МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

---

**Институт информационных технологий (ИИТ)**
**Кафедра прикладной математики (ПМ)**

# ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ

по дисциплине «Технологии и инструментарий машинного обучения»

## Практическая работа № 5

Студент группы ИМБО-01-21 Малкина В.В. _____ (подпись)

Старший преподаватель Высоцкая А.А. _____ (подпись)

Отчет представлен «___»_____2023г.

Москва 2023 г.

# 1 Предобработка данных

Выведем общую информацию о столбцах из набора (Рисунок 1).

```
import pandas as pd
data = pd.read_csv('/Users/vladamalkina/Downloads/Mall_Customers.csv')
```

```
data.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   Genre                   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
```

**Рисунок 1 — Вывод общей информации**

Исследуем данные на пропуски, выведем общую информацию (Рисунок 2).

```
data.isna().sum()
```

```
CustomerID                0
Genre                     0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

```
data = data.drop(columns=['CustomerID'])
```

**Рисунок 2 — Вывод пропусков**

Проверим численные признаки на выбросы:

```
numeric_cols = data.select_dtypes(exclude=["object"]).columns.tolist()
print(len(numeric_cols))
numeric_cols
```

```
3
['Age', 'Annual Income (k$)', 'Spending Score (1-100)']
```

```
outlier_percentage = {}
for feature in numeric_cols:
    tempData = data.sort_values(by=feature)[feature]
    Q1, Q3 = tempData.quantile([0.25, 0.75])
    IQR = Q3 - Q1
    Lower_range = Q1 - (1.5 * IQR)
    Upper_range = Q3 + (1.5 * IQR)

    outlier_count = ((tempData < Lower_range) | (tempData > Upper_range)
    outlier_perc = round((outlier_count / tempData.shape[0]) * 100, 2)
    outlier_percentage[feature] = outlier_perc

outlier_percentage
```

```
{'Age': 0.0, 'Annual Income (k$)': 1.0, 'Spending Score (1-100)': 0.0}
```

**Рисунок 3 — Выведем процентное содержание аномалий и выбросов**

Удалим выбросы:

```
import numpy as np
def out_iqr(df , column):
    global lower,upper
    q25, q75 = np.quantile(df[column], 0.25), np.quantile(df[column], 0.75)
    iqr = q75 - q25
    cut_off = iqr * 1.5
    lower, upper = q25 - cut_off, q75 + cut_off
    df1 = df[df[column] > upper]
    df2 = df[df[column] < lower]
    return 0

for col in numeric_cols:
    out_iqr(data, col)
    data = data[(data[col] < upper) | (data[col] > lower)]
```

**Рисунок 4 — Удаление выбросов**

Закодируем численные и категориальные признаки:

```
from sklearn.preprocessing import  LabelEncoder
label_encode = LabelEncoder()
data.Genre = label_encode.fit_transform(data.Genre)
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data[numeric_cols] = scaler.fit_transform(data[numeric_cols])
```

**Рисунок 5 — Кодирование признаков**

Для проверки пригодности данных для кластеризации проведем тест Хопкинса (Рисунок 6).

```
#pip install pyclustertend
```

```
from pyclustertend import hopkins
```

```
print(hopkins(data, data.shape[0]))
```

```
0.30052999583101786
```

**Рисунок 6 — Тест Хопкинса**

Применим метод Локтя для определения количества кластеров (Рисунок 7).

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
k_with_score = {}
wcss = []
for i in range(2,11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(data)
    wcss.append(kmeans.inertia_)
    k_with_score[i] = silhouette_score(data, kmeans.labels_)
```

```
import matplotlib.pyplot as plt
plt.plot(range(2,11), wcss, 'bx-')
plt.xlabel('number of clusters')
plt.ylabel('wcss')
plt.show()
```

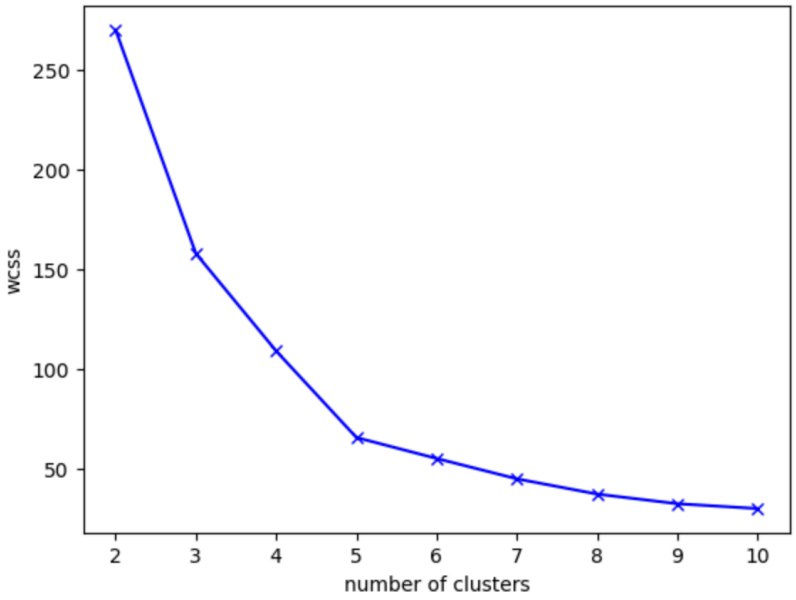


**Рисунок 7 — Метод Локтя**

Построим график зависимости значения метрики от количества кластеров (Рисунок 8).

```
plt.plot(k_with_score.keys() , k_with_score.values(), marker = 'o')
plt.title('K vs. Score')
plt.xlabel('K')
plt.ylabel('Score')
```

```
Text(0, 0.5, 'Score')
```

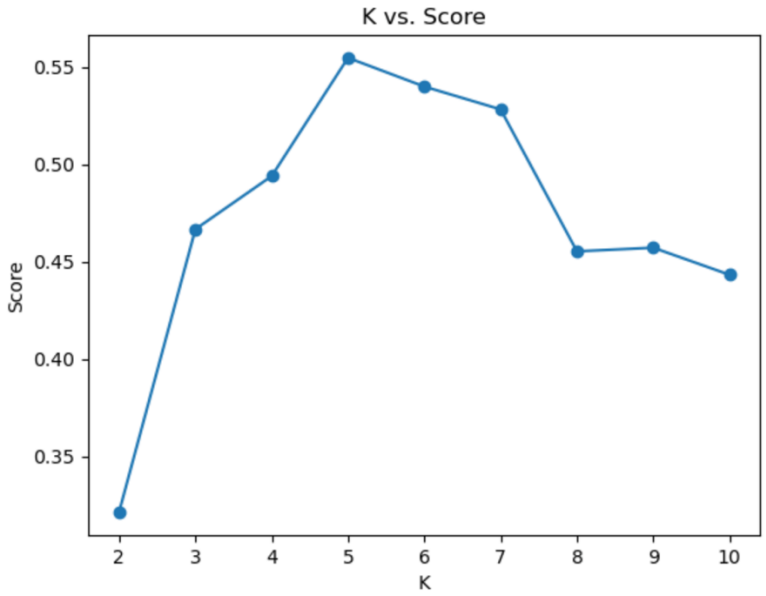


**Рисунок 8 — Метод Локтя**

Теперь рассмотрим коэффициент Силуэта для определения количества кластеров.

**silhouette method**

```
import matplotlib.cm as cm
from sklearn.metrics import silhouette_samples, silhouette_score
for n_clusters in range(2,11):
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(10, 3) #18 7
    ax1.set_xlim([-0.1, 1])
    ax1.set_ylim([0, len(data) + (n_clusters + 1) * 10])
    clusterer = KMeans(n_clusters=n_clusters, random_state=10)
    cluster_labels = clusterer.fit_predict(data)
    silhouette_avg = silhouette_score(data, cluster_labels)
    print("For n_clusters =", n_clusters,
          "silhouette_score is :", silhouette_avg)
    sample_silhouette_values = silhouette_samples(data, cluster_labels)
    y_lower = 10
    for i in range(n_clusters):
        ith_cluster_silhouette_values = \
            sample_silhouette_values[cluster_labels == i]
        ith_cluster_silhouette_values.sort()
        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i
        color = cm.nipy_spectral(float(i) / n_clusters)
        ax1.fill_betweenx(np.arange(y_lower, y_upper),
                          0, ith_cluster_silhouette_values,
                          facecolor=color, edgecolor=color, alpha=0.7)
        ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))
        y_lower = y_upper + 10
    ax1.set_title("График силуэта для кластеров")
    ax1.set_xlabel("Силуэт")
    ax1.set_ylabel("Номер кластера")
    ax1.axvline(x=silhouette_avg, color="red", linestyle="--")
```

**Рисунок 9 — Вычисление коэффициента Силуэта**

```
    ax1.set_yticks([])
    ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])
    colors = cm.nipy_spectral(cluster_labels.astype(float) / n_clusters)
    ax2.scatter(data[:, 0], data[:, 1], marker='.', s=30, lw=0, alpha=0.7,
                c=colors, edgecolor='k')
    centers = clusterer.cluster_centers_
    ax2.scatter(centers[:, 0], centers[:, 1], marker='o',
                c="white", alpha=1, s=200, edgecolor='k')
    for i, c in enumerate(centers):
        ax2.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,
                    s=50, edgecolor='k')
    ax2.set_title("Визуализация кластеров.")
    ax2.set_xlabel("Первый признак")
    ax2.set_ylabel("Второй признак")
    plt.suptitle(("Метод Силуэта"
                  "with n_clusters = %d" % n_clusters),
                 fontsize=14, fontweight='bold')
plt.show()
```

```
For n_clusters = 2 silhouette_score is : 0.3212707813918878
For n_clusters = 3 silhouette_score is : 0.46658474419000145
For n_clusters = 4 silhouette_score is : 0.4939069237513199
For n_clusters = 5 silhouette_score is : 0.5546571631111091
For n_clusters = 6 silhouette_score is : 0.5398800926790663
For n_clusters = 7 silhouette_score is : 0.5263454490712252
For n_clusters = 8 silhouette_score is : 0.4519906205874712
For n_clusters = 9 silhouette_score is : 0.4527137182637435
For n_clusters = 10 silhouette_score is : 0.45135229635551055
```

**Рисунок 10 — Коэффициент Силуэта для различных кластеров**

Графики визуализации кластеров и значения силуэта для кластеров.

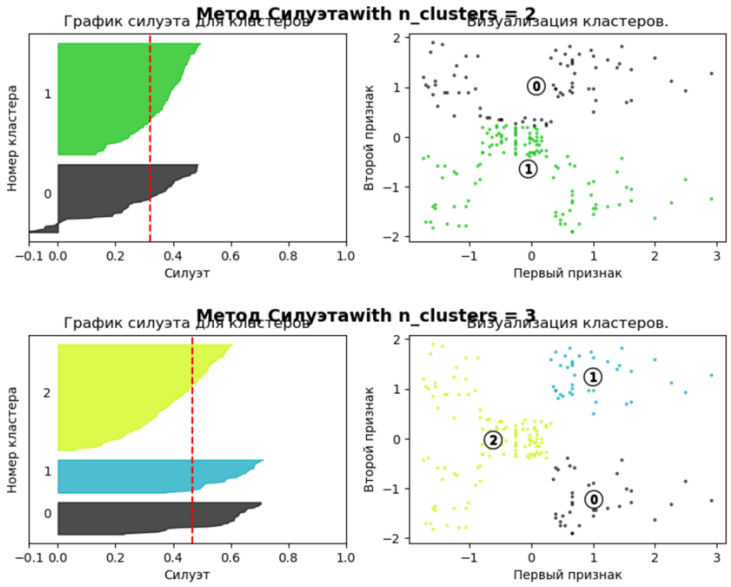


**Рисунок 11 — Визуализация коэффициента Силуэта**

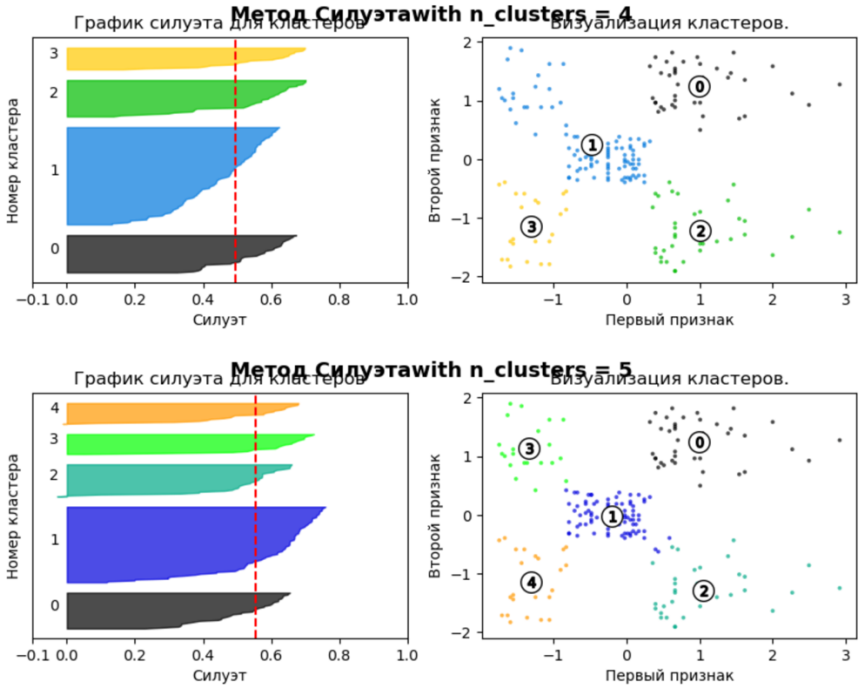


**Рисунок 12 — Визуализация коэффициента Силуэта**

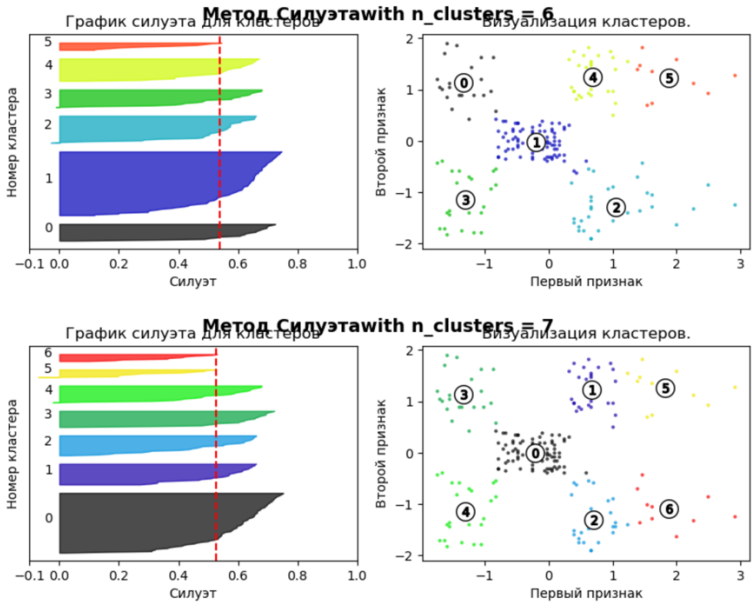


**Рисунок 13 — Визуализация коэффициента Силуэта**

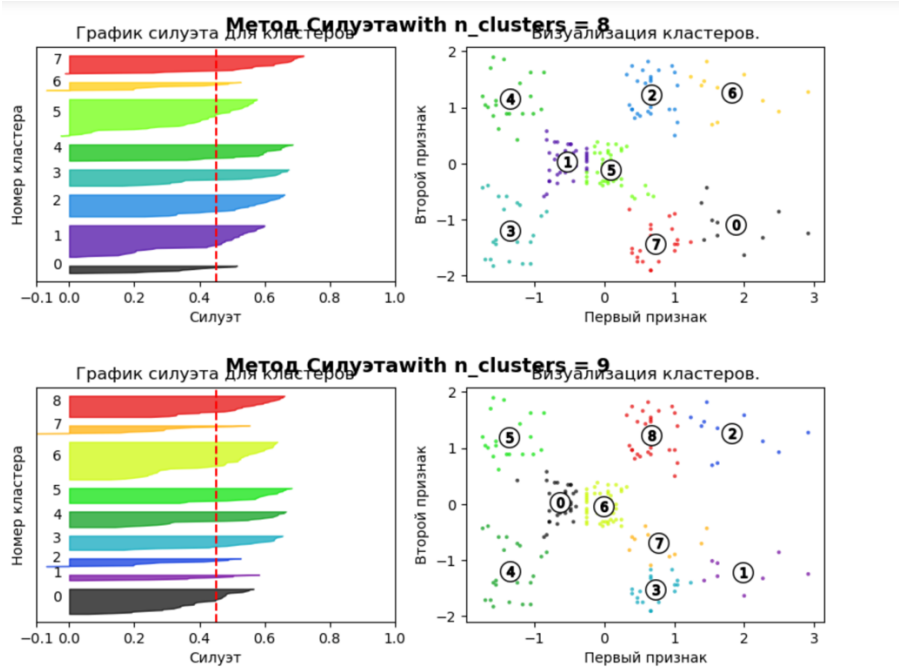


**Рисунок 14 — Визуализация коэффициента Силуэта**

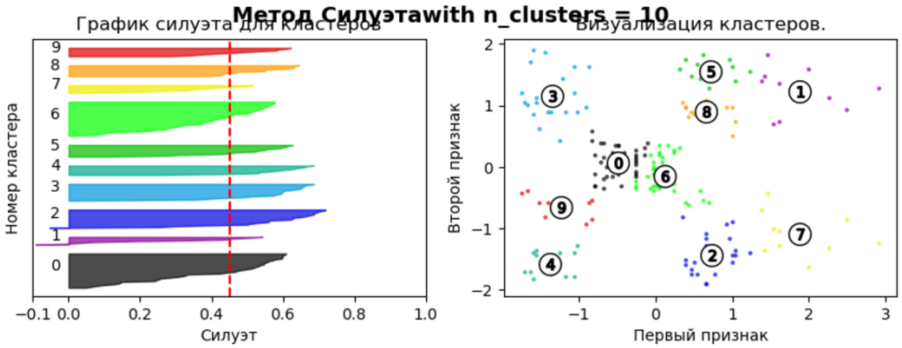


**Рисунок 15 — Визуализация коэффициента Силуэта**

# 2 Обучение моделей

Обучим KNN (Рисунок 15).

```
kmeans = KMeans(n_clusters=5, init = "k-means++", random_state=42)
y_kmeans = kmeans.fit_predict(data)
```

```
plt.scatter(data[y_kmeans==0,0], data[y_kmeans==0,1], s=60, c='red', label = 'CLuster 1')
plt.scatter(data[y_kmeans==1,0], data[y_kmeans==1,1], s=60, c='blue', label = 'CLuster 2')
plt.scatter(data[y_kmeans==2,0], data[y_kmeans==2,1], s=60, c='green', label = 'CLuster 3')
plt.scatter(data[y_kmeans==3,0], data[y_kmeans==3,1], s=60, c='violet', label = 'CLuster 4')
plt.scatter(data[y_kmeans==4,0], data[y_kmeans==4,1], s=60, c='yellow', label = 'CLuster 5')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 100, c = 'black', label = 'Centroids')
plt.legend()
plt.show()
```

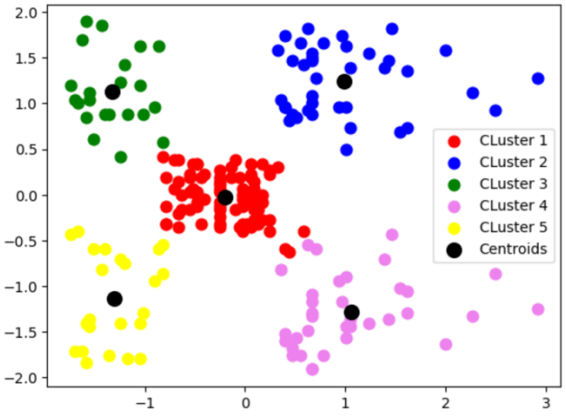


**Рисунок 16 — Обучение KNN**

Теперь обучим иерархическую кластеризацию (Рисунок 17).

**Hierarchy**

```
from sklearn.cluster import AgglomerativeClustering
```

```
import scipy.cluster.hierarchy as sch
import matplotlib.pyplot as plt
dendrogram = sch.dendrogram(sch.linkage(data, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.show()
```

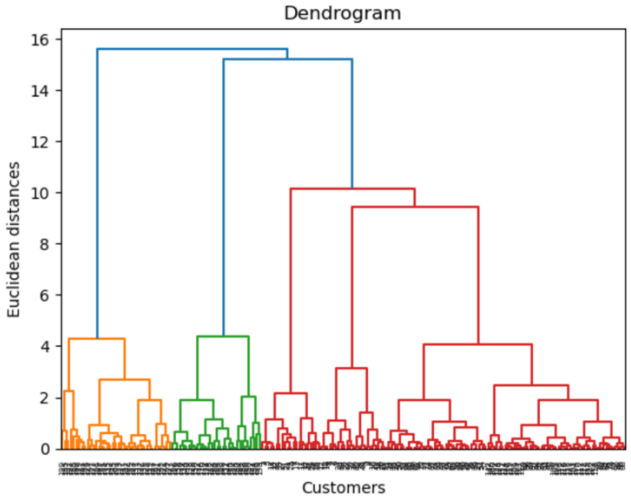


**Рисунок 17 — Дендрограмма для иерархической кластеризации**

Обучим агломеративную кластеризацию (Рисунок 18).

```
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
y_hc = hc.fit_predict(data)
```

```
# Visualising the clusters
plt.scatter(data[y_hc == 0, 0], data[y_hc == 0, 1], s = 100, c = 'red', label = 'first')
plt.scatter(data[y_hc == 1, 0], data[y_hc == 1, 1], s = 100, c = 'blue', label = 'second')
plt.scatter(data[y_hc == 2, 0], data[y_hc == 2, 1], s = 100, c = 'green', label = 'third')
plt.scatter(data[y_hc == 3, 0], data[y_hc == 3, 1], s = 100, c = 'cyan', label = 'fourth')
plt.scatter(data[y_hc == 4, 0], data[y_hc == 4, 1], s = 100, c = 'magenta', label = 'fifth')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```


**Рисунок 18 — Обучение иерархической кластеризации**

Теперь обучим DBSCAN (Рисунок 18).

## DBSCAN

```
data = data_st.iloc[:,[3,4]].values
```

```
from sklearn.cluster import DBSCAN
db = DBSCAN(eps = 3,min_samples = 4,metric = 'euclidean')
```

```
model = db.fit(data)
```

```
label = model.labels_
```

```
from sklearn import metrics

#identifying the points which makes up our core points
sample_cores = np.zeros_like(label,dtype=bool)

sample_cores[db.core_sample_indices_]=True

#Calculating the number of clusters

n_clusters = len(set(label))- (1 if -1 in label else 0)
print('No of clusters:',n_clusters)
```

```
No of clusters: 9
```

**Рисунок 19 — Обучение DBSCAN**

Визуализируем кластеры (Рисунок 19).

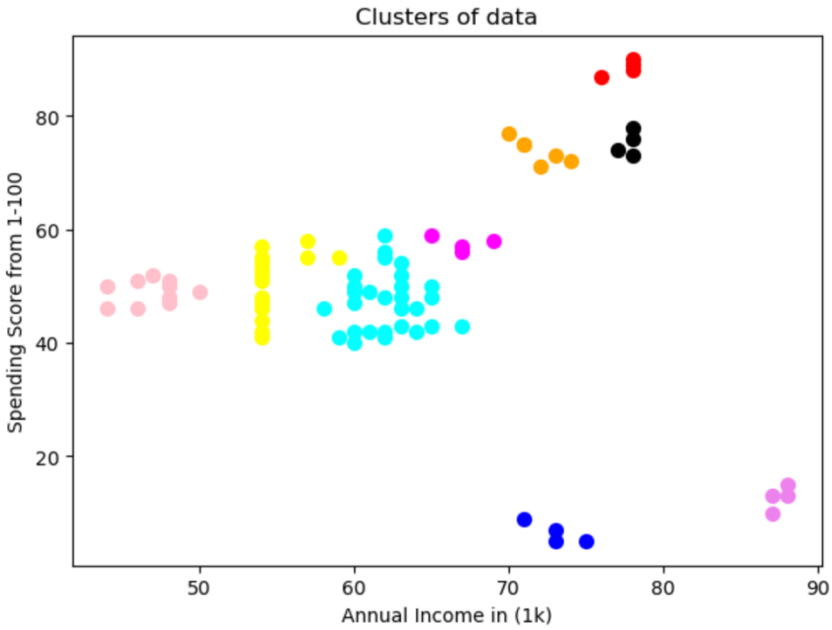


**Рисунок 20 — Визуализация кластеров**

# Выводы

Мы научились решать задачу кластеризации используя различные модели. Исследовали такие модели как KNN (метод ближайших соседей), DBSCAN и иерархическую кластеризацию. Для проверки данных на пригодность для кластеризации провели тест Хопкинса.