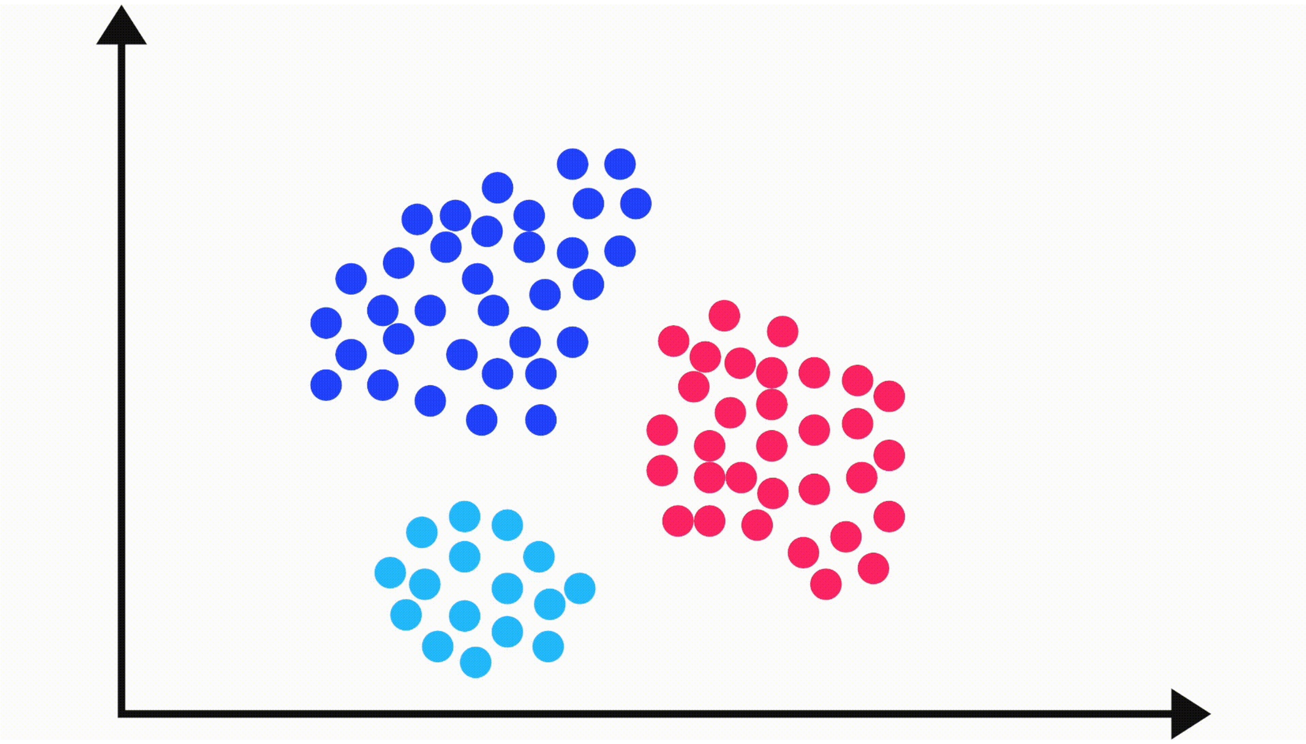
Clustering



In [1]:
```
import pandas as pd
import numpy as np
import seaborn as snn
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
data = pd.read_csv("weather.csv",parse_dates=True,index_col=0)
data.head()
```

Out[2]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	...	Hu
Date												
2008-02-01	19.5	22.4	15.6	6.2	0.0	NaN	NaN	S	SSW	17.0	...	
2008-02-02	19.5	25.6	6.0	3.4	2.7	NaN	NaN	W	E	9.0	...	
2008-02-03	21.6	24.5	6.6	2.4	0.1	NaN	NaN	ESE	ESE	17.0	...	
2008-02-04	20.2	22.8	18.8	2.2	0.0	NaN	NaN	NNE	E	22.0	...	
2008-02-05	19.7	25.7	77.4	NaN	0.0	NaN	NaN	NNE	W	11.0	...	

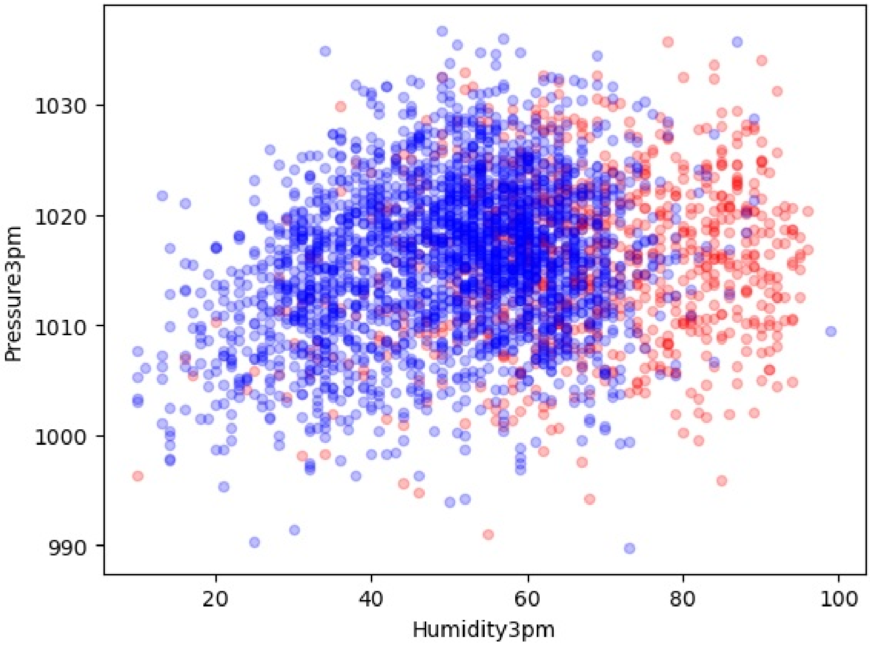
5 rows × 22 columns

In [3]:
```
dataset = data[["Humidity3pm",'Pressure3pm','RainTomorrow']]
dataset.head()
```

Out[3]:

	Humidity3pm	Pressure3pm	RainTomorrow
Date			
2008-02-01	84.0	1017.4	Yes
2008-02-02	73.0	1016.4	Yes
2008-02-03	86.0	1015.6	Yes
2008-02-04	90.0	1011.8	Yes
2008-02-05	74.0	1004.8	Yes

In [4]:
```
fig, ax = plt.subplots()

dataset[dataset['RainTomorrow'] == "Yes"].plot.scatter(x= 'Humidity3pm', y ="Pressure3pm" , c="r",ax = ax,alpha
dataset[dataset['RainTomorrow'] == "No"].plot.scatter(x= 'Humidity3pm', y ="Pressure3pm" , c="b",ax = ax, alpha
```



KNeighborsClassifier

In [5]:
```
dataset_clean = dataset.dropna()
```

In [6]:
```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

In [7]:
```
X = dataset_clean[['Humidity3pm','Pressure3pm']]
y = dataset_clean['RainTomorrow']
y = np.array([0 if value == 'No' else 1 for value in y])
y
```

Out[7]:
```
array([1, 1, 1, ..., 0, 0, 0])
```

In [8]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=.20,random_state=21)
```

In [9]:
```
neigh = KNeighborsClassifier()
neigh.fit(X_train,y_train)
y_pred = neigh.predict(X_test)
print(accuracy_score(y_pred,y_test))
```

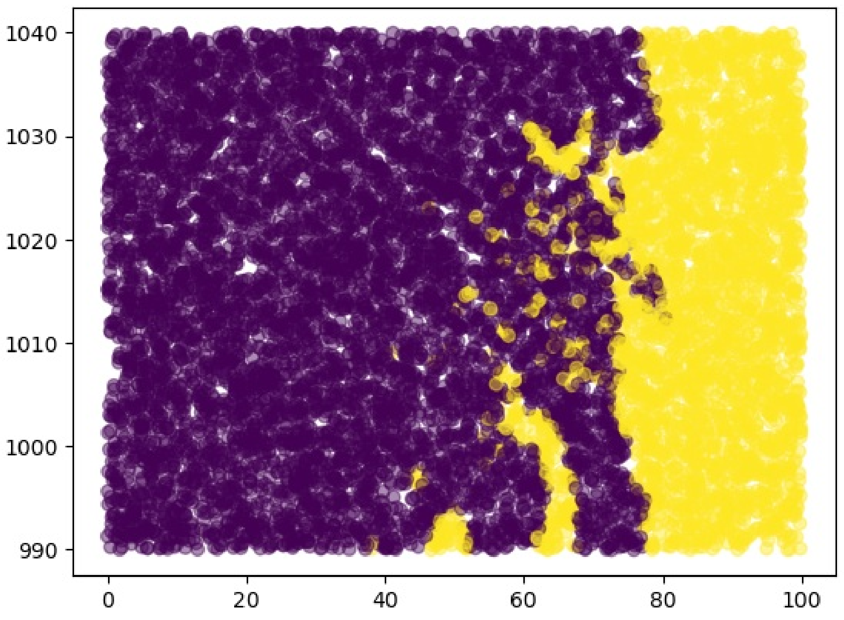
```
0.8157099697885196
```

In [10]:
```
X_map = np.random.rand(10000, 2)
X_map = X_map*(100, 50) + (0,990)
```

In [11]:
```
X_map
```

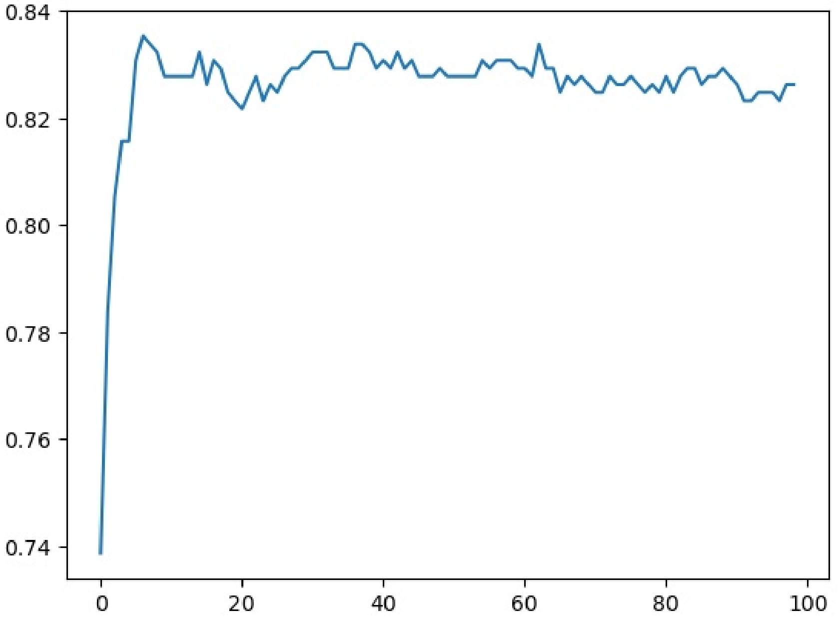
Out[11]:
```
array([[  61.14645808, 1002.87655364],
       [  31.24233326, 1001.08613587],
       [   9.1835381 , 1018.70484943],
       ...,
       [  37.57270749, 1026.40683756],
       [  69.01216072, 1031.25492823],
       [  23.19827575, 1009.43235919]])
```

In [12]:
```
fig, ax = plt.subplots()

y_map = neigh.predict(X_map)

ax.scatter(x=X_map[:,0], y= X_map[:,1],c=y_map,alpha=.45);
```



In [13]:
```
scores = []

for k in range (1,100):
    neigh = KNeighborsClassifier(n_neighbors=k)
    neigh.fit(X_train,y_train)
    y_pred = neigh.predict(X_test)
    score = accuracy_score(y_pred,y_test)
    scores.append(score)
```

In [14]:
```
fig, ax = plt.subplots()
ax.plot(scores);
```



K Nearest Neighbour Classifier

In [15]:
```
data = pd.read_csv("weather.csv",parse_dates=True,index_col=0)
data.head(2)
```

Out[15]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	...	Hu
Date												
2008-02-01	19.5	22.4	15.6	6.2	0.0	NaN	NaN	S	SSW	17.0	...	
2008-02-02	19.5	25.6	6.0	3.4	2.7	NaN	NaN	W	E	9.0	...	

2 rows × 22 columns

In [16]:
```
columns = ["Humidity3pm",'Pressure3pm',"Cloud3pm",'RainTomorrow']
dataset = data[columns]
```

```
dataset.head(2)
```

Out[16]:

Date	Humidity3pm	Pressure3pm	Cloud3pm	RainTomorrow
2008-02-01	84.0	1017.4	8.0	Yes
2008-02-02	73.0	1016.4	7.0	Yes

In [17]:
```
dataset_clean = dataset.dropna()
```

In [18]:
```
len(dataset), len(dataset_clean)
```

Out[18]:
```
(3337, 2754)
```

In [19]:
```
X = dataset_clean.drop("RainTomorrow",axis=1)
y = dataset_clean['RainTomorrow']
```

In [20]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
le.fit_transform(y)
y = le.transform(y)
```

In [21]:
```
X_train,X_test,y_train,y_test =train_test_split(X,y,random_state=21)
```

In [22]:
```
neigh = KNeighborsClassifier(n_neighbors=k)
neigh.fit(X_train,y_train)
y_pred = neigh.predict(X_test)
score = accuracy_score(y_pred,y_test)
score.round(3)
```

Out[22]:
```
0.836
```

Preceptron

In [23]:
```
import pandas as pd
import numpy as np
import seaborn as snn
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings("ignore")
```

In [24]:
```
data = pd.read_csv("weather.csv",parse_dates=True,index_col=0)
data.head(2)
```

Out[24]:

Date	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	...	Hu
2008-02-01	19.5	22.4	15.6	6.2	0.0	NaN	NaN	S	SSW	17.0	...	
2008-02-02	19.5	25.6	6.0	3.4	2.7	NaN	NaN	W	E	9.0	...	

2 rows × 22 columns

In [25]:
```
dataset = data[["Humidity3pm",'Pressure3pm','RainTomorrow']].dropna()
dataset.head()
```

Out[25]:

Date	Humidity3pm	Pressure3pm	RainTomorrow
2008-02-01	84.0	1017.4	Yes
2008-02-02	73.0	1016.4	Yes
2008-02-03	86.0	1015.6	Yes
2008-02-04	90.0	1011.8	Yes
2008-02-05	74.0	1004.8	Yes

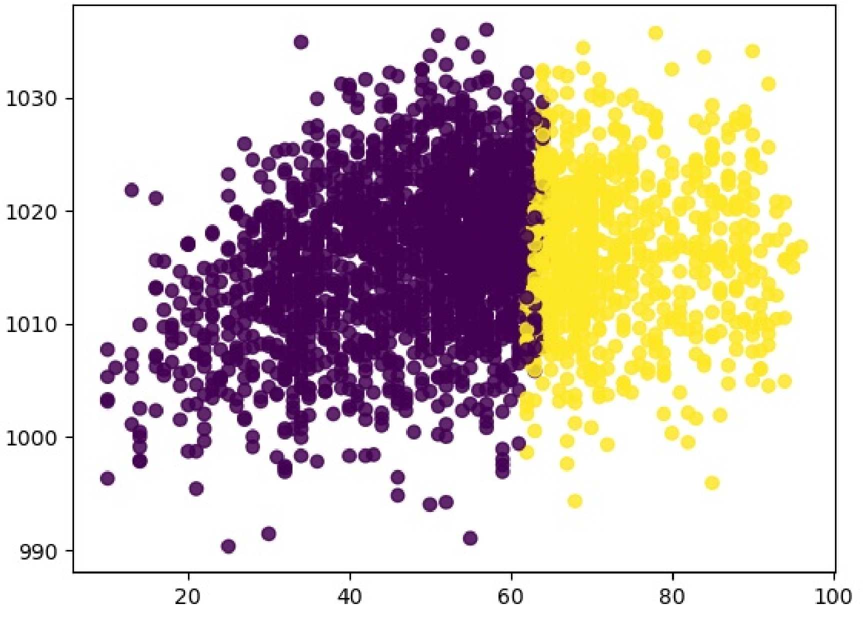
In [26]:
```
X = dataset_clean.drop("RainTomorrow",axis=1)
y = dataset_clean['RainTomorrow']

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
le.fit_transform(y)
y = le.transform(y)

X_train,X_test,y_train,y_test =train_test_split(X,y,random_state=21)
```

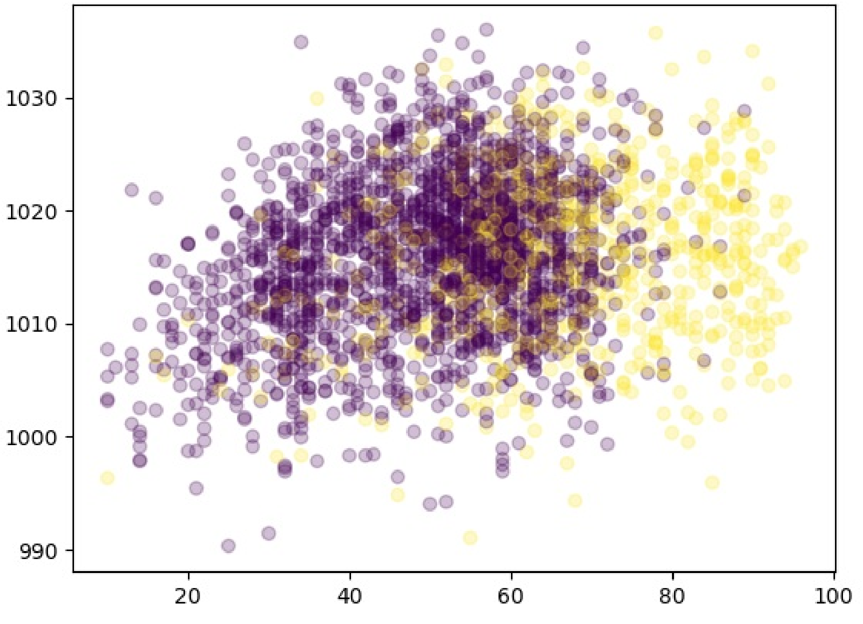
In [27]:
```
from sklearn.linear_model import Perceptron
clf = Perceptron(random_state=21)
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
accuracy_score(y_test,y_pred)
```

Out[27]:
```
0.7968069666182874
```

In [28]:
```
fig, ax =  plt.subplots()
X_data = X.to_numpy()

y_all = clf.predict(X)
ax.scatter(x=X['Humidity3pm'],y=X['Pressure3pm'],c = y_all,alpha=.85);
```



In [29]:
```
fig, ax =  plt.subplots()
ax.scatter(x=X['Humidity3pm'], y=X['Pressure3pm'], c = y, alpha = .25);
```



Multiple Perceptron

In [30]:
```
import pandas as pd
import numpy as np
import seaborn as snn
import matplotlib.pyplot as plt
from sklearn.linear_model import Perceptron
%matplotlib inline

import warnings
```

```
warnings.filterwarnings("ignore")
```

In [31]:
```
data = pd.read_csv("weather.csv",parse_dates=True,index_col=0)
data.head(2)
```

Out[31]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	...	Hu
Date												
2008-02-01	19.5	22.4	15.6	6.2	0.0	NaN	NaN	S	SSW	17.0	...	
2008-02-02	19.5	25.6	6.0	3.4	2.7	NaN	NaN	W	E	9.0	...	

2 rows × 22 columns

In [32]:
```
data.isnull().sum()
```

Out[32]:
```
MinTemp             3
MaxTemp             2
Rainfall            6
Evaporation        51
Sunshine           16
WindGustDir      1036
WindGustSpeed    1036
WindDir9am         56
WindDir3pm         33
WindSpeed9am       26
WindSpeed3pm       25
Humidity9am        14
Humidity3pm        13
Pressure9am        20
Pressure3pm        19
Cloud9am          566
Cloud3pm          561
Temp9am             4
Temp3pm             4
RainToday           6
RISK_MM             0
RainTomorrow        0
dtype: int64
```

In [33]:
```
dataset_clean = data.drop(["WindGustDir",'WindGustSpeed','WindGustSpeed','WindDir9am','WindDir3pm','RainToday']
dataset_clean.head()
```

Out[33]:

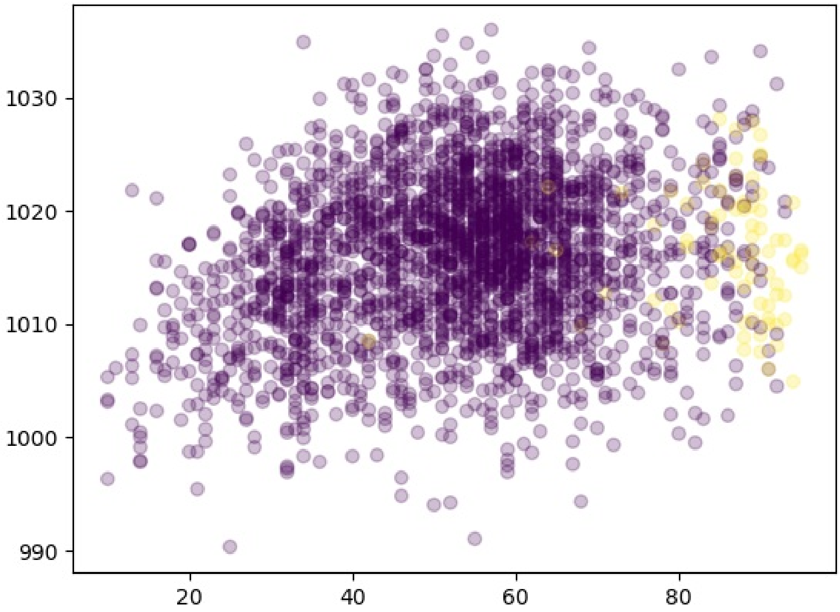
	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pres
Date											
2008-02-01	19.5	22.4	15.6	6.2	0.0	17.0	20.0	92.0	84.0	1017.6	
2008-02-02	19.5	25.6	6.0	3.4	2.7	9.0	13.0	83.0	73.0	1017.9	
2008-02-03	21.6	24.5	6.6	2.4	0.1	17.0	2.0	88.0	86.0	1016.7	
2008-02-04	20.2	22.8	18.8	2.2	0.0	22.0	20.0	83.0	90.0	1014.2	
2008-02-06	20.2	27.2	1.6	2.6	8.6	9.0	22.0	69.0	62.0	1002.7	

In [34]:
```
X = dataset_clean.drop("RainTomorrow",axis=1)
y = dataset_clean['RainTomorrow']

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

le.fit_transform(y)
y = le.transform(y)

X_train,X_test,y_train,y_test =train_test_split(X,y,random_state=21)
```

In [35]:
```
clf = Perceptron(random_state=21)
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
accuracy_score(y_pred,y_test)
```

Out[35]:
```
0.783661119515885
```

In [36]:
```
fig, ax = plt.subplots()

y_pred = clf.predict(X)
ax.scatter(x=X['Humidity3pm'], y=X['Pressure3pm'], c = y_pred, alpha =.25);
```



K-means Clustring

Why K-means clustering for this dataset?

In this scenario we will attempt to find groups which have not been explicitly labeled in the data.

- *Choose the number of K clusters*
- *Select random centroids*
- *Assign each data point the closest centroid*
- *Compute and place the new centroid of each cluster*
- *Reassign the data points to the new closest k centroid till no more reassignment*

In [37]:
```
# visualisation
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import matplotlib.pyplot as plt
```

In [38]:
```
data = pd.read_csv("weather.csv")
data.head(1)
```

Out[38]:

	Date	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	...	Humidity3pm	P
0	2008-02-01	19.5	22.4	15.6	6.2	0.0	NaN	NaN	S	SSW	...	84.0	

1 rows × 23 columns

In [39]:
```
df = data[['RainTomorrow',"Humidity3pm",'Pressure3pm',"WindSpeed3pm"]]
df = df.dropna()
df.head()
```

Out[39]:

	RainTomorrow	Humidity3pm	Pressure3pm	WindSpeed3pm
0	Yes	84.0	1017.4	20.0
1	Yes	73.0	1016.4	13.0
2	Yes	86.0	1015.6	2.0
3	Yes	90.0	1011.8	20.0
4	Yes	74.0	1004.8	6.0

In [40]:
```
df.describe()
```

Out[40]:

	Humidity3pm	Pressure3pm	WindSpeed3pm
count	3286.000000	3286.000000	3286.000000
mean	54.682897	1015.998019	19.325928
std	16.271008	7.021456	7.494735
min	10.000000	989.800000	0.000000
25%	44.000000	1011.300000	15.000000
50%	56.000000	1016.300000	19.000000
75%	64.000000	1020.800000	24.000000
max	99.000000	1036.700000	57.000000

In [41]:

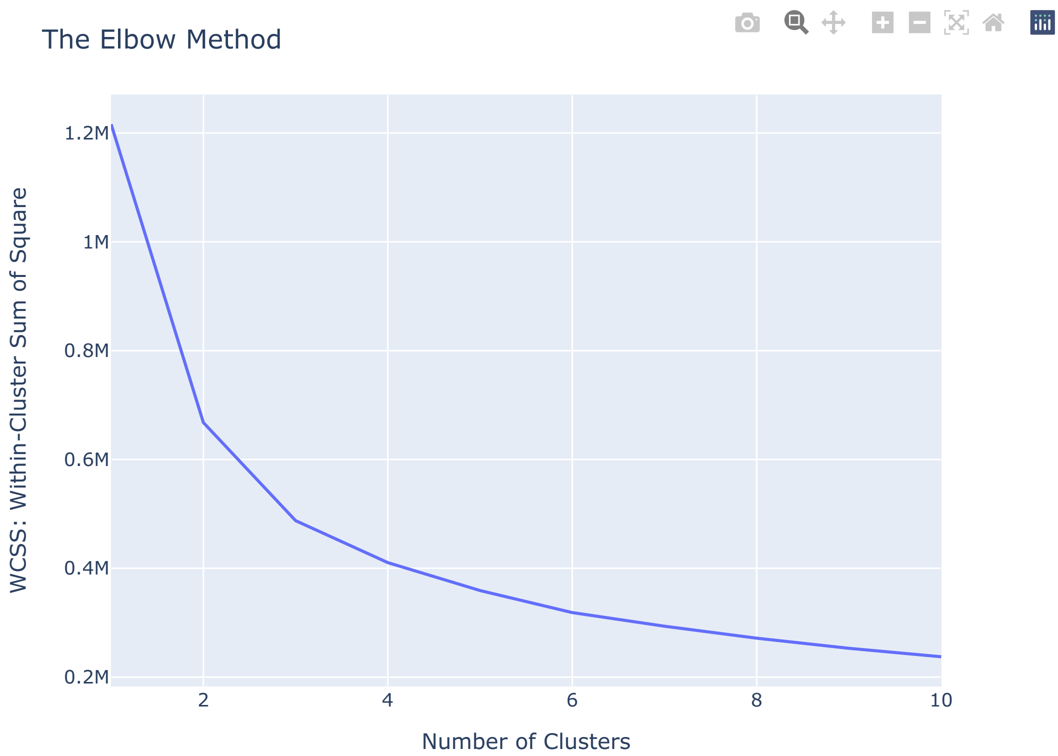
```
X = df.iloc[:, 1:4].values
X
```

Out[41]:

```
array([[  84. , 1017.4,   20. ],
       [  73. , 1016.4,   13. ],
       [  86. , 1015.6,    2. ],
       ...,
       [  56. , 1015. ,   13. ],
       [  35. , 1015.1,   19. ],
       [  32. , 1015.4,   13. ]])
```

In [42]:

```
from sklearn.cluster import KMeans

wcss = []

for i in range(1,11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 1)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

fig = px.line(x=range(1,11), y=wcss)

# edit the layout
fig.update_layout(title='The Elbow Method',
                   xaxis_title='Number of Clusters',
                   yaxis_title='WCSS: Within-Cluster Sum of Square')

fig.show()
```



In [43]:

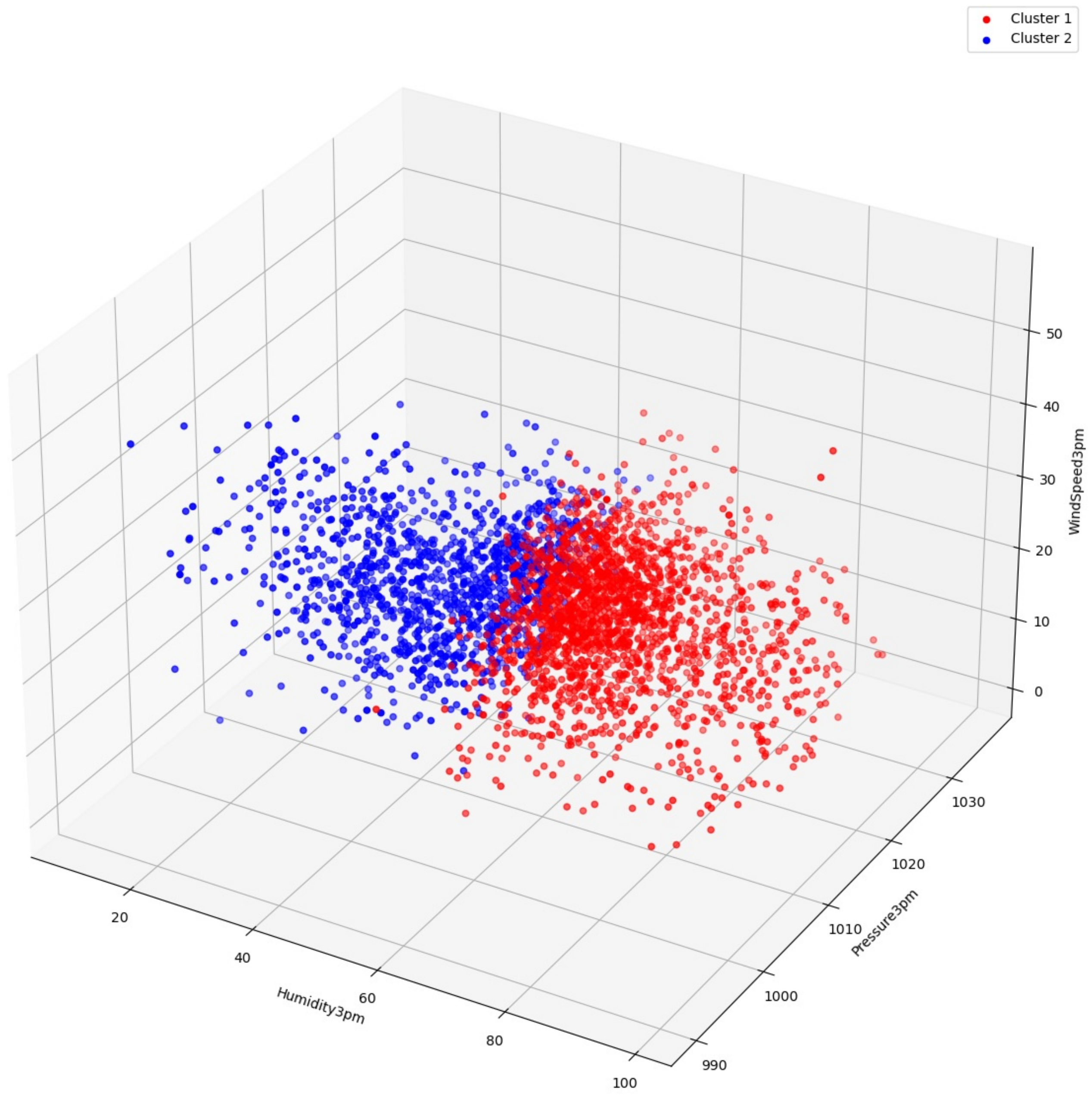
```
kmeans = KMeans(n_clusters = 2, init = 'k-means++', random_state = 1)
y_kmeans = kmeans.fit_predict(X)
print(y_kmeans)
```

```
[0 0 0 ... 0 1 1]
```

In [44]:

```
fig = plt.figure(figsize = (15,15), dpi=100)

ax = fig.add_subplot(111, projection='3d')
```

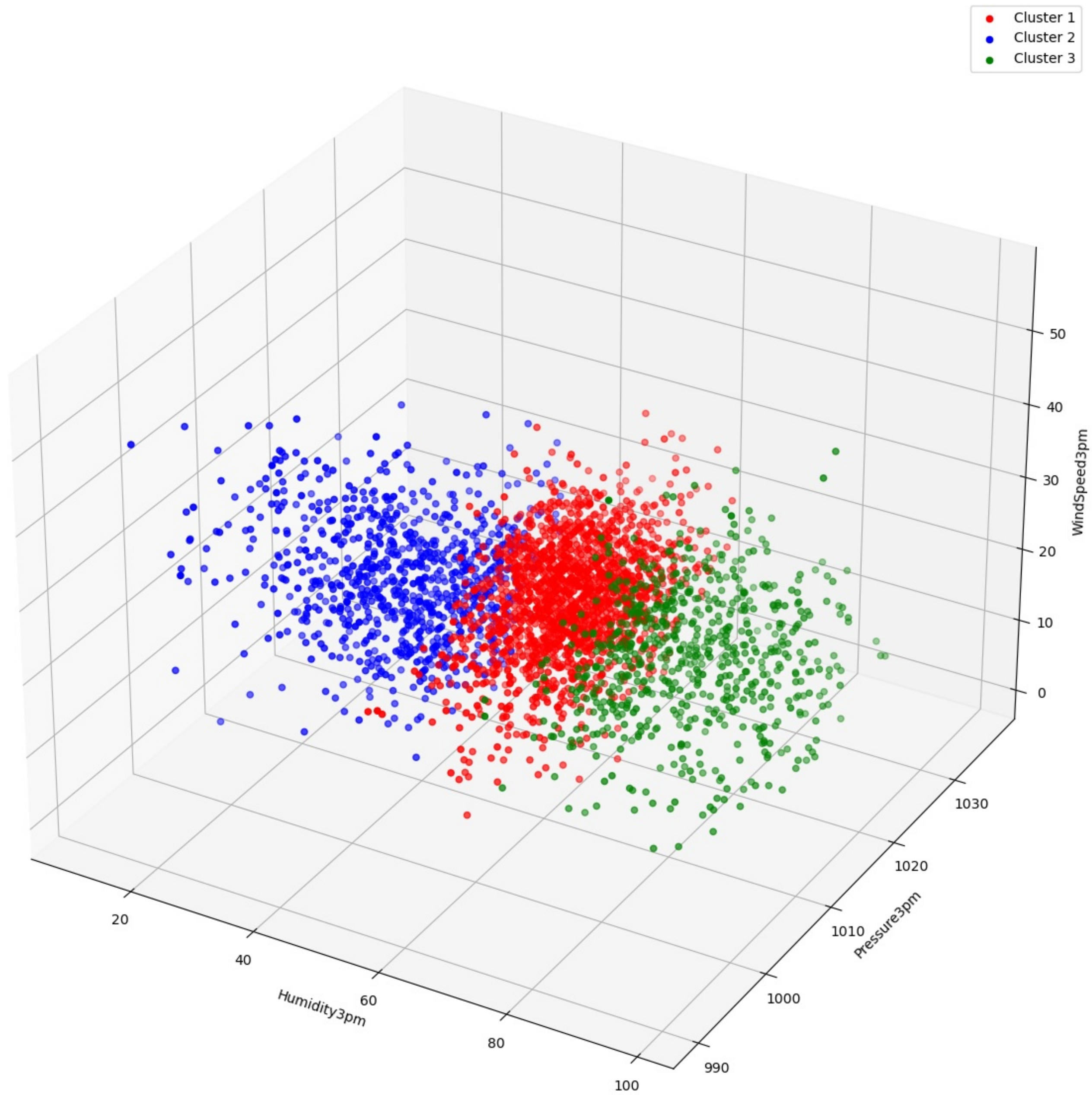
```
ax.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], X[y_kmeans == 0, 2], s = 20, c = 'Red', label = 'Cluster 1
ax.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], X[y_kmeans == 1, 2], s = 20, c = 'blue', label = 'Cluster
#ax.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], X[y_kmeans == 2, 2], s = 20, c = 'green', label = 'Cluste

ax.set_xlabel('Humidity3pm')
ax.set_ylabel('Pressure3pm')
ax.set_zlabel('WindSpeed3pm')

ax.legend()
plt.show()
```



In [45]:
```
kmeans = KMeans(n_clusters = 3, init = 'k-means++', random_state = 1)
y_kmeans = kmeans.fit_predict(X)

print(y_kmeans)
```

```
[2 2 2 ... 0 1 1]
```

In [46]:
```
fig = plt.figure(figsize = (15,15), dpi=100)

ax = fig.add_subplot(111, projection='3d')

ax.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], X[y_kmeans == 0, 2], s = 20, c = 'Red', label = 'Cluster 1
ax.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], X[y_kmeans == 1, 2], s = 20, c = 'blue', label = 'Cluster
ax.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], X[y_kmeans == 2, 2], s = 20, c = 'green', label = 'Cluster

ax.set_xlabel('Humidity3pm')
ax.set_ylabel('Pressure3pm')
ax.set_zlabel('WindSpeed3pm')

ax.legend()
plt.show()
```



Iris Data Set KMean Clustring

In [47]:
```
# visualisation
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import matplotlib.pyplot as plt
```

In [48]:
```
data = pd.read_csv("Iris.csv")
data = data.drop('Id',axis=1)
data.head(1)
```

Out[48]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa

In [49]:
```
X = data.iloc[:, 0:4].values
X
```

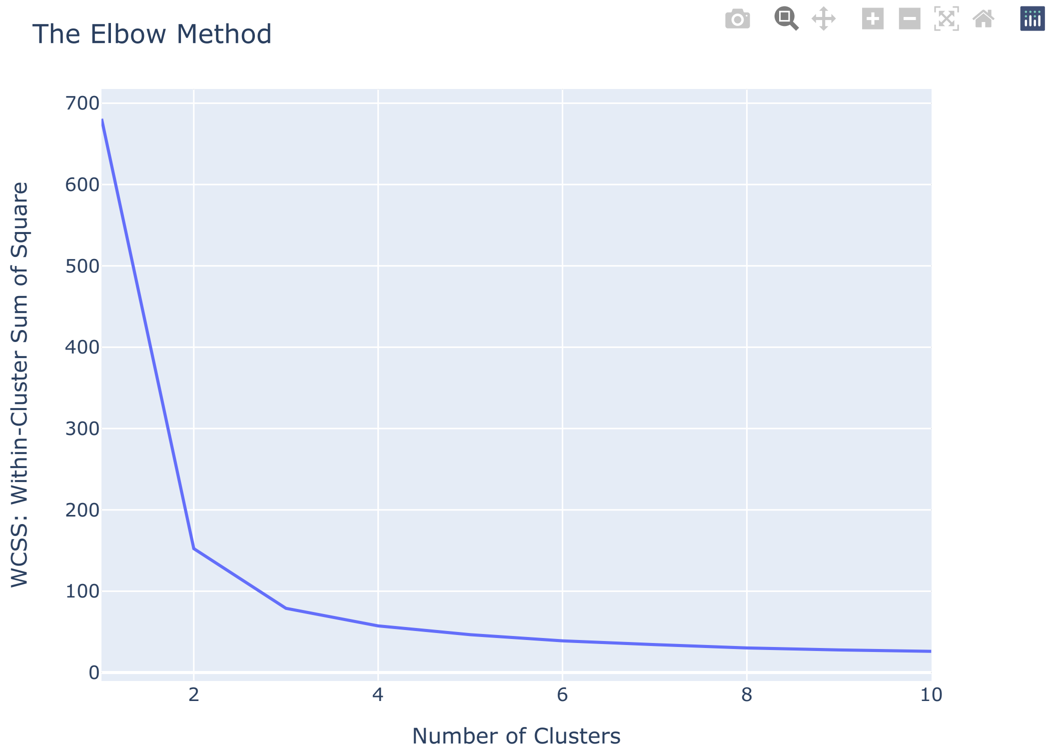
Out[49]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
```

```
[5.4, 3.7, 1.5, 0.2],
[4.8, 3.4, 1.6, 0.2],
[4.8, 3. , 1.4, 0.1],
[4.3, 3. , 1.1, 0.1],
[5.8, 4. , 1.2, 0.2],
[5.7, 4.4, 1.5, 0.4],
[5.4, 3.9, 1.3, 0.4],
[5.1, 3.5, 1.4, 0.3],
[5.7, 3.8, 1.7, 0.3],
[5.1, 3.8, 1.5, 0.3],
[5.4, 3.4, 1.7, 0.2],
[5.1, 3.7, 1.5, 0.4],
[4.6, 3.6, 1. , 0.2],
[5.1, 3.3, 1.7, 0.5],
[4.8, 3.4, 1.9, 0.2],
[5. , 3. , 1.6, 0.2],
[5. , 3.4, 1.6, 0.4],
[5.2, 3.5, 1.5, 0.2],
[5.2, 3.4, 1.4, 0.2],
[4.7, 3.2, 1.6, 0.2],
[4.8, 3.1, 1.6, 0.2],
[5.4, 3.4, 1.5, 0.4],
[5.2, 4.1, 1.5, 0.1],
[5.5, 4.2, 1.4, 0.2],
[4.9, 3.1, 1.5, 0.1],
[5. , 3.2, 1.2, 0.2],
[5.5, 3.5, 1.3, 0.2],
[4.9, 3.1, 1.5, 0.1],
[4.4, 3. , 1.3, 0.2],
[5.1, 3.4, 1.5, 0.2],
[5. , 3.5, 1.3, 0.3],
[4.5, 2.3, 1.3, 0.3],
[4.4, 3.2, 1.3, 0.2],
[5. , 3.5, 1.6, 0.6],
[5.1, 3.8, 1.9, 0.4],
[4.8, 3. , 1.4, 0.3],
[5.1, 3.8, 1.6, 0.2],
[4.6, 3.2, 1.4, 0.2],
[5.3, 3.7, 1.5, 0.2],
[5. , 3.3, 1.4, 0.2],
[7. , 3.2, 4.7, 1.4],
[6.4, 3.2, 4.5, 1.5],
[6.9, 3.1, 4.9, 1.5],
[5.5, 2.3, 4. , 1.3],
[6.5, 2.8, 4.6, 1.5],
[5.7, 2.8, 4.5, 1.3],
[6.3, 3.3, 4.7, 1.6],
[4.9, 2.4, 3.3, 1. ],
[6.6, 2.9, 4.6, 1.3],
[5.2, 2.7, 3.9, 1.4],
[5. , 2. , 3.5, 1. ],
[5.9, 3. , 4.2, 1.5],
[6. , 2.2, 4. , 1. ],
[6.1, 2.9, 4.7, 1.4],
[5.6, 2.9, 3.6, 1.3],
[6.7, 3.1, 4.4, 1.4],
[5.6, 3. , 4.5, 1.5],
[5.8, 2.7, 4.1, 1. ],
[6.2, 2.2, 4.5, 1.5],
[5.6, 2.5, 3.9, 1.1],
[5.9, 3.2, 4.8, 1.8],
[6.1, 2.8, 4. , 1.3],
[6.3, 2.5, 4.9, 1.5],
[6.1, 2.8, 4.7, 1.2],
[6.4, 2.9, 4.3, 1.3],
[6.6, 3. , 4.4, 1.4],
[6.8, 2.8, 4.8, 1.4],
[6.7, 3. , 5. , 1.7],
[6. , 2.9, 4.5, 1.5],
[5.7, 2.6, 3.5, 1. ],
[5.5, 2.4, 3.8, 1.1],
[5.5, 2.4, 3.7, 1. ],
[5.8, 2.7, 3.9, 1.2],
[6. , 2.7, 5.1, 1.6],
[5.4, 3. , 4.5, 1.5],
[6. , 3.4, 4.5, 1.6],
[6.7, 3.1, 4.7, 1.5],
[6.3, 2.3, 4.4, 1.3],
[5.6, 3. , 4.1, 1.3],
[5.5, 2.5, 4. , 1.3],
[5.5, 2.6, 4.4, 1.2],
[6.1, 3. , 4.6, 1.4],
[5.8, 2.6, 4. , 1.2],
[5. , 2.3, 3.3, 1. ],
[5.6, 2.7, 4.2, 1.3],
[5.7, 3. , 4.2, 1.2],
[5.7, 2.9, 4.2, 1.3],
[6.2, 2.9, 4.3, 1.3],
[5.1, 2.5, 3. , 1.1],
```

```
       [5.7, 2.8, 4.1, 1.3],
       [6.3, 3.3, 6. , 2.5],
       [5.8, 2.7, 5.1, 1.9],
       [7.1, 3. , 5.9, 2.1],
       [6.3, 2.9, 5.6, 1.8],
       [6.5, 3. , 5.8, 2.2],
       [7.6, 3. , 6.6, 2.1],
       [4.9, 2.5, 4.5, 1.7],
       [7.3, 2.9, 6.3, 1.8],
       [6.7, 2.5, 5.8, 1.8],
       [7.2, 3.6, 6.1, 2.5],
       [6.5, 3.2, 5.1, 2. ],
       [6.4, 2.7, 5.3, 1.9],
       [6.8, 3. , 5.5, 2.1],
       [5.7, 2.5, 5. , 2. ],
       [5.8, 2.8, 5.1, 2.4],
       [6.4, 3.2, 5.3, 2.3],
       [6.5, 3. , 5.5, 1.8],
       [7.7, 3.8, 6.7, 2.2],
       [7.7, 2.6, 6.9, 2.3],
       [6. , 2.2, 5. , 1.5],
       [6.9, 3.2, 5.7, 2.3],
       [5.6, 2.8, 4.9, 2. ],
       [7.7, 2.8, 6.7, 2. ],
       [6.3, 2.7, 4.9, 1.8],
       [6.7, 3.3, 5.7, 2.1],
       [7.2, 3.2, 6. , 1.8],
       [6.2, 2.8, 4.8, 1.8],
       [6.1, 3. , 4.9, 1.8],
       [6.4, 2.8, 5.6, 2.1],
       [7.2, 3. , 5.8, 1.6],
       [7.4, 2.8, 6.1, 1.9],
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [6.4, 3.1, 5.5, 1.8],
       [6. , 3. , 4.8, 1.8],
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [6.8, 3.2, 5.9, 2.3],
       [6.7, 3.3, 5.7, 2.5],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]])
```

In [50]:

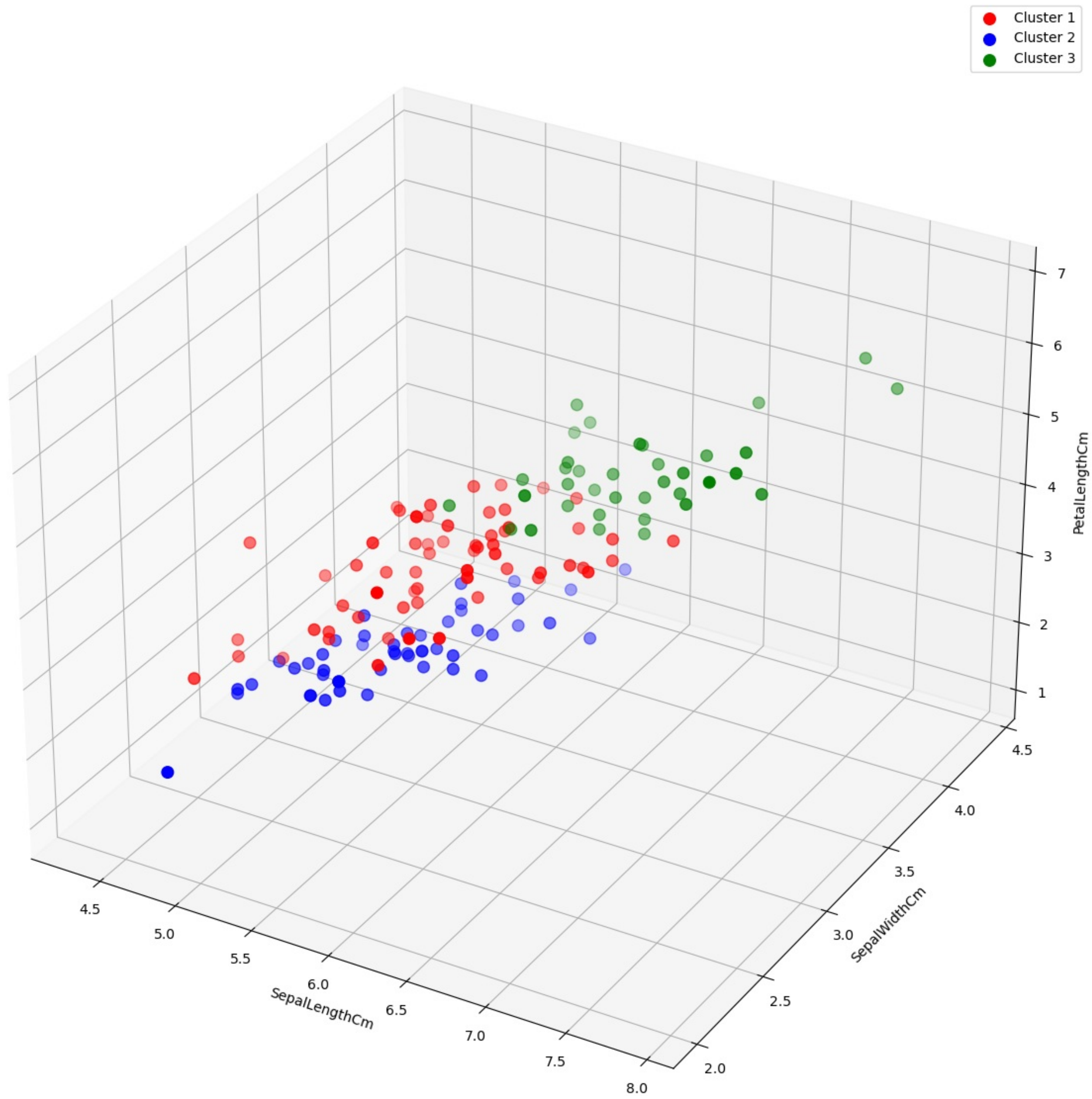
```
from sklearn.cluster import KMeans

wcss = []

for i in range(1,11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 1)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

fig = px.line(x=range(1,11), y=wcss)

# edit the layout
fig.update_layout(title='The Elbow Method',
                   xaxis_title='Number of Clusters',
                   yaxis_title='WCSS: Within-Cluster Sum of Square')

fig.show()
```



In [51]:
```
kmeans = KMeans(n_clusters = 3, init = 'k-means++', random_state = 1)
y_kmeans = kmeans.fit_predict(X)

print(y_kmeans)
```

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 2 2 2 0 2 2 2 2
 2 2 0 0 2 2 2 2 0 2 0 2 0 2 2 0 0 2 2 2 2 2 0 2 2 2 2 0 2 2 2 0 2 2 2 0 2
 2 0]
```

In [52]:
```
fig = plt.figure(figsize = (15,15), dpi=100)

ax = fig.add_subplot(111, projection='3d')

ax.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], X[y_kmeans == 0, 2], s = 70, c = 'Red', label = 'Cluster 1
ax.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], X[y_kmeans == 1, 2], s = 70, c = 'blue', label = 'Cluster
ax.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], X[y_kmeans == 2, 2], s = 70, c = 'green', label = 'Cluster

ax.set_xlabel('SepalLengthCm')
ax.set_ylabel('SepalWidthCm')
ax.set_zlabel('PetalLengthCm')

ax.legend()
plt.show()
```



In [53]:
```python
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import MinMaxScaler

data = pd.read_csv("Iris.csv")
data = data.drop('Id',axis=1)
data.head(1)
```

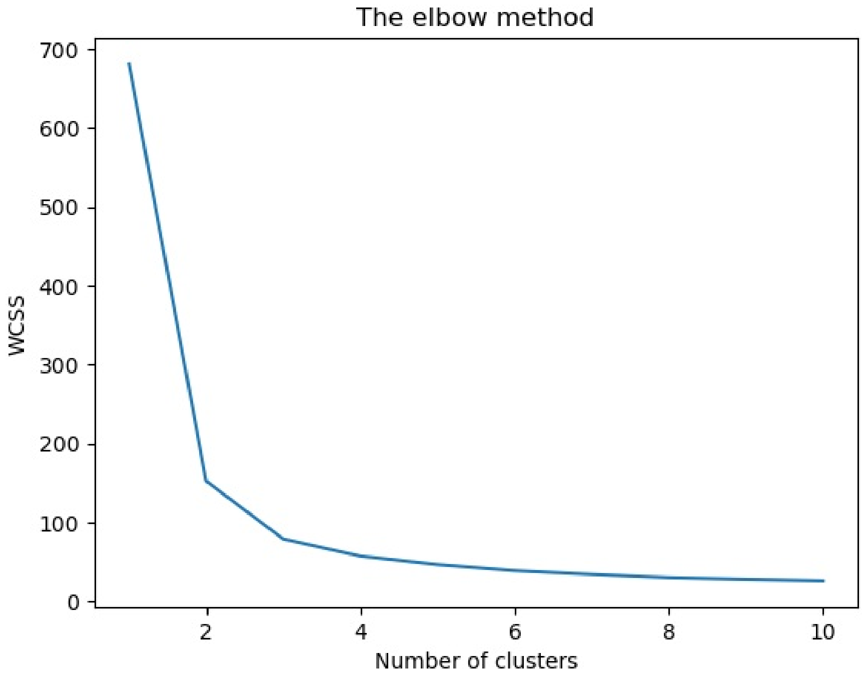
Out[53]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa

In [54]:
```python
x = data.iloc[:, [0, 1, 2, 3]].values
```

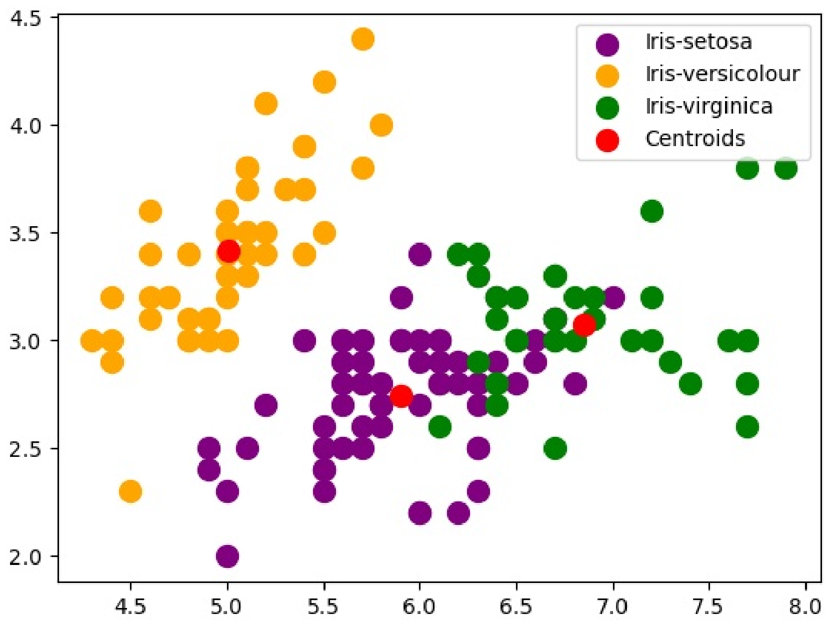
In [55]:
```python
#Finding the optimum number of clusters for k-means classification
from sklearn.cluster import KMeans
wcss = []

for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)
```

In [56]:
```python
plt.plot(range(1, 11), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') #within cluster sum of squares
plt.show()
```

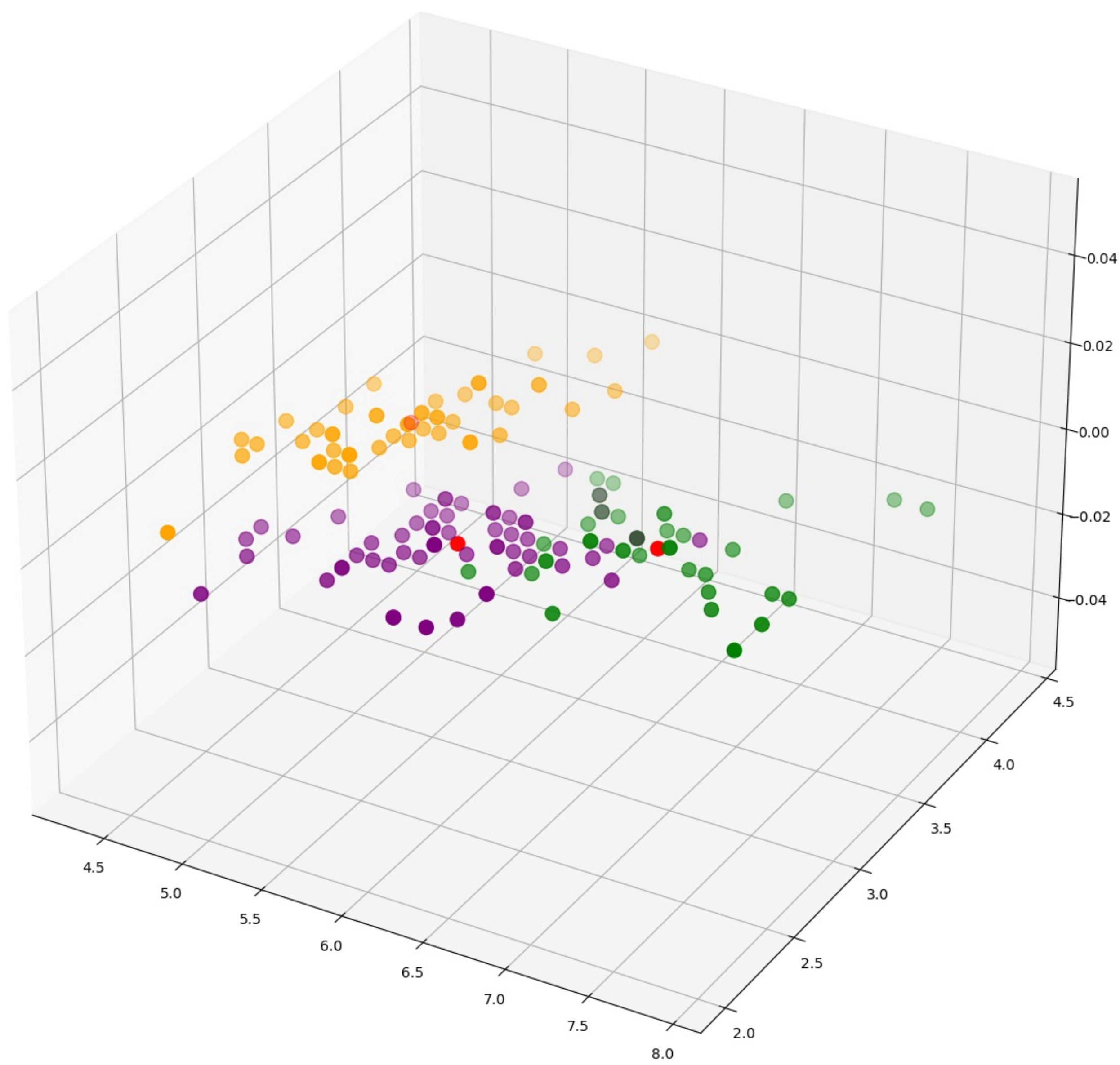


In [57]:
```
kmeans = KMeans(n_clusters = 3, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
y_kmeans = kmeans.fit_predict(x)
```

In [58]:
```
#Visualising the clusters
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 100, c = 'purple', label = 'Iris-setosa')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 100, c = 'orange', label = 'Iris-versicolour')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Iris-virginica')

#Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1], s = 100, c = 'red', label = 'Centroids

plt.legend();
```

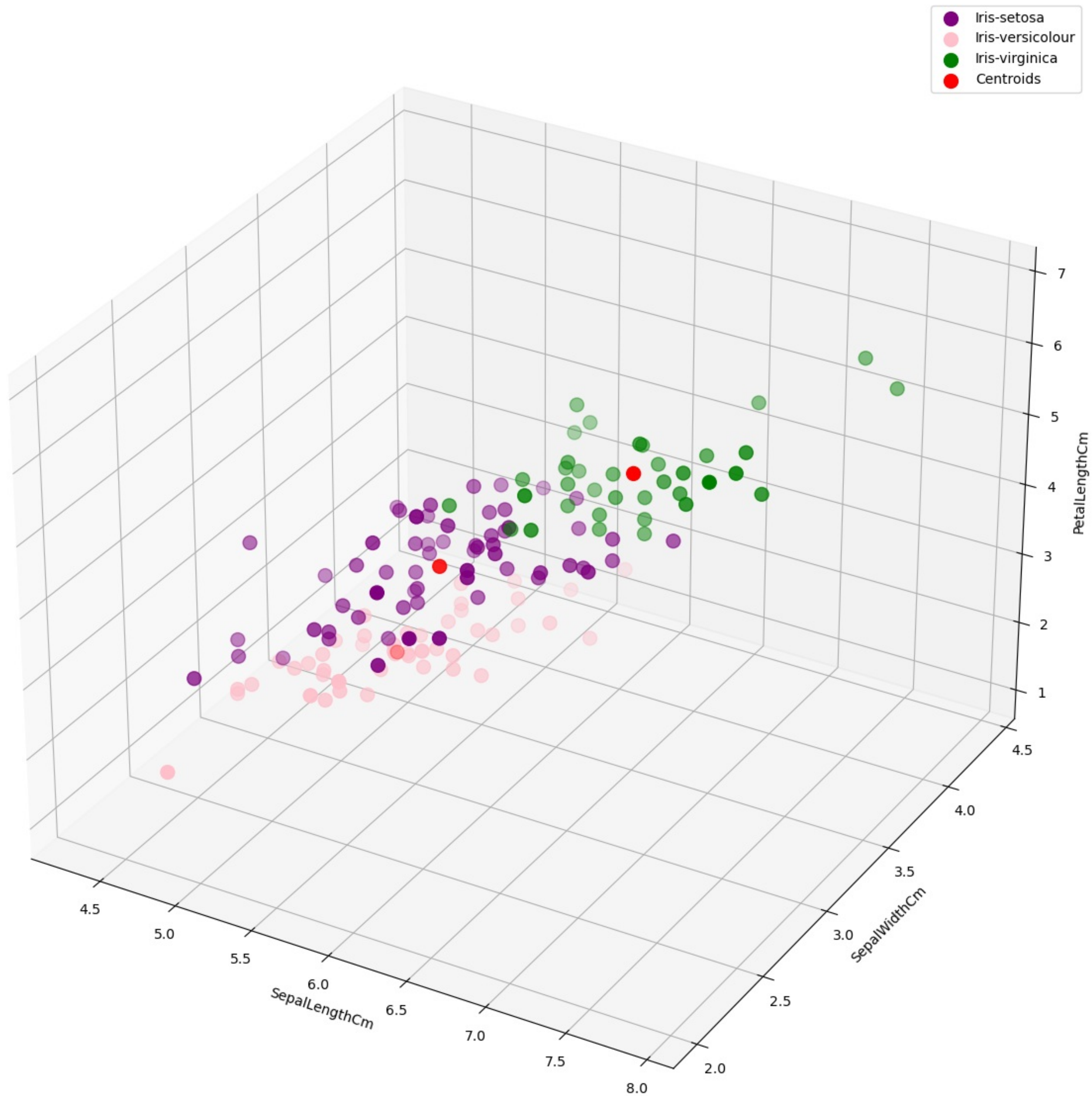


In [59]:
```
# 3d scatterplot using matplotlib

fig = plt.figure(figsize = (15,15))
ax = fig.add_subplot(111, projection='3d')
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 100, c = 'purple', label = 'Iris-setosa')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 100, c = 'orange', label = 'Iris-versicolour')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Iris-virginica')

#Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1], s = 100, c = 'red', label = 'Centroids
plt.show()
```



In [60]:

```
fig = plt.figure(figsize = (15,15), dpi=100)

ax = fig.add_subplot(111, projection='3d')


ax.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1],x[y_kmeans == 0, 2], s = 100, c = 'purple', label = 'Iris-s
ax.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1],x[y_kmeans == 1, 2], s = 100, c = 'pink', label = 'Iris-ver
ax.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1],x[y_kmeans == 2, 2], s = 100, c = 'green', label = 'Iris-vi

ax.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1],kmeans.cluster_centers_[:,2], s = 100, c

ax.set_xlabel('SepalLengthCm')
ax.set_ylabel('SepalWidthCm')
ax.set_zlabel('PetalLengthCm')

ax.legend()
plt.show()
```



K Mean Clustring Wine

In [61]:
```
# visualisation
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import matplotlib.pyplot as plt
```

In [62]:
```
wine = pd.read_csv('winequality-red.csv')
wine.head()
```

Out[62]:

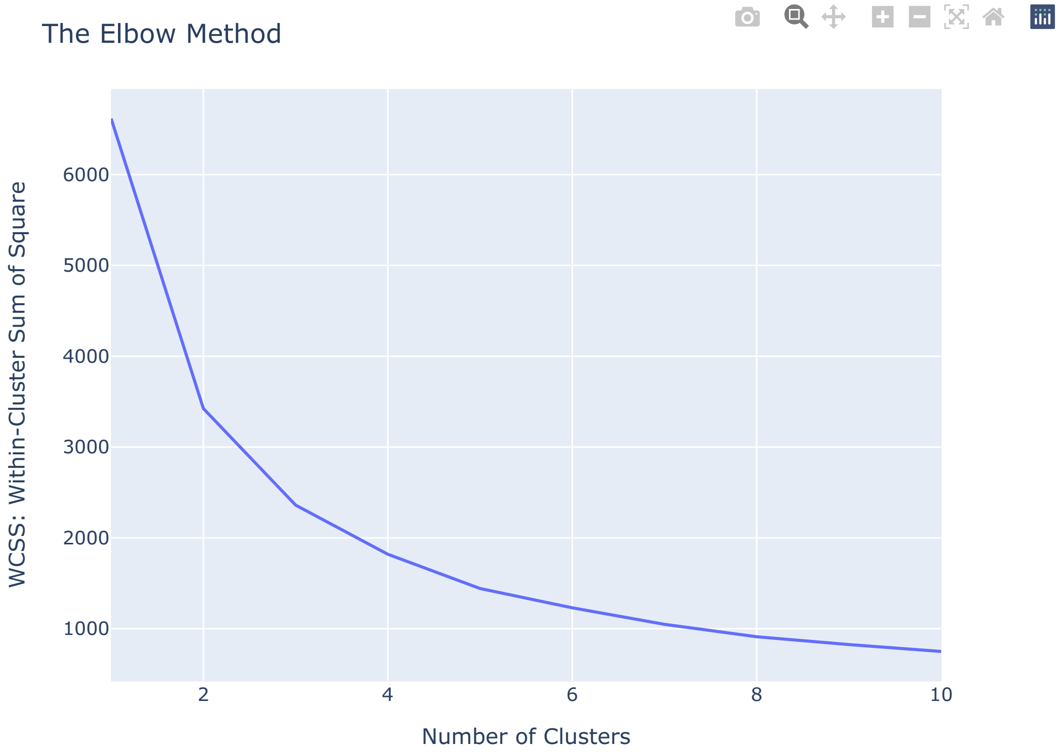
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

In [63]:
```
wine = wine[['alcohol','fixed acidity','pH']]
```

In [64]:
```
X = wine.iloc[:, 0:11].values
X
```

Out[64]:
```
array([[ 9.4 ,  7.4 ,  3.51],
       [ 9.8 ,  7.8 ,  3.2 ],
       [ 9.8 ,  7.8 ,  3.26],
       ...,
       [11.  ,  6.3 ,  3.42],
       [10.2 ,  5.9 ,  3.57],
       [11.  ,  6.  ,  3.39]])
```

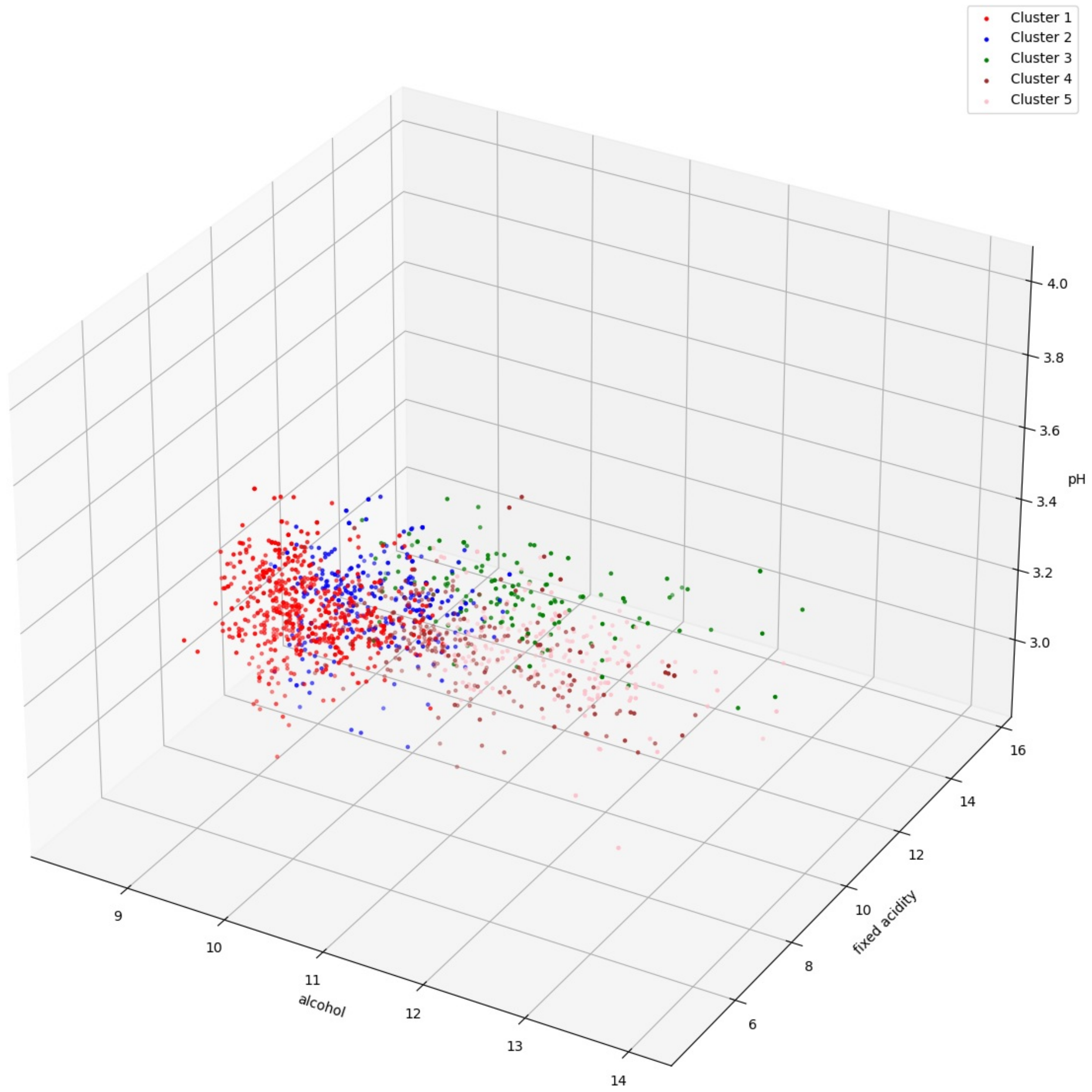
In [65]:
```
from sklearn.cluster import KMeans

wcss = []

for i in range(1,11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 1)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

fig = px.line(x=range(1,11), y=wcss)

# edit the layout
fig.update_layout(title='The Elbow Method',
                   xaxis_title='Number of Clusters',
                   yaxis_title='WCSS: Within-Cluster Sum of Square')

fig.show()
```



In [66]:
```
kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 1)
y_kmeans = kmeans.fit_predict(X)

print(y_kmeans)
```

```
[0 0 0 ... 3 3 3]
```

In [67]:
```
fig = plt.figure(figsize = (15,15), dpi=100)

ax = fig.add_subplot(111, projection='3d')

ax.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], X[y_kmeans == 0, 2], s = 5, c = 'Red', label = 'Cluster 1'
ax.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], X[y_kmeans == 1, 2], s = 5, c = 'blue', label = 'Cluster 2
ax.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], X[y_kmeans == 2, 2], s = 5, c = 'green', label = 'Cluster
ax.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], X[y_kmeans == 3, 2], s = 5, c = 'Brown', label = 'Cluster
ax.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], X[y_kmeans == 4, 2], s = 5, c = 'Pink', label = 'Cluster 5

ax.set_xlabel('alcohol')
ax.set_ylabel('fixed acidity')
ax.set_zlabel('pH')

ax.legend()
plt.show()
```


Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
4.0
3.8
3.6
3.4
3.2
3.0
pH
16
14
12
10
8
6
fixed acidity
9
10
11
12
13
14
alcohol

In [ ]:

In [ ]: