigency)
print(round(p,3))
contigency= pd.crosstab(df['Growth_direction'], df['Ceo'])
c, p, dof, expected = stats.chi2_contingency(contigency)
print(round(p,3))
contigency= pd.crosstab(df['Growth_direction'], df['Ranking_group'])
c, p, dof, expected = stats.chi2_contingency(contigency)
print(round(p,3))
contigency= pd.crosstab(df['Hqstate'], df['Ranking_group'])
c, p, dof, expected = stats.chi2_contingency(contigency)
print(round(p,3))
contigency= pd.crosstab(df['Ceo'], df['Ranking_group'])
c, p, dof, expected = stats.chi2_contingency(contigency)
print(round(p,3))
contigency= pd.crosstab(df['Industry'], df['Ranking_group'])
c, p, dof, expected = stats.chi2_contingency(contigency)
print(round(p,3))
contigency= pd.crosstab(df['Sector'], df['Ranking_group'])
c, p, dof, expected = stats.chi2_contingency(contigency)
print(round(p,3))
contigency= pd.crosstab(df['Sector'], df['Industry'])
c, p, dof, expected = stats.chi2_contingency(contigency)
print(round(p,3))
```

[ ]:
```
# prepare data for regression

data = df.drop(['Title', 'Website','Hqlocation', 'Hqaddr', 'Hqcity',
               'Ceo', 'Ceo-title','Address', 'Ticker', 'Fullname'], axis=1)

# drop outliers
data = data.drop(394)
data = data.drop(data['Revenues'].idxmax())
data = data.drop(data['Revenue_per_employee'].idxmax())

# changing qualitative data to quantitative
dummies = pd.get_dummies(data[['Sector', 'Industry','Hqstate']])

# preparing numerical columns
X_numerical = data.drop(['Sector', 'Industry', 'Hqstate', 'Hqstate', 'Hqzip',
                      'Hqtel', 'Revenues','Ranking_group', 'Growth_direction',
                       'Assets', 'Rank', 'Profits'],
                        axis=1).astype('float64')
list_numerical = X_numerical.columns

# putting it all together
X = pd.concat([X_numerical, dummies], axis=1)
```

```
# what will be predicted
y = data['Revenues']

# splitting train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=10)

# normalizing data
scaler = StandardScaler().fit(X_train[list_numerical])
X_train[list_numerical] = scaler.transform(X_train[list_numerical])
X_test[list_numerical] = scaler.transform(X_test[list_numerical])
```

[ ]:
```
#  MAPE calculation function

def MAPE(Y_actual,Y_Predicted):
    mape = np.mean(np.abs((Y_actual - Y_Predicted)/Y_actual))*100
    return mape
```

[ ]:
```
# lasso regression with alpha = 1

reg = Lasso(alpha=1)
reg.fit(X_train, y_train)

print('R squared training set', round(reg.score(X_train, y_train)*100, 2))
print('R squared test set', round(reg.score(X_test, y_test)*100, 2))

pred_train = reg.predict(X_train)
rmse_train = math.sqrt(mean_squared_error(y_train, pred_train))
print('RMSE training set', round(rmse_train, 2))

pred = reg.predict(X_test)
rmse_test = math.sqrt(mean_squared_error(y_test, pred))
print('RMSE test set', round(rmse_test, 2))
```

[ ]:
```
# plot of lasso with different alphas

alphas = np.linspace(0.01,500,100)
lasso = Lasso(max_iter=10000)
coefs = []

for a in alphas:
    lasso.set_params(alpha=a)
    lasso.fit(X_train, y_train)
    coefs.append(lasso.coef_)

ax = plt.gca()
```

```
ax.plot(alphas, coefs)
ax.set_xscale('log')
plt.axis('tight')
plt.xlabel('alpha')
plt.ylabel('Standardized Coefficients')
plt.savefig('lasso_coeff_alpha.png',bbox_inches='tight')
```

[ ]:
```
# lasso with best alpha

# Lasso with 5 fold cross-validation
model = LassoCV(cv=5, random_state=0, max_iter=10000)

# Fit model
model.fit(X_train, y_train)

LassoCV(cv=5, max_iter=10000, random_state=0)

print('Alpha', model.alpha_)

lasso_best = Lasso(alpha=model.alpha_)
lasso_best.fit(X_train, y_train)

print('R squared training set', round(lasso_best.score(X_train, y_train)*100,␣
 ↪2))
print('R squared test set', round(lasso_best.score(X_test, y_test)*100, 2))



lasso_predict = reg.predict(X_test)
lasso_predict_best = lasso_best.predict(X_test)
Lasso_MAPE = MAPE(y_test,lasso_predict)
Lasso_MAPE_best = MAPE(y_test,lasso_predict_best)
print("MAPE value: ",Lasso_MAPE)
print("MAPE value best: ",Lasso_MAPE_best)
Accuracy = 100 - Lasso_MAPE
Accuracy_best = 100 - Lasso_MAPE_best
print('Accuracy of Lasso Regression: {:0.2f}%.'.format(Accuracy))
print('Accuracy of Lasso Regression best: {:0.2f}%.'.format(Accuracy_best))

pred_train = lasso_best.predict(X_train)
rmse_train = math.sqrt(mean_squared_error(y_train, pred_train))
print('RMSE best training set', round(rmse_train, 2))

pred = lasso_best.predict(X_test)
rmse_test = math.sqrt(mean_squared_error(y_test, pred))
print('RMSE best test set', round(rmse_test, 2))
```

[ ]:
```
# print coefficients

ser = pd.Series(lasso_best.coef_, index = X_train.columns)
print(ser[ser > 0])
print(ser[ser < 0])
```

[ ]:
```
# ridge regression with alpha = 1
rig = Ridge(alpha=1)
rig.fit(X_train, y_train)
print('R squared training set', round(rig.score(X_train, y_train)*100, 2))
print('R squared test set', round(rig.score(X_test, y_test)*100, 2))

pred_train = rig.predict(X_train)
rmse_train = math.sqrt(mean_squared_error(y_train, pred_train))
print('RMSE training set', round(rmse_train, 2))

pred = rig.predict(X_test)
rmse_test = math.sqrt(mean_squared_error(y_test, pred))
print('RMSE test set', round(rmse_test, 2))
```

[ ]:
```
# plot of lasso with different alphas

alphas = np.linspace(0.01,500,100)
ridge = Ridge(max_iter=10000)
coefs = []


for a in alphas:
    ridge.set_params(alpha=a)
    ridge.fit(X_train, y_train)
    coefs.append(ridge.coef_)

ax = plt.gca()

ax.plot(alphas, coefs)
ax.set_xscale('log')
plt.axis('tight')
plt.xlabel('alpha')
plt.ylabel('Standardized Coefficients')
plt.savefig('ridge_coeff_alpha.png',bbox_inches='tight')
```

[ ]:
```
# ridge with best alpha

# ridge with 5 fold cross-validation
model = RidgeCV(cv=5)

# Fit model
```

```
model.fit(X_train, y_train)

RidgeCV(cv=5)

print('Alpha', model.alpha_)

ridge_best = Ridge(alpha=model.alpha_)
ridge_best.fit(X_train, y_train)

print('R squared training set', round(ridge_best.score(X_train, y_train)*100,␣
 ↪2))
print('R squared test set', round(ridge_best.score(X_test, y_test)*100, 2))

ridge_predict = rig.predict(X_test)
ridge_predict_best = ridge_best.predict(X_test)
ridge_MAPE = MAPE(y_test,ridge_predict)
ridge_MAPE_best = MAPE(y_test,ridge_predict_best)
print("MAPE value: ",ridge_MAPE)
print("MAPE value best: ",ridge_MAPE_best)
Accuracy = 100 - ridge_MAPE
Accuracy_best = 100 - ridge_MAPE_best
print('Accuracy of Ridge Regression: {:0.2f}%.'.format(Accuracy))
print('Accuracy of Ridge Regression best: {:0.2f}%.'.format(Accuracy_best))

pred_train = ridge_best.predict(X_train)
rmse_train = math.sqrt(mean_squared_error(y_train, pred_train))
print('RMSE best training set', round(rmse_train, 2))

pred = ridge_best.predict(X_test)
rmse_test = math.sqrt(mean_squared_error(y_test, pred))
print('RMSE best test set', round(rmse_test, 2))
```

```
[ ]: # ridge and lasso comparison graph

features = X.columns
target = 'Revenues'
print(features)
print(target)

plt.figure(figsize = (25, 10))
plt.plot(features,ridge_best.coef_,alpha=0.
 ↪7,linestyle='none',marker='*',markersize=5,
         color='red',label=r'Ridge; $\alpha = 10$',zorder=7)
plt.plot(lasso_best.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,
         color='blue',label=r'lasso; $\alpha = 2019.8$')

plt.xticks(rotation = 90)
plt.legend()
plt.savefig('lasso_ridge.png',bbox_inches='tight')
```