ECM3420 Coursework Assessment

A. Main objective of the analysis that specifies whether your model will be focused on prediction or interpretation.

In this analysis I will focus on prediction of the dataset in an attempt to be able to predict the profits of Fortune 500 companies with minimal error. I will attempt to find the factors that have the largest impact on the profits of a company and use two different regression models to attempt to make predictions about it.

B. Brief description of the data set you chose and a summary of its attributes.

The dataset I choose was the 2017 Fortune 500 dataset available at [1], the Fortune 500 is a list of the 500 American companies that generated the most revenue in that year, ranked by the revenue generated. It is a dataset of shape 500×23 that contains the data for every Fortune 500 company in 2017 the columns are as follows:

- rank – the ranking of the company in the Fortune 500 list.
- title – the name of the company.
- website – the website of the company.
- employees – the number of people the company employee.
- sector – the business sector the company is involved in.
- industry – the main work that the company does.
- hqlocation – the town/city and state where the headquarters of the company is located.
- hqaddr – the address of the company headquarters.
- hqcity – the city the headquarters are located in.
- hqstate – the state in which the company headquarters are located.
- hqzip – the zip code of the company headquarters.
- hqtel – the telephone number of the company headquarters.
- ceo – the name of the CEO of the company.
- ceo_title – the title of the company CEO.
- address – the company address.
- ticker – the company/stock ticker used to uniquely identify publicly traded shares.
- fullname – the full name of the company.
- revenues – the revenue of the company in million USD.
- revchange – the percentage change in revenue of the company from the previous year.
- profits – the profit of the company in million USD.
- prftchange – the percentage change in profits of the company from the previous year.
- assests – the asset value of the company in million USD.
- totshequity - represents the net worth of a company, which is the amount that would be returned to shareholders if a company's total assets were liquidated, and all of its debts repaid in million USD.

C. Brief summary of data exploration and actions taken for data cleaning and feature engineering.

Data cleaning

The data is cleaned by first checking for any duplicates.

```
Check for duplicates

In [27]: df['title'].is_unique # Returns True if all values of the 'title' column are unique.
Out[27]: True
```

Every entry in the ‘title’ column is unique so that means there is no duplicated data.

Next every data entry is checked to make sure it is in the appropriate format so that there are no structural errors.

Unwanted outliers are checked for. The only outlier in the dataset is the entry for ‘Walmart’ which has hugely higher values for certain indexes including revenue and employees, however this is still valid data and despite being an outlier is not unwanted. In some instances, in future this entry may have to be temporarily ignored in order too not affect results or visualisation too much.

The dataset is checked for any blank values and none found.

The dataset is reviewed to make sure everything about it makes sense and a new pandas dataframe object is created with the following columns dropped:

- website
- hqlocation
- hqaddr
- hqzip
- hqtel
- ceo
- ceo_title
- fullname

These are dropped to reduce the size of the dataframe object that the analysis will occur on as there is no point in keeping them as they will not be used in any of the analysis. They are however still kept in the original dataset.

Data exploration

The full exploration is visible in the Jupyter notebook file.

First, I made a histogram [Figure 1] to explore the number of companies that were at each data level with the histogram bars split into sections to show the sectors that the companies are involved in. It can be seen that the vast majority of companies are on the far left of the graph. On the far right of the graph is one datapoint with a far higher revenue than any other in the dataset, this datapoint is the 1st ranked company; Walmart who had a yearly revenue of 485,873 Million USD, compared to 2nd place; Berkshire Hathaway at 223,604 Billion USD, a difference of 117.3% rounded to 4 significant figures. In order to get a closer view of the rest of the data I created a new variable 'walmart_removed_df' and assigned it the value of the original dataframe with the first row removed using the pandas.DataFrame.drop function. I then plotted the graph as before but using this new dataframe so the Walmart data was not included [Figure2]

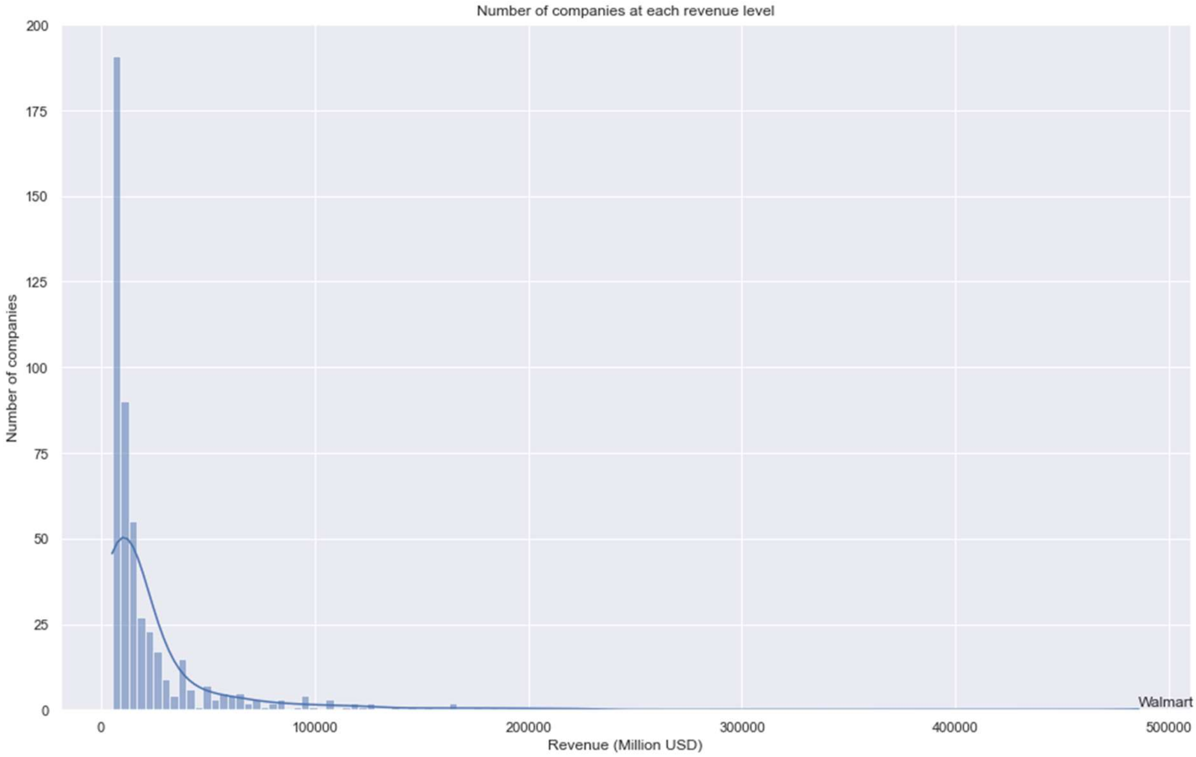


Figure 1 - Histogram of the number of companies at each revenue level

Figure 1 - Histogram of the number of companies at each revenue level

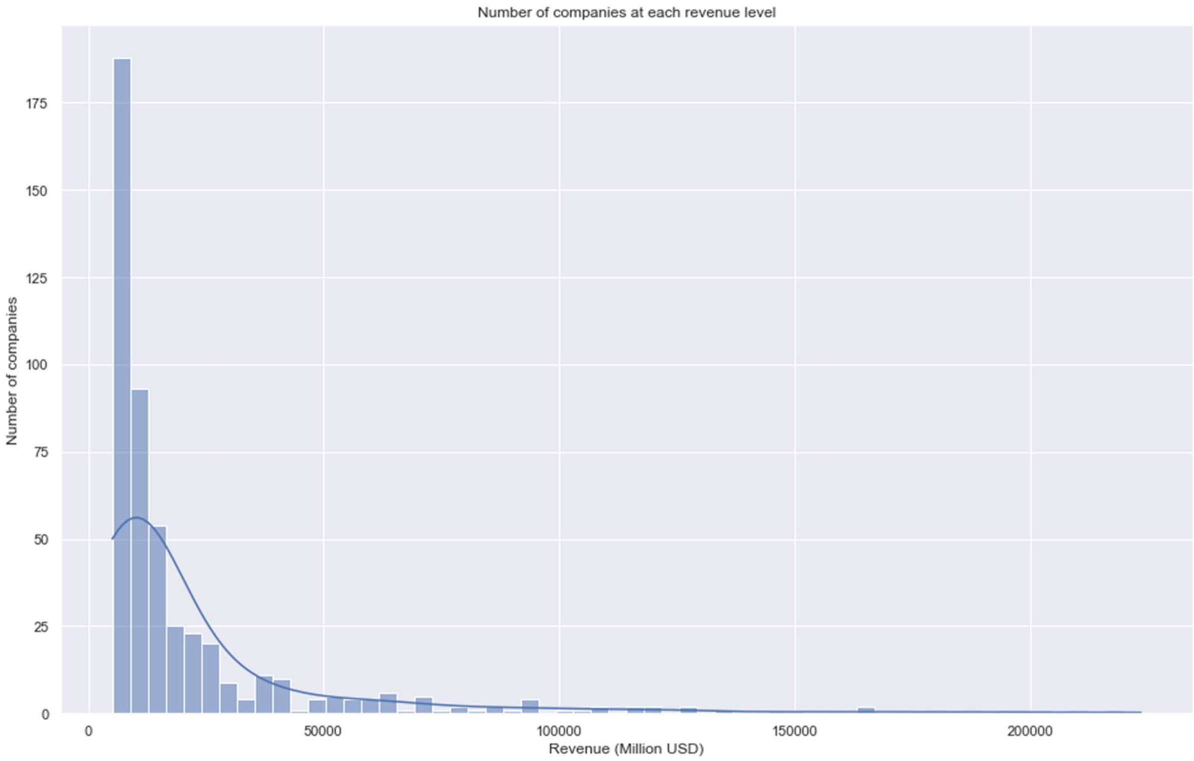


Figure 2 - Histogram of the number of companies at each revenue level with Walmart data removed

Figure 2 provides the same data as Figure 1 but without the larger scale needed to display the Walmart data, allowing the rest of the data to be seen more clearly.

Next is [Figure 3], a bar graph using seaborn countplot to count the number of companies that are in each industry sector and [Figure 4], a pie chart showing the percentage distribution of the sectors in the dataset.

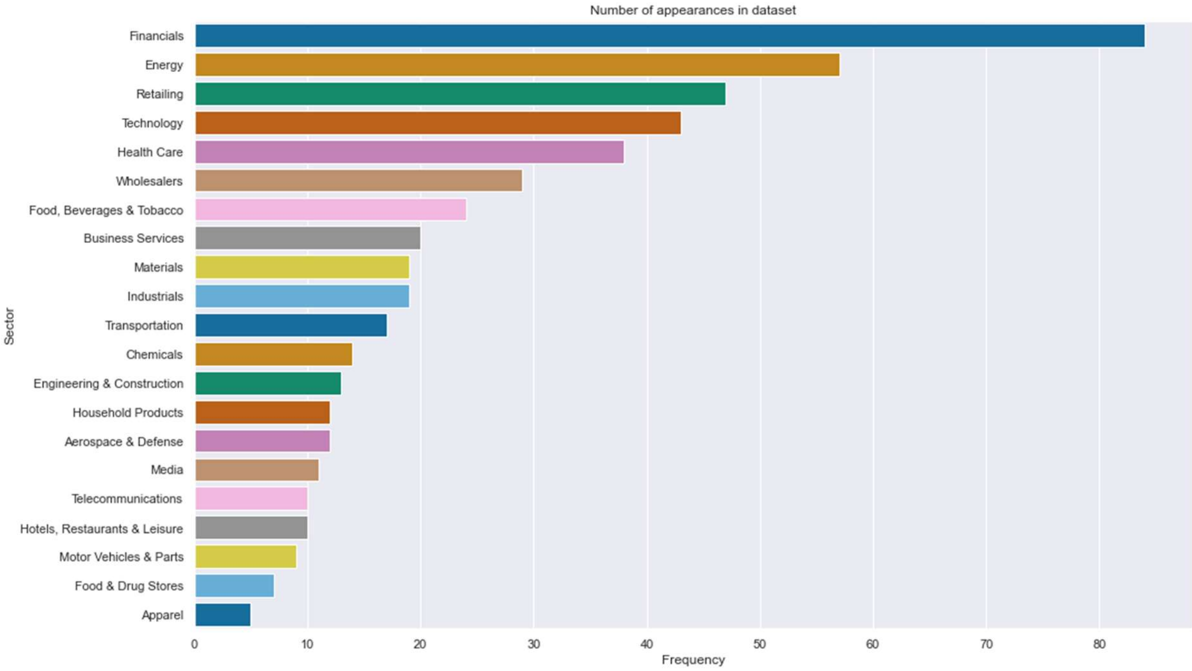


Figure 3 - Bar graph of the number of appearances in the dataset for each sector

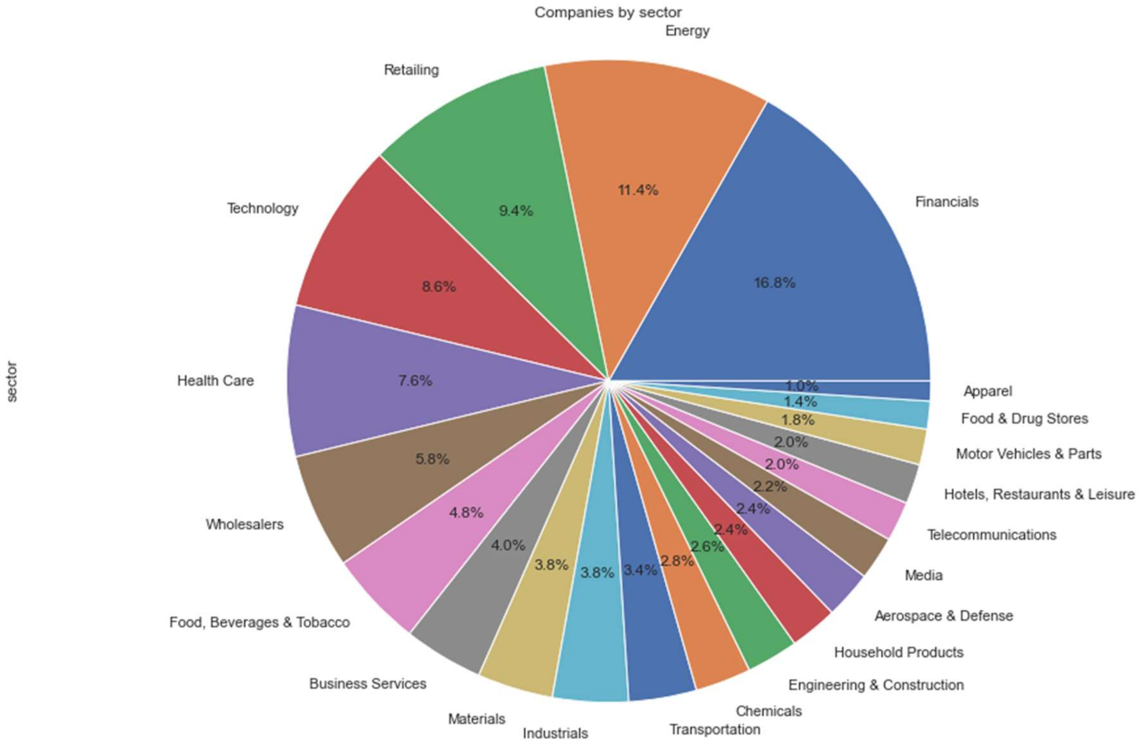


Figure 4 - Pie chart of the distribution of sectors

From these two figures it can be clearly seen that the most common sectors, and only ones above 5% of the dataset are Financials, Energy, Retailing, Technology, Health Care and Wholesalers.

The next exploration step was comparing the revenues of each sector. I did this in 2 ways, first, I made a seaborn bar graph of the total revenue generated by each sector, ordering the graph by the frequency at which the sectors occurred [Figure 5]. I then made another bar graph, this time comparing the mean average revenue for each sector [Figure 6], I choose the mean average as I did not want any data point to be ignored in this average.

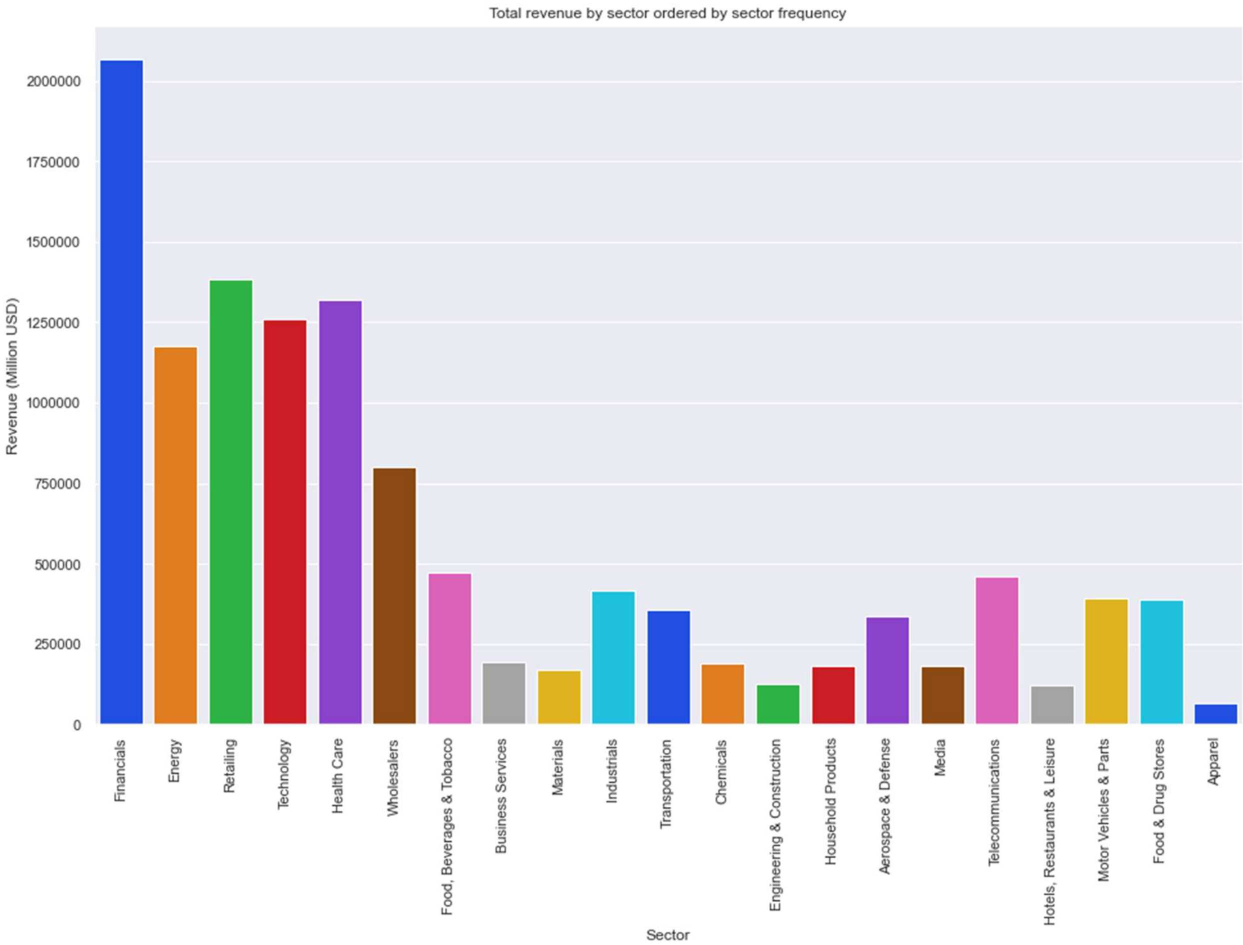


Figure 5 - Total revenue per sector

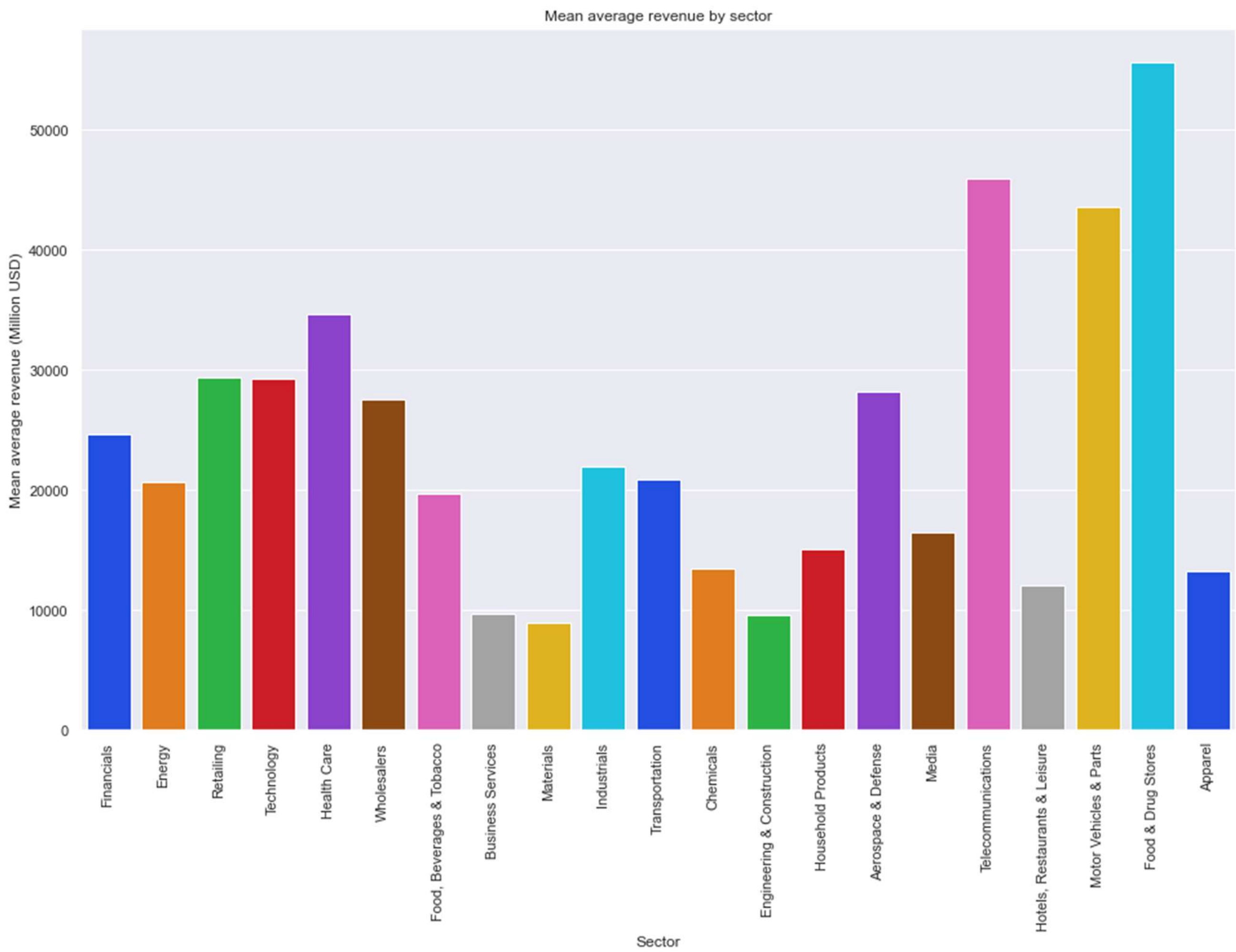


Figure 6 - Mean average revenue per sector

From these Figure 5 it can be seen that usurpingly the overall most revenue is generated by Financials companies which is also the most frequently occurring sector and the least overall revenue is generated by apparel companies, the least frequently occurring. Figure 6 shows that on average the most revenue is generated by companies in the Food & Drug Store sector, the 2nd least frequent and from Figure 3 it can be seen that only 7 of these companies exist in the dataset.

The next exploration I performed was comparing the number of companies that were based in each state. I did this by making a seaborn countplot bar graph to count the number of companies that had each value of ‘hqstate’ [Figure 7].

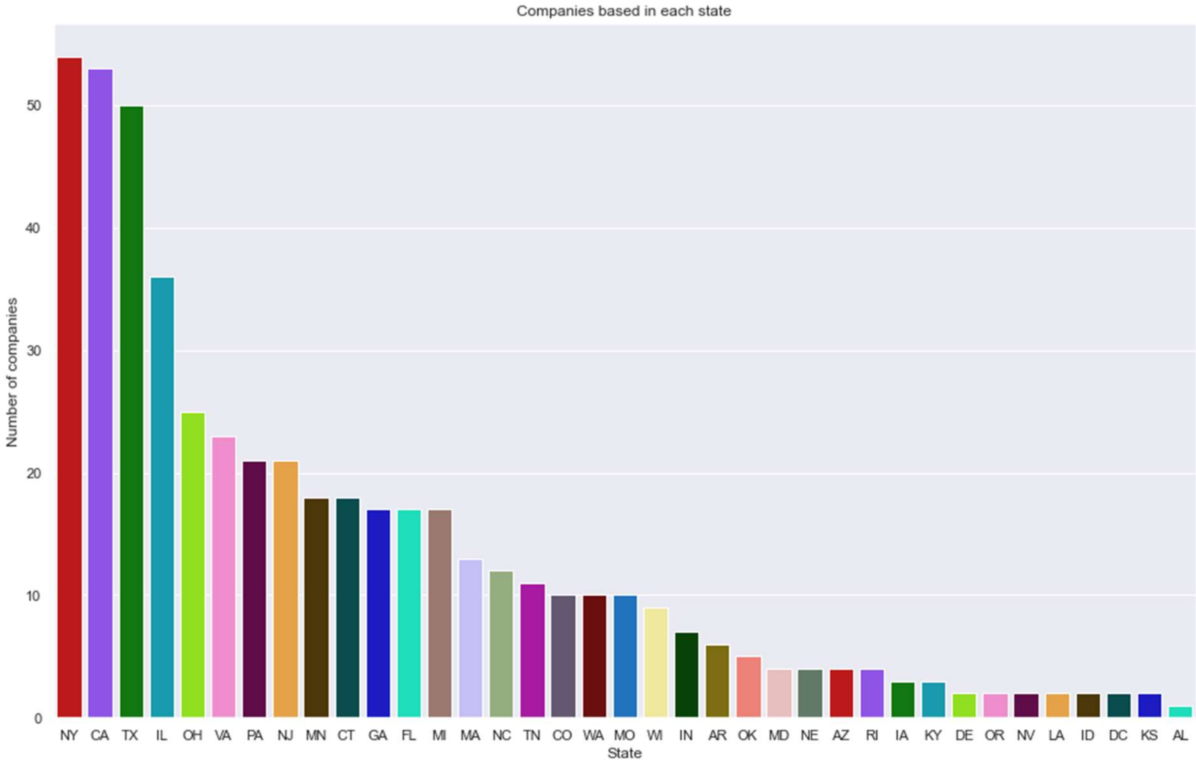


Figure 7 - Bar graph of number of companies bases in each state

From Figure 7 it can be seen that the most popular headquarter state is New York, followed by California and Texas, and the least popular state to be based in is Alabama.

The next graph I made was a scatter graph using seaborn scatterplot was exploring the relationship between the number of employees of a company and the revenue that a company makes.

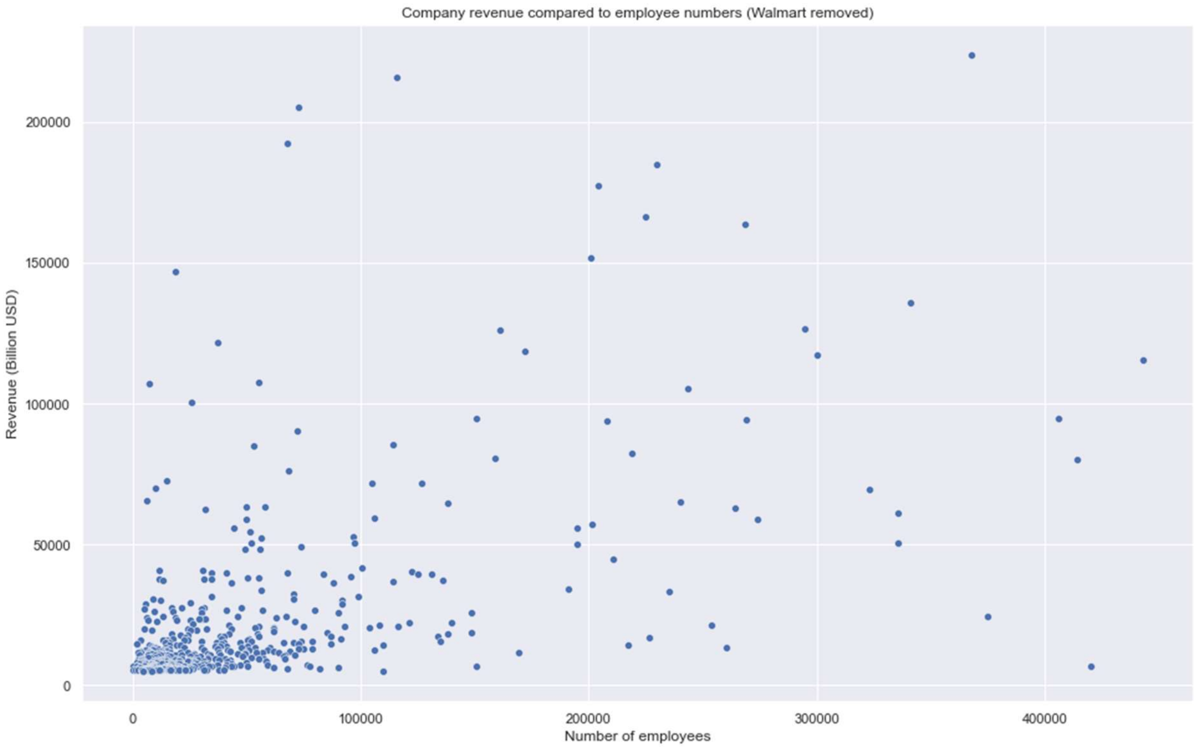


Figure 8 - Scatter graph of number of employees compared to company revenue

The final exploration I did was to use the seaborn.pairplot function to quickly produce grapghs listing all of the numerical columns against each other in scatter graph form [Figure 9].

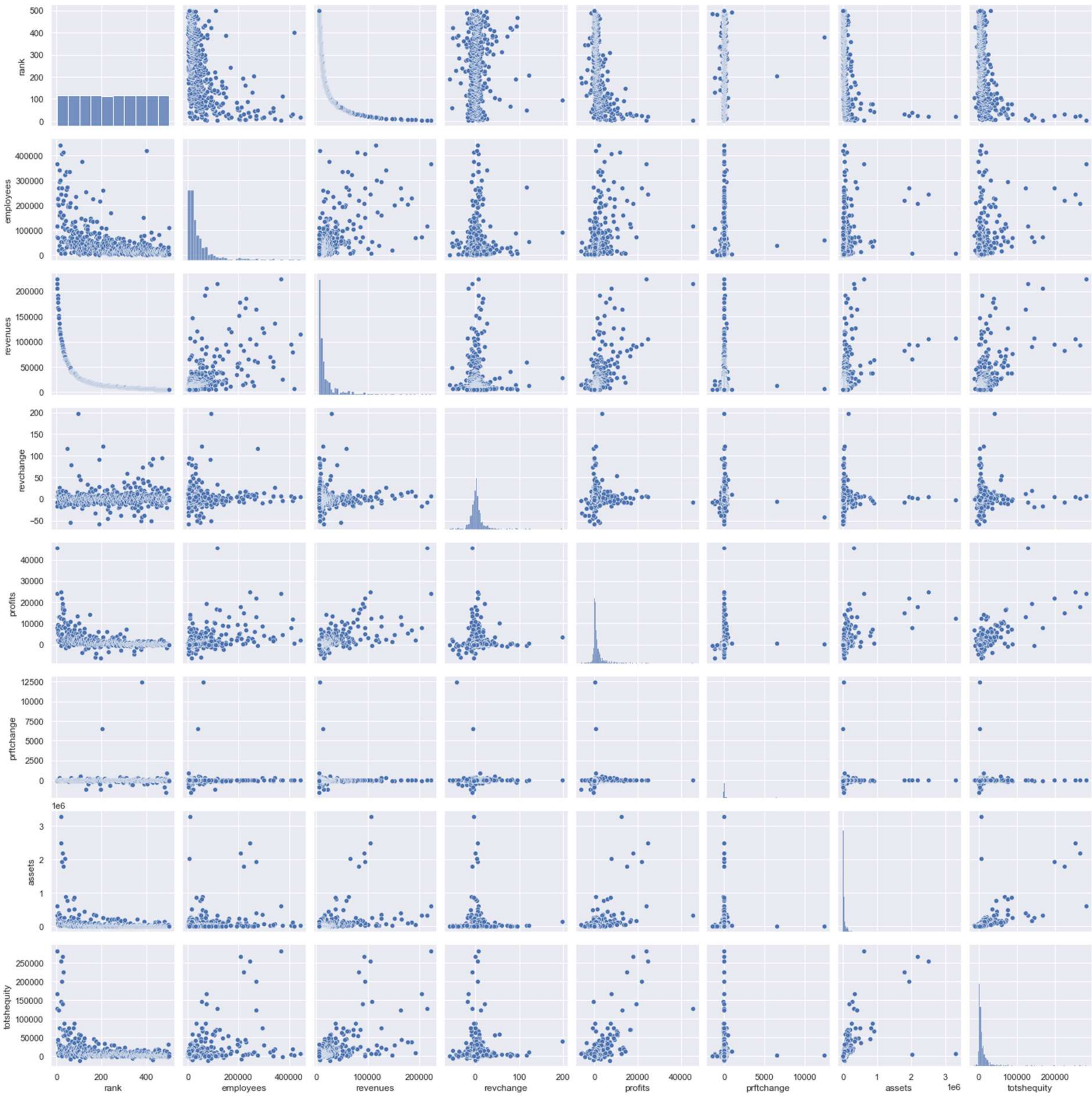


Figure 9 - Pair plot of the numerical columns of the dataset

<u>D. Summary of training two machine learning models and F. Summary Key Findings and Insights, which walks your reader through the main drivers of your model and insights from your data derived from your models.</u>

I chose to perform regression on this dataset to use as prediction, so I selected the variables I was going to use as 'x' and the variable I was going to try and predict as 'y'. I chose profits as y and for 'x' I chose to use *employees, revenues,, prftchange, assets, revchange* and *totshequity* as from the heatmap [Figure 10] as these were the numeric values in the dataset and I could see that all of them had a correlation with profits, I chose to exclude rank due to seeing that it had a negative correlation on profits.

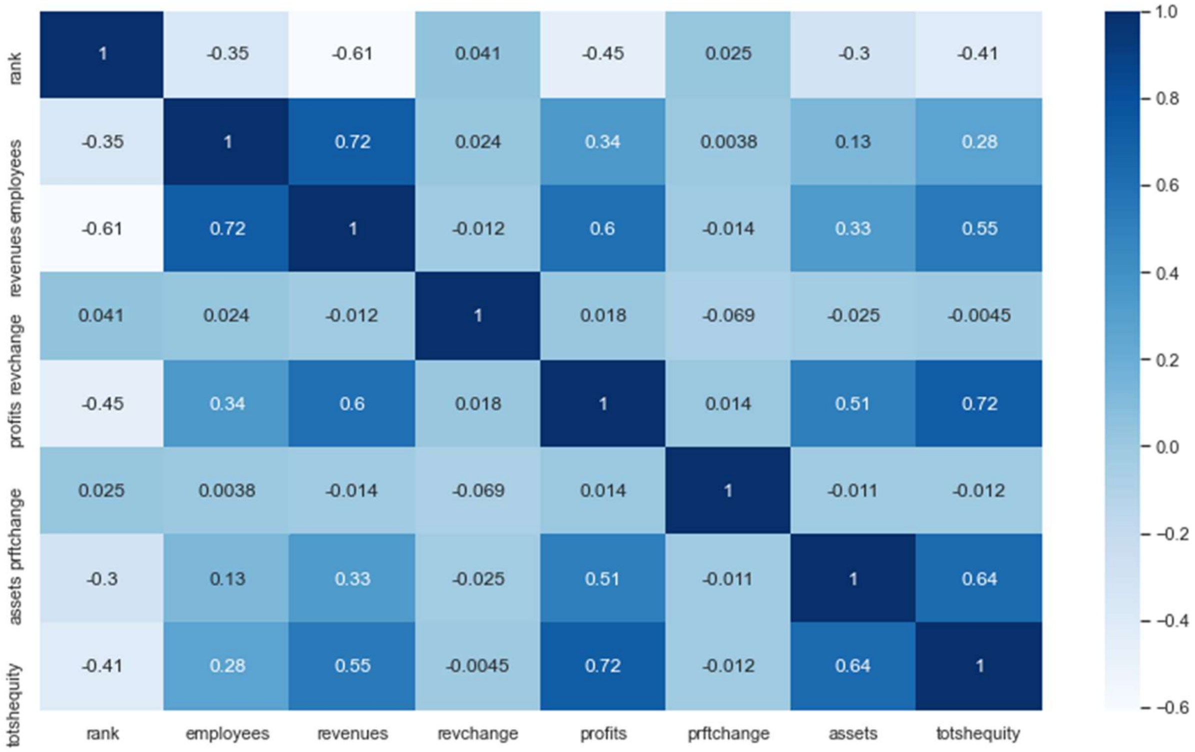


Figure 10 - Heatmap of feature correlation

Regression models describe the relationship between variables, linear regression models use a straight line. Multiple linear regression attempts to model the relationship between two or more explanatory variables and a response variable by fitting a linear equation to observed data. The dependent variable must be a continuous value, e.g. profits in this case. The independent variables may be either continuous or binary, in this case all are continuous.

Singular linear regression can also be easily plotted using the seaborn regplot function which plots the data given to it along with a linear regression fit. Plotting profits against the single factor that has the highest impact, which from the heatmap we can see is totshequity, we get the [Figure 11] showing the relationship between the 2 features and a fit line that can be used to read predictions from the graph.

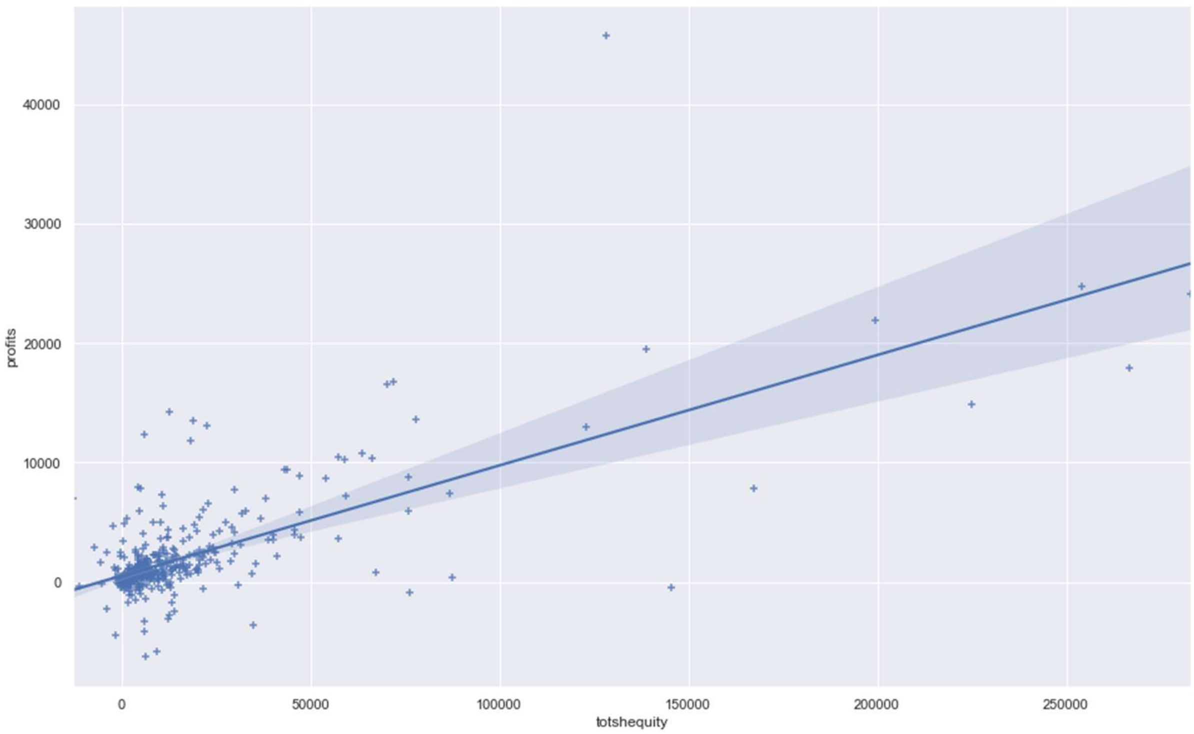


Figure 11 - Single linear regression graph

Multiple linear regression

In linear regression a data split is required, splitting both the independent variables and the dependent variables into two sets, one set is the training data; used to fit the regression model and the other is the test data; used to test the fit of the regression model on for analysis. I did this by using the scikit-learn model_selection function; train_test_split with a test size of 0.3, meaning that 30% of the data is used for testing and the other 70% is used to train the fit of the regression.

The fit is then applied using scikit-learn's LinearRegression() and fit() functions. Following this the test data for the independent variables can then be used with scikit-learn's predict() function to predict the corresponding values of the dependent variable.

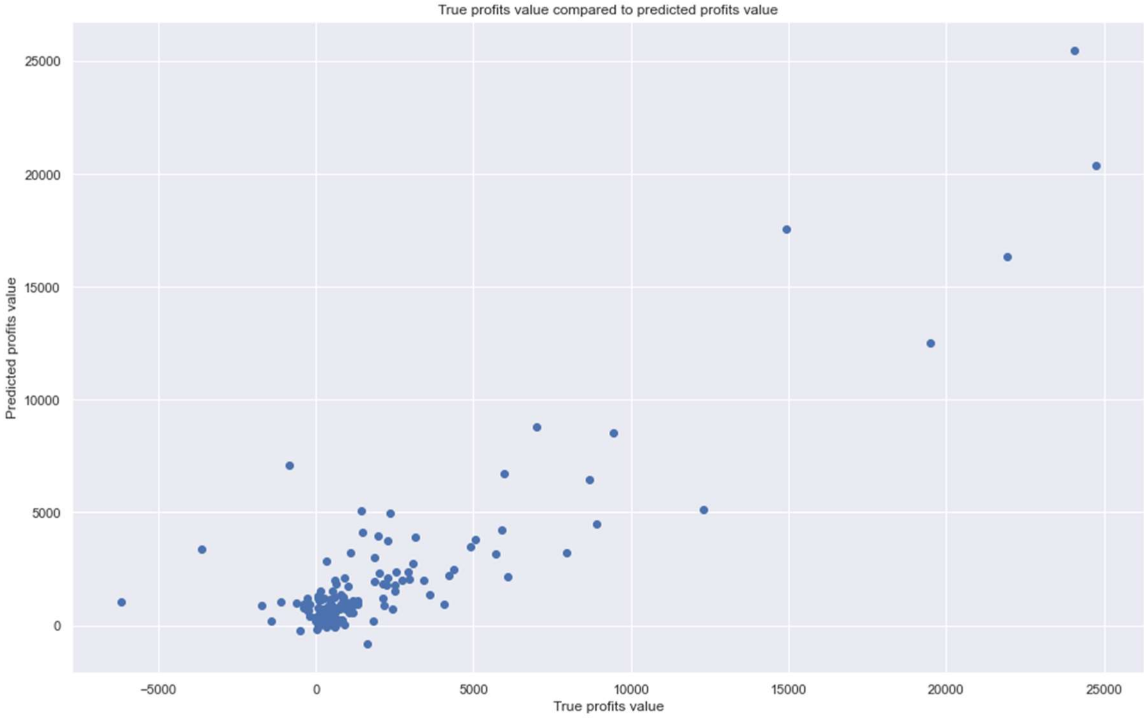


Figure 12 - Scatter graph of the predicted values compared to true values

The results of the predictions compared to the true values can be seen in [Figure 12], and it can be seen that the predicted value is often close to the actual value, however there are several values which have less accurate predictions and a small number that have very inaccurate predictions. Two further figures, [Figures 13, 14] also show this.

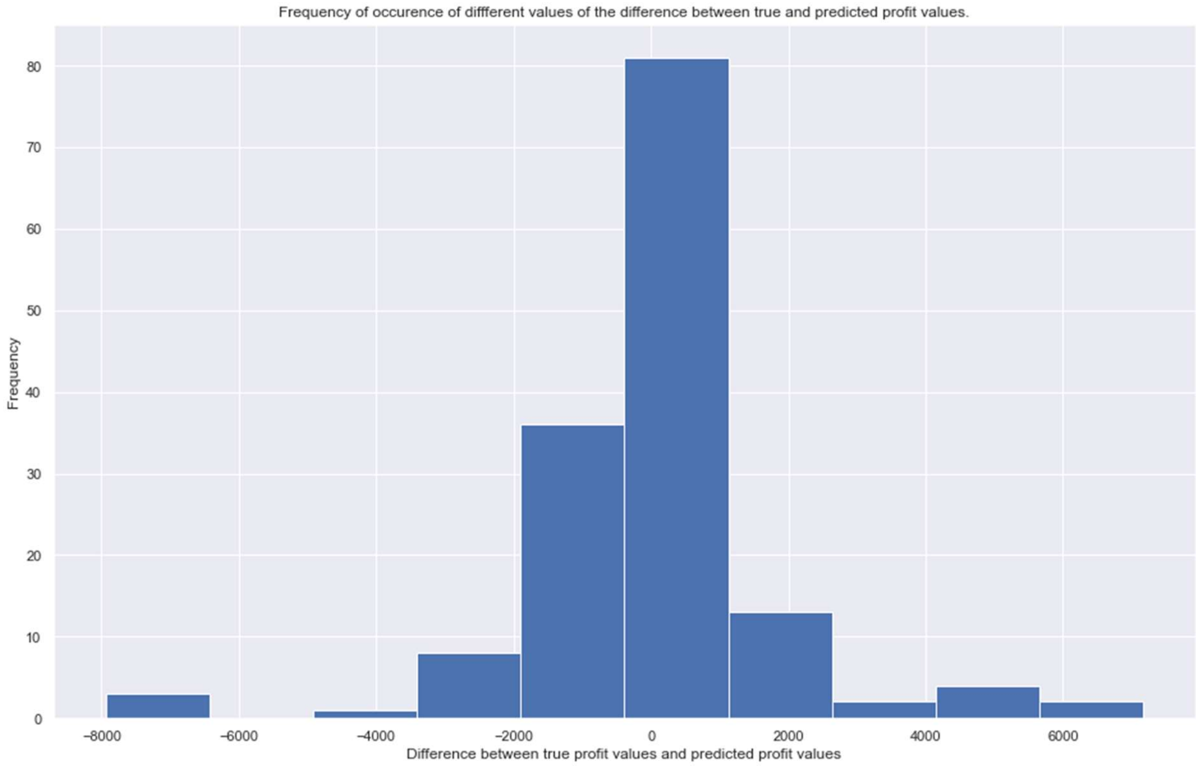


Figure 13 - Histogram of the difference between the true and predicted profit values

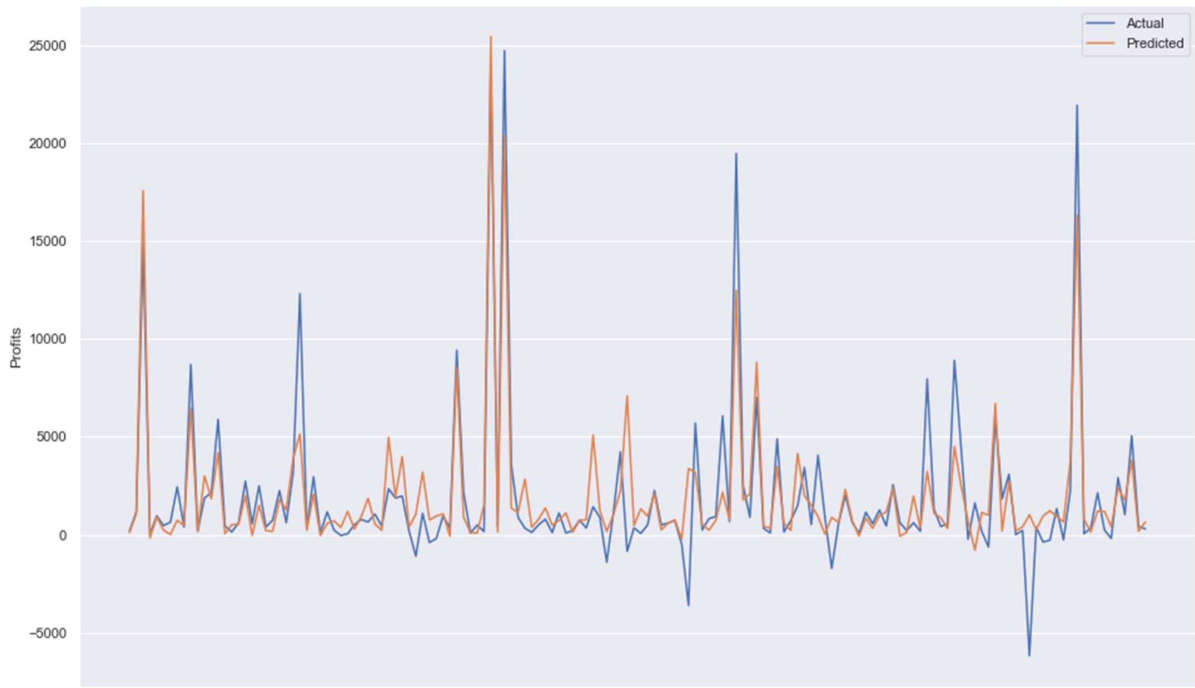


Figure 14 - Line graph of the true and predicted profit values for each item in the testing set

To calculate how accurate the predictions of this regression fit were, I used the mean absolute error (MAE) and the mean squared error (MSE). These are two ways of measuring the error of predictions, MAE measures the absolute difference between the true and predicted values, is non differentiable and is not highly sensitive to outliers. MSE measures the variance in predictions to true values, is differentiable and is highly sensitive to outliers. Figure 15 shows that values I got for these measures.

```
mae = metrics.mean_absolute_error(y_test, predictions)
mse = metrics.mean_squared_error(y_test, predictions)
print('Mean absolute error = ',(mae))
print('##############################################')
print('Mean squared error = ',(mse))
print('##############################################')
r2_linreg = metrics.r2_score(y_test, predictions)
print('R2 score = ',(r2_linreg))
print('##############################################')
```

```
Mean absolute error =  1358.5135112556482
##############################################
Mean squared error =  12266589.642697483
##############################################
R2 score =  0.48337490563151286
##############################################
```

Figure 15 - MAE, MSE and R2 of the regression fit

From the MAE value it can be seen that the average difference between the predicted and true profit values was the relatively low, it varied each time the code is run but it was usually between the values of 1200 - 1550 (Million USD). The MSE however is usually very large, indicating a large variance in results, this could be in part due to the few predictions that were very wrong as MSE is highly sensitive to any outliers. I also got an R2 score of 0.4834 (4 S.F) implying that 48.34% of the data fit the regression, the R2 also changed every time the code was run, I ran it 30 times and the lowest score I saw it at was 0.3836 and the highest score was 0.7998. I think this variance in metrics was due to the randomly selected split of the training and testing data.

The multiple linear regression fit may be used to predict the profit values of a specific item in the dataset, for example, using the first item in the dataset, Walmart in this instance the model predicted a profit of 28883 (Million USD) and the real value was 13643 (Million USD) giving an error in this prediction of 15240 (Million USD).

Lasso Regression

LASSO stands for **L**east **A**bsolute **S**hrinkage and **S**election **O**perator, it is a statistical formula for that performs variable selection and regularisation of data. Lasso uses L1 regularisation which adds a penalty equal to the absolute value of the magnitude of coefficients. This regularisation can result in models with only a few coefficients as in the process, some coefficients may become zero and therefore be eliminated from the model.

I made this model using the scikit-learn LassoCV function to find the optimum lambda value, the value by which the tuning is done, as the lambda is increased the level of shrinkage applied is increased, meaning that coefficients trend towards zero. This function applies iterative fitting along a regularization path suing cross validation. As before due to random data split into training and test data this value varies but, in my model, I found it to be usually around the value of 990. I then fit the lasso model with this optimum value to my training data and used the model to predict the values of my test data. Doing this gave me the metrics shown in [Figure 16].

```
lasso_MSE = metrics.mean_squared_error(y_test,y_pred_tuned)
lasso_MAE = metrics.mean_absolute_error(y_test, y_pred_tuned)
r2_lasso = metrics.r2_score(y_test, y_pred_tuned)

print('Mean absolute error = ',(lasso_MAE))
print('###############################################')
print('Mean squared error = ',(lasso_MSE))
print('###############################################')
print('R2 score = ',(r2_lasso))
print('###############################################')
```

```
Mean absolute error =  1357.357997981189
###############################################
Mean squared error =  12264230.77818813
###############################################
R2 score =  0.48347425260856547
###############################################
```

Figure 16 - MAE, MSE and R2 score for lasso regression

I then compared these values to those of [Figure 15] and got the results show in [Figure 17].

```
mae_diff = mae - lasso_MAE
mse_diff = mse - lasso_MSE
r2_diff = r2_linreg - r2_lasso

print('Mean absolute error difference = ',(mae_diff))
print('###############################################')
print('Mean squared error difference = ',(mse_diff))
print('###############################################')
print('R2 score difference = ',(r2_diff))
print('###############################################')
```

```
Mean absolute error difference =  1.1555132744590537
###############################################
Mean squared error difference =  2358.864509353414
###############################################
R2 score difference =  -9.934697705260565e-05
###############################################
```

Figure 17 - Differences between the MAE, MSE and R2 score for my multiple linear regression model and my lasso regression model

E. A paragraph explaining which of your models you recommend as a final model that best fits your needs in terms of accuracy and explainability.

I recommend using my lasso regression model as the final model as in the example run of the models shown in [Figure 15, 16, 17], the majority of iterations of the code provided lasso regression to have slightly slower MAE and MSE and slightly higher R2 score which are all positives. However, this improved accuracy is still only a very small increase compared to multiple linear regression.

G. Suggestions for next steps in analysing this data, which may include suggesting revisiting this model adding specific data features to achieve a better explanation, a better prediction, etc.

For next steps analysing this data I do not think adding any more data features is applicable as I used all the data features of the dataset that had a positive correlation with the value I was attempting to predict, this could potentially be improved by using a dataset that contained more data features.

I am not sure of the impact that multicollinearity had on this model but if I were to attempt this again, I may try to instead use a different model that is less vulnerable to the effects of it such as Principal component regression.

I would like to try to increase the reliability of the model by decreasing the variance of results I got as a result of the random data split. One method of this could be to increase the size of the dataset to instead be Fortune 1000 companies or even more instead of Fortune 500.

Bibliography

[1] "Fortune 500 - 2017 - dataset by aurielle," 2017. [Online]. Available: https://data.world/aurielle/fortune-500-2017.

Appendix: Jupyter notebook code

Jupyter Notebook Code

1 ECM3420 Coursework

1.1 Data cleaning and exploration

Import all the required libraries and set sttings as desired.

[1]:
```
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.model_selection import train_test_split, RepeatedKFold,␣
↪cross_val_score, GridSearchCV
from sklearn.linear_model import LinearRegression, Lasso, LassoCV
from sklearn import metrics
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from mpl_toolkits.mplot3d import Axes3D
import colorcet as cc
from IPython.display import Markdown as md


sns.set(rc={'figure.figsize':(16,10)})

palette = sns.color_palette(cc.glasbey, n_colors=25)
```

Read the dataset csv file into a pandas dataset and print it.

[2]:
```
df = pd.read_csv('fortune500.csv')

print(df)
```

```
     rank               title                         website  employees  \
0       1             Walmart          http://www.walmart.com    2300000
1       2  Berkshire Hathaway  http://www.berkshirehathaway.com   367700
2       3               Apple            http://www.apple.com    116000
3       4         Exxon Mobil       http://www.exxonmobil.com     72700
4       5            McKesson         http://www.mckesson.com     68000
```

```
..   …             …                           …       …
495  496    Michaels Cos.        http://www.michaels.com    31000
496  497    Toll Brothers    http://www.tollbrothers.com     4200
497  498            Yahoo           http://www.yahoo.com     8500
498  499    Vistra Energy    http://www.vistraenergy.com     4431
499  500   ABM Industries             http://www.abm.com   110000

                          sector                                  industry  \
0                      Retailing                    General Merchandisers
1                     Financials  Insurance: Property and Casualty (Stock)
2                     Technology            Computers, Office Equipment
3                         Energy                      Petroleum Refining
4                    Wholesalers                Wholesalers: Health Care
..                             …                                         …
495                    Retailing              Specialty Retailers: Other
496  Engineering & Construction                             Homebuilders
497                   Technology         Internet Services and Retailing
498                       Energy                                   Energy
499            Business Services         Diversified Outsourcing Services

             hqlocation                 hqaddr         hqcity hqstate  …  \
0      Bentonville, AR    702 S.W. Eighth St.    Bentonville      AR  …
1            Omaha, NE         3555 Farnam St.          Omaha      NE  …
2        Cupertino, CA         1 Infinite Loop      Cupertino      CA  …
3           Irving, TX  5959 Las Colinas Blvd.         Irving      TX  …
4    San Francisco, CA             1 Post St.  San Francisco      CA  …
..                   …                      …              …       …  …
495         Irving, TX    8000 Bent Branch Dr.         Irving      TX  …
496        Horsham, PA      250 Gibraltar Rd.         Horsham      PA  …
497      Sunnyvale, CA         701 First Ave.       Sunnyvale      CA  …
498         Dallas, TX         1601 Bryan St.          Dallas      TX  …
499       New York, NY        1 Liberty Plaza        New York      NY  …

                                          ceo_title  \
0    President, Chief Executive Officer & Director
1              Chairman & Chief Executive Officer
2              Chief Executive Officer & Director
3              Chairman & Chief Executive Officer
4    Chairman, President & Chief Executive Officer
..                                               …
495            Chairman & Chief Executive Officer
496            Chief Executive Officer & Director
497  President, Chief Executive Officer & Director
498  President, Chief Executive Officer & Director
499  President, Chief Executive Officer & Director

                                      address ticker  \
0    702 S.W. Eighth St., Bentonville, AR 72716    WMT
```

```
1             3555 Farnam St., Omaha, NE 68131   BRKA
2         1 Infinite Loop, Cupertino, CA 95014   AAPL
3      5959 Las Colinas Blvd., Irving, TX 75039    XOM
4          1 Post St., San Francisco, CA 94104    MCK
..                                         …    …
495      8000 Bent Branch Dr., Irving, TX 75063    MIK
496        250 Gibraltar Rd., Horsham, PA 19044    TOL
497          701 First Ave., Sunnyvale, CA 94089   YHOO
498             1601 Bryan St., Dallas, TX 75201    VST
499          1 Liberty Plaza, New York, NY 10006    ABM

                          fullname revenues revchange  profits  prftchange  \
0             Wal-Mart Stores, Inc.   485873       0.8  13643.0        -7.2
1           Berkshire Hathaway Inc.   223604       6.1  24074.0         0.0
2                       Apple, Inc.   215639      -7.7  45687.0       -14.4
3           Exxon Mobil Corporation   205004     -16.7   7840.0       -51.5
4              McKesson Corporation   192487       6.2   2258.0        53.0
..                              …      …       …        …         …
495  The Michaels Companies, Inc.     5197       5.8    378.2         4.2
496            Toll Brothers, Inc.     5170      23.9    382.1         5.2
497                    Yahoo! Inc.     5169       4.0   -214.3         5.2
498            Vistra Energy Corp.     5164       4.0   -214.3         5.2
499   ABM Industries Incorporated     5145      -2.8     57.2       -25.0

     assets  totshequity
0    198825      77798.0
1    620854     283001.0
2    321686     128249.0
3    330314     167325.0
4     56563       8924.0
..      …          …
495    2148      -1698.0
496    9737       4229.0
497   48083      31049.0
498   15167       6597.0
499    2281        974.0

[500 rows x 23 columns]
```

Check for duplicates

[3]:
```
df.duplicated().sum()
```

[3]: 0

It could easily be seen that many of the columns in this dataset would not be used for any anylsis of the data so I removed the following columns from the pandas dataframe using pandas.DataFrame.drop:

- website
- hqlocation
- hqaddr
- hqzip
- hqtel
- ceo
- ceo_title
- fullname

This reduces the dataframe to 500 x 15.

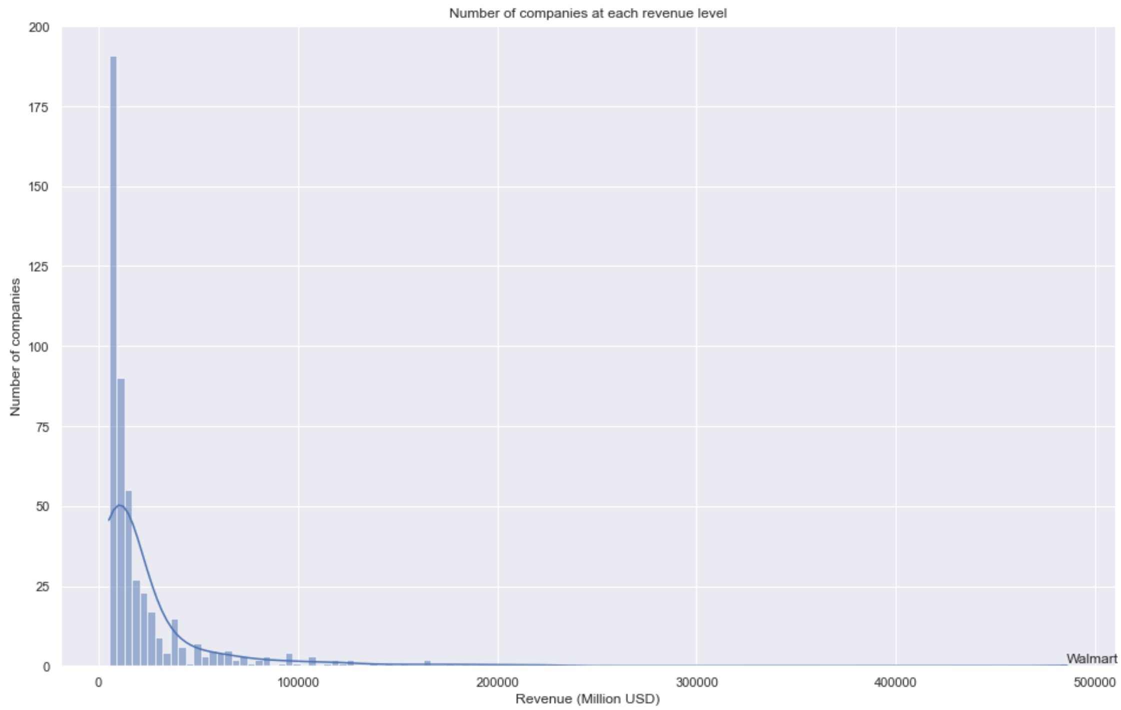
[4]:
```
df.
↪drop(columns=['website','hqlocation','hqaddr','hqzip','hqtel','ceo','ceo_title','fullname']
↪inplace=True)
print(df)
```

```
     rank              title  employees                      sector  \
0       1            Walmart    2300000                   Retailing
1       2  Berkshire Hathaway    367700                  Financials
2       3              Apple     116000                  Technology
3       4        Exxon Mobil      72700                      Energy
4       5           McKesson      68000                 Wholesalers
..    …                  …        …                          …
495   496       Michaels Cos.     31000                   Retailing
496   497       Toll Brothers      4200  Engineering & Construction
497   498              Yahoo       8500                  Technology
498   499      Vistra Energy       4431                      Energy
499   500     ABM Industries    110000           Business Services

                                   industry         hqcity hqstate  \
0                      General Merchandisers    Bentonville      AR
1     Insurance: Property and Casualty (Stock)         Omaha      NE
2                Computers, Office Equipment      Cupertino      CA
3                          Petroleum Refining         Irving      TX
4                  Wholesalers: Health Care  San Francisco      CA
..                                         …              …     …
495               Specialty Retailers: Other         Irving      TX
496                               Homebuilders        Horsham      PA
497         Internet Services and Retailing      Sunnyvale      CA
498                                     Energy         Dallas      TX
499          Diversified Outsourcing Services       New York      NY

                                     address ticker  revenues  revchange  \
0    702 S.W. Eighth St., Bentonville, AR 72716    WMT    485873        0.8
1             3555 Farnam St., Omaha, NE 68131   BRKA    223604        6.1
2           1 Infinite Loop, Cupertino, CA 95014   AAPL    215639       -7.7
3       5959 Las Colinas Blvd., Irving, TX 75039    XOM    205004      -16.7
4          1 Post St., San Francisco, CA 94104    MCK    192487        6.2
```

```
..                                              ...    ...     ...      ...
495      8000 Bent Branch Dr., Irving, TX 75063    MIK      5197        5.8
496        250 Gibraltar Rd., Horsham, PA 19044    TOL      5170       23.9
497         701 First Ave., Sunnyvale, CA 94089   YHOO      5169        4.0
498            1601 Bryan St., Dallas, TX 75201    VST      5164        4.0
499         1 Liberty Plaza, New York, NY 10006    ABM      5145       -2.8

     profits  prftchange  assets  totshequity
0    13643.0        -7.2  198825      77798.0
1    24074.0         0.0  620854     283001.0
2    45687.0       -14.4  321686     128249.0
3     7840.0       -51.5  330314     167325.0
4     2258.0        53.0   56563       8924.0
..       ...         ...     ...          ...
495    378.2         4.2    2148      -1698.0
496    382.1         5.2    9737       4229.0
497   -214.3         5.2   48083      31049.0
498   -214.3         5.2   15167       6597.0
499     57.2       -25.0    2281        974.0

[500 rows x 15 columns]
```

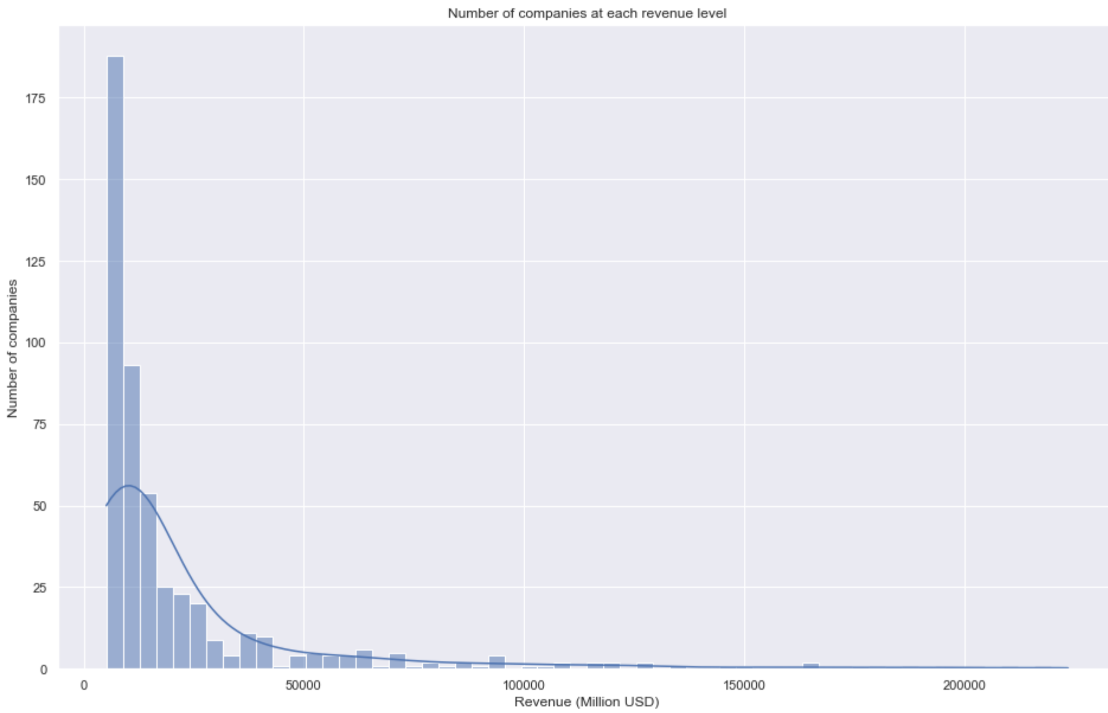
Create a histogram of the revenue data with bars split by sector the company is involved with.

[5]:
```
sns.histplot(data=df, x="revenues", palette='bright', kde=True)
plt.title('Number of companies at each revenue level')
plt.xlabel('Revenue (Million USD)')
plt.ylabel('Number of companies')
plt.text(x=df.revenues[df.revenues==df.revenues.max()], y=1, s='Walmart')
plt.show()
```



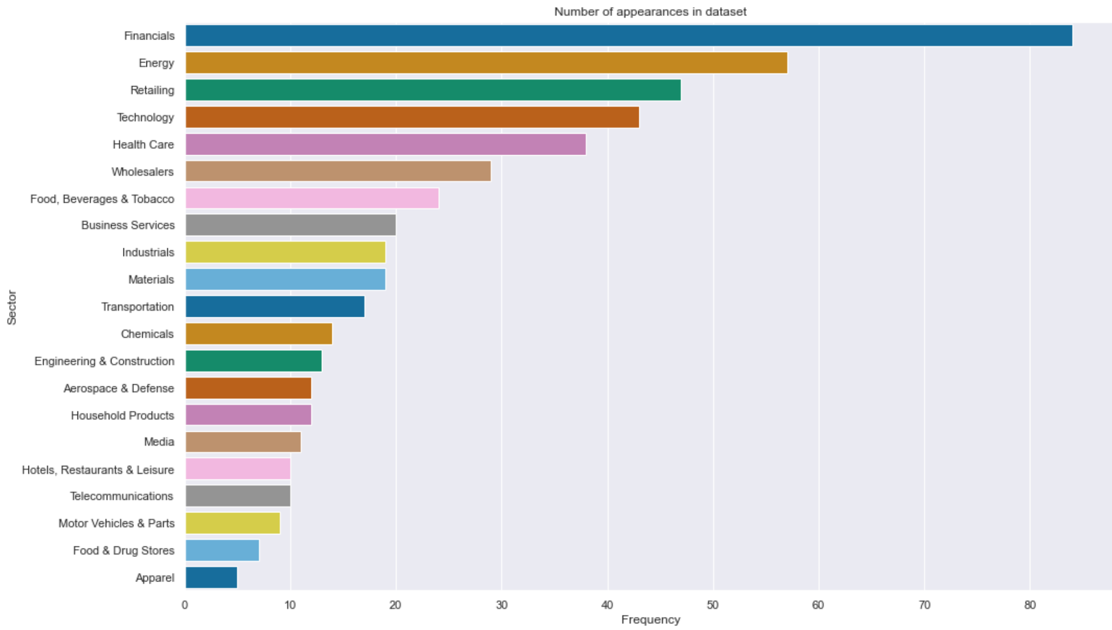
In the previous graph it could be seen that one data point had a far higher revenue than any other in the dataset, this datapoint is the 1st ranked company; Walmart who had a yearly revenue of 485,873 Million USD, compared to 2nd place; Berkshire Hathaway at 223,604 Million USD, a difference of 117.3% to 4 significant figures. In order to get a closer view of the rest of the data I created a new variable 'walmart_removed_df' and assigned it the value of the original dataframe with the first row removed using the pandas.DataFrame.drop function. I then plotted the graph as before.

[6]:
```
walmart_removed_df = df.drop(0)

sns.histplot(data=walmart_removed_df, x="revenues", palette='bright', kde=True,
 ↪stat='count')
plt.title('Number of companies at each revenue level')
plt.xlabel('Revenue (Million USD)')
plt.ylabel('Number of companies')
plt.show()
```



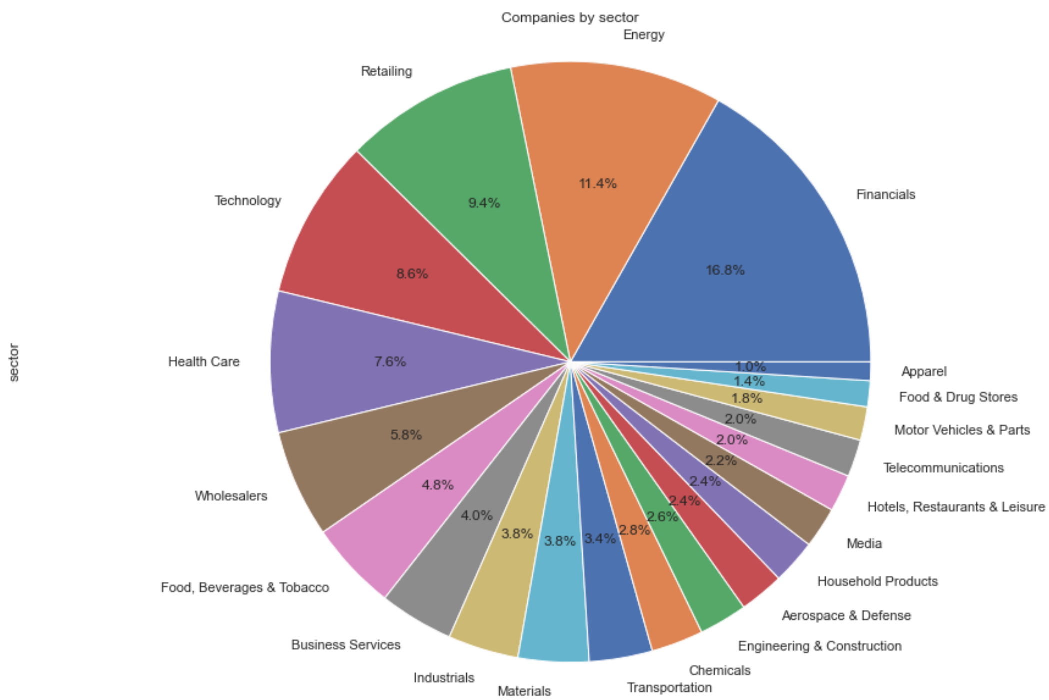
I used a seaborn countplot to plot the number of occurences of each industry sector in the dataset

[7]:
```
sns.countplot(data=df, y='sector', order=df.sector.value_counts().index,␣
 ↪palette='colorblind')
plt.title('Number of appearances in dataset')
plt.xlabel('Frequency')
plt.ylabel('Sector')
plt.show()
```



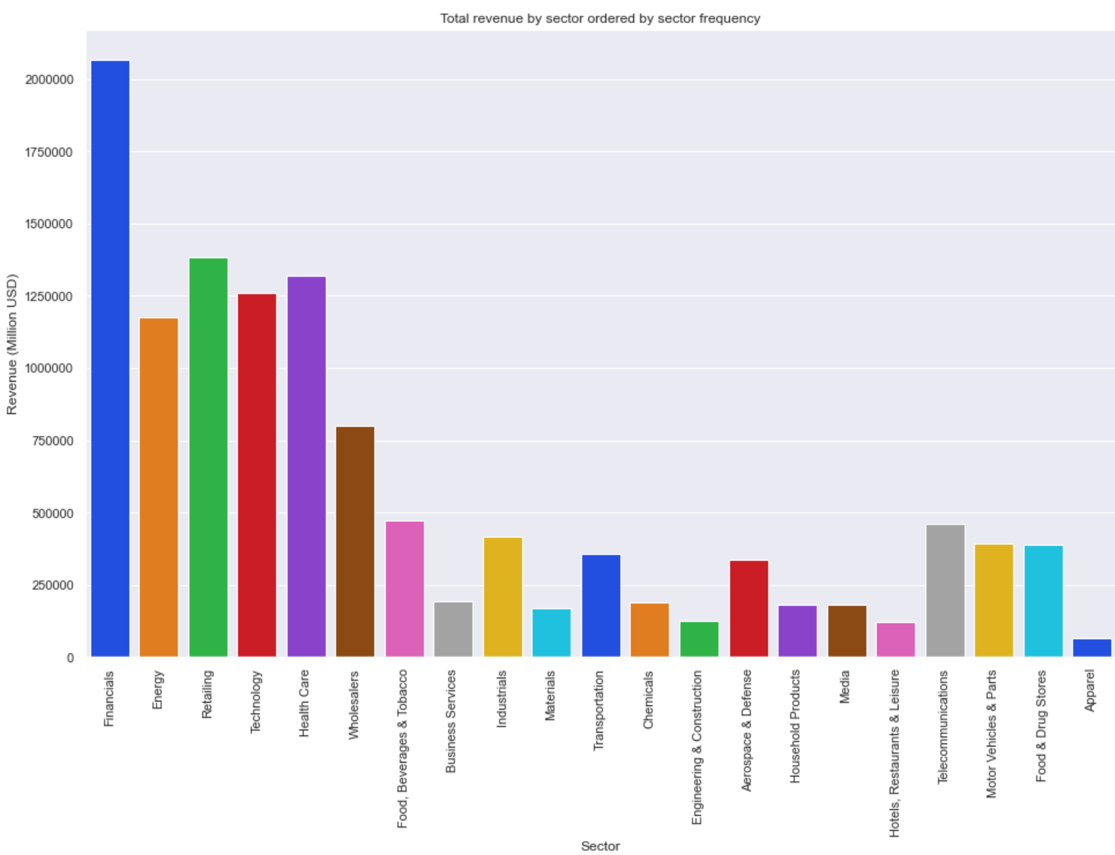
And a pie chart to show the same data as a percentage.

[8]:
```
df.sector.value_counts().plot(kind='pie',autopct='%1.1f%%')
plt.axis('equal')
plt.title("Companies by sector")
plt.show()
```



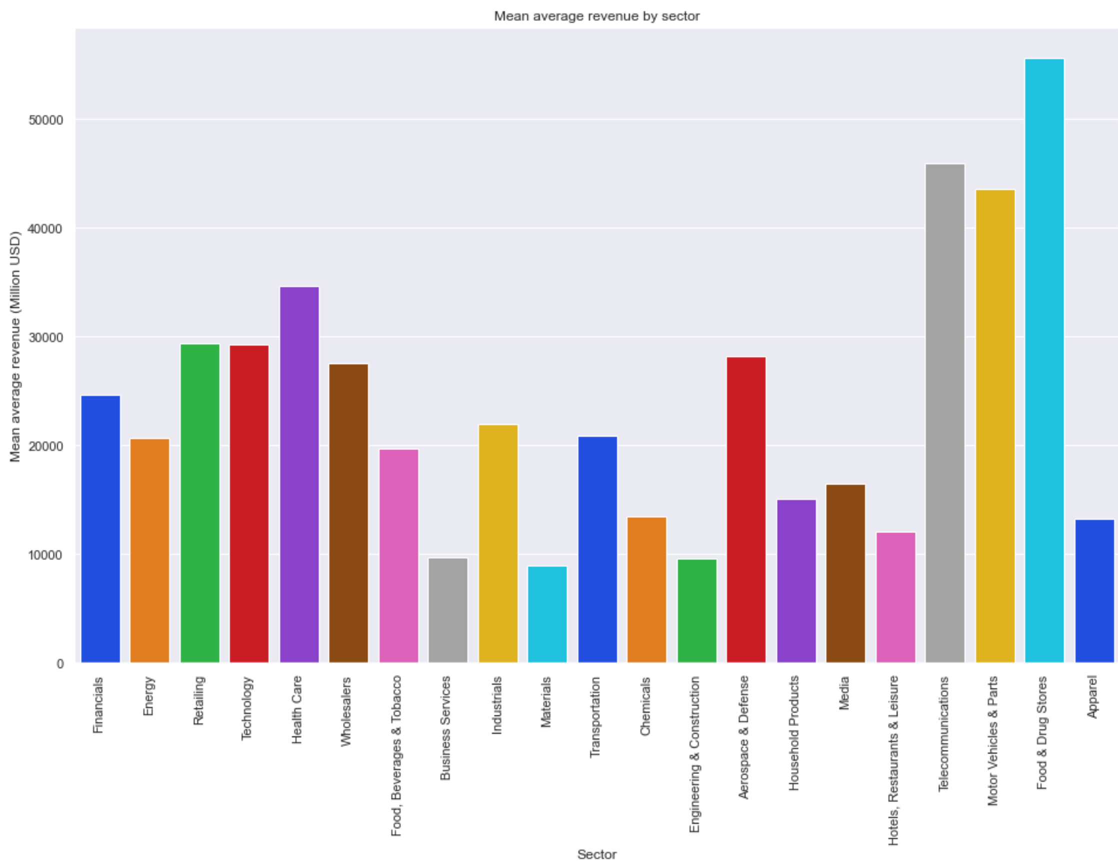
I then used a seaborn barplot to display the total revenue for each sector, ordering it by the frequency in which the sector occurs in the dataset.

[9]:
```
sns.barplot(x="sector", y="revenues", data=df, estimator=sum, palette='bright',␣
↪ci=None,
            order=df.sector.value_counts().index)
plt.xticks(rotation=90)
plt.title("Total revenue by sector ordered by sector frequency")
plt.xlabel("Sector")
plt.ylabel("Revenue (Million USD)")
plt.ticklabel_format(style='plain', axis='y') # Use normal notation instead of␣
↪scientific
plt.show()
```



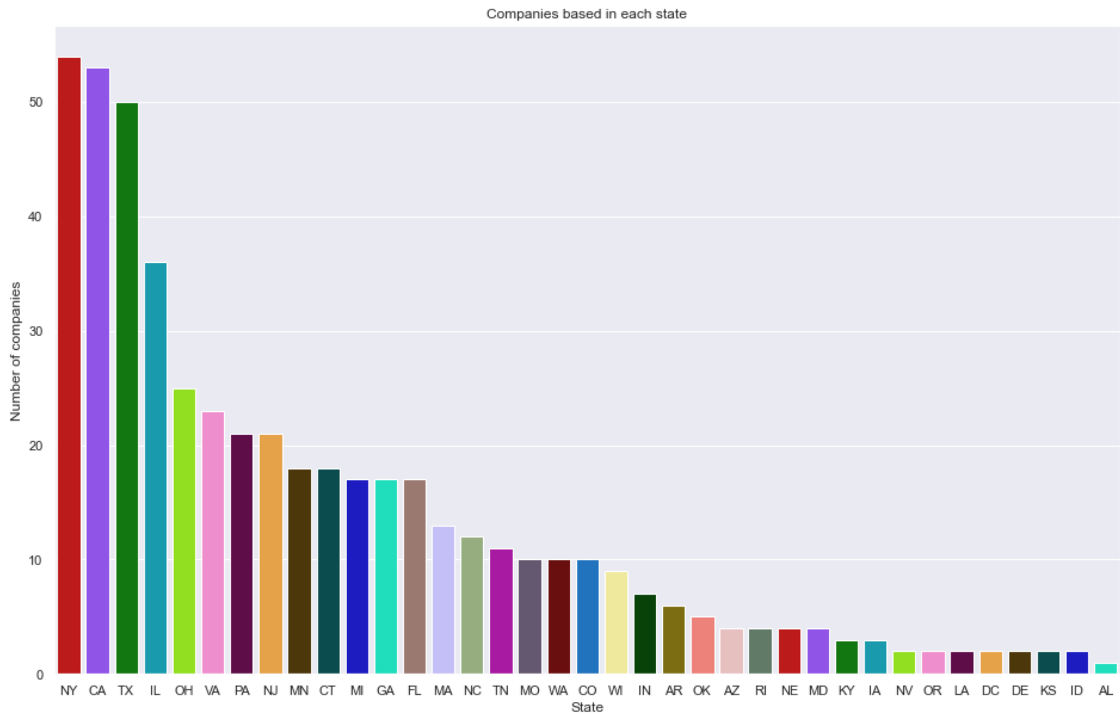
And then the same as above but instead of the total revenue it is the mean revenue for each sector still in the saem oder as before.

```
[10]: sns.barplot(x="sector", y="revenues", data=df, estimator=np.mean,␣
  ↪palette='bright', ci=None,
           order=df.sector.value_counts().index)
plt.xticks(rotation=90)
plt.title("Mean average revenue by sector")
plt.xlabel("Sector")
plt.ylabel("Mean average revenue (Million USD)")
plt.show()
```



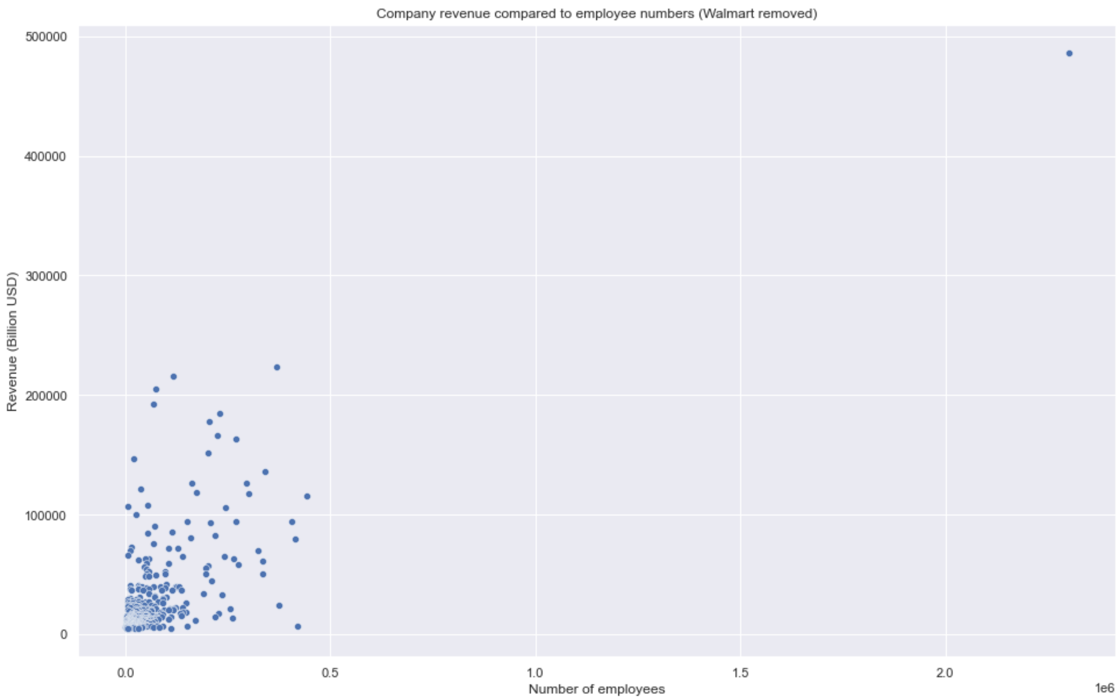
I used another seaborn countplot to count the number of companies with headquarters in each state.

```
[11]: sns.countplot(data=df, x='hqstate', order=df.hqstate.value_counts().index,␣
  ↪palette=palette, dodge=False)
plt.title("Companies based in each state")
plt.xlabel("State")
plt.ylabel("Number of companies")
plt.show()
```

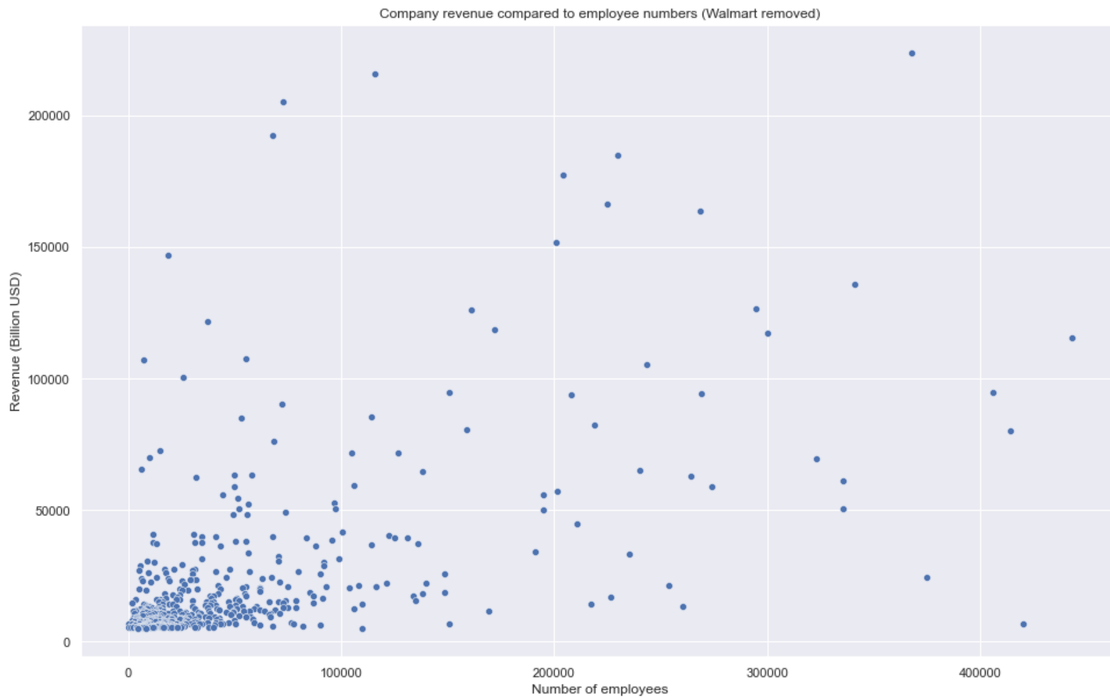


Using a seaborn scatterplot to graph the company revenue compared to the number of employees. The employee numbers of Walmart compared to the company that employees the 2nd most people is huge; 2,300,000 compared to Kroger with 443,000. This difference is so huge it dramatically reduces the usefulness of the graph so another one is made using the datafram with walmart removed that was prepared earlier.

[12]:
```
sns.scatterplot(data=df, x='employees', y='revenues', palette='bright')
plt.title("Company revenue compared to employee numbers (Walmart removed)")
plt.xlabel("Number of employees")
plt.ylabel("Revenue (Billion USD)")
plt.ticklabel_format(style='plain', axis='y')
plt.show()
```



[13]:
```
sns.scatterplot(data=walmart_removed_df, x='employees', y='revenues',␣
 ↪palette='bright')
plt.title("Company revenue compared to employee numbers (Walmart removed)")
plt.xlabel("Number of employees")
plt.ylabel("Revenue (Billion USD)")
plt.ticklabel_format(style='plain', axis='y')
plt.show()
```



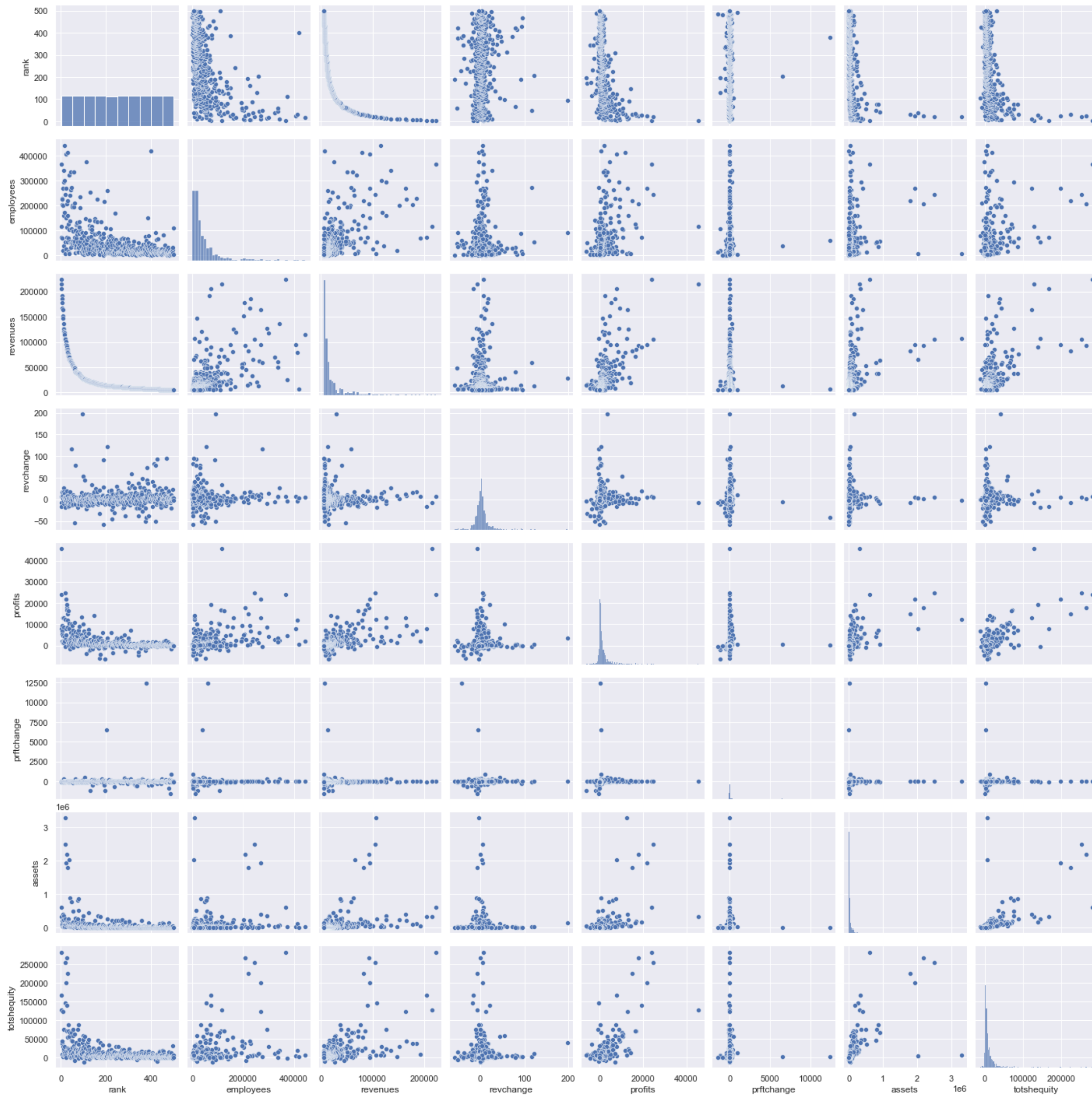
[14]:
```
df.sort_values('employees', ascending=False)
```

[14]:
```
     rank                  title  employees                          sector  \
0       1                Walmart    2300000                       Retailing
17     18                 Kroger     443000              Food & Drug Stores
398   399      Yum China Holdings     420000  Hotels, Restaurants & Leisure
31     32                    IBM     414400                      Technology
22     23             Home Depot     406000                       Retailing
..    …                      …        …                             …
333   334        Global Partners       1770                     Wholesalers
188   189           INTL FCStone       1464                      Financials
479   480      Delek US Holdings       1326                          Energy
471   472    Host Hotels & Resorts       220                      Financials
394   395   A-Mark Precious Metals        83                       Materials

                              industry        hqcity hqstate  \
0                 General Merchandisers   Bentonville      AR
17                 Food and Drug Stores    Cincinnati      OH
398                       Food Services         Plano      TX
31     Information Technology Services        Armonk      NY
22          Specialty Retailers: Other       Atlanta      GA
..                                   …             …       …
333           Wholesalers: Diversified       Waltham      MA
188            Diversified Financials      New York      NY
```

```
479              Petroleum Refining     Brentwood      TN
471                     Real Estate      Bethesda      MD
394                   Miscellaneous  Santa Monica      CA

                                           address ticker  revenues  \
0        702 S.W. Eighth St., Bentonville, AR 72716    WMT    485873
17              1014 Vine St., Cincinnati, OH 45202     KR    115337
398             7100 Corporate Dr., Plano, TX 75024   YUMC      6752
31              1 New Orchard Rd., Armonk, NY 10504    IBM     79919
22          2455 Paces Ferry Rd., Atlanta, GA 30339     HD     94595
..                                              …      …        …
333                800 South St., Waltham, MA 02453    GLP      8240
188              708 Third Ave., New York, NY 10017   INTL     14755
479           7102 Commerce Way, Brentwood, TN 37027     DK      5414
471           6903 Rockledge Dr., Bethesda, MD 20817    HST      5488
394  429 Santa Monica Blvd., Santa Monica, CA 90401   AMRK      6784

     revchange  profits  prftchange  assets  totshequity
0          0.8  13643.0        -7.2  198825      77798.0
17         5.0   1975.0        -3.1   36505       6698.0
398        4.0    502.0       -13.4    3727       2377.0
31        -3.1  11872.0       -10.0  117470      18246.0
22         6.9   7957.0        13.5   42966       4333.0
..         …      …          …      …           …
333      -20.1   -199.4      -557.8    2564        393.0
188      -57.5     54.7        -1.8    5951        434.0
479       -6.0   -153.7      -892.3    2985        992.0
471        1.9    762.0        36.6   11408       6994.0
394       11.8      9.3        31.5     437         63.0

[500 rows x 15 columns]
```

[15]:

```
#sns.pairplot(df)
sns.pairplot(walmart_removed_df, diag_kind='hist')
plt.show()
```



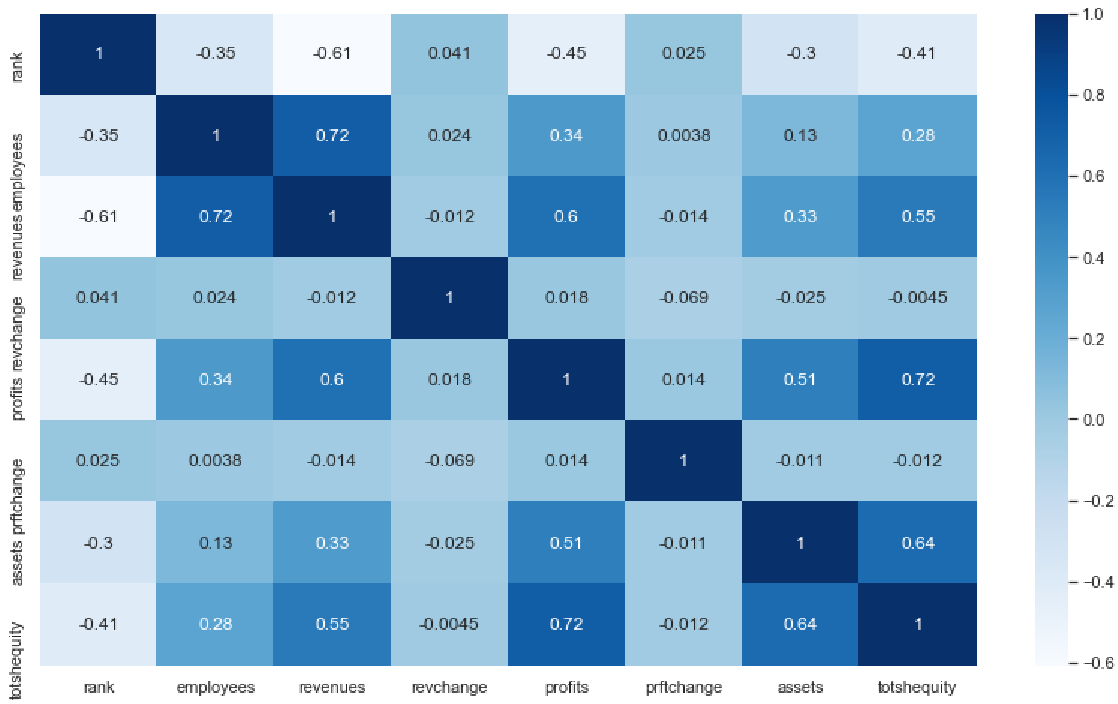
1.2 Machine learning methods

I chose to perform regression on this dataset to use as prediction, so I selected the variables I was going to use as 'x' and the variable I was going to try and predict as 'y'. I chose profits as y and for 'x' I chose to use employees, revenues, assets and totshequity as from the heatmap I could see that these were the 4 factors that had the largest impact on profits.

[241]:

```
plt.figure(figsize=(14,8))
corr = df.corr()
heatmap = sns.heatmap(corr, annot=True, cmap="Blues")
plt.show()
```

```python
x = df[['employees', 'revenues', 'assets', 'totshequity', 'revchange',␣
 ↪'prftchange']]
y = df['profits']

print(x)
print('########################################################')
print(y)
```




```
     employees  revenues  assets  totshequity  revchange  prftchange
0      2300000    485873  198825      77798.0        0.8        -7.2
1       367700    223604  620854     283001.0        6.1         0.0
2       116000    215639  321686     128249.0       -7.7       -14.4
3        72700    205004  330314     167325.0      -16.7       -51.5
4        68000    192487   56563       8924.0        6.2        53.0
..         …         …      …           …          …         …
495      31000      5197    2148      -1698.0        5.8         4.2
496       4200      5170    9737       4229.0       23.9         5.2
497       8500      5169   48083      31049.0        4.0         5.2
498       4431      5164   15167       6597.0        4.0         5.2
499     110000      5145    2281        974.0       -2.8       -25.0

[500 rows x 6 columns]
########################################################
0      13643.0
```

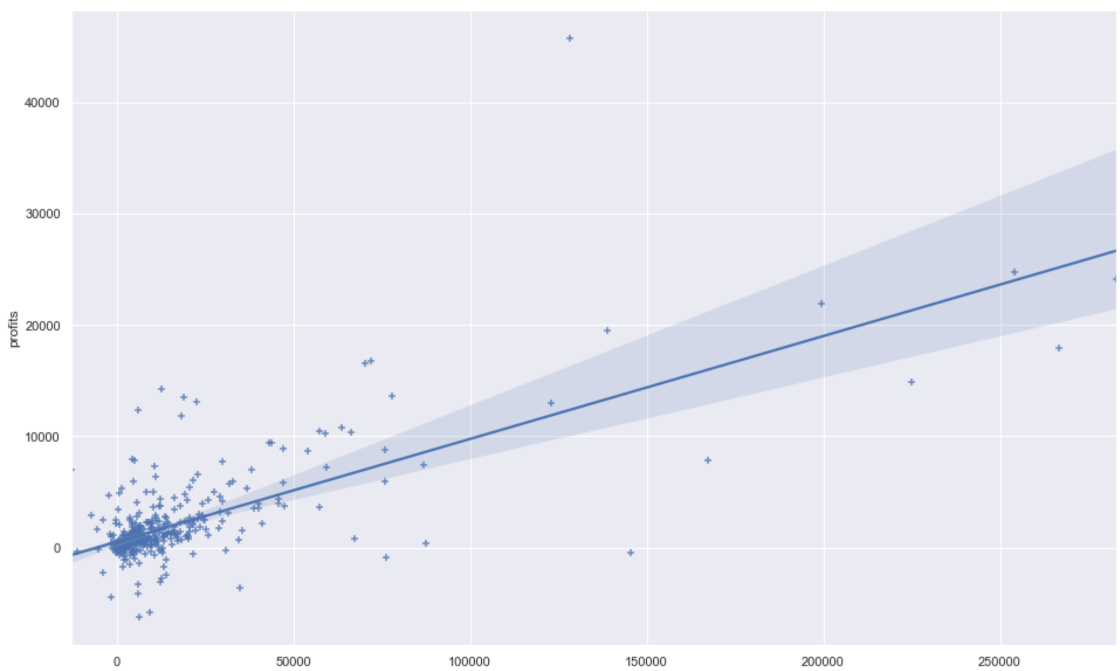
```
1        24074.0
2        45687.0
3         7840.0
4         2258.0
          …
495        378.2
496        382.1
497       -214.3
498       -214.3
499         57.2
Name: profits, Length: 500, dtype: float64
```

1.2.1 Multiple linear regression

Regression models describe the relationship between variables, linear regression models use a straight line. Multiple linear regression attempts to model the relationship between two or more explanatory variables and a response variable by fitting a linear equation to observed data. The dependent variable must be a continuous vlaue, e.g. profits in this case. The independent variables may be either continuous or binary, in this case all are continuous.

Singular linear regression can also be easily plotted using the seaborn regplot function which plots the data given to it along with a linear regression fit. Plotting profits against the single factor that has the highest impact, which from the heatmap we can see is totshequity, we get the following graph showing the relationship between the 2 fetaures and a fit line that can be used to read predictions from the graph.

[263]:
```
sns.regplot(y=df.profits, x=df.totshequity, marker="+")
plt.show()
```



Multiple linear regression requires more work which is done as follows. Split the data into training and test data using scikit-learn allowing control over the size of the test data. I chose to use a split with the test data being 30% of the dataset and the other 70% used for training.

[264]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3)

print(x_train)
print('350 rows')
print('############################################################')
print(x_test)
print('150 rows')
```

```
     employees  revenues  assets  totshequity  revchange  prftchange
486       6600      5369    2164        781.0        0.1        82.5
431      67800      6063    6908       1224.0        5.8        25.8
41       58000     63476  898764      67309.0       -9.3       -84.9
82       88000     36556   40140       8659.0       -3.5       -69.2
471        220      5488   11408       6994.0        1.9        36.6
..         …         …       …           …          …          …
433      18000      6004   12578       4578.0        8.4       -16.6
462      40000      5591    2575        -50.0        9.0        43.5
111     375000     24622   31024      -2204.0       -3.1         3.5
443      26400      5853    3419       -848.0      -18.4        85.7
74       50000     38308  125592      20366.0       -2.9        95.7

[350 rows x 6 columns]
350 rows
############################################################
     employees  revenues  assets  totshequity  revchange  prftchange
358       8900      7625    4094       1292.0       15.6        -5.6
34      126400     71890  141208      70418.0        2.6         7.3
421      90000      6366    5478      -5656.0      -51.4        25.2
265       6700     10782  214235      12994.0       -4.9      -204.8
255      11476     11107   41155      11021.0        0.7        14.1
..         …         …       …           …          …          …
498       4431      5164   15167       6597.0        4.0         5.2
291      57500      9568    8315       2916.0       -1.7        -2.9
219      59100     12574    8170       -729.0        3.5        -7.6
108      90000     25923   61538      25161.0      -12.5       -77.2
354      28100      7651    5354       3339.0        0.2       -36.2

[150 rows x 6 columns]
150 rows
```

Fit the scikit-learn linear regression model to the data.

[265]:
```
LinReg = LinearRegression()
LinReg.fit(x_train, y_train)
```

[265]:
```
LinearRegression()
```

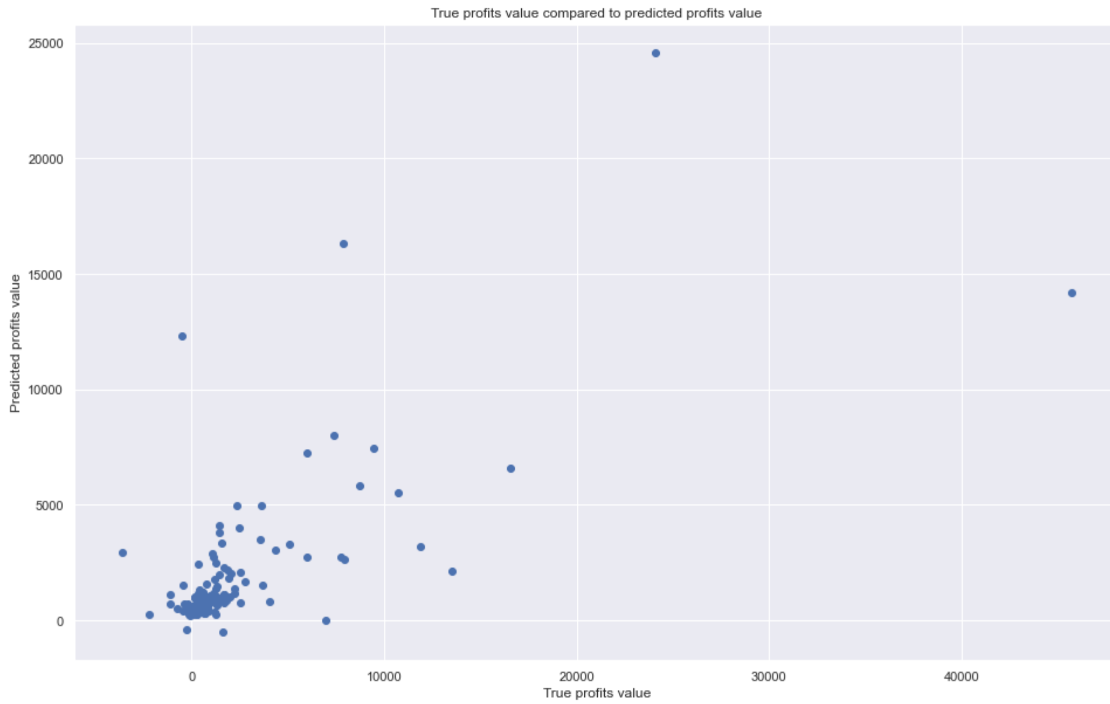
Use the trained model to make predictions usinng scikit-learn.predict() method on the x_test variable then compare the predictions to the y_test variable to evaluate model's performance.

This can be done by either plotting the predicted values of against the true values or by plotting a histogram of the difference between the two values or on a line graph with one line for the predicted and one line for the actual data.

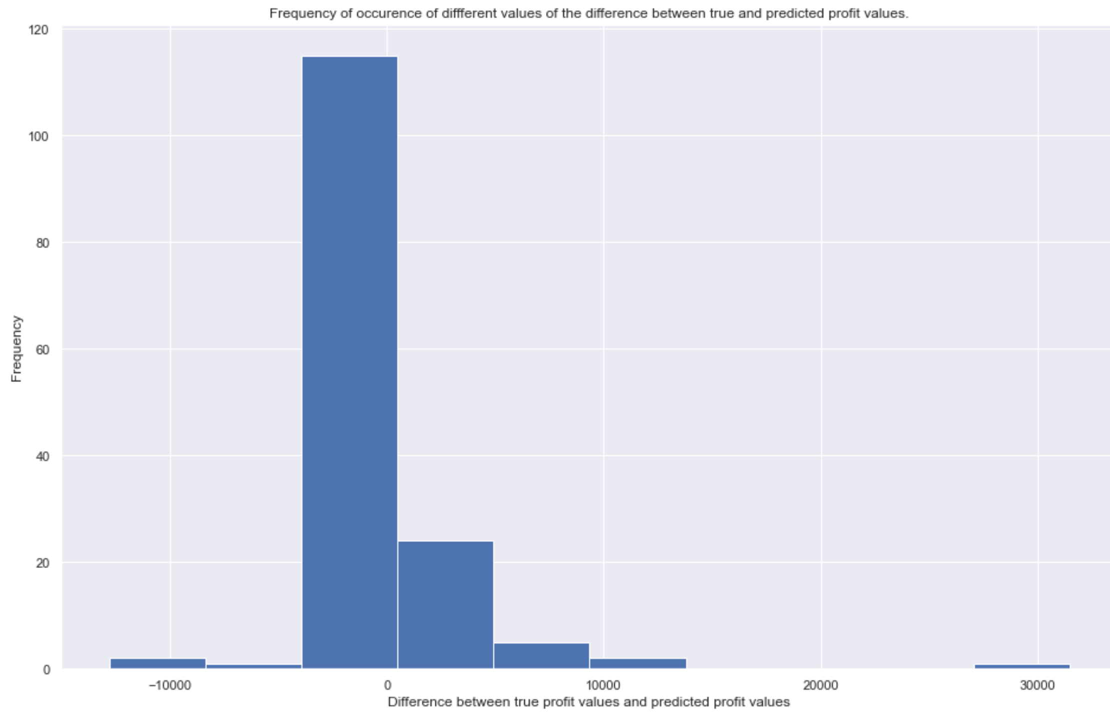
[266]:
```
predictions = LinReg.predict(x_test)

pd.DataFrame(LinReg.coef_, x.columns, columns = ['Coefficient']) # large␣
↪coefficients indicate that
                                                                # the␣
↪particular variable impacts the prediction signifcantly
```

[266]:
```
              Coefficient
employees       -0.000695
revenues         0.025887
assets           0.001733
totshequity      0.062983
revchange        6.629733
prftchange       0.161249
```

[267]:
```
plt.scatter(y_test, predictions)
plt.title('True profits value compared to predicted profits value')
plt.xlabel('True profits value')
plt.ylabel('Predicted profits value')
plt.show()
```

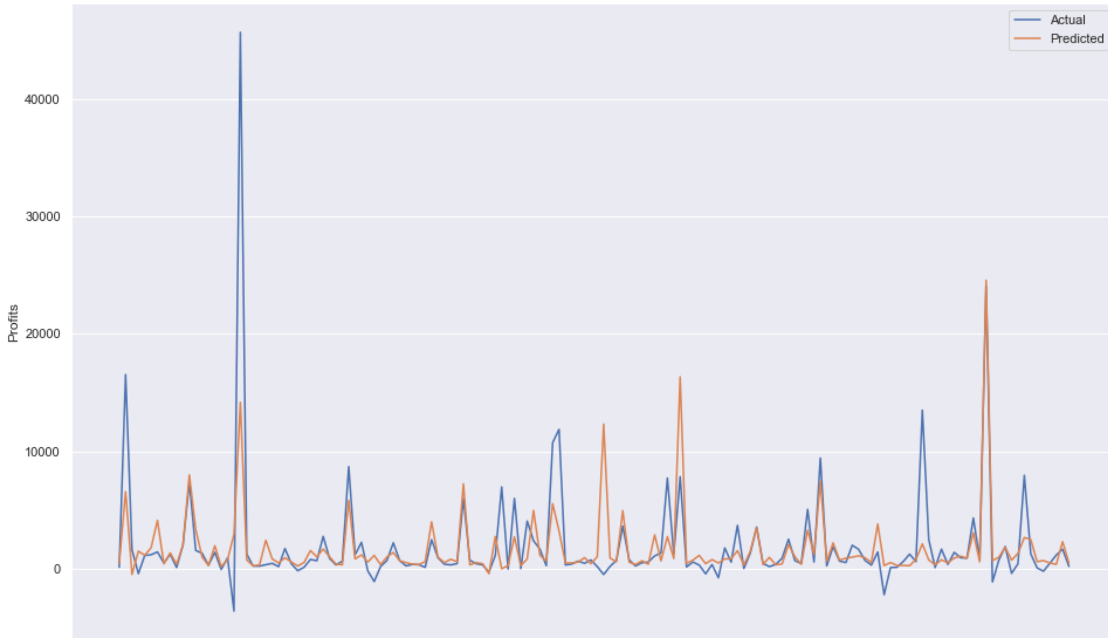


[268]:
```
plt.hist(y_test - predictions)
plt.title('Frequency of occurence of diffferent values of the difference␣
↪between true and predicted profit values.')
plt.xlabel('Difference between true profit values and predicted profit values')
plt.ylabel('Frequency')
plt.show()
```



[269]:

```
pred_df = pd.DataFrame(predictions)
x_df = pd.DataFrame(x_test.iloc[:,0])

plt.plot(range(len(y_test)), y_test)
plt.plot(range(len(predictions)), predictions)
plt.ylabel('Profits')
plt.legend(['Actual', 'Predicted'])
plt.xticks(())

plt.show()
```



From these graphs it can be seen that the predicted value is often close to the actual value, however there are some values which have less accurate predictions and usually a small number that have very inaccurate predictions.

Evaluate the model using either the mean absolute error (MAE) or the mean squared error (MSE).

MAE captures the average absolute difference between predictions and true values.

MSE encapsulates the variance.

MSE is differentiable, unlike MAE, however, MAE is less sensitive to outliers.

[270]:
```
mae = metrics.mean_absolute_error(y_test, predictions)
mse = metrics.mean_squared_error(y_test, predictions)
print('Mean absolute error = ',(mae))
print('###############################################')
print('Mean squared error = ',(mse))
print('###############################################')
r2_linreg = metrics.r2_score(y_test, predictions)
print('R2 score = ',(r2_linreg))
print('###############################################')
```

```
Mean absolute error =  1358.5135112556482
###############################################
Mean squared error =  12266589.642697483
###############################################
R2 score =  0.48337490563151286
###############################################
```

From these values it can be seen that the average difference between the predicted and true profit values was the value of mae which varied each time the code is run but it usually between the values of 1300 - 1550 (Million USD). The MSE however is usually very large indicating a large variance in results, this could be in part due to few predictions that were very wrong as MSE is highly sensitive to any outliers. The R2 score shows how much of the data fits the regression.

[271]:
```
testing = x.drop(df.index[1:])
real = y.drop(df.index[1:])
```

[272]:
```
print(testing)
print(int(real))
```

```
   employees  revenues  assets  totshequity  revchange  prftchange
0    2300000    485873  198825      77798.0        0.8        -7.2
13643
```

[273]:
```
walmartpred = LinReg.predict(testing)
```

[274]:
```
walmartpred
```

[274]:
```
array([16321.36781824])
```

[275]:
```
int(walmartpred) - int(real)
```

[275]:
```
2678
```

1.2.2 Lasso regression

Using the LassoCV method to find the optimum lambda value.

[276]:
```
lasso_cv_model = LassoCV(alphas = np.random.randint(0,1000,100), cv = 10,␣
 ↪max_iter = 100000).fit(x_train,y_train)
```

Find the alpha value of the Lasso model established with Cross-Validation.

[277]:
```
lasso_cv_model.alpha_
```

[277]:
```
412
```

Fit the Corrected Lasso model with this optimum alpha value. Then print the predicted values over the test set to y_pred_tuned.

[278]:
```
lasso_tuned = Lasso().set_params(alpha = lasso_cv_model.alpha_).
 ↪fit(x_train,y_train)

y_pred_tuned = lasso_tuned.predict(x_test)

print(y_pred_tuned)
```

```
[ 4.65559008e+02  6.56698312e+03 -4.40576378e+02  1.49958761e+03
  1.14804732e+03  1.79025844e+03  4.11503654e+03  4.20234656e+02
```

```
 1.35117924e+03  3.92796319e+02  2.01318678e+03  7.99334923e+03
 3.35168580e+03  1.00501960e+03  2.70597737e+02  1.96411248e+03
 1.81056044e+02  7.55820200e+02  2.95539606e+03  1.41858044e+04
 7.49613211e+02  2.58483913e+02  3.26462881e+02  2.42178366e+03
 8.34830235e+02  4.93308562e+02  9.19476846e+02  5.75187816e+02
 2.40728863e+02  5.16333389e+02  1.53246132e+03  1.01361264e+03
 1.68969222e+03  1.00727635e+03  3.99562070e+02  3.36973847e+02
 5.82503094e+03  8.28140402e+02  1.17871148e+03  5.69531515e+02
 1.14625823e+03  3.46264122e+02  9.99461135e+02  1.38345090e+03
 6.84435878e+02  4.94821498e+02  3.77527720e+02  3.61765257e+02
 5.93749465e+02  3.99058259e+03  9.97447722e+02  5.17790726e+02
 7.67737540e+02  6.00323830e+02  7.24233187e+03  3.30161979e+02
 5.21396994e+02  4.37031752e+02 -4.24142253e+02  2.74068133e+03
-5.68251885e+00  2.69865569e+02  2.71117567e+03  2.66989284e+02
 8.23456076e+02  4.94237194e+03  1.14263484e+03  6.60661744e+02
 5.54132345e+03  3.21199428e+03  4.85014699e+02  4.97707163e+02
 5.70884779e+02  9.22028651e+02  4.31270427e+02  1.02665989e+03
 1.23414949e+04  9.12115491e+02  6.07178884e+02  4.90790872e+03
 5.48091271e+02  3.67091506e+02  6.78781997e+02  3.89313263e+02
 2.89501146e+03  6.85842848e+02  2.73204782e+03  8.64339179e+02
 1.63599337e+04  4.58181750e+02  7.10411801e+02  1.11767801e+03
 4.36099916e+02  7.71912048e+02  5.01018574e+02  8.34880081e+02
 8.52850195e+02  1.53314745e+03  3.15168904e+02  1.46778195e+03
 3.49676211e+03  3.61367076e+02  9.47776841e+02  3.51212751e+02
 4.10797471e+02  2.08150546e+03  9.94071412e+02  3.80994344e+02
 3.29045552e+03  1.18683441e+03  7.44388933e+03  6.29413192e+02
 2.19536822e+03  7.38930326e+02  9.10569563e+02  9.73674081e+02
 1.08667118e+03  9.37598813e+02  5.37752989e+02  3.79662853e+03
 2.79593297e+02  5.20437775e+02  2.89618103e+02  2.83798177e+02
 2.54569147e+02  7.69522571e+02  2.12139266e+03  8.00638897e+02
 3.31862137e+02  7.66477997e+02  4.94596818e+02  9.22029550e+02
 1.06616327e+03  9.01198570e+02  3.06444408e+03  6.01535382e+02
 2.45696666e+04  6.79629987e+02  9.99697896e+02  1.81096718e+03
 7.27760381e+02  1.32029455e+03  2.65330511e+03  2.49820559e+03
 5.95996574e+02  6.93105373e+02  4.92443620e+02  3.68460328e+02
 2.31274285e+03  4.90338801e+02]
```

Print the correlation values as a dataframe.

[279]:
```
# large coefficients indicate that the particular variable impacts the␣
↪prediction signifcantly

pd.DataFrame(lasso_tuned.coef_, x.columns, columns = ['Coefficient'])
```

[279]:
```
           Coefficient
employees    -0.000689
revenues      0.025866
assets        0.001730
```

```
totshequity     0.063015
revchange       5.778685
prftchange      0.158495
```

Calculate MAE, MSE and R2 scores.

[280]:
```
lasso_MSE = metrics.mean_squared_error(y_test,y_pred_tuned)
lasso_MAE = metrics.mean_absolute_error(y_test, y_pred_tuned)
r2_lasso = metrics.r2_score(y_test, y_pred_tuned)

print('Mean absolute error = ',(lasso_MAE))
print('##############################################')
print('Mean squared error = ',(lasso_MSE))
print('##############################################')
print('R2 score = ',(r2_lasso))
print('##############################################')
```

```
Mean absolute error =  1357.357997981189
##############################################
Mean squared error =  12264230.77818813
##############################################
R2 score =  0.48347425260856547
##############################################
```

Calculate difference between MAE, MSE and R2 scores for multiple linear regression and lasso regression.

[281]:
```
mae_diff = mae - lasso_MAE
mse_diff = mse - lasso_MSE
r2_diff = r2_linreg - r2_lasso

print('Mean absolute error difference = ',(mae_diff))
print('##############################################')
print('Mean squared error difference = ',(mse_diff))
print('##############################################')
print('R2 score difference = ',(r2_diff))
print('##############################################')
```

```
Mean absolute error difference =  1.1555132744590537
##############################################
Mean squared error difference =  2358.864509353414
##############################################
R2 score difference =  -9.934697705260565e-05
##############################################
```

[282]:
```
walmartpred_lasso = lasso_tuned.predict(testing)
```

[283]:
```
int(walmartpred_lasso) - int(real)
```

[283]: 2687

```
[ ]:
```