Project title	**CLUSTERING MODEL USING K-MEANS**
Module Name	**NICF Principals of Machine Learning (SF)**
Qualification Name	**NICF Diploma in Infocomm Technology (Data)**

Student name	**Assessor name**	
TAN DAI JUN	SHANTI SEKHAR	
Date issued	**Completion date**	**Submitted on**
4 APRIL, 2022	5 APRIL, 2022	5 APRIL, 2022

Project title	**CLUSTERING MODEL USING K-MEANS**

Learner declaration
I certify that the work submitted for this assignment is my own and research sources are fully acknowledged. Student signature: DJ Date: 5 APRIL, 2022

Table of Contents

1. Introduction

Unsupervised machine learning is a type of machine learning that uses algorithms to analyse and cluster unlabelled dataset. Unlabelled dataset can be understood as dataset that only has features and without labels nor target to predict. In unsupervised machine learning model, clustering is one of the techniques to understand a dataset by grouping unlabelled features. Out of many methods in clustering, K-means clustering is the method famous for its simplicity, guarantees converges (*loss decreases over time*) and more. In this project, we will be using this model to identify similarity within different kinds of problem registered in a technical support dataset.

The technical support dataset consists of different kinds of information that reflects the expenditure of a company in maintaining customer satisfaction, from total workforce in customer service to free parts replacement for products under warranty. It is crucial for a company to be aware of the performance of their product in terms of net profit in order to make a swift adjustment and maximize the profit margin throughout its life cycle.

Having understand the problem statement, the main objective of this project is to group all these registered problems from dataset into smaller groups and study those groups' descriptive statistics in order to gain insight from it and minimizes company's expenditure while remains to maintain customer satisfaction.

2. Methodology

As the methodology for this project, we first extract the "technical_spuuport_data" dataset which is in csv format. Since we do not have prior knowledge regarding on the dataset, it is best to explore the dataset from understanding its column and data type before model fitting.

Noted that this entire project will be done using Python in Jupyter Notebook. Table below showcases some of the useful libraries used in this project.

Library	Description
numpy	House with large collection of mathematical functions to operate arrays.
pandas	Built on top of numpy, offers data structure and operations for manipulating numerical table and time series.
sklearn	House with large collection of machine learning computation and operation functions.
matplotlib	A cross-platform, data visualization and graphical plotting library.
seaborn	Built on top of matplotlib, provides high level interface for drawing attractive and informative statistical graphics.
scipy	House with large collection of scientific and technical computing functions.

```
1  import pandas as pd
2  import numpy as np
3
4  import matplotlib.pylab as plt
5  %matplotlib inline
6
7  import seaborn as sns
8  from sklearn.model_selection  import train_test_split
9  from sklearn.cluster import KMeans
10
11 from scipy.stats import zscore
```

Figure 2.0: Libraries imported for this project

3. Project Execution and Discussions

With all the necessary libraries imported, we are ready to execute this project. Noted that for every crucial execution, there will be a short summary and explanation throughout this section.

3.1 Data overview

In this section, we will read the "technical_support_data" dataset using pandas from Python. We can notice that the dataset contains **23 observations** and **8 features**.

```
1 tech_supp_df = pd.read_csv("technical_support_data.csv")
```

Figure 3.1.0: Extract dataset and assign to variable "tech_supp_df"

```
1 tech_supp_df.shape
(23, 8)
```

Figure 3.1.1: Dimension of the dataset

We can eyeball the first 5 rows of observations from the dataset.

```
1 tech_supp_df.head()
```

	PROBLEM_TYPE	no_of_cases	Avg_pending_calls	Avg_resol_time	recurrence_freq	Replace_percent	In_warranty_percent	Post_warranty_percent
0	Temperature control not working	170	1.3	32	0.04	0.0	75	25
1	power chord does not tightly fit	12	2.0	150	0.01	0.5	5	95
2	Fan swing not working	5	1.0	35	0.02	0.2	90	10
3	Main switch does not on	3	2.0	8	0.01	0.7	5	95
4	Forgot mobile app password	45	2.3	54	0.15	0.0	99	1

Figure 3.1.2: First 4 rows of observations from the dataset

In order to understand the dataset, table below describes each of the variable and their characteristics.

Column	Description
PROBLEM_TYPE	Description of issue of any category of any device/service.
no_of_cases	Count of the specific type of complain received.
Avg_pending_calls	Average remaining calls of the complain to resolve “Avg_resol_time”.
Avg_resol_time	Average time to resolve a specific type of complain/issue.
recurrence_freq	How many time the same issue raised.
Replace_percent	To solve the issue/complain how many times the replacement is done.
In_warranty_percent	How many times the issue raised by customer falls under warranty period.
Post_warranty_percent	How many times the issue raised by customer falls after warranty period.

Based on the data description and figure 3.1.2, we can understand that each record in dataset contains each unique problem type. It has corresponding metrics for each type like count, average calls to resolve, average resolution time and more.

Besides, we can further understand the dataset by check out its datatype.

```
1  tech_supp_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23 entries, 0 to 22
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   PROBLEM_TYPE          23 non-null     object
 1   no_of_cases           23 non-null     int64
 2   Avg_pending_calls     23 non-null     float64
 3   Avg_resol_time        23 non-null     int64
 4   recurrence_freq       23 non-null     float64
 5   Replace_percent       23 non-null     float64
 6   In_warranty_percent   23 non-null     int64
 7   Post_warranty_percent 23 non-null     int64
dtypes: float64(3), int64(4), object(1)
memory usage: 1.6+ KB
```

Figure 3.1.3: Data types of each variable

3.2 Model fitting

In this section, we will begin with model fitting. Before the actual implementation of model fitting, it is crucial for us to have a basic understanding regarding on K-means clustering model.

3.2.1 K-mean clustering

As mentioned in earlier part, k-mean clustering is one of the techniques used in unsupervised machine learning model to cluster data in dataset into several groups based on similarity, or technically Euclidean distance.

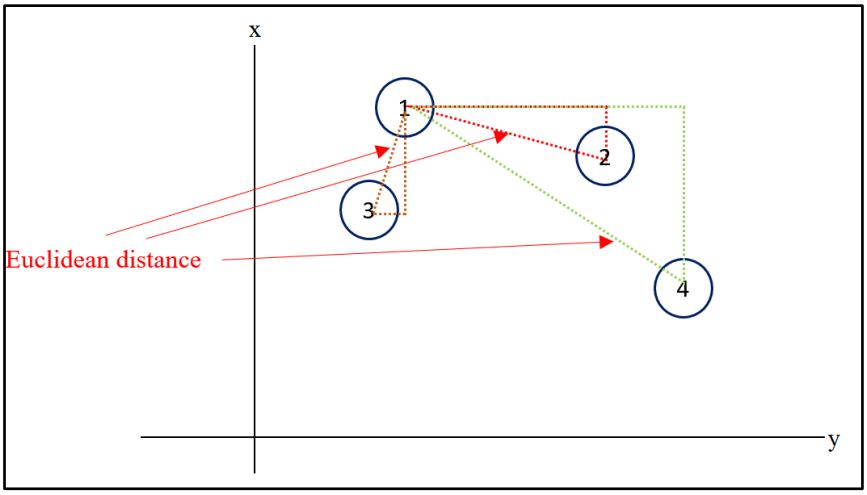

Figure 3.2.1.0: Conceptual diagram of Euclidean distance

Based on figure 3.2.1.0, we can understand how machine model finding the "similarity" between two data points by using Euclidean distance. In k-means model, the alphabet *k* means that total number of groups we would like to cluster from the dataset.

The working principle of K-mean clustering model can be summarized as:

1. Randomly select *k* number of data points.
2. Calculate distance between the remaining unselected data points and each selected *k* number of data points.
3. Data point with shortest distance to one of the *k* number data points will be clustered under that particular group. This process continues until exhausted all unselected data points.
4. Step 2 to 3 will be repeated based on user's configuration; the number of time k-mean will run with different centroid (*randomly select another k number of data points*). Noted that default value in sklearn is 10 times.
5. Once total number of time k-mean run with different centroid is exhausted, machine will choose the model with least total variance within each cluster.

Now that we have a general idea on how k-mean works, the main question will be how can we define the number of *k*. In order to compute for the optimum number of *k*, we will run from step 1 to 5 mentioned above with number of k from 1 to 10. As the number of *k* increases, the total sum of squares within clusters will be reduced. We can use this phenomenon to plot a line graph, known as Elbow method; and use it as a tool to select optimum value for *k*.

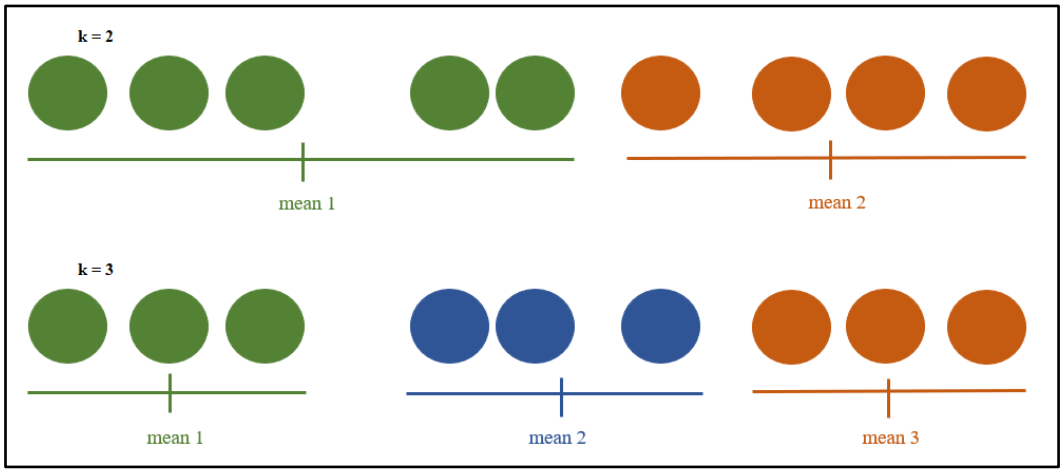

Figure 3.2.1.1: Illustration of sum of squares within cluster with different total number of clusters

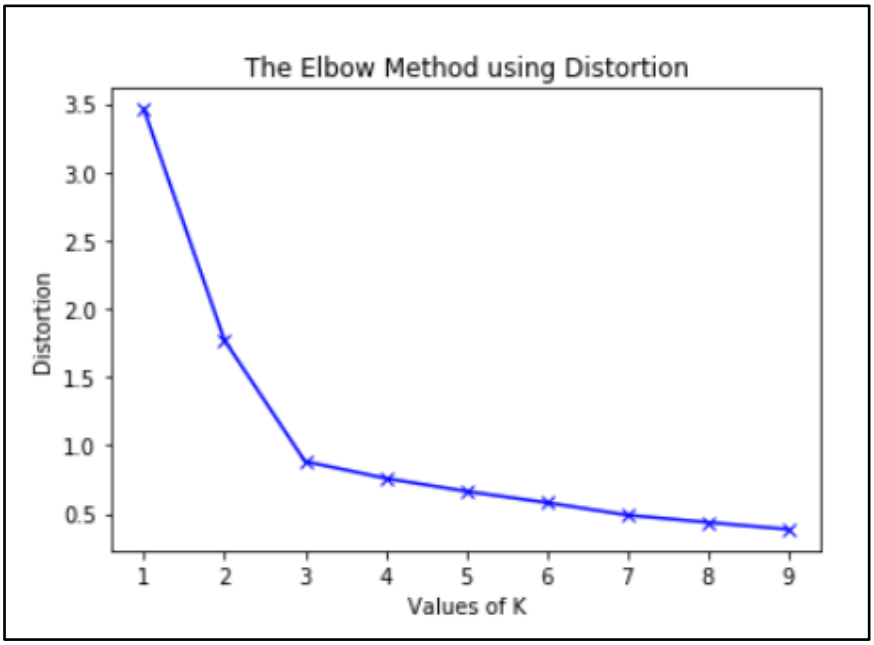

Figure 3.2.1.2: Example of plot from Elbow method

The main idea of Elbow method is to observe the trend of the plot and select the number of *k* on the x-axis where there's a drastic change in the gradient. Take figure 3.2.1.2 as an example, we can see that gradient of slope decreased drastically between *k* of 2 and 3 and *k* of 3 and 4, this *k* value of 3 is the optimum value for *k* in that context. Now that we have understood the K-means clustering method, it is time to work it out in this project.

3.2.2 Data transformation and visualization

In order to understand each variable better in the dataset, we construct a pair plot and study the distribution of each of them. On the other hand, we will scale down all of the variables into same unit of measurement to prevent bias towards particular variable during modelling.

Noted that in this project, we will be using z score transformation to scale down all variables

```
techSuppAttr=tech_supp_df.iloc[:,1:]
techSuppScaled=techSuppAttr.apply(zscore)
sns.pairplot(techSuppScaled,diag_kind='kde')
```

Figure 3.2.2.0: Source code for pair plot and z-score transformation

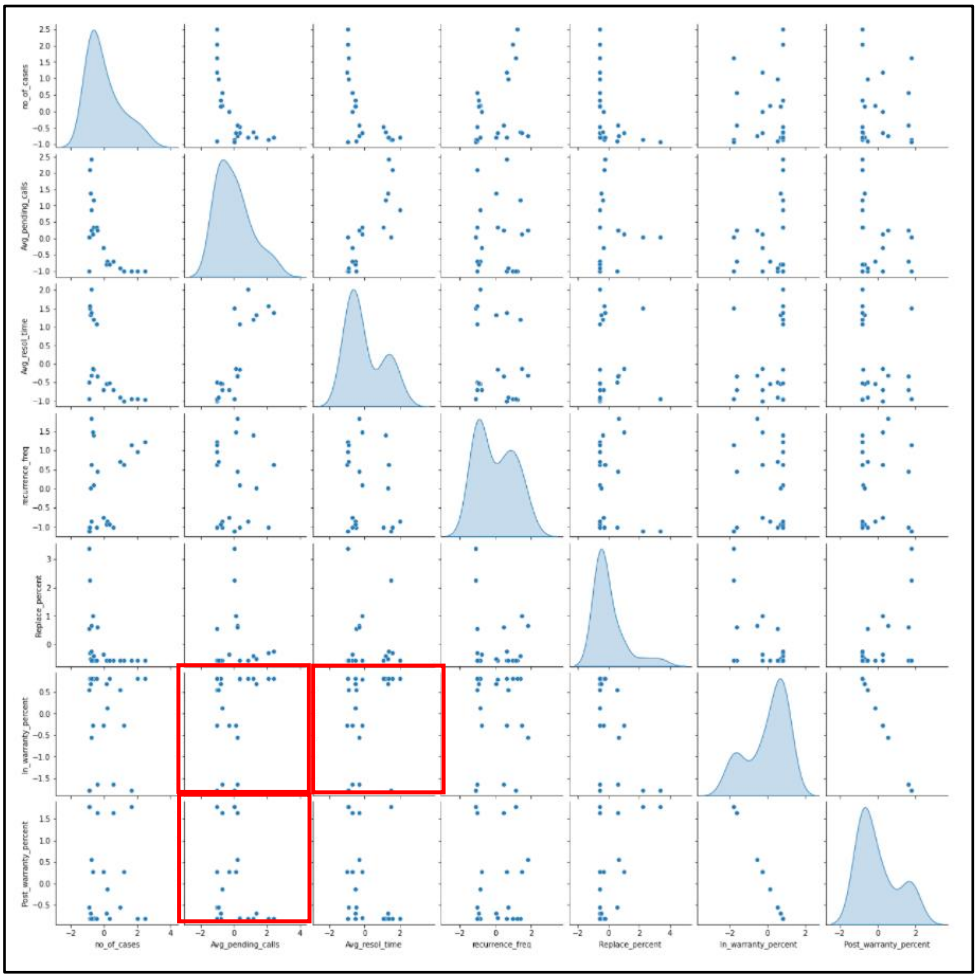

Figure 3.2.2.1: Pair plot

Based on figure 3.2.2.1, we can observe that some of the scatter plots has three distinct groups which indirectly means that the ideal value of *k* will be 3.

3.2.3 Elbow method

```
#Finding optimal no. of clusters
from scipy.spatial.distance import cdist
clusters=range(1,10)
meanDistortions=[]

for k in clusters:
    model=KMeans(n_clusters=k)
    model.fit(techSuppScaled)
    prediction=model.predict(techSuppScaled)
    meanDistortions.append(sum(np.min(cdist(techSuppScaled,
                                            model.cluster_centers_,
                                            'euclidean'), axis=1)) / techSuppScaled.shape[0])


plt.plot(clusters, meanDistortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Average distortion')
plt.title('Selecting k with the Elbow Method');
```

Figure 3.2.3.0: Source code for Elbow method

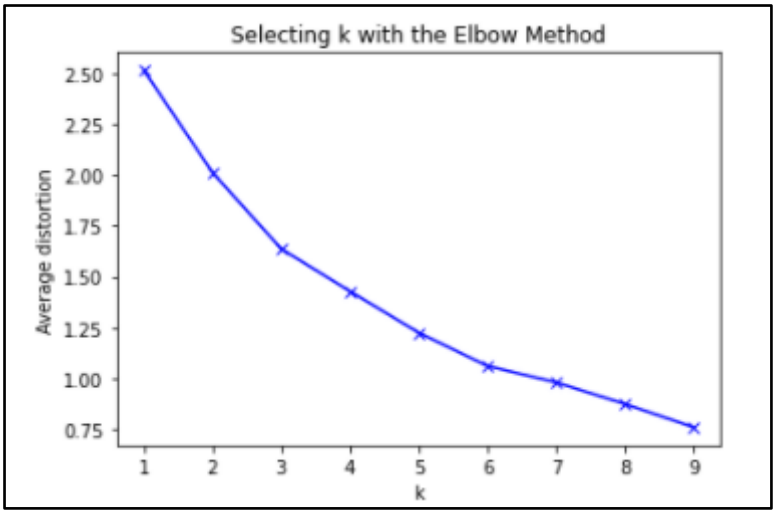

Figure 3.2.3.1: Plot generated by Elbow method

Based on figure 3.2.3.1, we can't obviously notice the trend and each gradient of slope does not change drastically. Therefore, we will pick the *k* of 3 and 5 to do clustering in later section.

3.2.4 Clustering with $k = 3$

In this section, we will begin the model fitting with 3 clusters.

```
# Let us first start with K = 3
final_model=KMeans(3)
final_model.fit(techSuppScaled)
prediction=final_model.predict(techSuppScaled)

#Append the prediction
tech_supp_df["GROUP"] = prediction
techSuppScaled["GROUP"] = prediction
print("Groups Assigned : \n")
tech_supp_df.head()
```

Figure 3.2.4.0: Source code for clustering with 3 groups

	PROBLEM_TYPE	no_of_cases	Avg_pending_calls	Avg_resol_time	recurrence_freq	Replace_percent	In_warranty_percent	Post_warranty_percent	GROUP
0	Temperature control not working	170	1.3	32	0.04	0.0	75	25	0
1	power chord does not tightly fit	12	2.0	150	0.01	0.5	5	95	1
2	Fan swing not working	5	1.0	35	0.02	0.2	90	10	0
3	Main switch does not on	3	2.0	8	0.01	0.7	5	95	1
4	Forgot mobile app password	45	2.3	54	0.15	0.0	99	1	2

Figure 3.2.4.1: First 5 rows of observations with newly added column “group” as prediction made

Once we are done with prediction, we can group the dataset by clustered 3 groups and study its descriptive statistics and gain some business insights.

```
techSuppClust = tech_supp_df.groupby(['GROUP'])
techSuppClust.mean()
```

GROUP	no_of_cases	Avg_pending_calls	Avg_resol_time	recurrence_freq	Replace_percent	In_warranty_percent	Post_warranty_percent
0	250.444444	1.166667	20.666667	0.125556	0.026667	85.555556	14.444444
1	112.142857	1.828571	47.857143	0.165714	0.272857	20.714286	79.285714
2	35.000000	3.157143	132.571429	0.125714	0.021429	99.142857	0.857143

Figure 3.2.4.2: Descriptive statistics for each group

Based on figure 3.2.4.2, we can notice that

- Group 0 has the lowest average resolution time for a complain and replace percent.
- Group 1 has the highest number of post warranty claim, recurrence frequency and replace percent.
- Group 2 has the average pending calls after a complain, average resolution time and warranty claim within warranty period.

Noted that asides from displaying descriptive statistics for each cluster in a table, we can visualize them using a box plot.

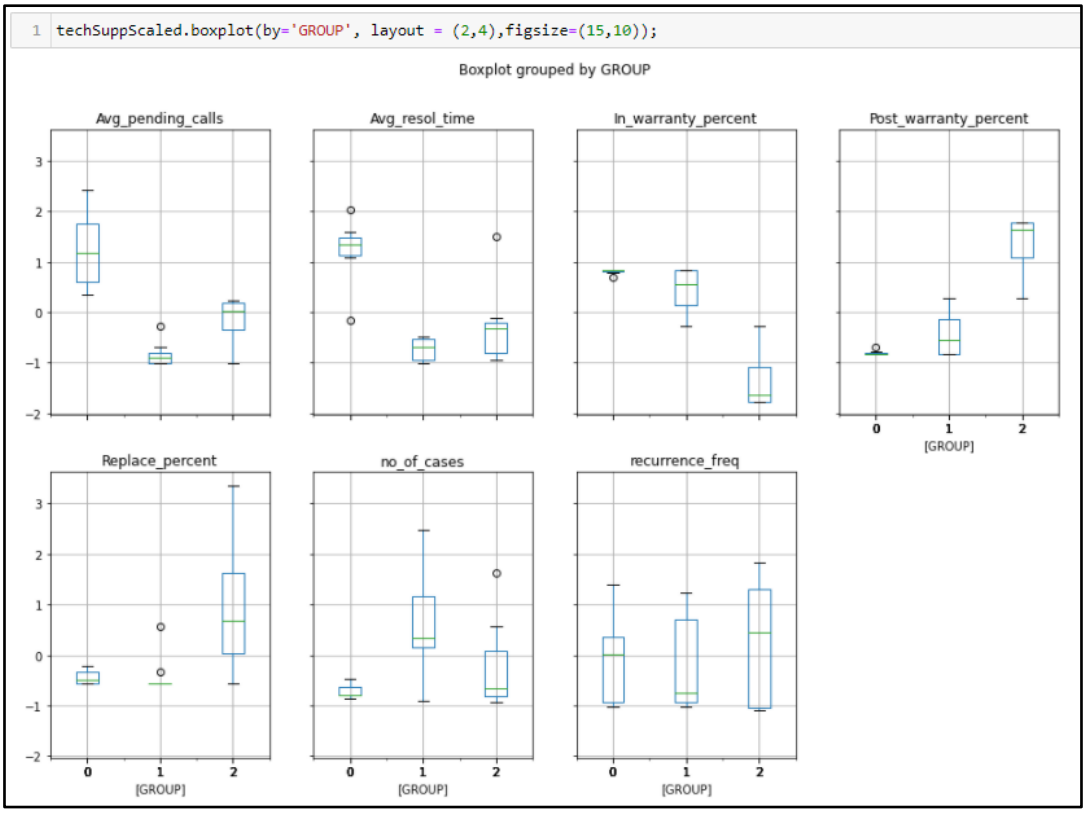

Figure 3.2.4.3: Box plot of each variable explained with cluster

3.2.5 Clustering with $k = 5$

In this section, we will begin with model fitting with 5 clusters.

```
# Let us first start with K = 5
final_model=KMeans(5)
final_model.fit(techSuppScaled)
prediction=final_model.predict(techSuppScaled)

#Append the prediction
tech_supp_df["GROUP"] = prediction
techSuppScaled["GROUP"] = prediction
print("Groups Assigned : \n")
tech_supp_df.head()
```

Figure 3.2.5.0: Source code for clustering with 5 groups

	PROBLEM_TYPE	no_of_cases	Avg_pending_calls	Avg_resol_time	recurrence_freq	Replace_percent	In_warranty_percent	Post_warranty_percent	GROUP
0	Temperature control not working	170	1.3	32	0.04	0.0	75	25	2
1	power chord does not tightly fit	12	2.0	150	0.01	0.5	5	95	0
2	Fan swing not working	5	1.0	35	0.02	0.2	90	10	2
3	Main switch does not on	3	2.0	8	0.01	0.7	5	95	0
4	Forgot mobile app password	45	2.3	54	0.15	0.0	99	1	3

Figure 3.2.5.1: First 5 rows of observations with newly added column “group” as prediction made

Once we are done with prediction, we can group the dataset by clustered 5 groups and study its descriptive statistics and gain some business insights.

```
techSuppClust = tech_supp_df.groupby(['GROUP'])
techSuppClust.mean()
```

GROUP	no_of_cases	Avg_pending_calls	Avg_resol_time	recurrence_freq	Replace_percent	In_warranty_percent	Post_warranty_percent
0	7.50	2.000000	79.000000	0.010000	0.600000	5.000000	95.000000
1	135.00	1.875000	38.500000	0.280000	0.177500	31.250000	68.750000
2	150.50	1.283333	29.666667	0.031667	0.040000	71.666667	28.333333
3	35.00	3.157143	132.571429	0.125714	0.021429	99.142857	0.857143
4	395.25	1.025000	7.750000	0.240000	0.000000	87.500000	12.500000

Figure 3.2.5.2: Descriptive statistics for each group

Based on figure 3.2.5.2, we can notice that

- Group 0 has the highest post warranty claim count, replace precent and frequency of occurrences of only 7.5 times.
- Group 1 has the highest number of recurrence frequency. The frequency of occurrences is 135 times.
- Group 2 has the second lowest in average resolution time. The frequency of occurrences is about 151 times.
- Group 3 has the highest average pending calls, average resolution time, number of warranties claim within warranty period and lowest post warranty claim. The frequency of occurrences is about 36 times.
- Group 4 has the lowest average resolution time, replace percentage. The frequency of occurrences is the highest, around 395 times.

Lastly, we can visualize the descriptive statistics in figure 3.2.5.2 using boxplots.

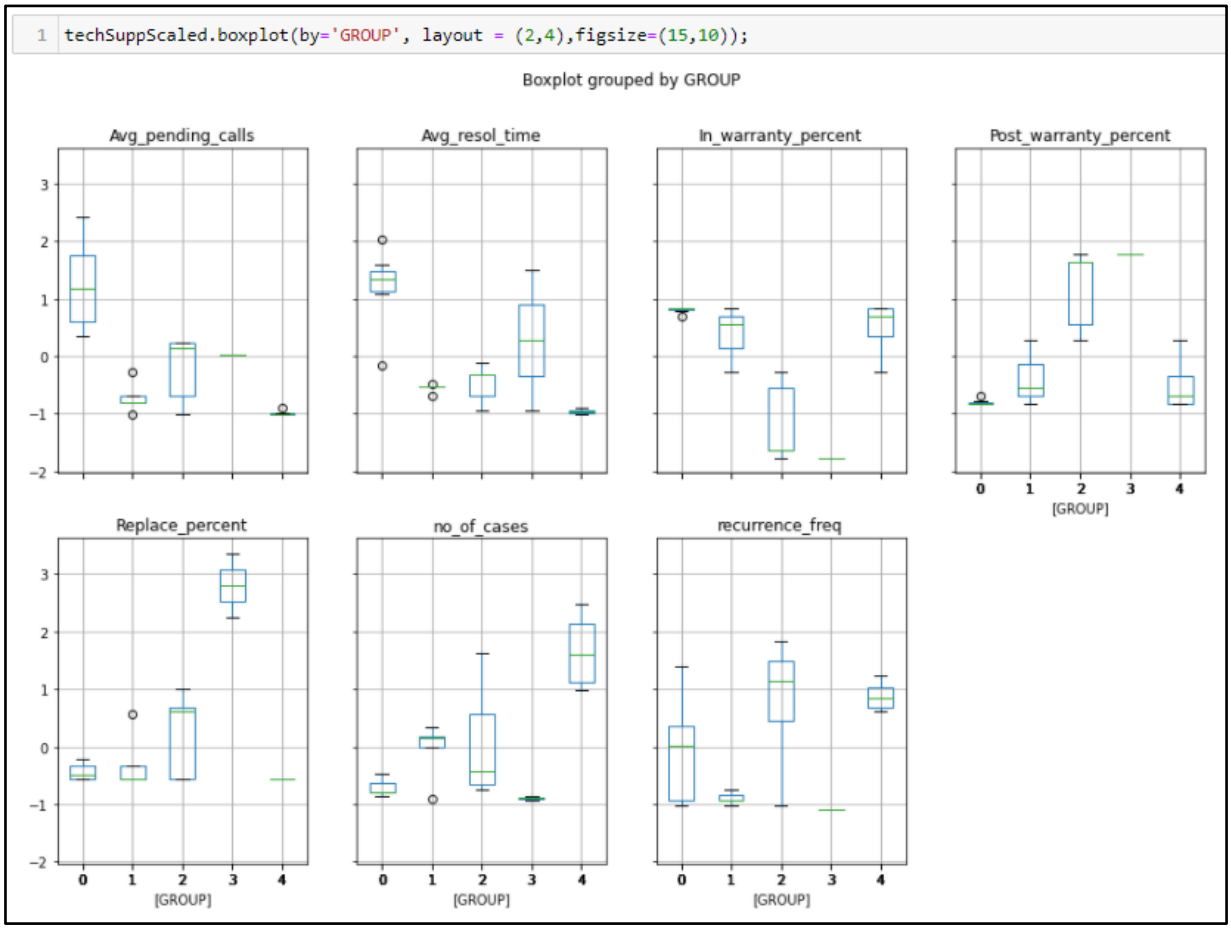

Figure 3.2.5.3: Box plot of each variable explained with cluster

4. Conclusion

As a conclusion for this project, we have learnt the application of unsupervised machine learning model in grouping unlabelled data and gain insights from it. In particular, we have built up some fundamentals in K-means clustering model and how different value of cluster will affect the output.