K-Means Clustering Project

Value Difference vs Total Cost

Aim: To show whether resulting clusters show a pattern during selected months.

Introduction:

The purpose of the project is to explore the stock loss data by running a K-means Clustering model to divide data points into natural grouping (clusters) based on a relationship between value difference and total cost, and whether there are patterns between the clusters and months.

About the data:

The data is for the last 8 months (Aug 18 - March 19) at a weekly, site, product level. To minimize the amount of data weeks have been aggregated to month level and product to category level. Data has also been cleansed pre-hand to ensure no rows are present which include null values, data now sits at 30,814 rows and saved as a CSV file.

Preparing the Data

Once data is loaded, it must be checked and prepared before running it against the model.

In [32]:
```
# import pandas library to read, transform and manipulate data
import pandas as pd
# import matplotlib to visualise the data
import matplotlib.pyplot as plt
# import KMeans machine learning algorithm only from the sklearn library
from sklearn.cluster import KMeans
# import scale preprocessing function to scale data from sklearn
from sklearn.preprocessing import scale
# import seaborn to visulaise the data
import seaborn as sns
```

In [33]:
```
# import CSV using pandas read_csv function
contents = pd.read_csv('K-MeansStockLoss.csv')
```

In [34]:
```
# head() function to display the first 5 rows of the data
contents.head()
```

Out[34]:

	Month	Site	Category	Cost Value	Value difference	Return Value %
0	F - Jan	12	F20A	0.0	-807.82	-1.0
1	F - Jan	12	F33A	0.0	-358.72	-1.0
2	F - Jan	12	F38C	0.0	-731.04	-1.0
3	F - Jan	14	F20A	0.0	-1376.66	-1.0
4	F - Jan	14	F21A	0.0	-26.95	-1.0

In [35]:
```
# checking the columns in the data table
contents.columns
```

Out[35]:
```
Index(['Month', 'Site', 'Category', 'Cost Value', 'Value difference',
       'Return Value %'],
      dtype='object')
```

In [36]:
```
# renaming the columns
contents.columns = ['month','site','category','cost','difference','return']
```

In [37]:
```
# info() function to display information about the data table
contents.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30814 entries, 0 to 30813
Data columns (total 6 columns):
month         30814 non-null object
site          30814 non-null int64
category      30814 non-null object
cost          30814 non-null float64
difference    30814 non-null float64
return        30814 non-null float64
dtypes: float64(3), int64(1), object(2)
memory usage: 1.4+ MB
```

In [38]:
```
# drop() function to drop columns not needed in the model
contents=contents.drop(['return','site','category'], axis=1)
```

In [39]:
```
# checking the data table after previous action
contents.head()
```

Out[39]:

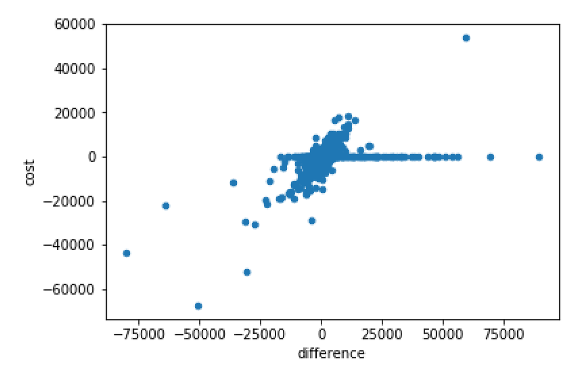
	month	cost	difference
0	F - Jan	0.0	-807.82
1	F - Jan	0.0	-358.72
2	F - Jan	0.0	-731.04
3	F - Jan	0.0	-1376.66
4	F - Jan	0.0	-26.95

Visulaising the Data

Visualising is an important step to help give a complete overview of the distribution of all the data points in the data set, this is achieved using a scatter plot.

In [40]:
```
# to plot scatter graph from matplotlib, must define what the X and Y axis are in ''
contents.plot.scatter(x='difference', y='cost')
```

Out[40]: `<matplotlib.axes._subplots.AxesSubplot at 0xb010d30>`



Total cost is plotted against the y-axis where positive is a gain and negative is a loss, difference in value is plotted along the x-axis where positive is a gain in value and negative is a loss in value.

From the scatter plot above most of the data points are grouped around the coordinates of (0,0), meaning the total cost and value difference are very close to the value of zero.

It is also clear to see that several data points are spread within the negative and positive regions of the graph, these points cannot be grouped with the majority.

Model

Now the data has been visualised the next step is to run the KMeans clustering machine learning algorithm from sklearn on the two variables chosen, difference and cost. Initially 4 cluster groups have been chosen, this will be tested further down to understand if this is the correct number of clusters to represent the data set.

In [41]:
```
# running the KMeans machine learning algorithm from sklearn
data = []
for index, row in contents.iterrows():
    # defining the variables in ''
    difference = row['difference']
    cost = row['cost']
    data.append([difference,cost])
# selecting how many clusters
model = KMeans(n_clusters = 4)
# fitting the KMeans model to the data
model.fit(scale(data))
# add cluster column to the data table
contents['cluster'] = model.labels_
```

Its quite important to always scale the data especially if you have large distances between data points which could influence the clustering model, this is accomplished using the 'scale' function from sklearn in the above code.

Now that the Kmeans model has run, the head() function below allows the change to the data table to be seen. As the table below in out put 86, a new column has been added to the data table called 'cluster' and each data point (row) has been assigned to a cluster number (0,1,2,3 or 4).

In [42]:
```
# checking the data table after previous action
contents.head()
```

Out[42]:

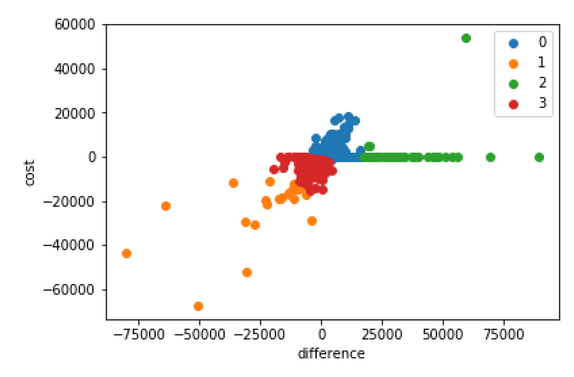
	month	cost	difference	cluster
0	F - Jan	0.0	-807.82	0
1	F - Jan	0.0	-358.72	0
2	F - Jan	0.0	-731.04	0
3	F - Jan	0.0	-1376.66	0
4	F - Jan	0.0	-26.95	0

Visualising Model Output

Now the KMeans model has run and the cluster output has been added to our data set as a column, it can now be visualised in the same way as before as scatter plot but in this instance an additional step is required, grouping by the clusters and colour coding to show clear identification of each cluster.

In [43]:
```
# grouping by clusters the model created
groups = contents.groupby('cluster')
# plot the clusters
fig, ax  = plt.subplots()
for name, group in groups:
    ax.plot(group.difference,group.cost, marker = 'o',linestyle = '', label = name)
# plot axis and legend
plt.xlabel('difference')
plt.ylabel('cost')
plt.legend()
```

Out[43]: `<matplotlib.legend.Legend at 0xae59fd0>`



It is clearly seen in the scatter plot above how the clusters are clearly visualised with a different colour. Its also clear that the four clusters seem to be split in half where two cluster (0 and 2) seem to be representing gains and two clusters (3 and 1) seem to be representing losses.

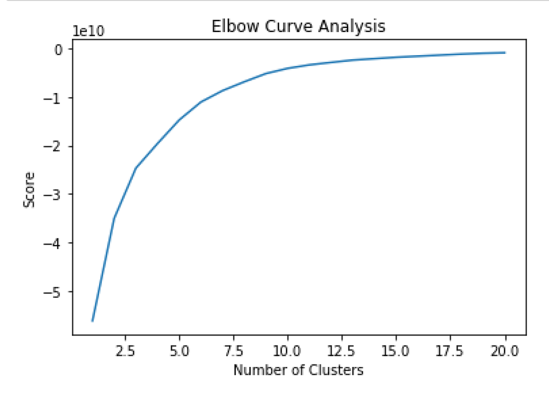
Optimising the Model

Understanding whether the number of clusters selected is the right number of clusters will help determine how optimal the model is, for this an Elbow test is required.

The Elbow test calculates the sum of squared errors for every k value, the curve represents an arm and at the point of the elbow is the best K value, but the aim is to choose a small value of K that still has a low SSE.

In [44]:
```
# ploting an elbow curve analysis for optimum number of clusters
x = contents[['difference']]
y = contents[['cost']]
# select the number of custers to allow to see a full curve
num_clusters = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
kmeans = [ KMeans(n_clusters=i) for i in num_clusters ]
score = [kmeans[i-1].fit(y).score(y) for i in num_clusters]

plt.plot(num_clusters,score)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve Analysis')
plt.show()
```



In certain situations, the data may not be very clustered and may not have a clear elbow due to a smooth curve. It can be unclear to which K value to choose.

From the elbow test above 6 clusters seems to be the optimal number of clusters.

Understanding the Clusters

To help further understand the clusters, a group by function is used to count the number of data points in each cluster.

In [45]:
```
# groupby function by cluster for all months
contents.groupby('cluster')['month'].count()
```

Out[45]:
```
cluster
0    27454
1       28
2       41
3     3291
Name: month, dtype: int64
```

Looking closely at the above results, clusters 0 and 3 are way over represented with 99.8% of all the data points. Would changing the number of clusters to the optimal number make any difference?

Analysis

Clustering the data is beneficial but, in this instance, doesn't give enough insight. To further analyse, months will be introduced. The idea is to detect if the total difference once split by month will show any pattern.

Firstly, the current data will be pivoted to show months as columns and the clusters as rows and the value as the sum of the difference. Once complete the outcome will be stored as a table called matrix.

In [46]:
```
# pivoting the data, naming it and showing the results
matrix=contents.pivot_table(index='cluster',columns='month',values='difference',aggfunc='sum')
matrix
```

Out[46]:

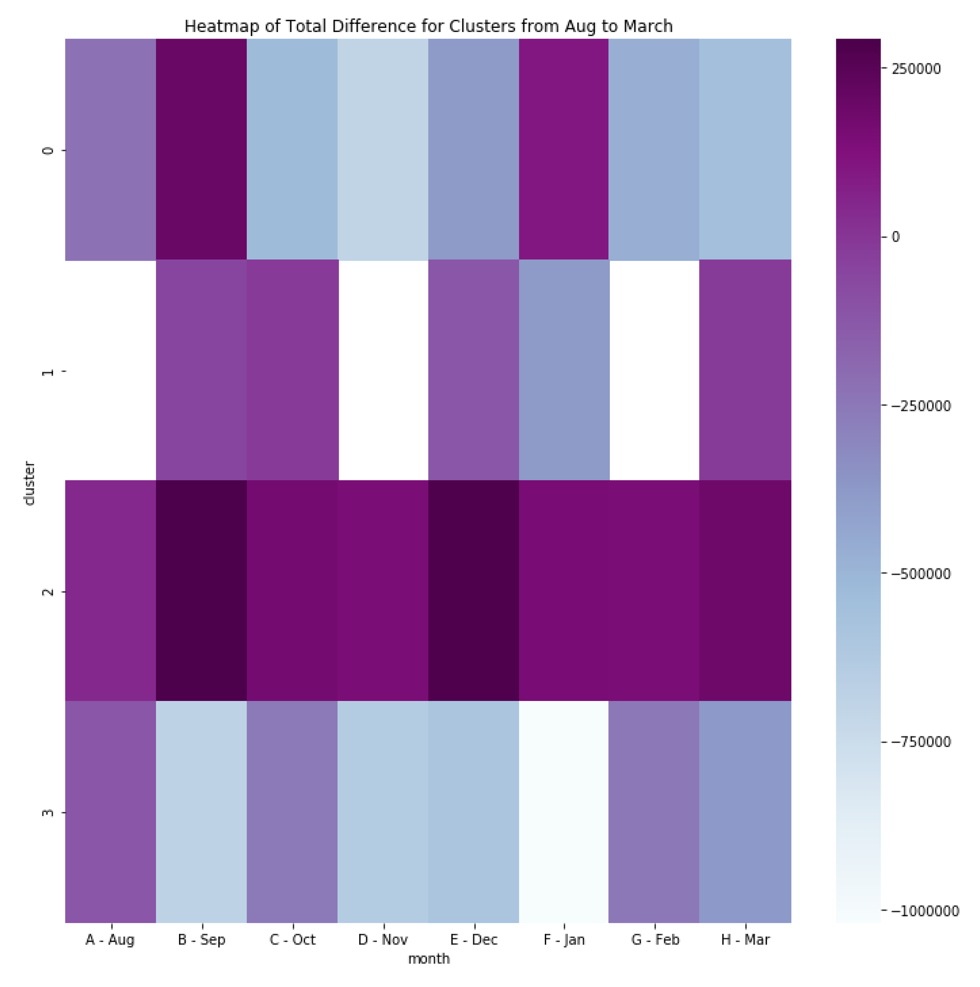
month	A - Aug	B - Sep	C - Oct	D - Nov	E - Dec	F - Jan	G - Feb	H - Mar
cluster								
0	-229986.76	204573.21	-524663.22	-694456.43	-381156.79	102130.24	-468135.29	-549086.14
1	NaN	-51663.70	-12696.19	NaN	-121171.88	-381108.71	NaN	-16526.35
2	41296.86	292063.71	165906.14	145575.90	278623.59	149591.58	147723.53	188792.63
3	-118488.25	-686165.44	-257425.83	-639680.08	-590797.70	-1019630.46	-247223.38	-374514.70

Visualising Matrix Output

The best way to detect a pattern is for the data to be visualised and in this instance a heat map will be used to reflect the total difference with shading.

In [47]:
```
# visualising the pivot table by plotting heat map
fig = plt.figure(figsize=(12,12))
r = sns.heatmap(matrix, cmap='BuPu')
r.set_title("Heatmap of Total Difference for Clusters from Aug to March")
```

Out[47]: `Text(0.5,1,'Heatmap of Total Difference for Clusters from Aug to March')`



Evaluation

Looking at the Heat map initially there is no clear patterns. Looking closely at each cluster there are some highlights to call out:

Custer 0 – Every month recorded a negative total difference except the months of Sep and Jan. Why? Could there be a stock count due to range change in Sep and Jan.

Cluster 1 – All months recorded a negative total difference except 4 months which recorded no data points. Data was only recorded for Sep and Oct, could the reason be range change and Halloween and Dec and Jan which is peak months.

Cluster 2 solid pattern of positive total difference every month.

Cluster 3 solid pattern of negative total difference every month with three of its four worst months recorded in Nov, Dec and Jan with the latter being the worst month, this is all leading and including peak period.

Improving Project

Initially, the data was cleansed outside the use of Python in Excel when testing the suitability of the data set. With the realisation that all is capable within Python, the intention is to import the data with no amendments in the next project and completely prepare the data in Jupyter Notebook.

The second improvement is to re-rum the model with the optimal number of clusters and to see the difference in the results.

The third improvement would be to gather a full year set of data and really explore the patterns.

Last improvement would have been to use code to be able to order the months in correct financial year sequence once the table was pivoted. Instead the raw data was manipulating with an alphabet character code in front of each month, so the order was done automatically.