K-means Clustering and Agglomerative Clustering

In [10]:
```python
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
```

In [26]:
```python
dataset = pd.read_csv("Mall_Customers.csv")
dataset[:5]
```

Out[26]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

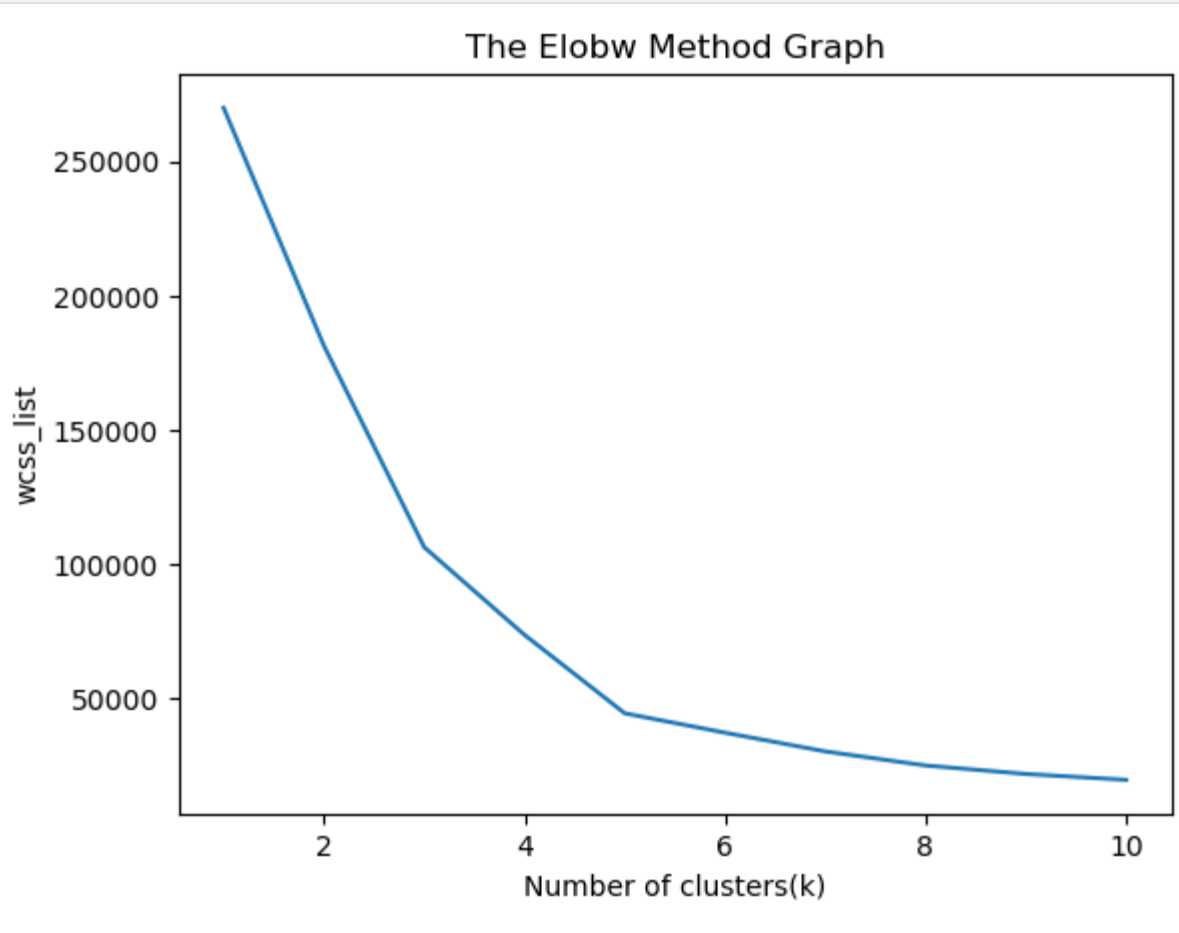
In [13]:
```python
# Extracting independent variables
x = dataset.iloc[:, [3, 4]].values
```

In [25]:
```python
x[:5]
```

Out[25]:
```
array([[15, 39],
       [15, 81],
       [16,  6],
       [16, 77],
       [17, 40]])
```

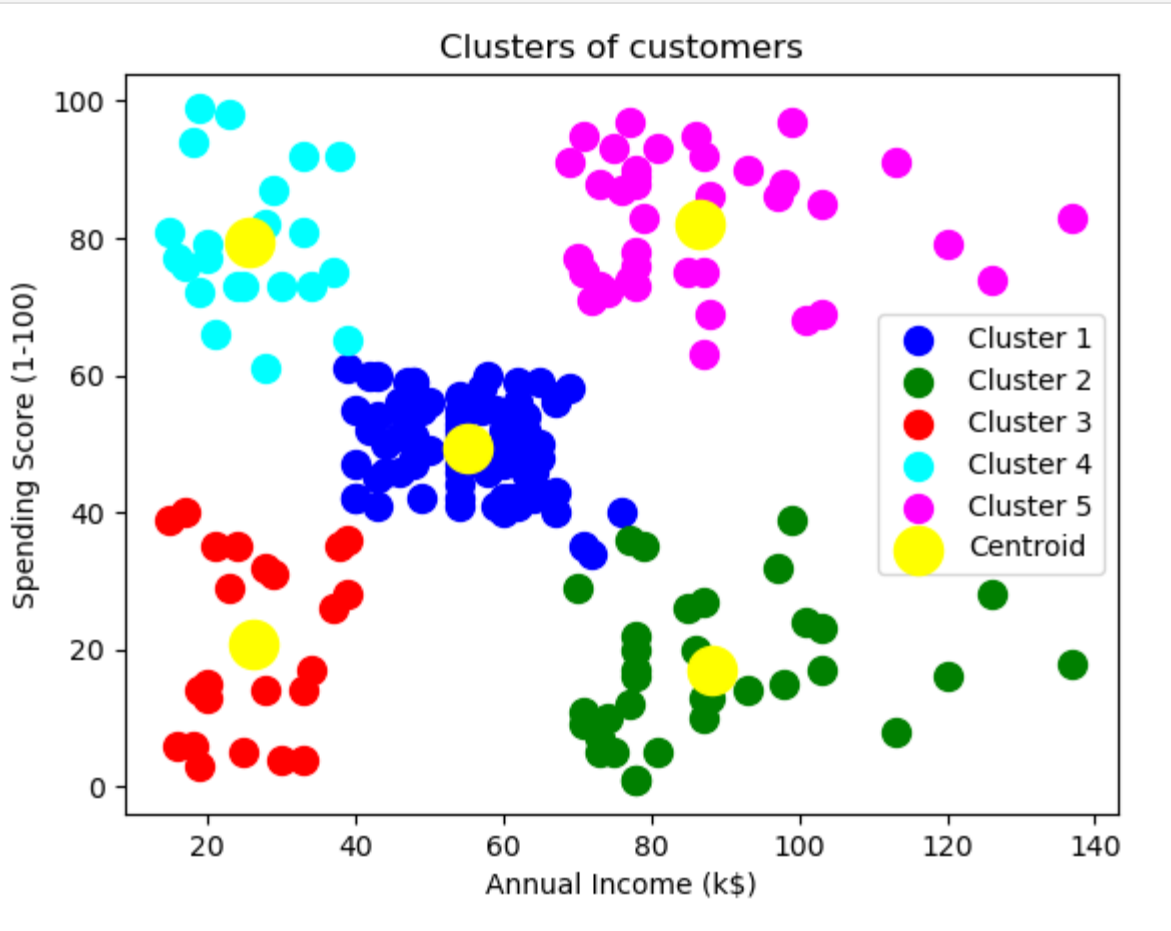
In [14]:
```python
#finding optimal number of clusters using the elbow method
from sklearn.cluster import KMeans
wcss_list= []  #Initializing the list for the values of WCSS

#Using for loop for iterations from 1 to 10.
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42)
    kmeans.fit(x)
    wcss_list.append(kmeans.inertia_)
mtp.plot(range(1, 11), wcss_list)
mtp.title('The Elobw Method Graph')
mtp.xlabel('Number of clusters(k)')
mtp.ylabel('wcss_list')
mtp.show()
```



In [15]:
```python
#training the K-means model on a dataset
kmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)
y_predict= kmeans.fit_predict(x)
```

In [16]:
```python
#visulaizing the clusters
mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label = 'Cluster 1') #for first cl
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Cluster 2') #for second
mtp.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label = 'Cluster 3') #for third clus
mtp.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4') #for fourth c
mtp.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5') #for fifth
mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centr
mtp.title('Clusters of customers')
mtp.xlabel('Annual Income (k$)')
mtp.ylabel('Spending Score (1-100)')
mtp.legend()
mtp.show()
```

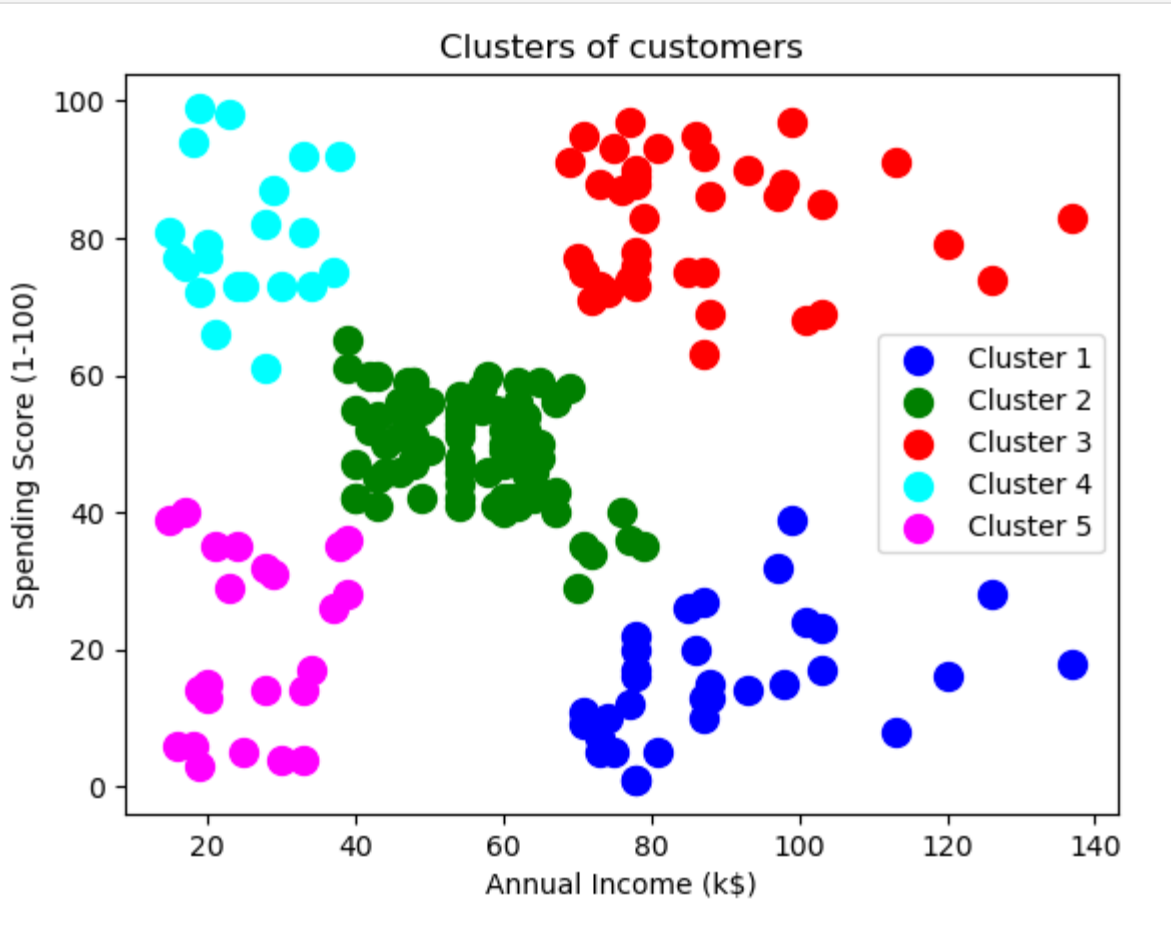


In [17]:
```python
# import
from sklearn.cluster import AgglomerativeClustering
```

In [18]:
```python
aggloclus = AgglomerativeClustering(n_clusters = 5)
```

In [20]:
```python
y_predict = aggloclus.fit_predict(x)
```

In [23]:
```python
mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label = 'Cluster 1') #for first cl
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Cluster 2') #for second
mtp.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label = 'Cluster 3') #for third clus
mtp.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4') #for fourth c
mtp.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5') #for fifth
mtp.title('Clusters of customers')
mtp.xlabel('Annual Income (k$)')
mtp.ylabel('Spending Score (1-100)')
mtp.legend()
mtp.show()
```



linkage measure

In [55]:
```python
import matplotlib.pyplot as plt
```

In [31]:
```python
dataset = pd.read_csv("new_mall_customers.csv")
```

In [32]:
```python
dataset[:5]
```

Out[32]:

	Annual Income(k$)	Spending Score (1-100)
0	75	9
1	50	17
2	74	73
3	40	63
4	66	59

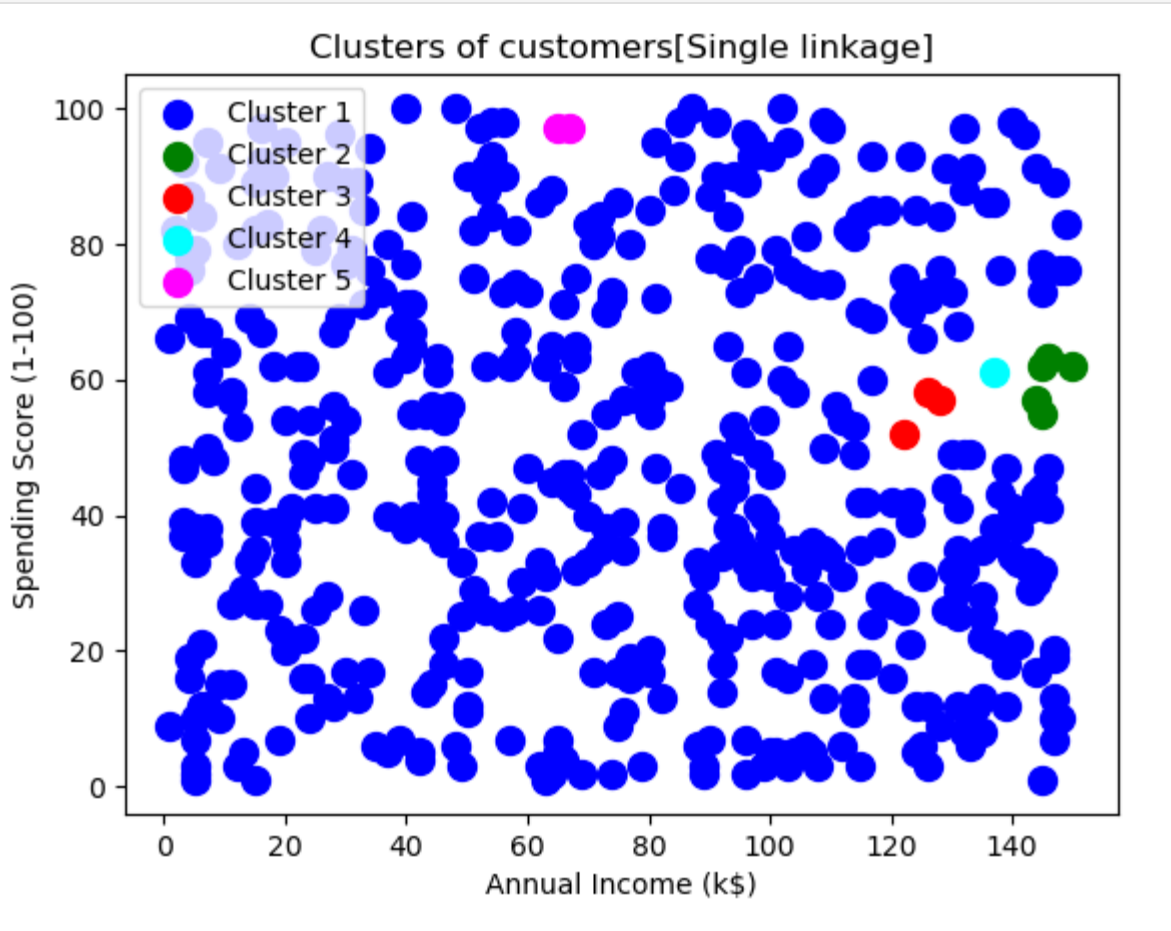
In [33]:
```python
x = dataset.iloc[:, :].values
x[:5]
```

Out[33]:
```
array([[75,  9],
       [50, 17],
       [74, 73],
       [40, 63],
       [66, 59]])
```

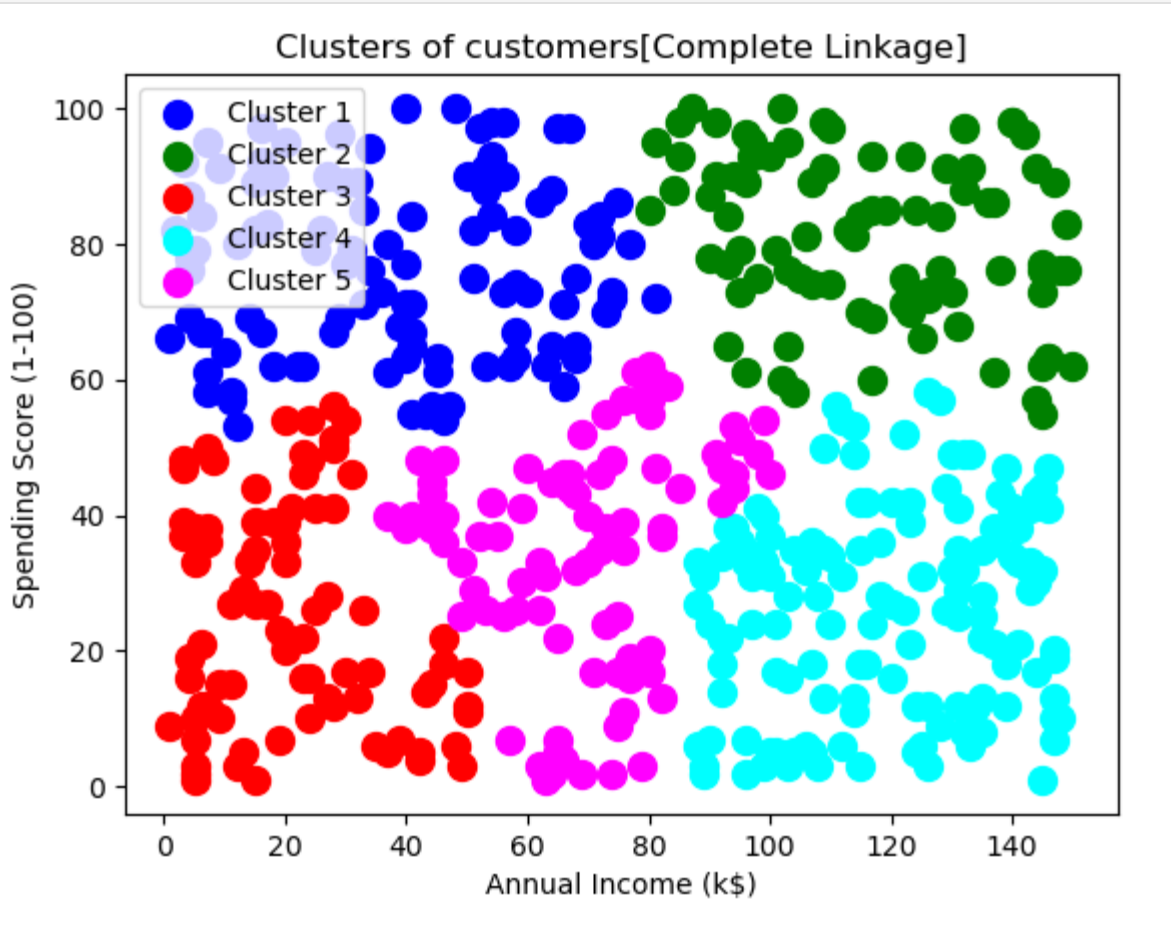
In [50]:
```python
model_single = AgglomerativeClustering(n_clusters = 5, linkage="single")
model_complete = AgglomerativeClustering(n_clusters = 5, linkage="complete")
model_average = AgglomerativeClustering(n_clusters = 5, linkage="average")
```

In [51]:
```python
single_predict = model_single.fit_predict(x)
complete_predict = model_complete.fit_predict(x)
average_predict = model_average.fit_predict(x)
```

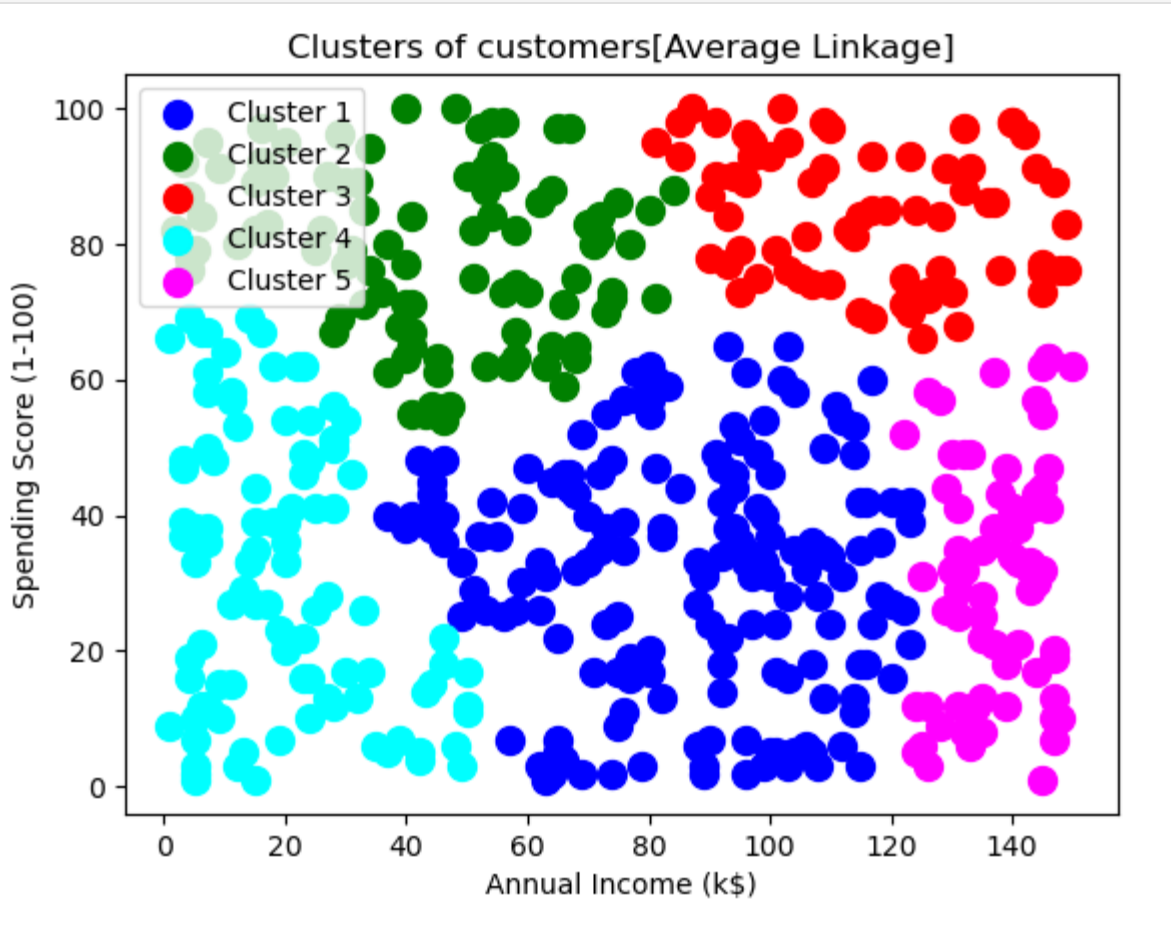
In [52]:
```python
plt.scatter(x[single_predict == 0, 0], x[single_predict == 0, 1], s = 100, c = 'blue', label = 'Cluster 1') #fo
plt.scatter(x[single_predict == 1, 0], x[single_predict == 1, 1], s = 100, c = 'green', label = 'Cluster 2') #f
plt.scatter(x[single_predict== 2, 0], x[single_predict == 2, 1], s = 100, c = 'red', label = 'Cluster 3') #for
plt.scatter(x[single_predict == 3, 0], x[single_predict == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4') #fo
plt.scatter(x[single_predict == 4, 0], x[single_predict == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
plt.title('Clusters of customers[Single linkage]')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```



In [53]:
```python
plt.scatter(x[complete_predict == 0, 0], x[complete_predict == 0, 1], s = 100, c = 'blue', label = 'Cluster 1')
plt.scatter(x[complete_predict == 1, 0], x[complete_predict == 1, 1], s = 100, c = 'green', label = 'Cluster 2'
plt.scatter(x[complete_predict== 2, 0], x[complete_predict == 2, 1], s = 100, c = 'red', label = 'Cluster 3') #
plt.scatter(x[complete_predict == 3, 0], x[complete_predict == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
plt.scatter(x[complete_predict == 4, 0], x[complete_predict == 4, 1], s = 100, c = 'magenta', label = 'Cluster
plt.title('Clusters of customers[Complete Linkage]')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```



In [54]:
```python
plt.scatter(x[average_predict == 0, 0], x[average_predict == 0, 1], s = 100, c = 'blue', label = 'Cluster 1') #
plt.scatter(x[average_predict == 1, 0], x[average_predict == 1, 1], s = 100, c = 'green', label = 'Cluster 2')
plt.scatter(x[average_predict== 2, 0], x[average_predict == 2, 1], s = 100, c = 'red', label = 'Cluster 3') #fo
plt.scatter(x[average_predict == 3, 0], x[average_predict == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4') #
plt.scatter(x[average_predict == 4, 0], x[average_predict == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5'
plt.title('Clusters of customers[Average Linkage]')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```



Birch Clustering

In [3]:
```python
# Import
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import Birch
```

In [4]:
```python
# Generating 600 samples using make_blobs
dataset, clusters = make_blobs(n_samples = 600, centers = 8, cluster_std = 0.75, random_state = 0)
```

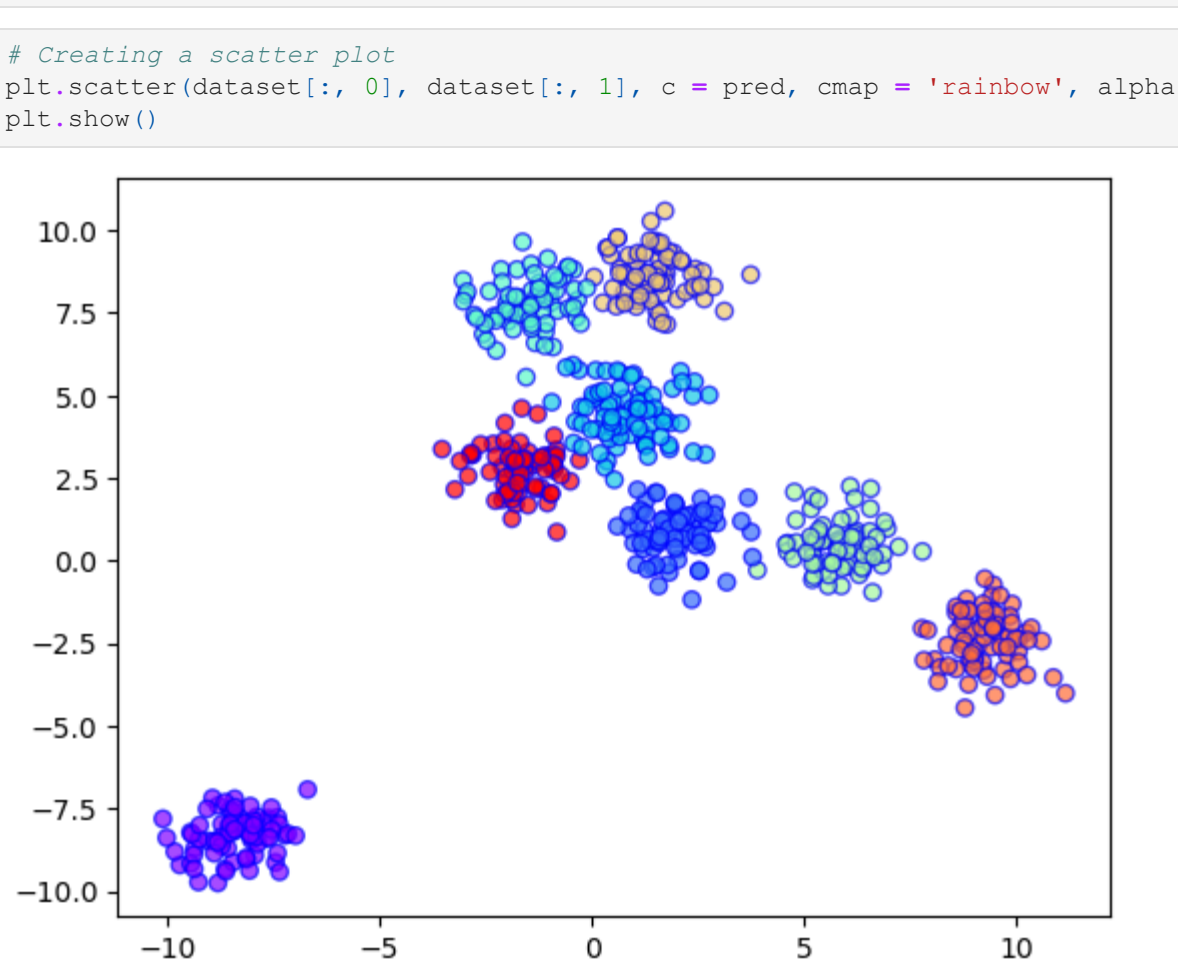
In [5]:
```python
# Creating the BIRCH clustering model
model = Birch(branching_factor = 50, n_clusters = None, threshold = 1.5)
```

In [6]:
```python
# Fit the data (Training)
model.fit(dataset)
```

Out[6]:
```
Birch
Birch(n_clusters=None, threshold=1.5)
```

In [7]:
```python
# Predict the same data
pred = model.predict(dataset)
```

In [8]:
```python
# Creating a scatter plot
plt.scatter(dataset[:, 0], dataset[:, 1], c = pred, cmap = 'rainbow', alpha = 0.7, edgecolors = 'b')
plt.show()
```



Bagging and Boosting

In [57]:
```python
import pandas as pd
import numpy as np
from numpy import loadtxt
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings("ignore")
```

In [60]:
```python
pima = pd.read_csv("diabetes.csv", header=0)
pima.head()
```

Out[60]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [61]:
```python
# split data into X and y
X = pima.iloc[:,0:8]
Y = pima.iloc[:,8]
```

In [62]:
```python
X.head()
```

Out[62]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

In [63]:
```python
# split data into train and test sets
seed = 42
test_size = 0.30
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=test_size, random_state=seed)
```

In [64]:
```python
model = XGBClassifier()
model.fit(X_train, y_train)
```

Out[64]:
```
XGBClassifier
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, feature_types=None, gamma=0, gpu_id=-1,
              grow_policy='depthwise', importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_bin=256, max_cat_threshold=64, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0, ...)
```

In [65]:
```python
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]
```

In [66]:
```python
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Accuracy: 73.59%
```

In [67]:
```python
from sklearn.ensemble import AdaBoostClassifier
```

In [76]:
```python
model2 = AdaBoostClassifier()
model2.fit(X_train, y_train)
```

Out[76]:
```
AdaBoostClassifier
AdaBoostClassifier()
```

In [77]:
```python
y_pred2 = model2.predict(X_test)
predictions2 = [round(value) for value in y_pred2]
```

In [78]:
```python
accuracy2 = accuracy_score(y_test, predictions2)
print("Accuracy: %.2f%%" % (accuracy2 * 100.0))
```

```
Accuracy: 74.46%
```

In [79]:
```python
from sklearn.ensemble import GradientBoostingClassifier
```

In [80]:
```python
model3 = GradientBoostingClassifier()
model3.fit(X_train, y_train)
```

Out[80]:
```
GradientBoostingClassifier
GradientBoostingClassifier()
```

In [81]:
```python
y_pred3 = model3.predict(X_test)
predictions3 = [round(value) for value in y_pred3]
```

In [82]:
```python
accuracy3 = accuracy_score(y_test, predictions3)
print("Accuracy: %.2f%%" % (accuracy3 * 100.0))
```

```
Accuracy: 74.89%
```