Import Libraries

```
from google.colab import files
import numpy as np
import pandas as pd
from numpy import unique
from numpy import where
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import KMeans
from matplotlib import pyplot
from sklearn import preprocessing
import scipy.cluster.hierarchy as shc
import plotly.express as px
```

Fetch Data

```
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Indian_IPO_Data.csv to Indian_IPO_Data.csv

Create Raw Dataframe

```
raw_df = pd.read_csv('Indian_IPO_Data.csv')
```

Define Constant Variable for Future Use

```
# LIST OF ATTRIBUTES THAT ARE RELEVANT FOR CLUSTERING
selected_list = ['Srno','Company_Name','Face_Value','Bid_Price_From','Bid_Price_To','
# LIST OF ATTRIBUTES THAT ARE RELEVANT FOR BIDDING PRICE
bid_average_list = ['Bid_Price_From', 'Bid_Price_To']
# LIST OF ATTRIBUTES THAT ARE RELEVANT FOR OPENING PRICE
open_average_list = ['NSE_Listing_day_open', 'NSE_SME_Listing_day_open','BSE_Listing_
# LIST OF ATTRIBUTES THAT ARE RELEVANT FOR CLOSING PRICE
close_average_list = ['NSE_Listing_day_last_trade', 'NSE_SME_Listing_day_last_trade',
# LIST OF ATTRIBUTES THAT ARE RELEVANT FOR HIGH ON FIRST DAY OF LISTING
high_average_list = ['NSE_Listing_day_high', 'NSE_SME_Listing_day_high','BSE_Listing_
# LIST OF ATTRIBUTES THAT ARE RELEVANT FOR LOW ON FIRST DAY OF LISTING
low_average_list = ['NSE_Listing_day_low', 'NSE_SME_Listing_day_low','BSE_Listing_day
# LIST OF ATTRIBUTES THAT ARE RELEVANT FOR OVERALL VOLUME IN MARKET
volume_average_list = ['NSE_Listing_day_volume', 'NSE_SME_Listing_day_volume','BSE_Li

# LIST OF ATTRIBUTES THAT ARE AFFECTING STOCK PERFORMANCE OVERALL
summarised_feature_columns = ['Bid_Price_Average','Open_Price_Average','High_Price_Av
# LIST OF ATTRIBUTES THAT ARE MOST PROMINENT FOR AFFECTING STOCK PERFORMANCE
```

```
# LIST OF ATTRIBUTES THAT ARE MOST PROMINENT FOR AFFECTING STOCK PERFORMANCE
selected_summarised_feature_columns = [summarised_feature_columns[0],summarised_featu
scaler = preprocessing.MinMaxScaler()
```

Filter Raw Dataframe

```
# DATAFRAME IS SELECTED ONLY FOR RELEVANT ATTRIBUTES
selected_df = raw_df.filter(selected_list, axis=1)
```

Pre-Process Data

```
# AMPUTING NAN WITH 0
selected_df = selected_df.replace(np.NaN,float(0))

# PRE-PROCESSING ATTRIBUTE FOR TYPE CASTING INTO FLOAT
selected_df['NSE_Listing_day_open'] = [float(str(i).replace(",", "")) for i in select
selected_df['BSE_Listing_day_open'] = [float(str(i).replace(",", "")) for i in select
selected_df['NSE_Listing_day_last_trade'] = [float(str(i).replace(",", "")) for i in
selected_df['BSE_Listing_day_last_trade'] = [float(str(i).replace(",", "")) for i in
selected_df['NSE_Listing_day_high'] = [float(str(i).replace(",", "")) for i in select
selected_df['BSE_Listing_day_high'] = [float(str(i).replace(",", "")) for i in select
selected_df['NSE_Listing_day_low'] = [float(str(i).replace(",", "")) for i in selecte
selected_df['BSE_Listing_day_low'] = [float(str(i).replace(",", "")) for i in selecte
```

Create Bid_Price Column

```
# FINDING AVERAGE VALUE OF BIDDING PRICE AND SCALING IT ACCORDINGLY
selected_df[summarised_feature_columns[0]] = selected_df[bid_average_list].mean(axis=
selected_df[summarised_feature_columns[0]] = scaler.fit_transform(selected_df[[summar
```

Create Average_Open_Price Column

```
# FINDING AVERAGE VALUE OF OPENING PRICE AND SCALING IT ACCORDINGLY
selected_df[summarised_feature_columns[1]] = selected_df[open_average_list].mean(axis
selected_df[summarised_feature_columns[1]] = scaler.fit_transform(selected_df[[summar
```

Create Average_High_Price Column

```
# FINDING AVERAGE VALUE OF HIGH PRICE ON LISTING DAY AND SCALING IT ACCORDINGLY
selected_df[summarised_feature_columns[2]] = selected_df[high_average_list].mean(axis
selected_df[summarised_feature_columns[2]] = scaler.fit_transform(selected_df[[summar
```

Create Average_Low_Price Column

```
# FINDING AVERAGE VALUE OF LOW PRICE ON LISTING DAY AND SCALING IT ACCORDINGLY
selected_df[summarised_feature_columns[3]] = selected_df[low_average_list].mean(axis=
selected_df[summarised_feature_columns[3]] = scaler.fit_transform(selected_df[[summar
```

Create Average_Close_Price Column

```
# FINDING AVERAGE VALUE OF CLOSING PRICE(LAST TRADE PRICE) ON LISTING DAY AND SCALING
selected_df[summarised_feature_columns[4]] = selected_df[close_average_list].mean(axi
selected_df[summarised_feature_columns[4]] = scaler.fit_transform(selected_df[[summar
```

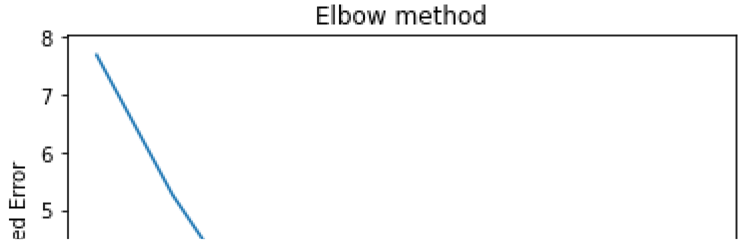
Create Total_Volume Column

```
# FINDING TOTAL VOLUME OF STOCK AS ON LISTING DAY IN MARKET AND SCALING IT ACCORDINGL
selected_df[[summarised_feature_columns[5]]] = selected_df[volume_average_list].sum(a
selected_df[summarised_feature_columns[5]] = scaler.fit_transform(selected_df[[summar
```

```
# FINDING PROFIT MARGIN ON LISTING DAY AND SCALING IT ACCORDINGLY
selected_df[summarised_feature_columns[6]] = selected_df[['Bid_Price_Average','Open_P
selected_df[summarised_feature_columns[6]] = selected_df[summarised_feature_columns[6
selected_df[summarised_feature_columns[6]] = scaler.fit_transform(selected_df[[summar
```

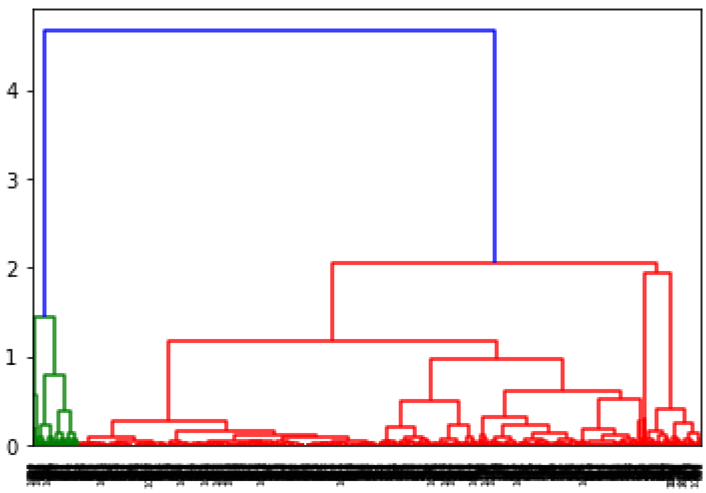
Finding Optimum Value for "K" using Elbow method

```
# COMPARING ERROR TO NUMBER OF CLUSTER USING KMEANS [VISUALLY] TO UNDERSTAND OPTIMUM
x = selected_df[selected_summarised_feature_columns].iloc[:,:].values
sum_sq_dist =[]
for i in range(2, 11):
    kmeans = KMeans(n_clusters = i)
    kmeans.fit(x)
    sum_sq_dist.append(kmeans.inertia_)
pyplot.plot(range(2, 11), sum_sq_dist)
pyplot.title('Elbow method')
pyplot.xlabel('No of clusters')
pyplot.ylabel('Sum Squared Error')
pyplot.show()
```




## Dendogram

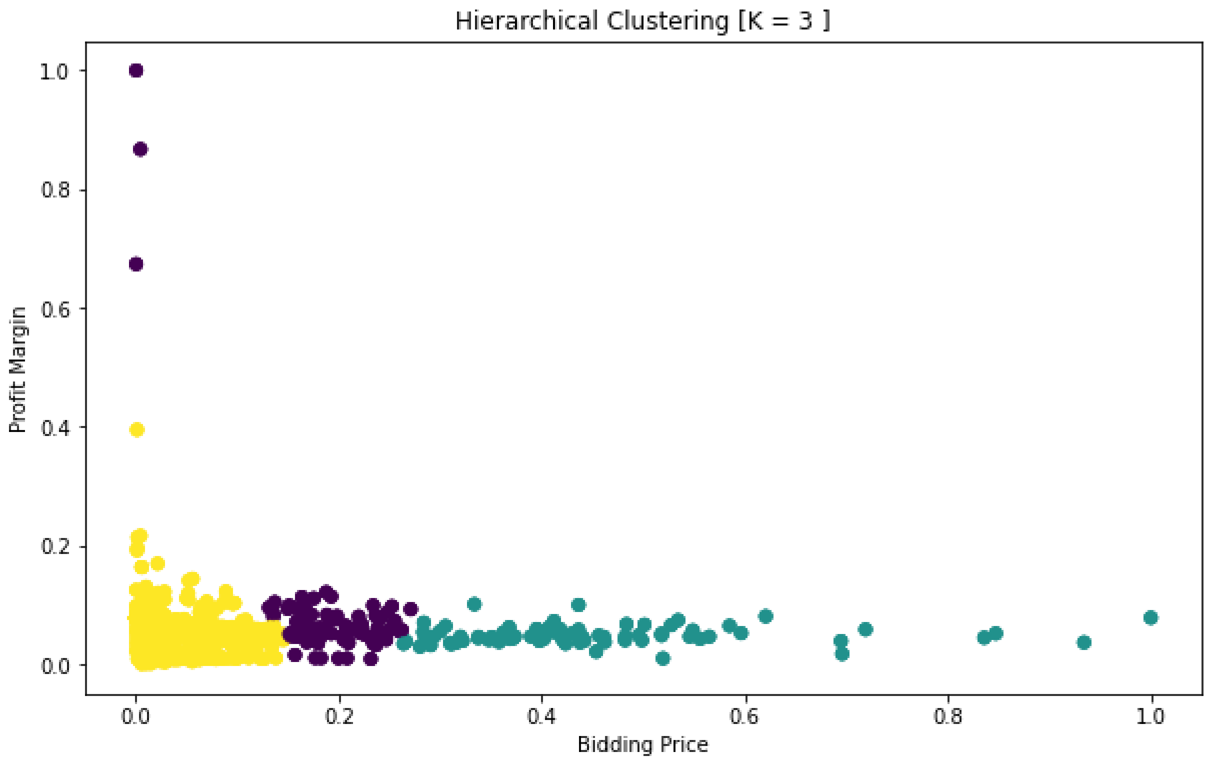
```
# CREATING DENDOGRAM FOR UNDERSTANDING CLUSTERING PROCESS AT EACH STEPS
Dendrogram = shc.dendrogram((shc.linkage(selected_df[selected_summarised_feature_colu
```

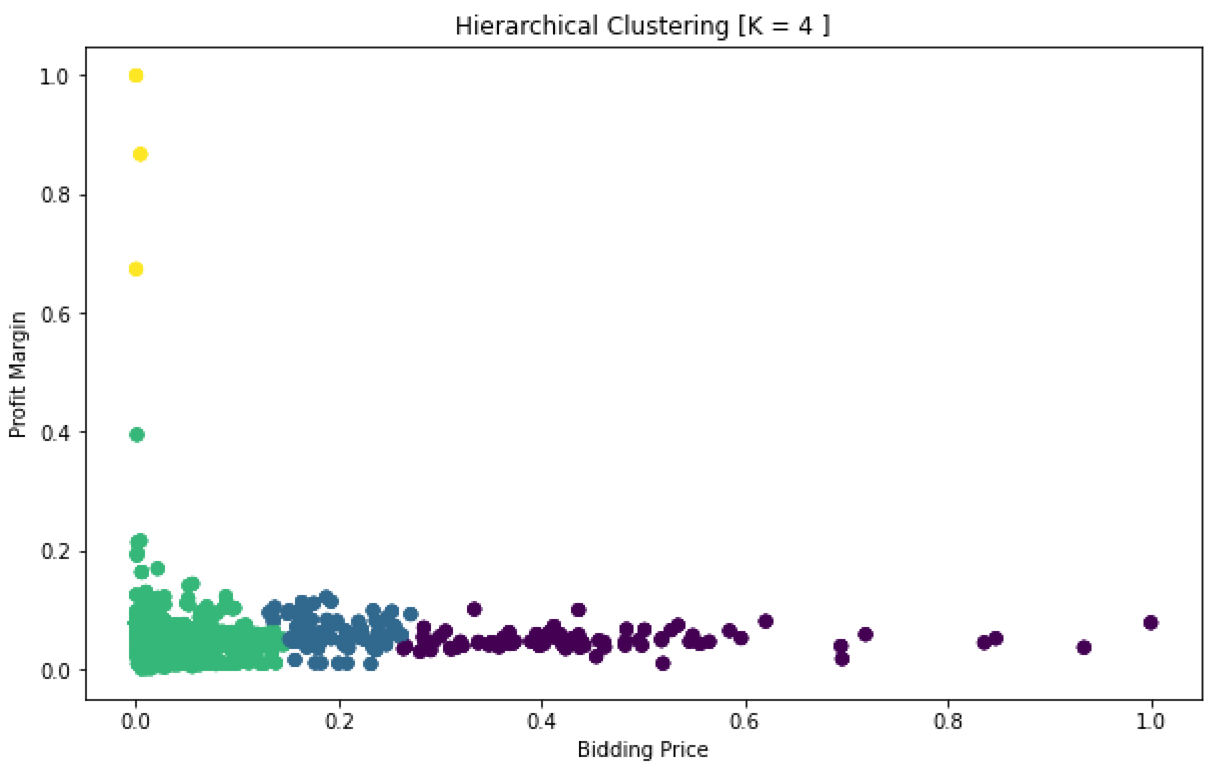


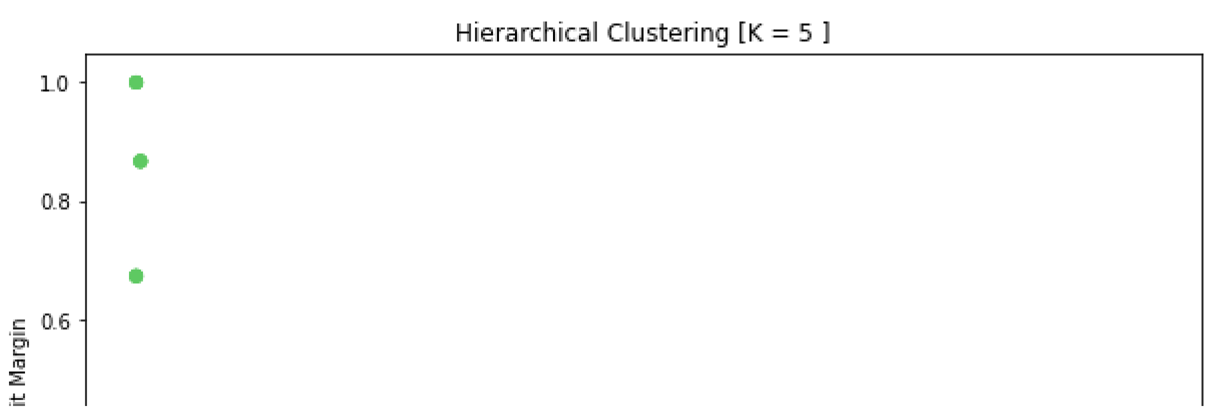
```
# SELECTING RANGE OF OPTIMUM VALUES FOR "K" BASED ON ELBOW METHOD
optimum_k = [3,4,5]
```

## Agglomerative Clustering

```
SUALISING DATA FOR AGGLOMERATIVE CLUSTERING ALGORITHM FOR VARIOUS VALUES OF "K"
k:

igsize =(10, 6))
rativeClustering(n_clusters=i)
odel.fit_predict(selected_df[selected_summarised_feature_columns])
ue(y_predicted)

clusters:
e(y_predicted == cluster)
r(selected_df[selected_summarised_feature_columns[0]], selected_df[selected_summarise
ierarchical Clustering [K = '+ str(i) +' ]')
Bidding Price')
Profit Margin')
```

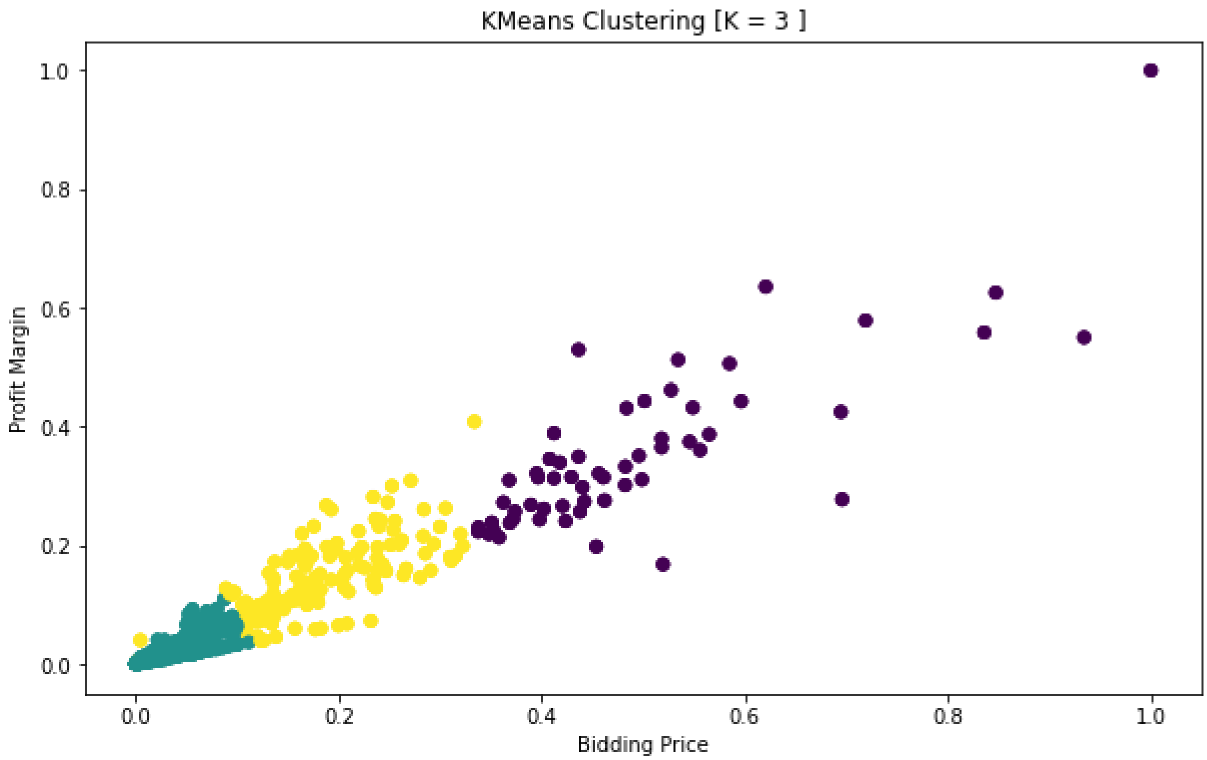

Hierarchical Clustering [K = 3 ]
Profit Margin
Bidding Price
0.0
0.2
0.4
0.6
0.8
1.0

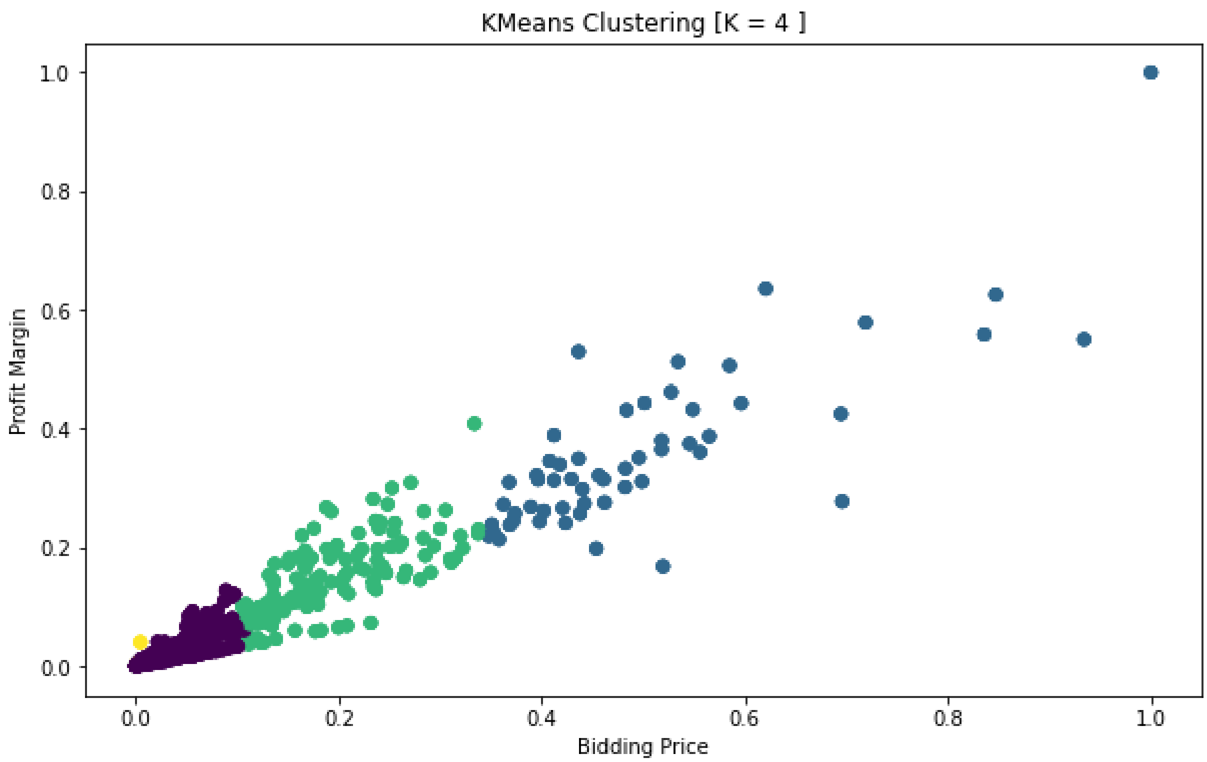

Hierarchical Clustering [K = 4 ]
Profit Margin
Bidding Price
0.0
0.2
0.4
0.6
0.8
1.0


Hierarchical Clustering [K = 5 ]
it Margin
0.6
0.8
1.0


K-Means Clustering

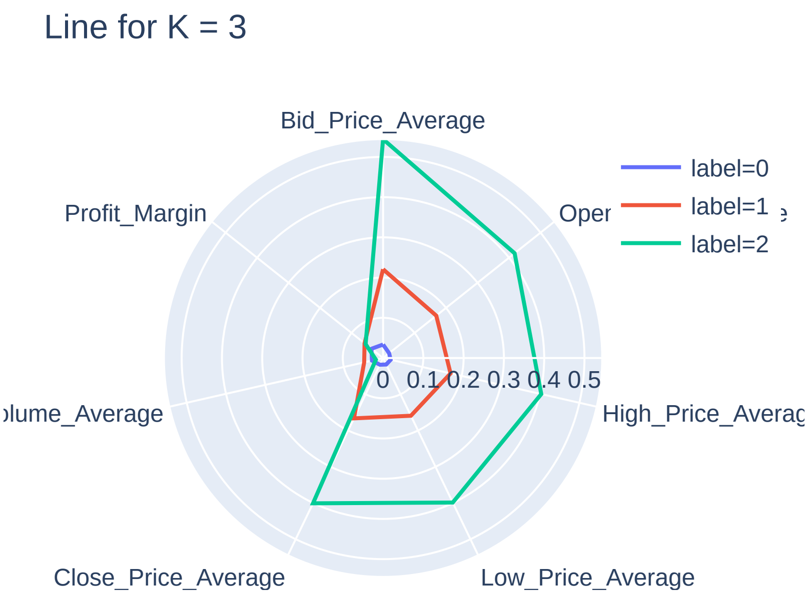
```
# APPLYING AND VISUALISING DATA FOR KMEANS CLUSTERING ALGORITHM FOR VARIOUS VALUES OF
for i in optimum_k:

  pyplot.figure(figsize =(10, 6))
  model = KMeans(n_clusters=i)
  model.fit(selected_df[selected_summarised_feature_columns])
  y_predicted = model.predict(selected_df[selected_summarised_feature_columns])

  clusters = unique(y_predicted)
  for cluster in clusters:
    row_ix = where(y_predicted == cluster)
    pyplot.scatter(selected_df[summarised_feature_columns[0]], selected_df[summarised
  pyplot.title('KMeans Clustering [K = '+ str(i) +' ]')
  pyplot.xlabel('Bidding Price')
  pyplot.ylabel('Profit Margin')
  pyplot.show()
```



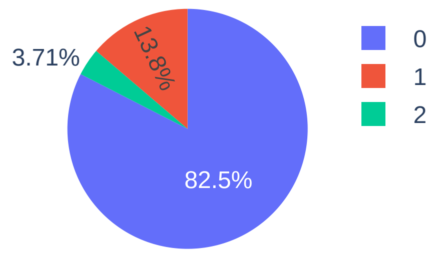

```
# APPLYING AND VISUALISING DATA FOR KMEANS CLUSTERING ALGORITHM [OVER MULTIPLE FEATUR
cluster_list = []
for i in optimum_k:

  model = KMeans(n_clusters=i)
  model.fit(selected_df[summarised_feature_columns])

  clusters=pd.DataFrame(selected_df[summarised_feature_columns],columns=summarised_fe
  clusters['label']=model.labels_
  polar=clusters.groupby("label").mean().reset_index()
```

```
    polar=pd.melt(polar,id_vars=["label"])
    fig4 = px.line_polar(polar, r="value", theta="variable", color="label", line_close=
    fig4.show()
    cluster_list.append([clusters,i])
```

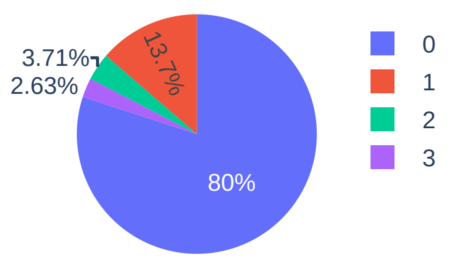



```
# VISUALISING DATA FOR STAKE OF EACH CLUSTER
color_list = ['blue','red','green','purple','orange']
for clusters in cluster_list:
  pie=clusters[0].groupby('label').size().reset_index()
  pie.columns=['label','value']
  fig = px.pie(pie,values='value',names='label',color=color_list[:clusters[1]],title=
  fig.show()
```

Pie Chart for K = 3




Pie Chart for K = 4




Pie Chart for K = 5