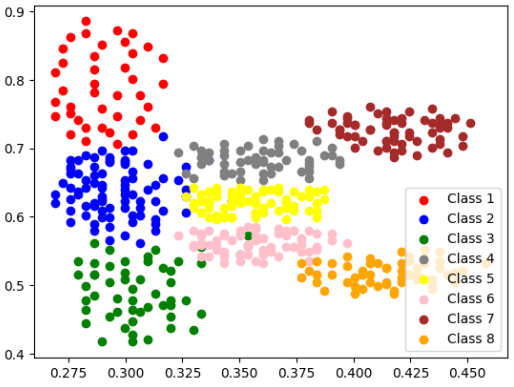
به نام خدا

مهدی فقهی

۴۰۱۷۲۲۱۳۶

Problem 6 :

برای حل سوال ششم ابتدا دیتاست مربوطه را load می‌کنیم که شکل زیر است :

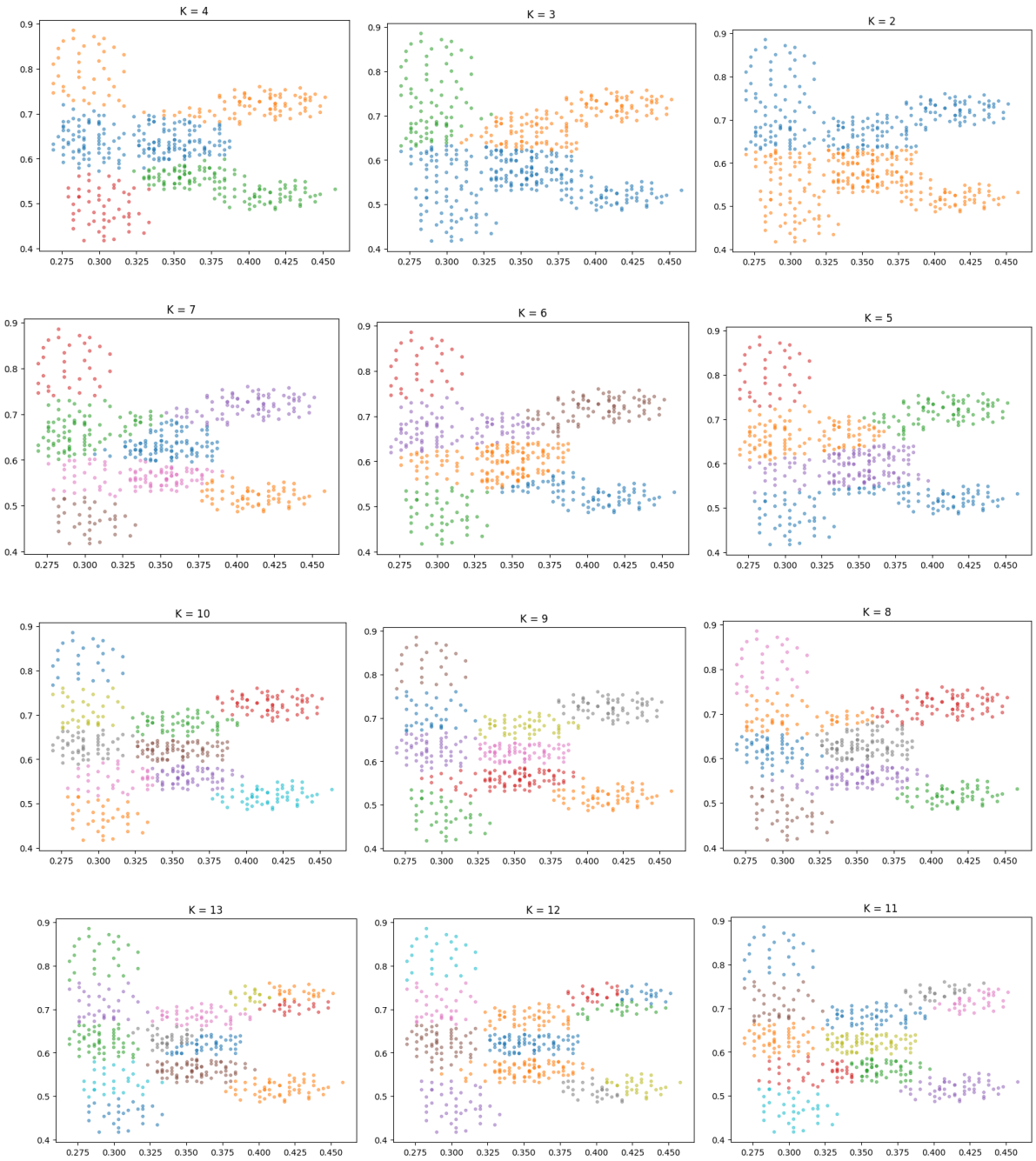


سپس الگوریتم K Means را پیاده سازی می‌کنیم .

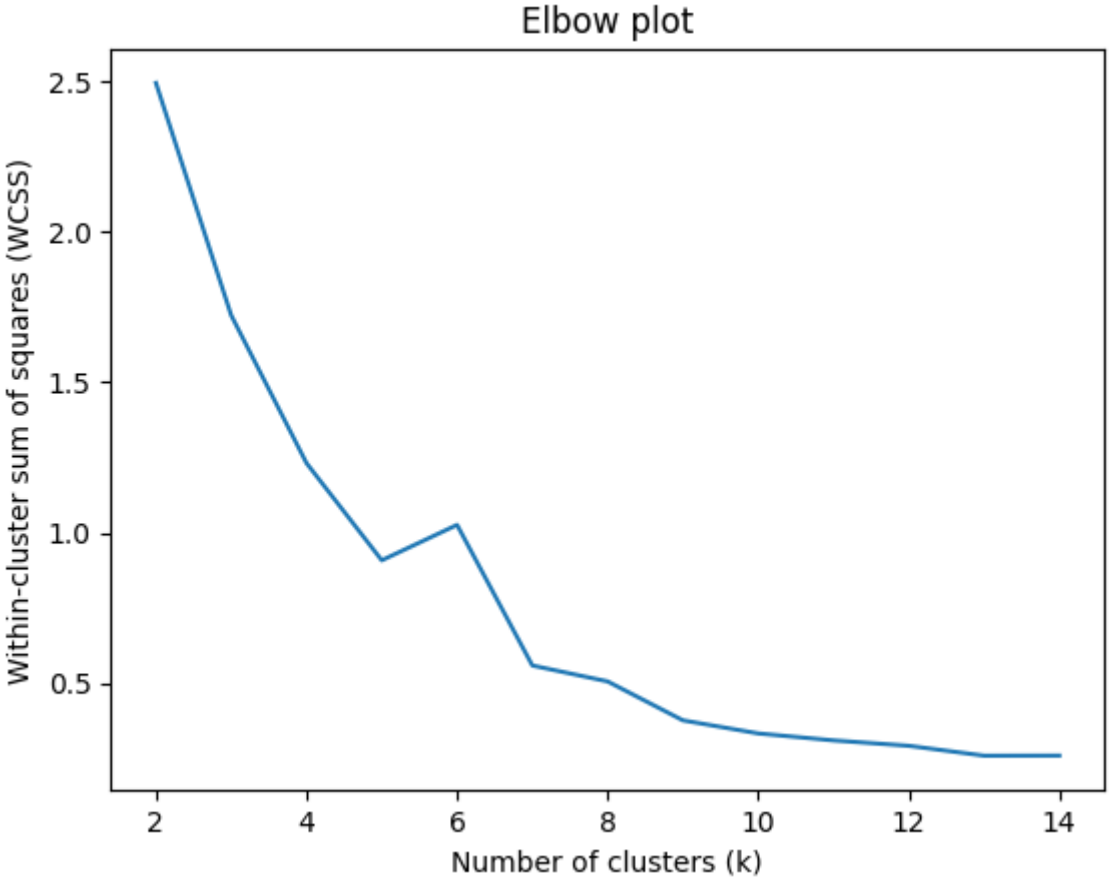
```
def k_means(data, k):
    # Initialize centroids randomly
    centroids = data[np.random.choice(len(data), size=k, replace=False)]

    # Iterate until convergence
    while True:
        # Assign each point to its closest centroid
        labels = np.argmin(np.sqrt(np.sum((data - centroids[:, np.newaxis])**2, axis=2)), axis=0)

        # Update centroid positions
        new_centroids = np.array([data[labels==i].mean(axis=0) for i in range(k)])

        # Check for convergence
        if np.allclose(new_centroids, centroids):
            break

        centroids = new_centroids

    return labels
```

ابتدا به صورت رندوم یکسری مرکز دسته برحسب تعداد دسته‌ای که در نظر گرفته‌ای نمونه را به آن اندازه دسته بندی کنیم، می‌سازیم سپس فاصله هر نقطه را تا آن مراکز بدست می‌آوریم و آنان را به کلاس‌ای که به آن تعلق دارند منتسب می‌کنیم و در نهایت مرکز دسته‌را براساس داده‌های آن خوشه دوباره بدست می‌آوریم و اینکار را اینقدر ادامه می‌دهیم که مرکز دسته‌های ما دیگر تغییر نکنند و در نهایت خوشه انتسابی به هر داده را برمی‌گردانیم .

این کار را به ازای K مختلف از ۲ تا ۱۴ انجام می‌دهیم و شکل‌ها را بررسی می‌کنیم.


K = 4
K = 3
K = 2
K = 7
K = 6
K = 5
K = 10
K = 9
K = 8
K = 13
K = 12
K = 11

سپس از قاعده Elbow برای پیدا کردن بهترین تعداد cluster استفاده می‌کنیم :



همانطور که مشاهده می‌کنیم نمودار ما در نقطه هشت به بعد دچار شکستگی می‌شود و سپس بعد از آن به شکل هموارتری به سمت صفر حرکت می‌کند پس می‌توانیم حدس بزنیم که به احتمال خوبی هشت، نه یا ۱۰ تا cluster داریم.

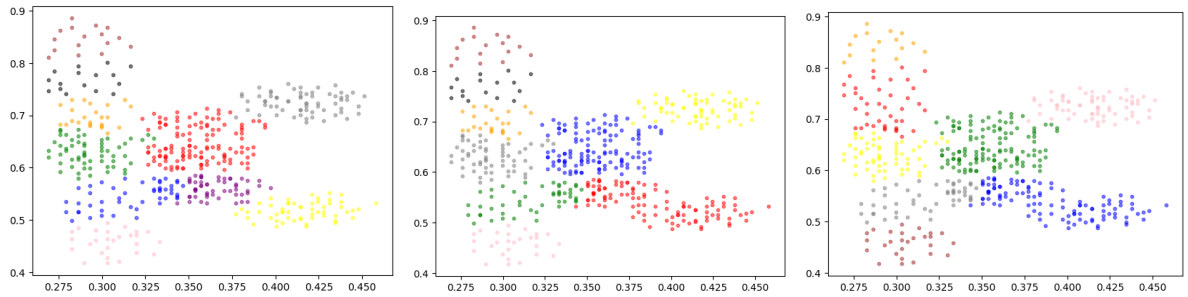
برای خوشه‌بندی Agglomerative Hierarchical Clustering می‌توانیم با در نظر گرفتن هر نقطه به عنوان یک خوشه جداگانه شروع کنیم و به طور مکرر نزدیک‌ترین خوشه‌ها را تا زمانی که یک شرط توقف برآورده شود، ادغام کنیم. چندین معیار پیوندی برای خوشه بندی تجمعی وجود دارد، مانند پیوند منفرد، پیوند کامل و پیوند متوسط. در اینجا چند کد نمونه برای انجام خوشه بندی بر اساس سیاست single link, complete link, average link آورده شده است پس از بررسی هر سه شیوه و نگاه کردن به حاصل‌های بدست آمده، متوجه شدم برای حل این مسئله خاص complete link بهتر کار می‌کند .

```python
def complete_link(ci, cj):
    return np.max([distance(vi, vj) for vi in ci for vj in cj])


def average_link(ci, cj):
    distances = [distance(vi, vj) for vi in ci for vj in cj]
    return np.sum(distances) / len(distances)


def get_distance_measure(M):
    if M == 0:
        return single_link
    elif M == 1:
        return complete_link
    else:
        return average_link
```

پیاده‌سازی این نوع از دسته‌بندی :

```python
def init_clusters(self):
    return {data_id: np.expand_dims(data_point, axis=0) for data_id, data_point in enumerate(self.data)}

def find_closest_clusters(self):
    min_dist = np.inf
    closest_clusters = None

    clusters_ids = list(self.clusters.keys())
    cluster_sizes = np.array([self.clusters[i].shape[0] for i in clusters_ids])

    for i, cluster_i in enumerate(clusters_ids[:-1]):
        for j, cluster_j in enumerate(clusters_ids[i+1:]):
            dist = self.measure(self.clusters[cluster_i], self.clusters[cluster_j])
            if dist < min_dist:
                min_dist, closest_clusters = dist, (cluster_i, cluster_j)
    return closest_clusters

def merge_and_form_new_clusters(self, ci_id, cj_id):
    new_clusters = {0: np.concatenate((self.clusters[ci_id], self.clusters[cj_id]), axis=0)}

    for cluster_id in self.clusters.keys():
        if (cluster_id == ci_id) or (cluster_id == cj_id):
            continue
        new_clusters[len(new_clusters.keys())] = self.clusters[cluster_id]
    return new_clusters

def run_algorithm(self):
    while len(self.clusters.keys()) > self.K:
        closest_clusters = self.find_closest_clusters()
        self.clusters = self.merge_and_form_new_clusters(*closest_clusters)

def print(self):
    for id, points in self.clusters.items():
        print("Cluster: {}".format(id))
        for point in points:
            print("    {}".format(point))
```

سپس به کمک این الگوریتم با فرض سه تا دسته شروع کردیم به دسته‌بندی و حاصل اینگونه شد برای دسته‌بندی هشت، نه و ده :



برای تعیین اینکه کدام الگوریتم خوشه‌بندی بهتر عمل کرده است، می‌توانیم شاخص رند تعدیل‌شده (ARI) را محاسبه کنیم، که معیاری از شباهت خوشه‌ای است که توافق شانس را در نظر می‌گیرد. ARI از 1- تا 1 است که مقدار 1 نشان دهنده تطابق کامل بین خوشه های واقعی و پیش بینی شده است و مقدار 0 نشان دهنده خوشه بندی تصادفی است. در اینجا چند کد نمونه برای محاسبه ARI آورده شده است:

```
from sklearn.metrics import adjusted_rand_score

# Load true labels from file
true_labels = np.loadtxt('artificial_data.txt', delimiter=',', usecols=2)

# Calculate ARI for k-means clustering]
k_labels = k_means(data[:, :2], k=8)
k_ari = adjusted_rand_score(true_labels, k_labels)
# Calculate ARI for agglomerative clustering
a_labels = preprocess_agglomerative_res(data[:, :2],clusters_8)
a_ari = adjusted_rand_score(true_labels, a_labels)

print('K-means ARI 8: {:.3f}'.format(k_ari))
print('Agglomerative ARI 8: {:.3f}'.format(a_ari))
```

```
K-means ARI 8: 0.809
Agglomerative ARI 8: 0.190
```

```
from sklearn.metrics import adjusted_rand_score

# Load true labels from file
true_labels = np.loadtxt('artificial_data.txt', delimiter=',', usecols=2)

# Calculate ARI for k-means clustering]
k_labels = k_means(data[:, :2], k=9)
k_ari = adjusted_rand_score(true_labels, k_labels)
# Calculate ARI for agglomerative clustering
a_labels = preprocess_agglomerative_res(data[:, :2],clusters_9)
a_ari = adjusted_rand_score(true_labels, a_labels)

print('K-means ARI 9: {:.3f}'.format(k_ari))
print('Agglomerative ARI 9: {:.3f}'.format(a_ari))
```

```
K-means ARI 9: 0.675
Agglomerative ARI 9: 0.144
```

```
[ ]  from sklearn.metrics import adjusted_rand_score

     # Load true labels from file
     true_labels = np.loadtxt('artificial_data.txt', delimiter=',', usecols=2)

     # Calculate ARI for k-means clustering]
     k_labels = k_means(data[:, :2], k=10)
     k_ari = adjusted_rand_score(true_labels, k_labels)
     # Calculate ARI for agglomerative clustering
     a_labels = preprocess_agglomerative_res(data[:, :2],clusters_10)
     a_ari = adjusted_rand_score(true_labels, a_labels)

     print('K-means ARI 10: {:.3f}'.format(k_ari))
     print('Agglomerative ARI 10: {:.3f}'.format(a_ari))


     K-means ARI 10: 0.618
     Agglomerative ARI 10: 0.164
```

همانطور که مشاهده می‌کنید بهترین جواب هر دو روش بر روی تعداد cluster هشت که تعداد cluster واقعی ما بوده‌اند حادث شده‌اند هر چند برای این نوع داده روش K Means به مراتب بهتر از روش agglomerative عمل کرده است .

Problem 7 :

به کمک این لینک ابتدا داده‌ها شروع به بررسی کردن کردم :

بعد از لود کردن دیتا به این شکل متوجه شدم که هیچ داده null نداریم و تایپ هر کدام از ستون‌ها چیست :

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

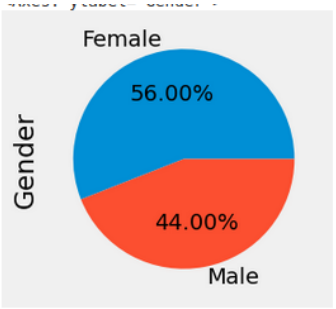
سپس با کوئری زیر متوسط درآمد زن و مردان مشتری را محاسبه کردم :

```
print("Mean of Annual Income (k$) of Female:",dataset['Annual Income (k$)'].loc[dataset['Gender'] == 'Female'].mean())
print("Mean of Annual Income (k$) of Male:",dataset['Annual Income (k$)'].loc[dataset['Gender'] == 'Male'].mean())
```

```
Mean of Annual Income (k$) of Female: 59.25
Mean of Annual Income (k$) of Male: 62.22727272727273
```

همانطور که می‌بینیم مردان به صورت میانگین درآمد بیشتری دارند .

با اینکه درصد فراوانی زنان در این دیتاست بیشتر از مردان است (زنان درصد بیشتری از مشتریان ما هستند)

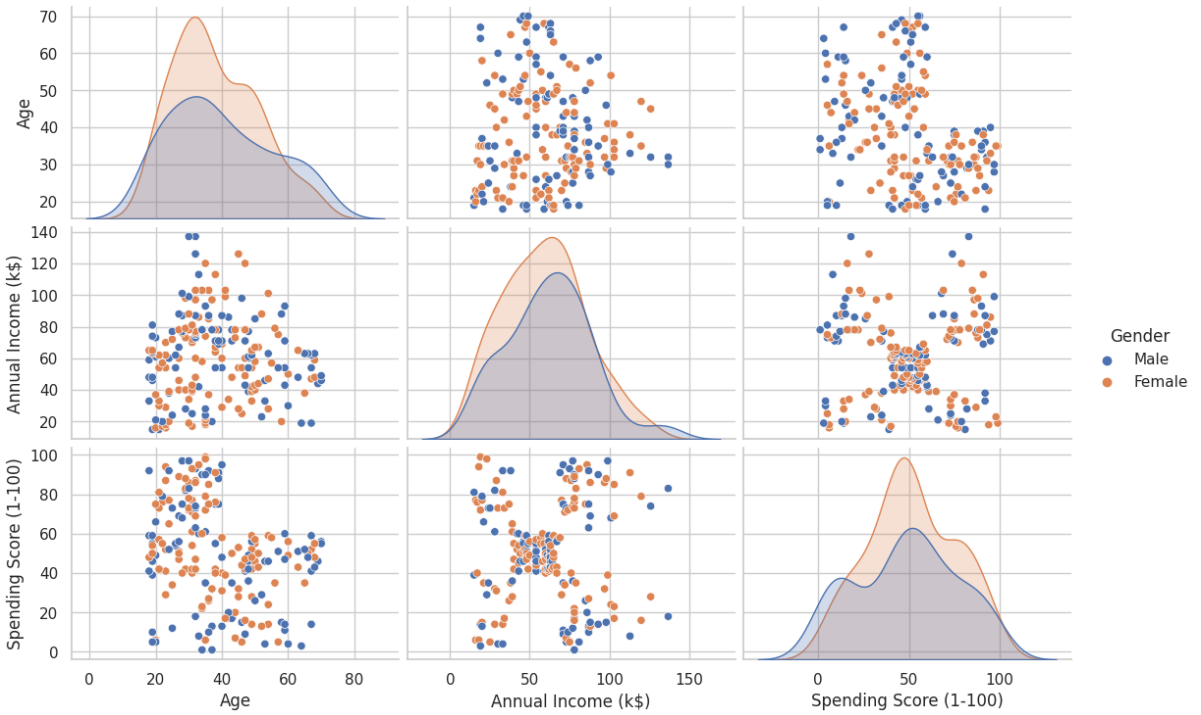


همین کار را برای بقیه ستون‌ها نیز انجام دادیم تا متوجه بشویم که زنان و مردان وضعیت میانگین‌شان در بقیه ستون‌های ما چگونه است :

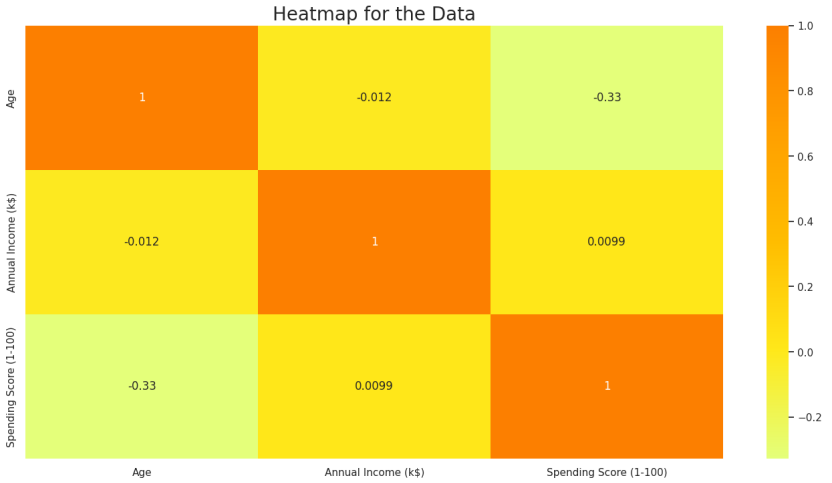
```
dataset.groupby('Gender').mean()
```

Gender	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
Female	97.562500	38.098214	59.250000	51.526786
Male	104.238636	39.806818	62.227273	48.511364

همانطور که مشاهده می‌کنید با آنکه متوسط درآمد خانم‌ها از آقایان کمتر است ولی خرج بیشتری در فروشگاه ما می‌کنند .



از نمودار بالا می بینیم که جنسیت هیچ ارتباط مستقیمی با تقسیم بندی مشتریان ندارد. به همین دلیل است که می‌توانیم آن را رها کنیم و با ویژگی‌های دیگر حرکت کنیم، به همین دلیل از این به بعد X را پارامتر می کنیم ، هر دو توزیع مرد و زن یکسان هستند و صرفا در میانگین و واریانس ممکن است تفاوت جزئی داشته باشند .



نمودار بالا برای نشان دادن همبستگی بین ویژگی‌های مختلف مجموعه داده‌های تقسیم‌بندی مشتری بازار، این نقشه گرمایی بیشترین همبستگی‌ها را با رنگ نارنجی و کمترین همبستگی ویژگی‌ها را با رنگ زرد نشان می‌دهد.

در ادامه مدل خوشه‌بندی خود یعنی ***Gaussian Mixture Model*** مدل را شرح می‌دهیم.

```
def gaussian(X, mu, sigma):
    d = X.shape[1]
    det = np.linalg.det(sigma)
    inv = np.linalg.inv(sigma)
    term1 = 1 / (np.sqrt((2 * np.pi)**d * det))
    term2 = np.exp(-0.5 * np.sum((X - mu).dot(inv) * (X - mu), axis=1))
    return term1 * term2
```

```
def gmm(X, num_clusters, max_iterations=100, tolerance=1e-4):
    n, d = X.shape
    # Initialize cluster means and covariance matrices
    means = X[np.random.choice(n, num_clusters, replace=False), :]
    covs = np.tile(np.diag(np.var(X, axis=0)), (num_clusters, 1, 1))
    # Initialize mixture weights
    weights = np.ones(num_clusters) / num_clusters

    # Initialize log-likelihood and iterate until convergence
    prev_log_likelihood = -np.inf
    for i in range(max_iterations):
        # E-Step: Compute the responsibilities for each data point
        p = np.array([weights[k] * gaussian(X, means[k], covs[k, :, :]) for k in range(num_clusters)]).T
        responsibilities = p / np.sum(p, axis=1, keepdims=True)

        # M-Step: Update the model parameters
        Nk = np.sum(responsibilities, axis=0)

        # Update mean
        means = responsibilities.T.dot(X) / Nk[:, np.newaxis]

        # Update covariance matrix
        for k in range(num_clusters):
            X_centered = X - means[k]
            covs[k] = (responsibilities[:, k, np.newaxis] * X_centered).T.dot(X_centered) / Nk[k]

        # Update mixture weights
        weights = Nk / n

        # Check for convergence
        log_likelihood = np.sum(np.log(np.sum(p, axis=1)))
        if log_likelihood - prev_log_likelihood < tolerance:
            break
        prev_log_likelihood = log_likelihood
    # Compute the BIC and AIC values for this model
    k = num_clusters
    n_params = k * (d + d*(d+1)/2 + 1)
    log_likelihood = np.sum(np.log(np.sum(p, axis=1)))
    bic = -2*log_likelihood + n_params*np.log(n)
    aic = -2*log_likelihood + n_params*2
    # Assign each point to its most likely cluster
    labels = np.argmax(responsibilities, axis=1)

    return labels, means, covs, weights, bic, aic
```

تابع gaussian با گرفتن ماتریس داده‌های X، میانگین و ماتریس کوواریانس mu و sigma، چگالی احتمال یک توزیع گاوسی چند متغیره را برای هر سطر از ماتریس X محاسبه می‌کند. به طور خاص، با دادن یک بردار میانگین mu و ماتریس کوواریانس sigma، احتمال هر سطر ماتریس X زیر توزیع گاوسی با میانگین و کوواریانس داده شده را محاسبه می‌کند.

تابع gmm یک پیاده‌سازی از مدل مخلوط گوسی با استفاده از الگوریتم Expectation-Maximization است. این تابع با گرفتن ماتریس داده‌های X، تعداد خوشه‌ها num_clusters برای یادگیری و آرگومان‌های اختیاری برای حداکثر تعداد تکرار و دقت همگرایی اجرا می‌شود. در ابتدا، این تابع میانگین خوشه‌ها و ماتریس‌های کوواریانس را به صورت تصادفی انتخاب می‌کند، و وزن‌های مخلوط را به طور یکنوا در ابتدا انتخاب شده است.

سپس بین دو مرحله E و M الگوریتم Expectation-Maximization تنظیم می‌کند. در مرحله E تعهدات هر نقطه را محاسبه می‌کند که احتمال نقطه تعلق به هر خوشه را با توجه به شرایط فعلی جداسازی می‌کند. در مرحله M پارامترهای مدل (میانگین، ماتریس کوواریانس ، وزن مخلوط) بر اساس تعهدات به‌روزرسانی می‌شوند.

تابع همچنین با محاسبه لگاریتم احتمال چندجمله‌ای در هر مرحله و مقایسه آن با مرحله قبل، بررسی کنونی را اجرای الگوریتم را پایش می‌کند. اگر تغییر لگاریتم احتمال کمتر از بازه مورد نظر باشد، الگوریتم پایان می‌یابد.

در پایان، توابع مدل پیچیدگی معیار اطلاعات(BIC) و معیار اطلاعات (AIC) را برای مدل برگزیده محاسبه می‌کند. اینها معیارهای استاندارد پیچیدگی مدل هستند که به لحاظ همزمان احتمال لگاریتم مدل و تعداد پارامترهای آن را در نظر می‌گیرند.

تابع برچسب‌ها (تعلق به هر خوشه) داده، میانگین‌های خوشه‌ها، ماتریس‌های کوواریانس، وزن‌های مخلوط، و مقادیر BIC و AIC را برمی‌گرداند. پس از این مرحله :

```python
import numpy as np
import matplotlib.pyplot as plt


# Define the range of number of clusters to try
K_range = range(1, 11)

# Initialize arrays to store BIC and AIC values for each model
bic_values = np.zeros(len(K_range))
aic_values = np.zeros(len(K_range))

# Fit a GMM to the data for each number of clusters and compute BIC and AIC values
for i, k in enumerate(K_range):
    labels, means, covs, weights, bic, aic = gmm(X, k)
    bic_values[i] = bic
    aic_values[i] = aic

# Plot BIC and AIC values versus number of clusters
fig, ax = plt.subplots(figsize=(8, 4))
ax.plot(K_range, bic_values, 'bo-', label='BIC')
ax.plot(K_range, aic_values, 'ro-', label='AIC')
ax.set_xlabel('Number of Clusters')
ax.set_ylabel('Model Complexity')
plt.legend()
plt.show()

# Select the number of clusters that minimizes the BIC or AIC
best_k = K_range[np.argmin(bic_values)]
```

از بین تعداد خوشه‌های مختلف تعداد خوشه ای را انتخاب می‌کنیم که دو معیار ذکر شده در بالا را به حداقل برساند .

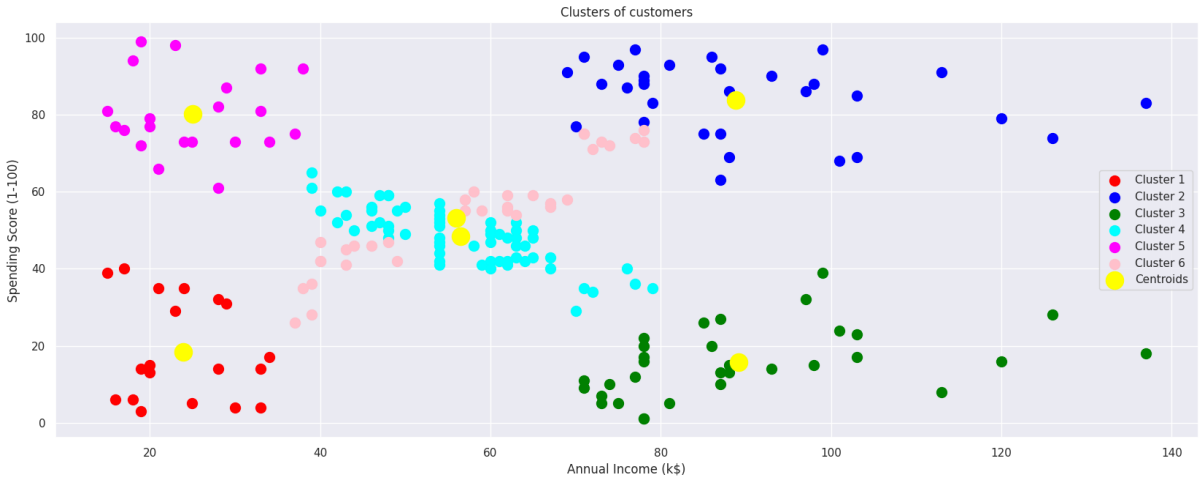
که در اینجا برابر با ۶ است .

حال حالت‌های مختلف clustering داده را انجام می‌دهیم و نتایج را بررسی می‌کنیم .

حالت اول (خوشه‌بندی صرفا با معیار Annual income and Spending Score) :

تعداد خوشه بهینه ۶ عدد

تحلیل بر اساس نمایش خوشه‌ها :



این تجزیه و تحلیل خوشه‌بندی بینش بسیار روشنی در مورد بخش‌های مختلف مشتریان در مرکز خرید به ما می‌دهد.

به وضوح پنج بخش از مشتریان وجود دارد که عبارتند از خوشه 1، خوشه 2، خوشه 3، خوشه 4، خوشه 5، خوشه 6 بر اساس درآمد سالانه و هزینه آنها در فروشگاه (بهترین عوامل / ویژگی ها برای تعیین بخش های یک مشتری در یک سیستم)

خوشه 1 (رنگ قرمز) -> درآمد کم، هزینه کمتر

خوشه 2 (رنگ آبی) -> درآمد بالا، هزینه زیاد [TARGET SET]

خوشه 3 (رنگ سبز) -> درآمد بالا اما هزینه کم [TARGET SET]

خوشه 4 (رنگ فیروزه ای) -> میانگین درآمد و هزینه غیرمستقیم برای کسب درآمد

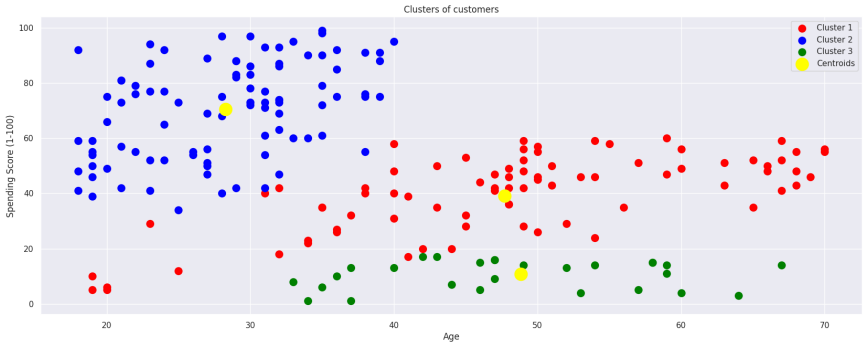
خوشه 5 (رنگ سرخابی) -> درآمد کمتر، هزینه زیاد [TARGET SET]

خوشه 6 (رنگ صورتی) -> میانگین درآمد و هزینه مستقیم به درآمد

ما می‌توانیم Cluster 2 را در همه بخش از سیستم‌های هشدار قرار دهیم که در آن ایمیل می‌تواند به صورت روزانه برای آنها ارسال شود زیرا آنان جز بهترین دسته از مشتریان ما هستند . همچنین می توانیم به خوشه 3 ایمیل بزنیم زیرا ممکن است این دسته از افراد ثروتمند با تبلیغات محصولات ما جذب ما شوند و سرمایه آنها نیز جذب فروشگاه ما شود اما باید مراقب باشیم که فقط در چند دوره به این گروه پیام ارسال کنیم. و اگر دوباره جذب نشدند از صرف هزینه برای آنها خودداری کنیم.

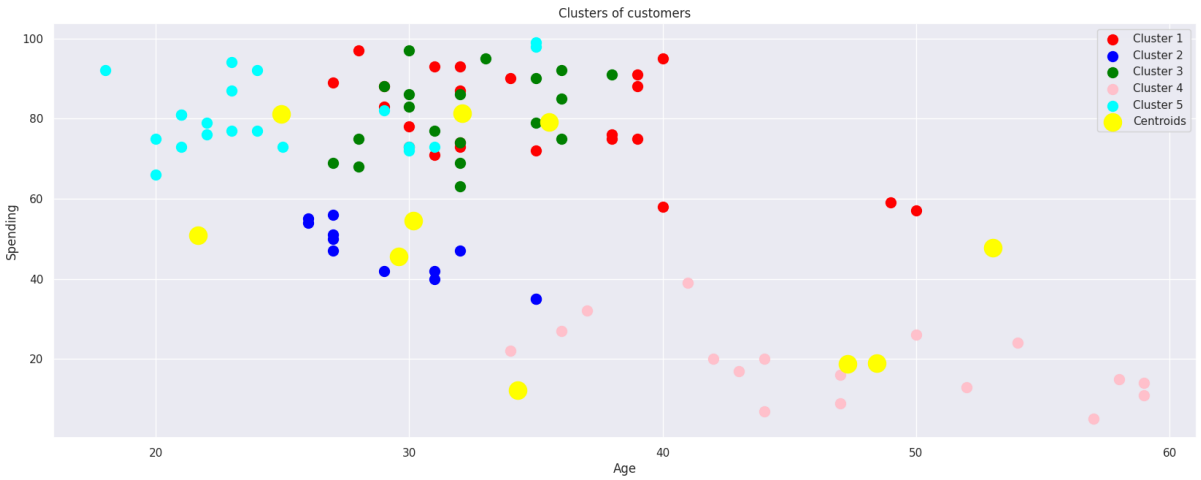
ما باید به خوشه 5 احترام بگذاریم و آنها را از دست ندهیم زیرا با وجود اینکه وضعیت خوبی ندارند سود خوبی به ما می دهند و همچنین می توان نتیجه گرفت که تبلیغات تاثیر بیشتری روی این قشر از جامعه دارد.

حال بیایم خوشه بندی را برحسب سن و میزان سود حاصله از مشتری انجام دهیم :



از این نوع مشاهدات می توان دریافت که گروه سنی جوانان که به رنگ آبی آمده اند، بیشترین خرید را دارند، بنابراین باید با بررسی محصولاتی که این گروه خریداری می کنند، محصولاتی که باید تبلیغ شوند، جهت جذب گروه آبی یعنی جوانان باشد.

خوشه بندی براساس سن ، درآمد و سود حاصل به فروشگاه :



همانطور که از این طبقه بندی مشخص شد، معمولاً گروه جوان با کمترین درآمد و بیشترین درآمد جزو مشتریان پرخرج فروشگاه محسوب می شوند و سعی می کنند بیشترین هزینه را داشته باشند، بنابراین بهتر است از نتایج به دست آمده استفاده شود و به عنوان تا جایی که می توانیم ایمیل باید طوری باشد که این قشر را نگه داریم.