CAPSTONE PROJECT 2:

Part I – Descriptive Statistics and EDA:

Cold Storage started its operations in Jan 2016. They are in the business of storing Pasteurized Fresh Whole or Skimmed Milk, Sweet Cream, Flavoured Milk Drinks. To ensure that there is no change of texture, body appearance, separation of fats the optimal temperature to be maintained is between 2 º - 4 º Centigrade. In the first year of business, they outsourced the plant maintenance work to a professional company with stiff penalty clauses. Average temperature data at the date level is given in the file "Cold_Storage_Temp_Data_.csv".

Tasks/ Questions to be Answered:

Data Summary:

1. Read the data set, check shape and info, and get familiar with the data.

- Import necessary libraries & packages
- Load Data set
- Check necessary details about data like shape, data types of variable, missing values etc

Data = pd.read_csv('cold_storage.csv') # Import the dataset named 'Admission_predict.csv'

Data.head()

	Season	Month	Date	Temperature
0	Winter	Jan	1	2.3
1	Winter	Jan	2	2.2
2	Winter	Jan	3	2.4
3	Winter	Jan	4	2.8
4	Winter	Jan	5	2.5

Data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 365 entries, 0 to 364
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Season       365 non-null    object
 1   Month        365 non-null    object
 2   Date         365 non-null    int64
 3   Temperature  365 non-null    float64
dtypes: float64(1), int64(1), object(2)
memory usage: 11.5+ KB
```

Data.shape

```
(365, 4)
```

2. Check the summary statistics of the data-frame and comment on your findings.

```
Data.describe()
```

	Date	Temperature
count	365.000000	365.000000
mean	15.720548	3.002466
std	8.808321	0.465832
min	1.000000	1.700000
25%	8.000000	2.700000
50%	16.000000	3.000000
75%	23.000000	3.300000
max	31.000000	4.500000

From the above data we can see mean of our temperatures which is three, this is also something we can see from our data and our maximum temperature is 4.5 C and the minimum temperature is 1.7 C. The total count of all of this is 365 days and temperature is captured for each of these days

3.Check for duplicates, unique and null values and clean the data using appropriate values. Provide comments on your approach for data imputation.

No Duplicate values.

```
dupes = Data.duplicated()
sum(dupes)
```

```
0
```

```
#no duplicates
```

```
pd.DataFrame( Data.isnull().sum(), columns= ['Number of missing values'])
```

	Number of missing values
Season	0
Month	0
Date	0
Temperature	0

No missing values, No Duplicate values,
are present so no imputations are required. We have also figured all the unique value(in the HTML)

Descriptive Statistics:

4. Find mean cold storage temperature for Summer, Winter, and Rainy Season. (hint: use appropriate plot)

```
Data.groupby('Season').mean()
```

Season	Date	Temperature
Rainy	15.754098	3.087705
Summer	15.525000	3.147500
Winter	15.878049	2.776423

Well as we can see we can find the mean without plotting a graph.

5. Find the overall mean temperature for the full year.
6. Find Standard Deviation of temperature for the full year.

```
print("Data:",Data.Temperature.mean())
```

```
Data: 3.0024657534246546
```

```
print(Data.Temperature.std())
```

```
0.4658319416510761
```

7. Check for distribution,

a. Assuming Normal distribution, what is the probability of temperature having fallen below 2º C?

```
z1=(2-3.0024657534246546)/0.4658319416510761
```

```
z1
```

```
-2.151990157376403
```

```
stats.norm.cdf(-2.151990157376403)
```

```
0.015699064791364483
```

b. Assume Normal distribution, what is the probability of temperature having gone above 4º C?

```
z2=(4-3.0024657534246546)/0.4658319416510761
```

```
z2
```

```
2.141403706752536
```

```
1 - stats.norm.cdf(2.14)
```

```
0.01617738337216612
```

8. What will be the penalty for the AMC Company? (Hint: Total probability of temperature being below 2 degree or above 4 degree)

```
Result: 0.031876448163530 61
```

<u>Part II – Inferential Statistics:</u>

Assume 3.9º C as the upper acceptable mean temperature and at alpha = 0.1 do you feel that there is a need f or some corrective action in the Cold Storage Plant or is it that the problem is from the procurement side from where Cold Storage is getting the Dairy Products. The data of the last 35 days is in "Cold_Storage_Mar2018_.cs v"

<u>Tasks/ Questions to be Answered:</u>

1. Read the data set, check shape and info, and get familiar with the data.

- Import necessary libraries & packages
- Load Data set
- Check necessary details about data like shape, data types of variable, missing values etc

```
data  = pd.read_csv('Cold_Storage_Mar2018_.csv')
data.head()
```

	Season	Month	Date	Temperature
0	Summer	Feb	11	4.0
1	Summer	Feb	12	3.9
2	Summer	Feb	13	3.9
3	Summer	Feb	14	4.0
4	Summer	Feb	15	3.8

For summer season cold storage temperature is measured and taken for 35 days.

```
data.shape
```

```
(35, 4)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35 entries, 0 to 34
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Season        35 non-null     object
 1   Month         35 non-null     object
 2   Date          35 non-null     int64
 3    Temperature  35 non-null     float64
dtypes: float64(1), int64(1), object(2)
memory usage: 1.2+ KB
```

2.)Check the summary statistics of the data-frame and comment on your findings.

```
data.describe()
```

	Date	Temperature
count	35.000000	35.000000
mean	14.400000	3.974286
std	7.389181	0.159674
min	1.000000	3.800000
25%	9.500000	3.900000
50%	14.000000	3.900000
75%	19.500000	4.100000
max	28.000000	4.600000

The mean temperature is 3.9 and the maximum temperature is 4.6 cold-storage temperature.Max date of temperature count is 28.

3.)Which Hypothesis test shall be performed to check if corrective action is needed at the cold storage plant? Justify your answer. (6 marks) (Descriptive)

We use the T - hypothesis testing, is, in one sample test, we compare the population parameter such as m ean of a single sample of data collected from a single population. We have evidence to reject the null hy pothesis since p value < Level of significance
Our one-sample t-test p-value= 0.009422395404264431.

4. Perform the Hypothesis Testing

a. State the Hypothesis (2 Marks)
   We will be performing T-testing for the hypothesis test. Reason as given above. A t-test is used to compare the mean of two given samples.

b. Perform necessary calculations to accept or reject the corresponding null hypothesis. (6 marks)

```
# one sample t-test

t_statistic, p_value = ttest_1samp(data.Temperature,3.9 )

print('One sample t test \nt statistic: {0} p value: {1} '.format(t_statistic, p_value))
```

```
One sample t test
t statistic: 2.752358609800241 p value: 0.009422395404264431
```

```
# p_value < 0.01 => alternative hypothesis:

alpha_value = 0.01 # Level of significance

print('Level of significance: %.2f' %alpha_value)

if p_value < alpha_value:

  print('We have evidence to reject the null hypothesis since p value < Level of significance')

else:

  print('We have no evidence to reject the null hypothesis since p value > Level of significance')

print ("Our one-sample t-test p-value=", p_value)
```

```
Level of significance: 0.01
We have evidence to reject the null hypothesis since p value < Level of
significance
Our one-sample t-test p-value= 0.009422395404264431
```

Part II – Inferential Statistics:

You are a part of an investment firm and your work is to do research about these 759 firms. You are provided with the dataset containing the sales and other attributes of these 759 firms. Predict the sales of these firms on the bases of the details given in the dataset so as to help your company in investing consciously. Also, provide them with 5 attributes that are most important.

Tasks/ Questions to be Answered:

Data Summary & Exploratory Data Analytics:

1. Read the data set, check shape and info, and get familiar with the data
   Import necessary libraries & packages
   Load Data set
   Check necessary details about data like shape, data types of variable, missing values etc.

   mydata = pd.read_csv('Firm_Level_Data.csv')

   mydata.head()

	Unnamed: 0	sales	capital	patents	randd	employment	sp500	tobinq	value	institutions
0	0	826.995050	161.603986	10	382.078247	2.306000	no	11.049511	1625.453755	80.27
1	1	407.753973	122.101012	2	0.000000	1.860000	no	0.844187	243.117082	59.02
2	2	8407.845588	6221.144614	138	3296.700439	49.659005	yes	5.205257	25865.233800	47.70
3	3	451.000010	266.899987	1	83.540161	3.071000	no	0.305221	63.024630	26.88
4	4	174.927981	140.124004	2	14.233637	1.947000	no	1.063300	67.406408	49.46

```
mydata.shape
```

```
(759, 10)
```

click to scroll output; double click to hide

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 759 entries, 0 to 758
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Unnamed: 0    759 non-null    int64
 1   sales         759 non-null    float64
 2   capital       759 non-null    float64
 3   patents       759 non-null    int64
 4   randd         759 non-null    float64
 5   employment    759 non-null    float64
 6   sp500         759 non-null    object
 7   tobinq        738 non-null    float64
 8   value         759 non-null    float64
 9   institutions  759 non-null    float64
dtypes: float64(7), int64(2), object(1)
memory usage: 59.4+ KB
```

We can see our data and check which are categorical and numerical values, we can also the number of rows and columns. We can also see our data types.

2. Check the summary statistics of the data-frame and comment on your findings.

	Unnamed: 0	sales	capital	patents	randd	employment	tobinq	value	institutions
count	759.000000	759.000000	759.000000	759.000000	759.000000	759.000000	738.000000	759.000000	759.000000
mean	379.000000	2689.705158	1977.747498	25.831357	439.938074	14.164519	2.794910	2732.734750	43.020540
std	219.248717	8722.060124	6466.704896	97.259577	2007.397588	43.321443	3.366591	7071.072362	21.685586
min	0.000000	0.138000	0.057000	0.000000	0.000000	0.006000	0.119001	1.971053	0.000000
25%	189.500000	122.920000	52.650501	1.000000	4.628262	0.927500	1.018783	103.593946	25.395000
50%	379.000000	448.577082	202.179023	3.000000	36.864136	2.924000	1.680303	410.793529	44.110000
75%	568.500000	1822.547366	1075.790020	11.500000	143.253403	10.050001	3.139309	2054.160386	60.510000
max	758.000000	135696.788200	93625.200560	1220.000000	30425.255860	710.799925	20.000000	95191.591160	90.150000

We are given data of sales and attribute and for all the data we can see its range, mean, median.

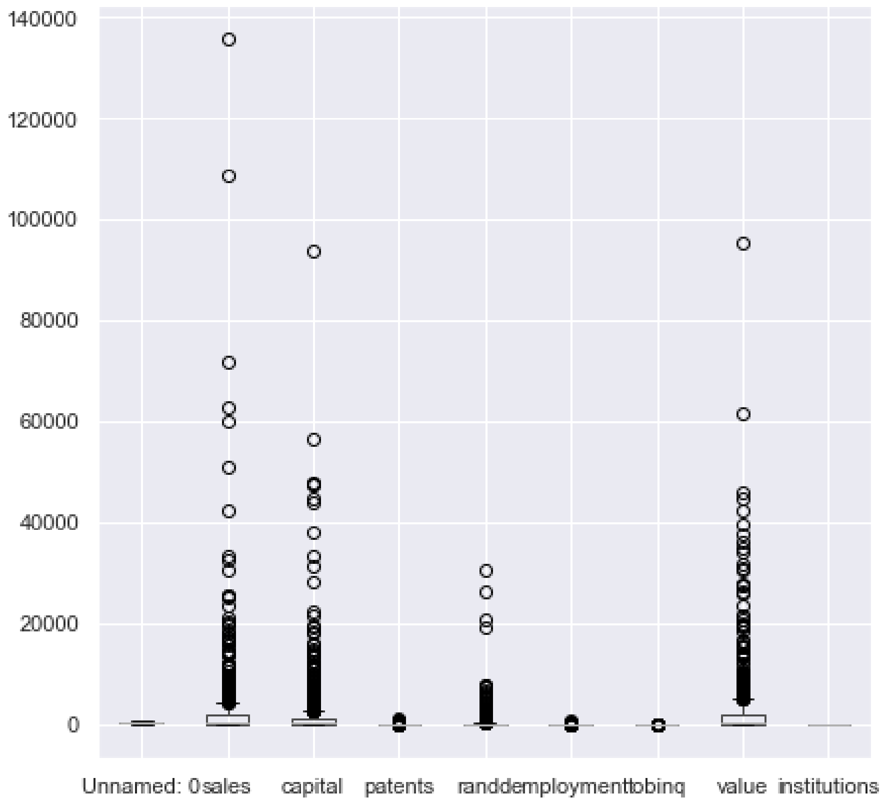
```python
# number of missing values (only the ones recognised as missing values) in each of the attributes
pd.DataFrame( mydata.isnull().sum(), columns= ['Number of missing values'])
```

	Number of missing values
Unnamed: 0	0
sales	0
capital	0
patents	0
randd	0
employment	0
sp500	0
tobinq	21
value	0
institutions	0

3. Describe the data briefly.

From the above findings we can see that sales, capital and patents etc and how they are related. We have Maximum sales is 135696.788 and minimum sales 0.138. We have 21 null values in tobinq: Tobin's q (also known as q ratio and Kaldor's v) is the ratio between a physical asset's market value and its replacement value. Sp500 has categorical values that needs to be encoded to get any further values. Maximum stock value 95191.59 for r and D stock value its 30425.255. The data shape is (759,10).Patent data types is integer as we count number of patents and not float. the As we can see we must replace null values with Either median or mean I have used median to replace NaN values. We must compare all the other attributes with sales according to the given problem statement.

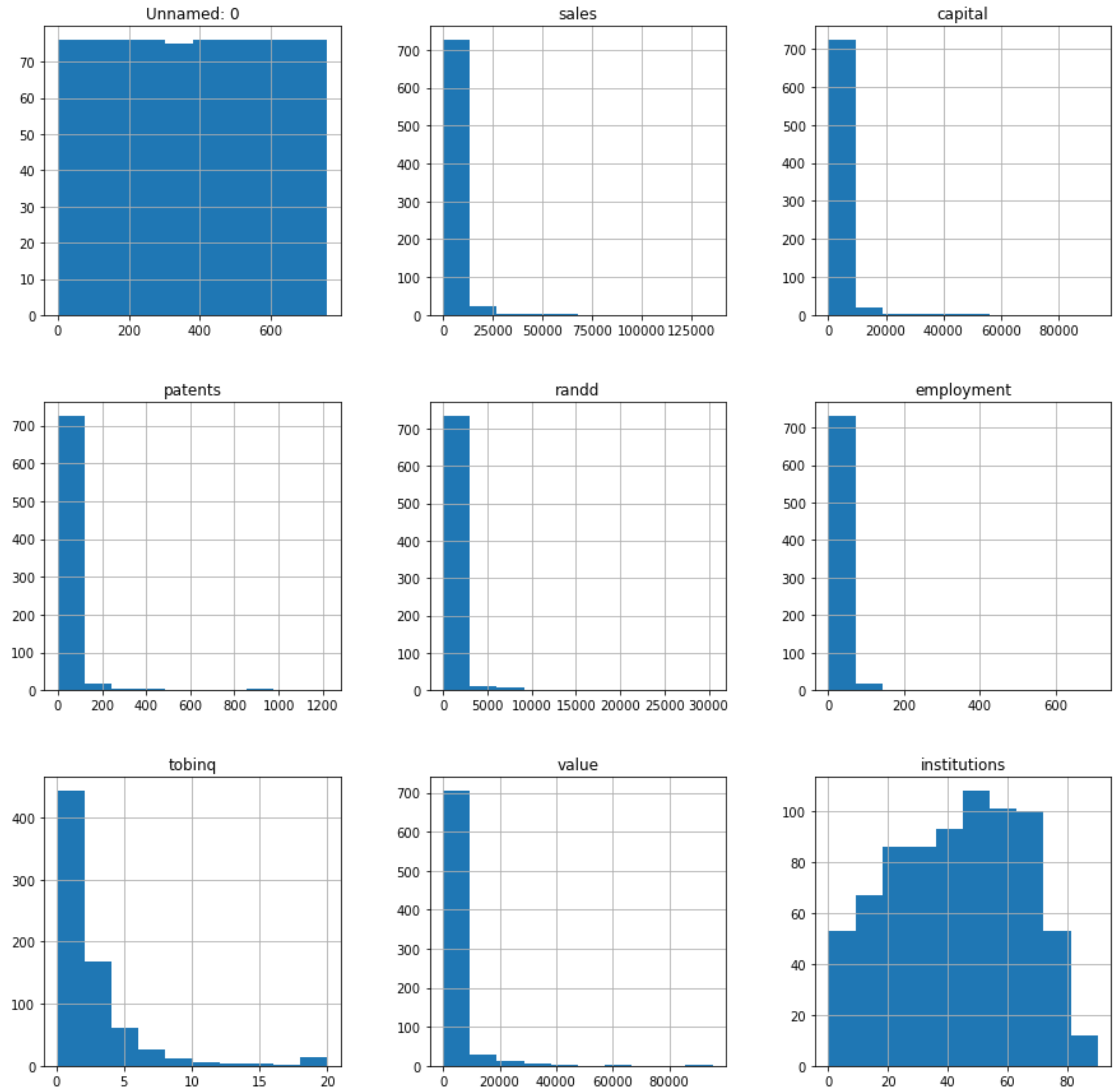
4. Perform Univariate Analysis
There are various methods to perform univariate analysis
Boxplot:

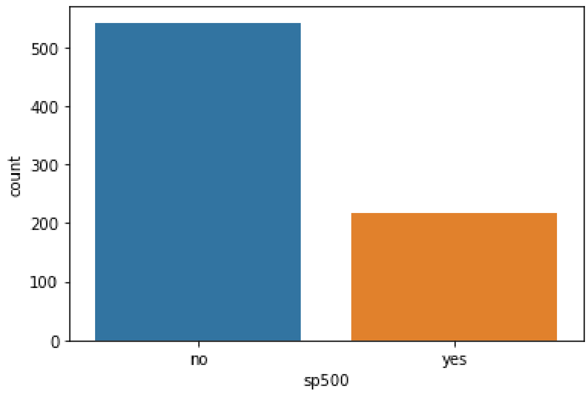


From the above boxplot we can see there are outliers which we must not remove since later the accuracy isn't good and we let the inconsistency remain.

From the below univariate sales, capital, patent, randd, employment right skewed, institution has normal distribution, tobinq also right skewed.

Plotting histogram analysis:



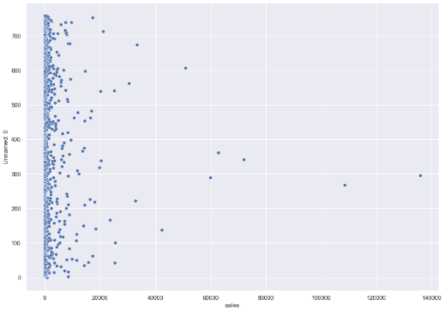


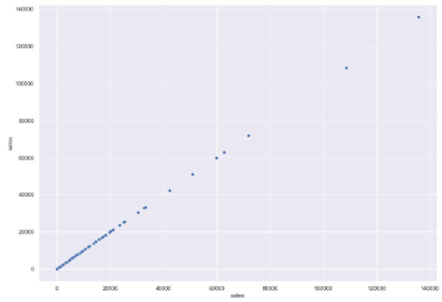
5. Bivariate Analysis

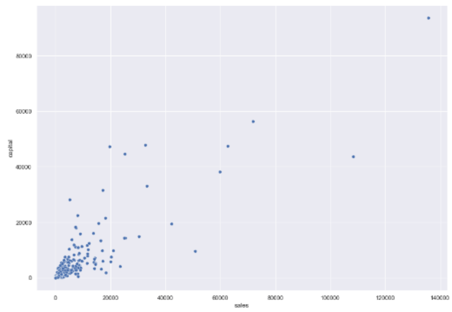
```
x=mydata["sales"]

for i in range(len(mydata.columns)):

    print("\033[1m\n Bivariate Analysis usr Vs", mydata.columns[i])

    plt.figure(figsize=(14,10))

    sns.scatterplot(x, mydata[mydata.columns[i]])

    plt.show()

    print("\033[1m\nInference & Observation:-")

    df1=mydata.loc[mydata[mydata.columns[i]].idxmax()]

    df1=df1["sales"]

    print("Maximum value of sales is ",df1,"when ",mydata.columns[i]," has maximum value
of",np.max(mydata[mydata.columns[i]]))
```

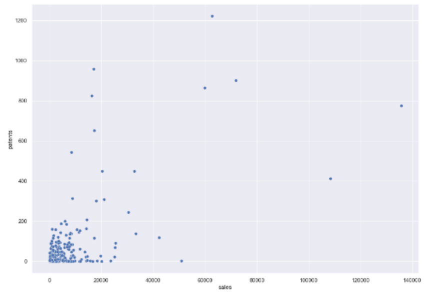
From the bivariate analysis we can see the various relation of all the given attributes with sales.

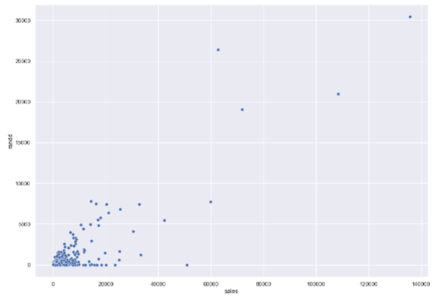
```
Maximum value of sales is  22.70199882 when  Unnamed: 0  has maximum va
lue of 758

Maximum value of sales is  135696.7882 when  sales  has maximum value o
f 135696.7882

Maximum value of sales is  135696.7882 when  capital  has maximum value
 of 93625.20056

Maximum value of sales is  62715.97381 when  patents  has maximum value
 of 1220

Maximum value of sales is  135696.7882 when  randd  has maximum value o
f 30425.25586

Maximum value of sales is  135696.7882 when  employment  has maximum va
lue of 710.7999253.
```

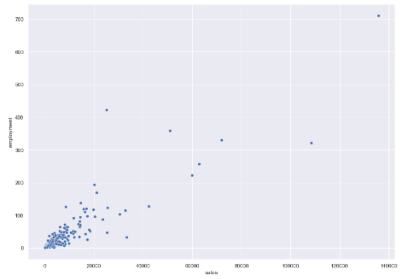


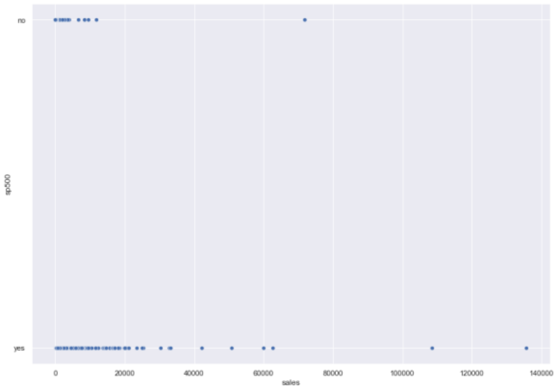


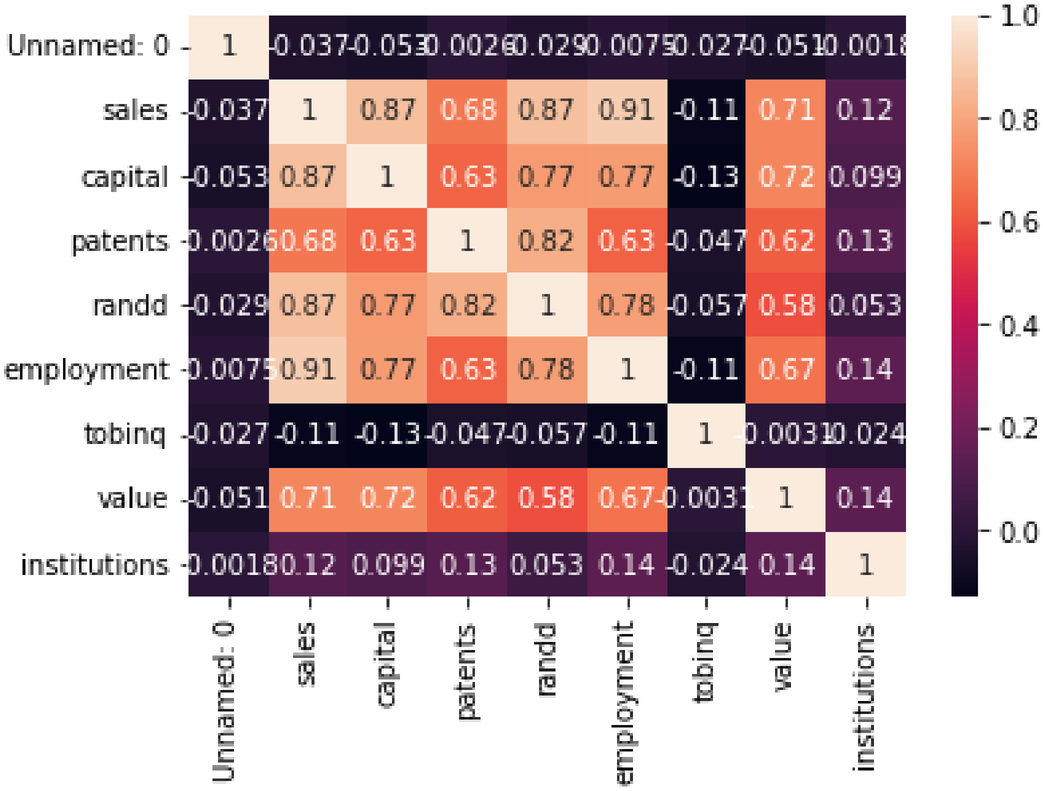












From the above bivariate analysis and heatmap we can find the multicollinearity capital and randd is highly correlated sales.We can remove highly correlated columns.

We can also do VIF treatment to do the same this helps in further calculation, and brings accuracy to the model.

```python
vif_data = pd.DataFrame()
vif_data["feature"] = X.columns

# calculating VIF for each feature
vif_data["VIF"] = [variance_inflation_factor(X.values, i)
                          for i in range(len(X.columns))]
vif_treatment=list(vif_data.loc[vif_data['VIF']>=4]['feature'])
print(vif_treatment)

vif_data.sort_values('VIF', ascending=False)
print(vif_data)
```

```
['capital', 'randd']
        feature       VIF
0       capital  4.119838
1       patents  3.885281
2         randd  6.229054
3    employment  3.888358
4        tobinq  1.566549
5         value  3.241414
6  institutions  2.447466
7     sp500_yes  2.220618
```

6.Impute null values if present?

```python
# Replacing NaN with a custom value
mydata['tobinq'].fillna(mydata.tobinq.median(), inplace = True)

# Replace NaN values with the mean of the column
# Data['tobinq'].fillna(Data.tobinq.mean(), inplace = True)

# Replace NaN values with the median of the column
# Data['tobinq'].fillna(Data.tobinq.median(), inplace = True)
mydata
```

	Unnamed: 0	sales	capital	patents	randd	employment	sp500	tobinq	value	institutions
0	0	826.995050	161.603986	10	382.078247	2.306000	no	11.049511	1625.453755	80.27
1	1	407.753973	122.101012	2	0.000000	1.860000	no	0.844187	243.117082	59.02
2	2	8407.845588	6221.144614	138	3296.700439	49.659005	yes	5.205257	25865.233800	47.70
3	3	451.000010	266.899987	1	83.540161	3.071000	no	0.305221	63.024630	26.88
4	4	174.927981	140.124004	2	14.233637	1.947000	no	1.063300	67.406408	49.46
...	...	...	...	...	...	...	...	...	...	...
754	754	1253.900196	708.299935	32	412.936157	22.100002	yes	0.697454	267.119487	33.50
755	755	171.821025	73.666008	1	0.037735	1.684000	no	1.680303	228.475701	46.41
756	756	202.726967	123.926991	13	74.861099	1.460000	no	5.229723	580.430741	42.25
757	757	785.687944	138.780992	6	0.621750	2.900000	yes	1.625398	309.938651	61.39
758	758	22.701999	14.244999	5	18.574360	0.197000	no	2.213070	18.940140	7.50

7.) Try test Scaling options and confirm if you think scaling is necessary in this case?

Yes, scaling is a good option since the data ranges are huge and we need to bring to a good level or range for further interpretation. Yes in this problem I have used scaling. It is a step of Data Pre Processing that is applied to independent variables or features of data.

```
scaler = MinMaxScaler()
columns=['capital','patents','randd','employment','tobinq','value','institutions']

df_scaled = scaler.fit_transform(df[columns].to_numpy())
df_scaled = pd.DataFrame(df_scaled,columns=columns)
print("Scaled Dataset Using MinMaxScaler")
df_scaled['sp500']=mydata['sp500']
df_scaled.head().T
```

```
Scaled Dataset Using MinMaxScaler
```

	0	1	2	3	4
capital	0.001725	0.001304	0.066447	0.00285	0.001496
patents	0.008197	0.001639	0.113115	0.00082	0.001639
randd	0.012558	0.0	0.108354	0.002746	0.000468
employment	0.003236	0.002608	0.069856	0.004312	0.002731
tobinq	0.549797	0.036476	0.255835	0.009367	0.047498
value	0.017055	0.002533	0.271703	0.000641	0.000687
institutions	0.890405	0.654687	0.529118	0.29817	0.548641
sp500	no	no	yes	no	no

Upon doing this we can remove the unnecessary data columns after Vif treatment, so that we can remove multicollinearity.

8.) Encode the data.

Encode

```
df = pd.get_dummies(mydata, prefix='sp500', columns=['sp500'])
df
```

	Unnamed: 0	sales	capital	patents	randd	employment	tobinq	value	institutions	sp500_no	sp500_yes
0	0	826.995050	161.603986	10	382.078247	2.306000	11.049511	1625.453755	80.27	1	0
1	1	407.753973	122.101012	2	0.000000	1.860000	0.844187	243.117082	59.02	1	0
2	2	8407.845588	6221.144614	138	3296.700439	49.659005	5.205257	25865.233800	47.70	0	1
3	3	451.000010	266.899987	1	83.540161	3.071000	0.305221	63.024630	26.88	1	0
4	4	174.927981	140.124004	2	14.233637	1.947000	1.063300	67.406408	49.46	1	0
...	...	...	...	...	...	...	...	...	...	...	...
754	754	1253.900196	708.299935	32	412.936157	22.100002	0.697454	267.119487	33.50	0	1
755	755	171.821025	73.666008	1	0.037735	1.684000	1.680303	228.475701	46.41	1	0
756	756	202.726967	123.926991	13	74.861099	1.460000	5.229723	580.430741	42.25	1	0
757	757	785.687944	138.780992	6	0.621750	2.900000	1.625398	309.938651	61.39	0	1
758	758	22.701999	14.244999	5	18.574360	0.197000	2.213070	18.940140	7.50	1	0

Encoding the data will convert all the categorical values to 1's and 0's.We are doing this for sp500_no, sp500_yes.

9.) Data Split: Split the data into test and train.

X = df.drop('sales', axis=1)

y = df[['sales']]

X.head()

	patents	employment	tobinq	value	institutions
0	10	2.306000	11.049511	1625.453755	80.27
1	2	1.860000	0.844187	243.117082	59.02
2	138	49.659005	5.205257	25865.233800	47.70
3	1	3.071000	0.305221	63.024630	26.88
4	2	1.947000	1.063300	67.406408	49.46

```
y.head()
```

	sales
0	826.995050
1	407.753973
2	8407.845588
3	451.000010
4	174.927981

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30 , random_state=123)
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
(531, 5) (531, 1)
(228, 5) (228, 1)
```

```
X.columns
```

```
Index(['patents', 'employment', 'tobinq', 'value', 'institutions'], dtype='object')
```

Model Building & Model Performance:

10.) Apply Linear regression:

```
regression_model = LinearRegression()

regression_model.fit(X_train, y_train)

# Let us explore the coefficients for each of the independent attributes


for idx, col_name in enumerate(X_train.columns):
   print("The coefficient for {} is {}".format(col_name, regression_model.coef_[0][idx]))
```

```
The coefficient for patents is 15.276584322633443
The coefficient for employment is 145.58806378577813
The coefficient for tobinq is -79.50630905099676
The coefficient for value is 0.17790393743248134
The coefficient for institutions is -10.543794183092317
```

This is the coefficient and the require attributes that will give us the best fit line.

11.) Check the performance of Predictions on Train and Test sets using performance metrices.

```
# Let us check the intercept for the model
intercept = regression_model.intercept_[0]


print("The intercept for our model is {}".format(intercept))
```

```
The intercept for our model is 493.2435872677197
```

```
# R square on training data
model1=regression_model.score(X_train, y_train)
model1
```

```
0.8521158031628298
```

```
#R square on testing data
model2=regression_model.score(X_test, y_test)
model2
```

```
0.8797374510772246
```

```
Our R square value in percent is 85% which is also similar to trained d
ata,
```

#RMSE on training data

predicted_train=regression_model.fit(X_train, y_train).predict(X_train)

np.sqrt(mean_squared_error(y_train,predicted_train))

```
3630.142501033774
```

##RMSE on testing data

predicted_test=regression_model.fit(X_train, y_train).predict(X_test)

np.sqrt(mean_squared_error(y_test,predicted_test))

```
2334.55672683095
```

Calculate MSE

mse = np.mean((lm1.predict(data_train.drop('sales',axis=1))-data_train['sales'])**2)

mse

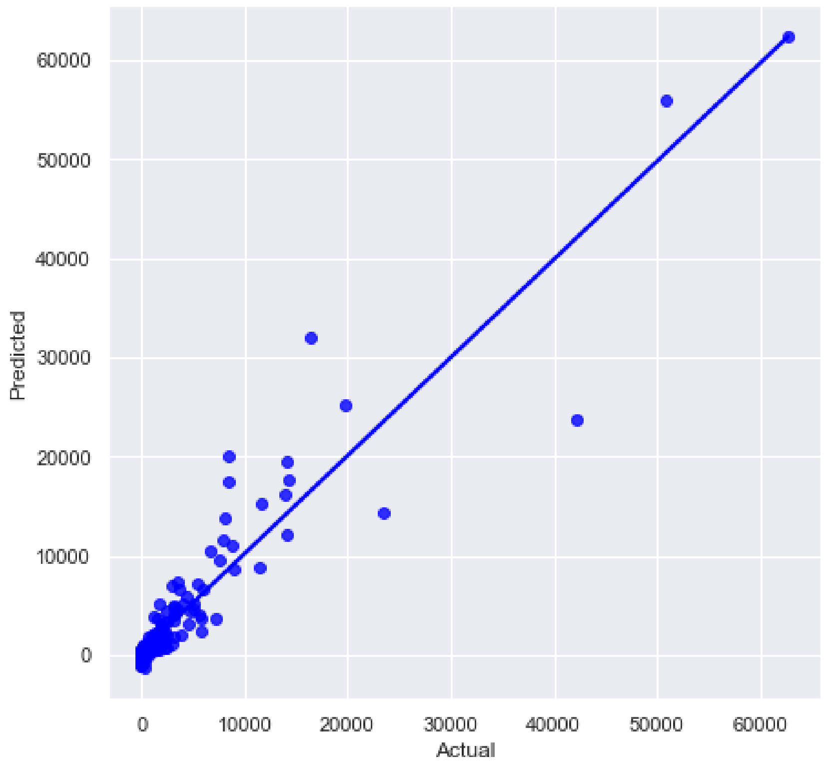
```
13177934.577811722 is the mse we achieved for our model
```

12.) Check for important features that impact the predictor and list them down.
We have found the important features previously patent, employment, tobinq, value, institution

```
The coefficient for patents is 15.276584322633443
The coefficient for employment is 145.58806378577813
The coefficient for tobinq is -79.50630905099676
The coefficient for value is 0.17790393743248134
The coefficient for institutions is -10.543794183092317
```

13).Drop unnecessary features and build a regressor for the best fit line.

```
sns.set(rc = {'figure.figsize':(7,7)})
ax=sns.regplot(x=y_test,y=y_pred,ci=None,color ='blue');
ax.set_xlabel('Actual')
ax.set_ylabel('Predicted')
```



The above we get the best fit line after scaling and removing unnecessary data from vif treatment and we get a linear regression plot .