In [ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

url='https://archive.ics.uci.edu/ml/machine-learning-databases/00267/data_banknote_
names = ['variance', 'skewness', 'curtosis', 'entropy', 'class']
dataset = pd.read_csv(url, names=names)

x = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

#Splitting dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test=train_test_split(x, y, test_size=0.25, random_stat
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc=  StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
#Training the logistic regression model the Training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)
y_pred=classifier.predict(X_test)

from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:",accuracy)
```

```
Accuracy: 0.9795918367346939
```

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas  as pd
```

In [ ]:

```
dataset = pd.read_csv('Mall_Customers.csv')
X = dataset.iloc[:, [3,4]].values
```

In [ ]:

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

silhouette_scores = []

for i in range(2, 11):
   kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
   kmeans.fit(X)
   silhouette_scores.append(silhouette_score(X, kmeans.labels_))

optimal_num_clusters = np.argmax(silhouette_scores) + 2
print("Optimal number of clusters: ", optimal_num_clusters)
```

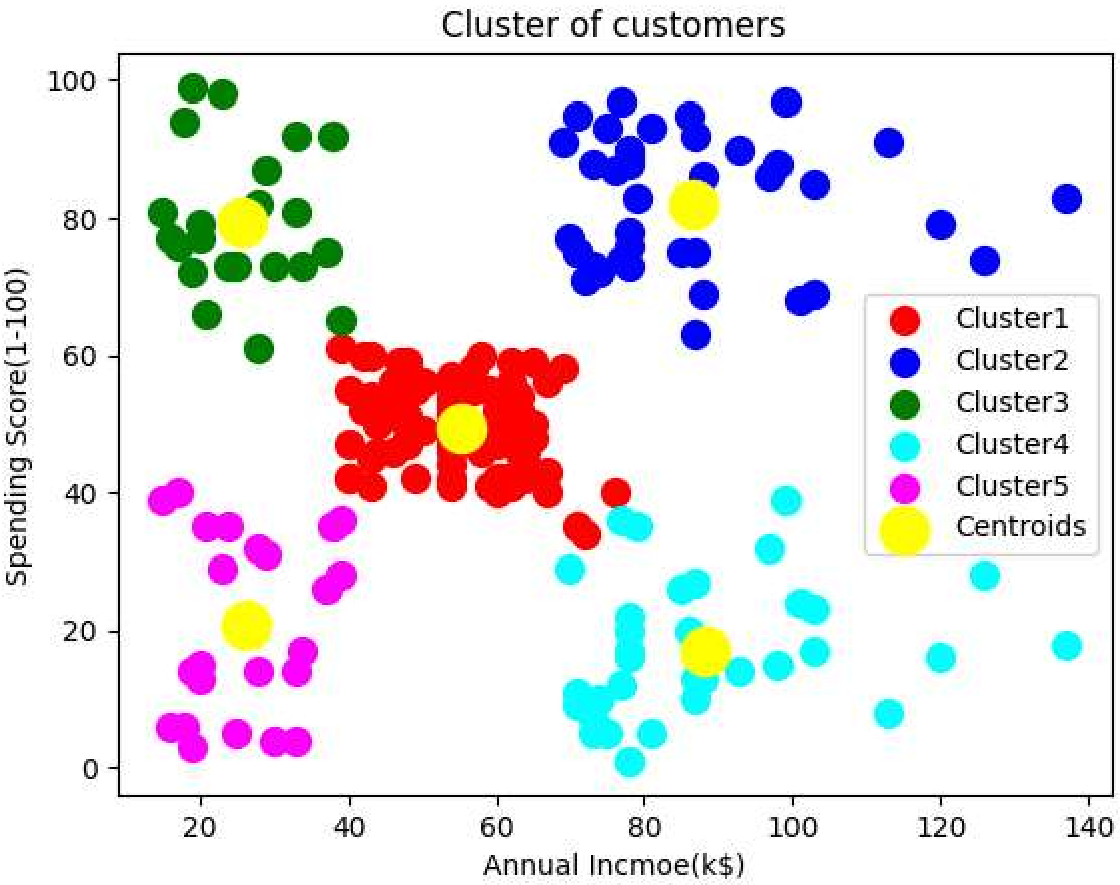
```
Optimal number of clusters:  5
```

In [ ]:

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

In [ ]:

```
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0,1], s=100, c='red', label ='Cluste
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1,1], s=100, c='blue', label ='Clust
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2,1], s=100, c='green', label ='Clus
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3,1], s=100, c='cyan', label ='Clust
plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4,1], s=100, c='magenta', label ='Cl
plt.scatter(kmeans.cluster_centers_[:, 0],kmeans.cluster_centers_[:,1],s=300, c='ye
plt.title('Cluster of customers')
plt.xlabel('Annual Incmoe(k$)')
plt.ylabel('Spending Score(1-100)')
plt.legend()
plt.show()
```


Cluster of customers
Spending Score(1-100)
Annual Incmoe(k$)
0
20
40
60
80
100
20
40
60
80
100
120
140
Cluster1
Cluster2
Cluster3
Cluster4
Cluster5
Centroids

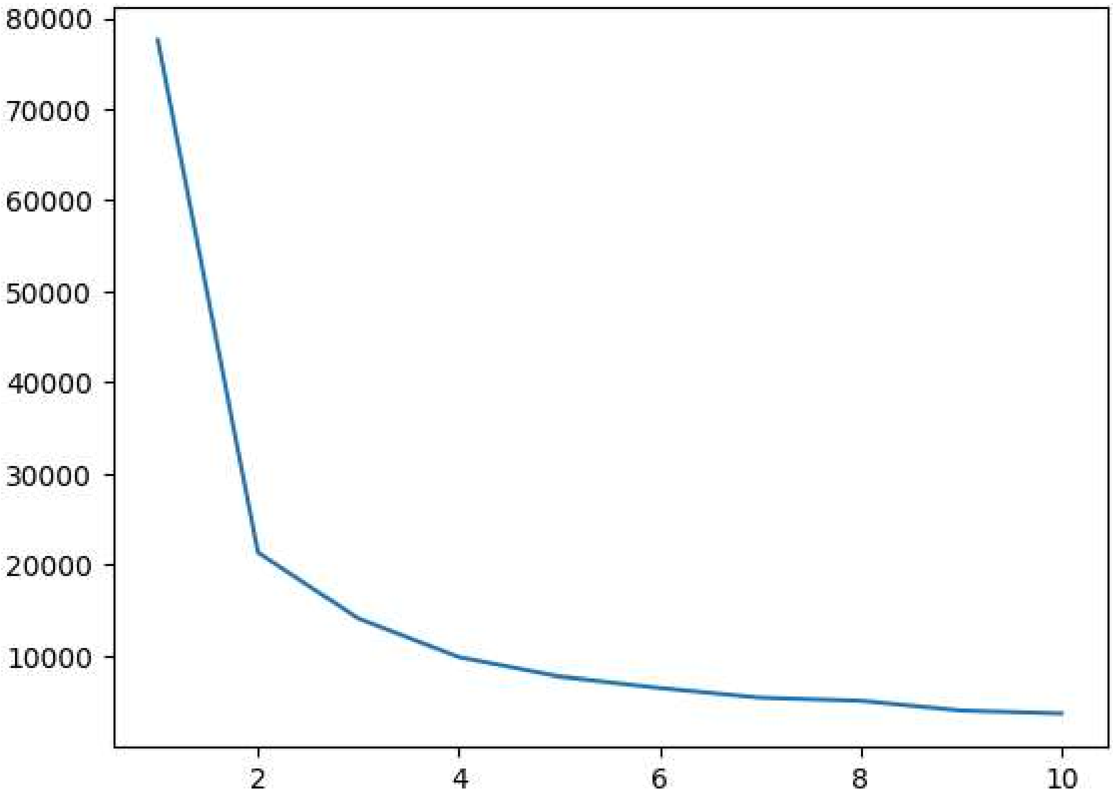
In [ ]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

df = pd.read_csv('penguins.csv')
print(df.shape) # (344, 9)
df = df[['bill_length_mm', 'flipper_length_mm']]
df = df.dropna(axis=0)
```

```
(344, 9)
```

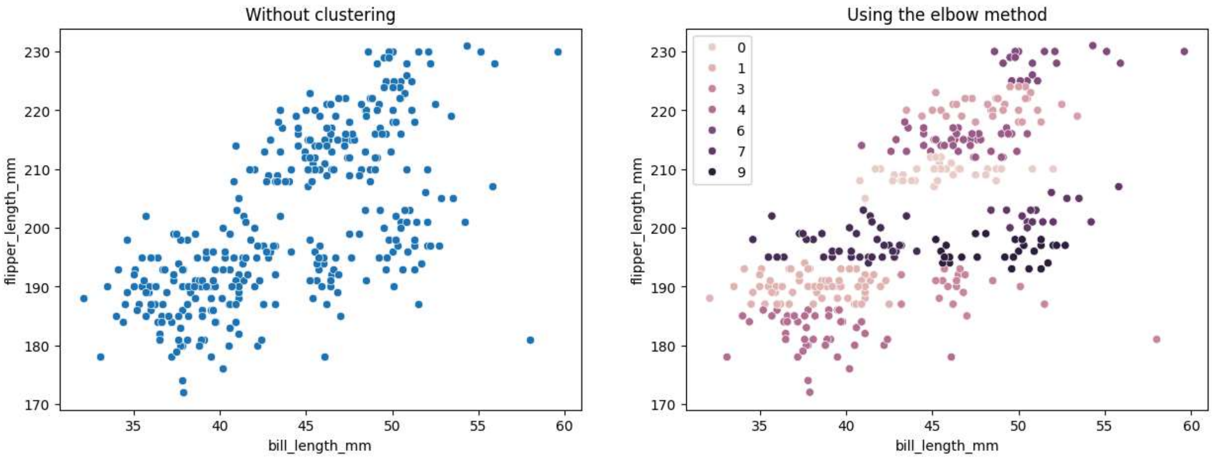
In [ ]:

```
wcss = []

for i in range(1, 11):
    clustering = KMeans(n_clusters=i, init='k-means++', random_state=42)
    clustering.fit(df)
    wcss.append(clustering.inertia_)

ks = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
sns.lineplot(x = ks, y = wcss);
```



In [ ]:

```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(15,5))
sns.scatterplot(ax=axes[0], data=df, x='bill_length_mm', y='flipper_length_mm').set
sns.scatterplot(ax=axes[1], data=df, x='bill_length_mm', y='flipper_length_mm', hue
```



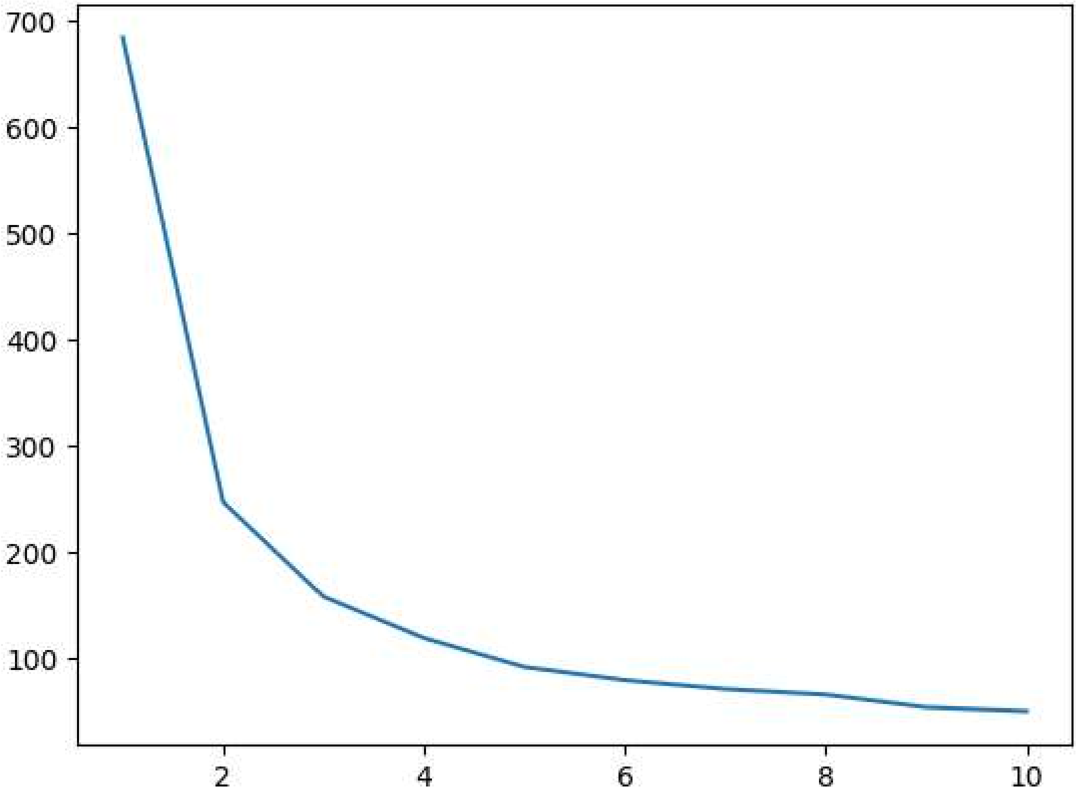
In [ ]:
```
df.describe().T
```

Out[ ]:

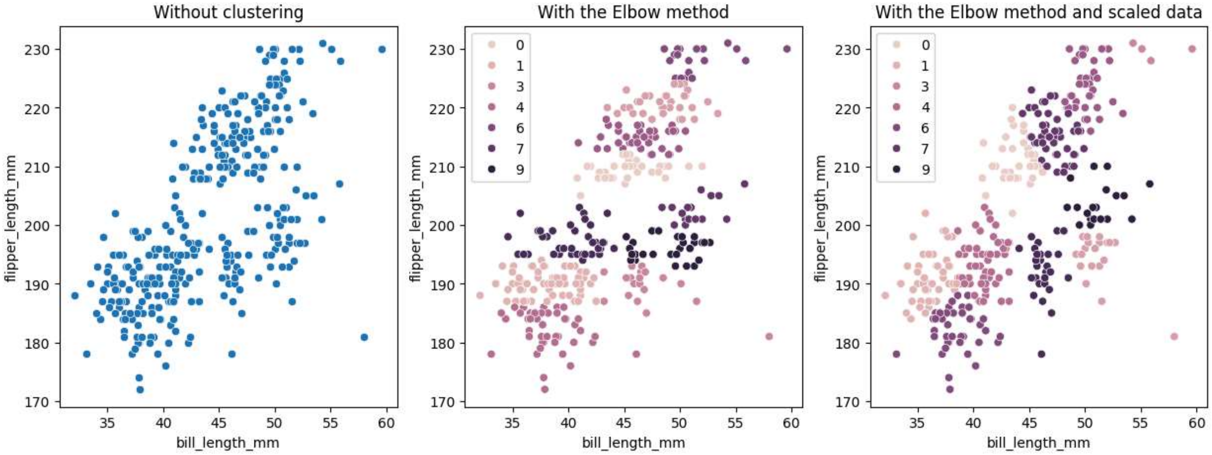
	count	mean	std	min	25%	50%	75%	max
bill_length_mm	342.0	43.921930	5.459584	32.1	39.225	44.45	48.5	59.6
flipper_length_mm	342.0	200.915205	14.061714	172.0	190.000	197.00	213.0	231.0

In [ ]:
```
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()
scaled = ss.fit_transform(df)
```

In [ ]:
```
wcss_sc = []

for i in range(1, 11):
    clustering_sc = KMeans(n_clusters=i, init='k-means++', random_state=42)
    clustering_sc.fit(scaled)
    wcss_sc.append(clustering_sc.inertia_)

ks = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
sns.lineplot(x = ks, y = wcss_sc);
```



In [ ]:
```
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15,5))
sns.scatterplot(ax=axes[0], data=df, x='bill_length_mm', y='flipper_length_mm').set
sns.scatterplot(ax=axes[1], data=df, x='bill_length_mm', y='flipper_length_mm', hue
sns.scatterplot(ax=axes[2], data=df, x='bill_length_mm', y='flipper_length_mm', hue
```


Without clustering
With the Elbow method
With the Elbow method and scaled data
flipper_length_mm
bill_length_mm