МИНОБРНАУКИ РО ССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра информационных систем

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Машинное обучение»
Тема: Знакомство с методом кластеризации K-mean с помощью пакета sklearn

Студентка гр. 2373 _____ Шавлохова А.А.

Преподаватель _____ Татчина Я.А.

Санкт-Петербург
2024

ЗАДАНИЕ

НА ЗНАКОМСТВО С МЕТОДОМ КЛАСТЕРИЗАЦИИ K-MEAN С ПОМОЩЬЮ ПАКЕТА SKLEARN ЛАБОРАТОРНУЮ

Студентка Шавлохова А.А.

Группа 2373

Тема лабораторной: Знакомство с методом кластеризации K-mean с помощью пакета sklearn

Задание на лабораторную:

1. К текущему датасету добавить новый аттрибут.

2. "Причесать" датасет: удалить выбросы и дубли, обработать пропущенные значения, найти кривые данные и т.п.

3. Построить графики зависимости одной переменной от другой (plt.scatter), описать какие кластеры (группы) можно увидеть. Построить несколько разных графиков, найти ярковыраженные группы.

4. Изучить перечисленные ноутбуки, по примеру, попробовать применить метод KMeans к вашему датасету (https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html)

https://www.kaggle.com/kushal1996/customer-segmentation-k-means-analysis

https://www.kaggle.com/karthickaravindan/k-means-clustering-project

https://www.kaggle.com/hellbuoy/online-retail-k-means-hierarchical-clustering

https://www.kaggle.com/sirpunch/k-means-clustering

5. Написать выводы

1 Часть

Для выполнения данной практической работы был выбран датасет с сайта: https://www.kaggle.com/datasets, который был предложен в Задании 1, лабораторной работы №1.

Был выбран датасет, используемый в предыдущей лабораторной работе, «Most Popular Programming Languages 2004-2024» (https://www.kaggle.com/datasets/muhammadroshaanriaz/most-popular-programming-languages-2004-2024). Каждая строка датасета представляет данные за определенный месяц, начиная с января 2004 года и заканчивая сентябрем 2024 года, отслеживая тенденции популярности представленных языков программирования во всем мире. В выбранном датасете представлены исключительно числовые данные (данные представлены в процентах).

Этот набор данных содержит следующие атрибуты:

- *Месяц*: дата (в формате год-месяц), когда данные были записаны.
- *Python Worldwide(%)*: процент глобальной популярности Python за этот месяц.
- *JavaScript Worldwide(%)*: процент глобальной популярности JavaScript.
- *Java Worldwide(%):* процент глобальной популярности Java.
- *C# Worldwide(%):* процент глобальной популярности C#.
- *PhP Worldwide(%):* процент глобальной популярности PhP.
- *Flutter Worldwide(%):* процент глобальной популярности Flutter.
- *React Worldwide(%):* процент глобальной популярности React.
- *Swift Worldwide(%):* процент глобальной популярности Swift.
- *TypeScript Worldwide(%):* процент глобальной популярности TypeScript.
- *Matlab Worldwide(%):* процент глобальной популярности Matlab.

2 Часть

1. К текущему датасету необходимо добавить новый атрибут, например, если есть дата рождения, то посчитать возраст или определить популярность фильма по количеству просмотров и т.п.

Для выполнения данного задания было решено добавить новый атрибут под названием «Distinction_Python_JavaScript(%)». Указанный атрибут показывает разницу популярности между Python Worldwide(%) и JavaScript Worldwide(%) в процентах.

```
#Атрибут, показывающий разницу популярности между Python Worldwide(%) и JavaScript Worldwide(%) в процентах
df['Distinction_Python_JavaScript(%)'] = df['Python Worldwide(%)'] - df['JavaScript Worldwide(%)']
df
```

Рис. 1.1 – Пример кода для создания нового атрибута под названием «Distinction_Python_JavaScript(%)»

Distinction_Python_JavaScript(%)
-68
-69
-72
-70
-63
...
44
44
42
40
48

Рис. 1.2 – Как выглядит новый атрибут под названием «Distinction_Python_JavaScript(%)» в датасете

2. "Причесать" датасет: удалить выбросы и дубли, обработать пропущенные значения, найти кривые данные и т.п.

Была сделана проверка на дубликаты, они не были найдены:

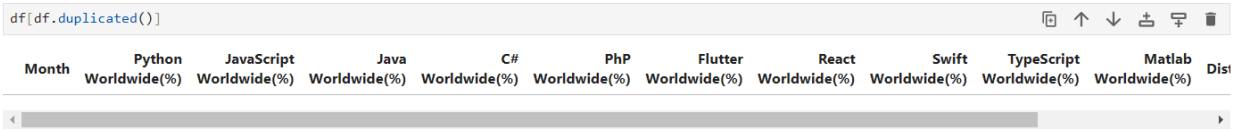


Рис. 2.1 – Пример кода для проверки на дубликаты и его выполнение

Был проведен поиск и удаление выбросов (Код был взят из первой лабораторной работы и модифицирован):

```python
def detect_and_clean_outliers_iqr(df):
    # Выбор числовых столбцов
    numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns

    outlier_mask = pd.Series(False, index=df.index)

    for column in numeric_cols:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1

        LowerBound = Q1 - 1.5 * IQR
        UpperBound = Q3 + 1.5 * IQR

        IQR_Outliers = (df[column] < LowerBound) | (df[column] > UpperBound)
        outlier_mask |= IQR_Outliers

        outlier_values = df[IQR_Outliers][column]

        if not outlier_values.empty:
            print(f"Выбросы в столбце {column}:")
            print(outlier_values)  # Выводим все выбросы
            number_of_outliers = outlier_values.shape[0]
            print(f"Количество выбросов в столбце {column}: {number_of_outliers}")
        else:
            print(f"Выбросов не обнаружено в столбце {column}.")

    # Удаляем выбросы
    df_cleaned = df[~outlier_mask]

    return df_cleaned.reset_index(drop=True)  # Возвращаем очищенный
```

Рис. 2.2 – Пример кода для поиска и удаления выбросов

```
Очистка завершена.
       Month  Python Worldwide(%)  JavaScript Worldwide(%)  Java Worldwide(%)  \
0    2005-12                   24                       69                 72
1    2006-01                   25                       67                 67
2    2006-02                   26                       68                 71
3    2006-04                   24                       68                 64
4    2006-05                   24                       68                 67
..       ...                  ...                      ...                ...
185  2021-08                   54                       26                 11
186  2021-09                   62                       28                 13
187  2021-10                   66                       28                 13
188  2021-11                   66                       31                 13
189  2021-12                   64                       28                 13

     C# Worldwide(%)  PhP Worldwide(%)  Flutter Worldwide(%)  \
0                 89                74                     5
1                 87                78                     6
2                 90                80                     5
3                 91                75                     5
4                 95                74                     6
..               ...               ...                   ...
185               29                21                    47
186               33                21                    49
187               32                21                    48
188               33                20                    51
189               31                20                    49

     React Worldwide(%)  Swift Worldwide(%)  TypeScript Worldwide(%)  \
0                     1                   9                        3
1                     2                  10                        4
2                     1                  10                        3
3                     1                  10                        3
4                     1                  10                        3
..                  ...                 ...                      ...
185                  47                  25                       44
186                  48                  23                       47
187                  51                  23                       47
188                  54                  41                       49
189                  52                  27                       47

     Matlab Worldwide(%)  Distinction_Python_JavaScript(%)
0                     64                               -45
1                     61                               -42
2                     68                               -42
3                     71                               -44
4                     67                               -44
..                   ...                               ...
185                   41                                28
186                   52                                34
187                   55                                38
188                   54                                35
189                   50                                36

[190 rows x 12 columns]
```

Рис. 2.3 – Датасет после удаления выборосов

Была проведена проверка на пропущенные значения. Они не были обнаружены:

```
print('Количество пропущенных значений:',df['Python Worldwide(%)'].isnull().sum())
print('Количество пропущенных значений:',df['JavaScript Worldwide(%)'].isnull().sum())
print('Количество пропущенных значений:',df['Java Worldwide(%)'].isnull().sum())
print('Количество пропущенных значений:',df['C# Worldwide(%)'].isnull().sum())
print('Количество пропущенных значений:',df['PhP Worldwide(%)'].isnull().sum())
print('Количество пропущенных значений:',df['Flutter Worldwide(%)'].isnull().sum())
print('Количество пропущенных значений:',df['React Worldwide(%)'].isnull().sum())
print('Количество пропущенных значений:',df['Swift Worldwide(%)'].isnull().sum())
print('Количество пропущенных значений:',df['TypeScript Worldwide(%)'].isnull().sum())
print('Количество пропущенных значений:',df['Matlab Worldwide(%)'].isnull().sum())
```

```
Количество пропущенных значений: 0
Количество пропущенных значений: 0
Количество пропущенных значений: 0
Количество пропущенных значений: 0
Количество пропущенных значений: 0
Количество пропущенных значений: 0
Количество пропущенных значений: 0
Количество пропущенных значений: 0
Количество пропущенных значений: 0
Количество пропущенных значений: 0
```

Рис. 2.4 – Пример кода и его выполнение на поиск количества пропущенных значений

Для поиска выбивающихся значений был выбран ящичный метод. Кривых данных не обнаружено:

```
def detect_and_visualize_outliers_horizontal(df):
    # Выбор числовых столбцов
    numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns

    if len(numeric_cols) == 0:
        print("Нет числовых столбцов для визуализации.")
        return

    num_cols = len(numeric_cols)
    num_rows = (num_cols // 4) + (num_cols % 4 > 0)
    plt.figure(figsize=(15, num_rows * 4))

    for i, col in enumerate(numeric_cols):
        plt.subplot(num_rows, 4, i + 1)  # Определяем строки и столбцы
        sns.boxplot(y=df[col], width=0.3)
        plt.title(f'Ящичный график:\n{col}')

    plt.tight_layout()
    plt.show()
```

```
detect_and_visualize_outliers_horizontal(df_cleaned)
```

Рис. 2.5 – Пример кода построения ящиков

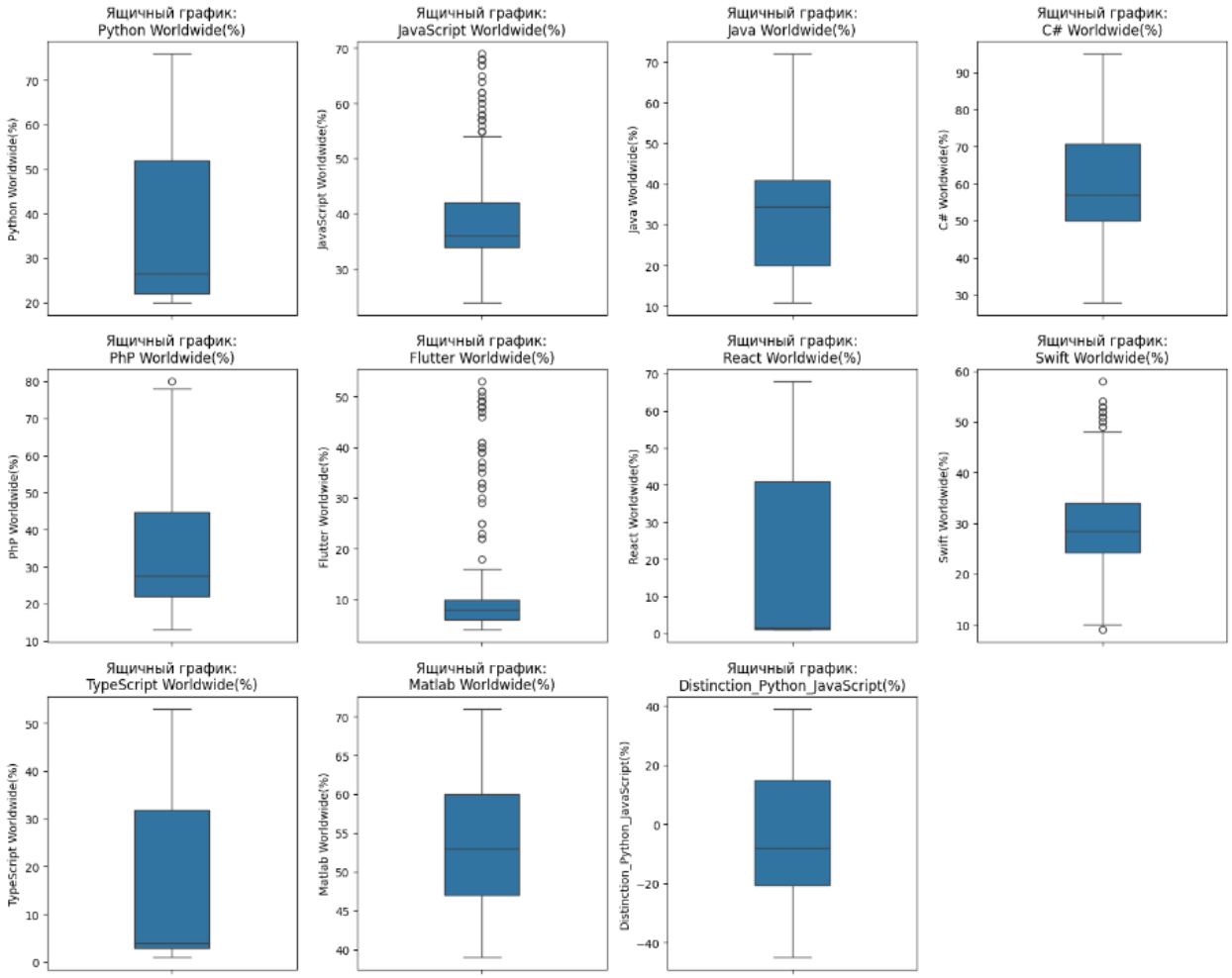


Рис. 2.6 – Выполнение кода для построения ящиков

3. Построить графики зависимости одной переменной от другой (plt.scatter), описать какие кластеры (группы) вы видите. Построить несколько разных графиков, найти ярковыраженные группы.

Для выполнения данного пункта были построены разнообразные графики и гистограммы.

Гистограммы месяца и популярности определенного языка программирования:

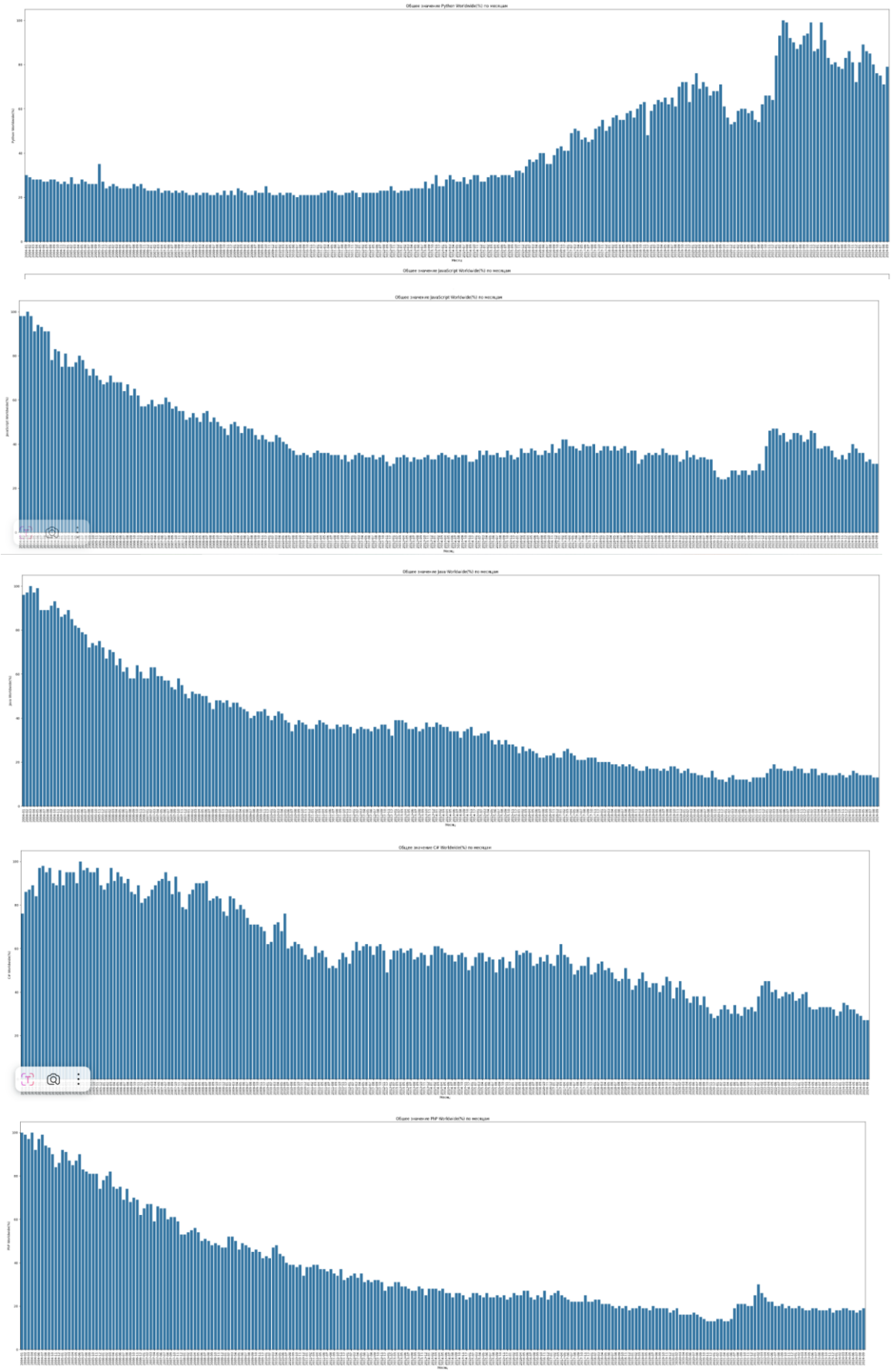


Рис. 3.1-5 – Гистограммы месяца и популярности определенного языка программирования

Были созданы гистограммы для «Python Worldwide(%)», «JavaScript Worldwide(%)», «Java Worldwide(%)», «C# Worldwide(%)», «PhP Worldwide(%)». Можно сразу заметить изменения популярности языков программирования (В Jupyter notebook можно приблизить каждый график).

Были рассмотрены графики зависимости для «Python Worldwide(%)», «JavaScript Worldwide(%)», «Java Worldwide(%)», «C# Worldwide(%)», «PhP Worldwide(%)» друг от друга:

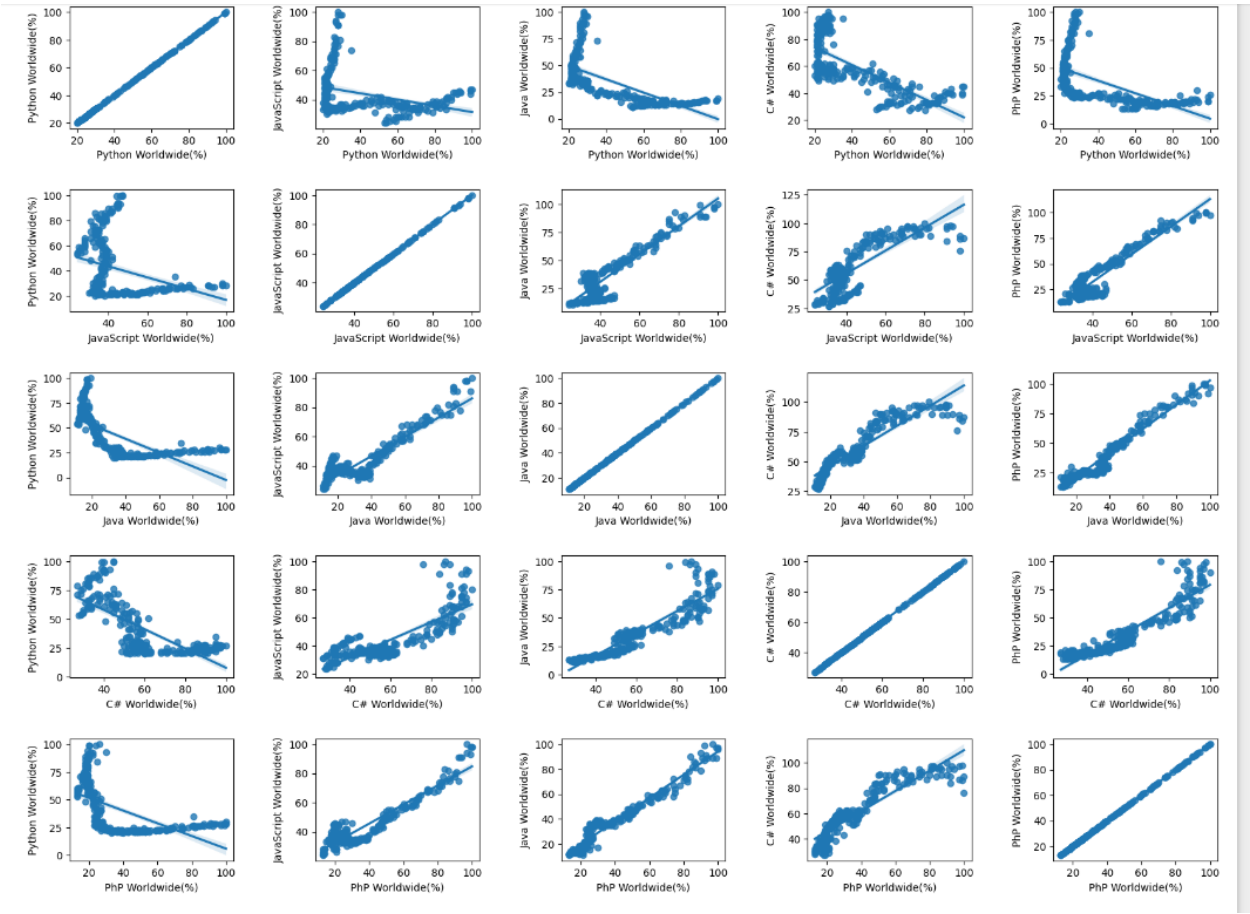


Рис. 3.6 – Графики зависимости языков программирования

Был также рассмотрен метод (plt.scatter), однако он не дал необходимых для выявления ярковыраженных групп результаты, так как выбранный датасет не имеет атрибутов прямо относящихся друг к другу (например при дополнительном параметре, где указывается кто чаще использует тот или иной язык программирования: дети, студенты или взрослые; то можно было построить зависимость исходя из статуса человека)

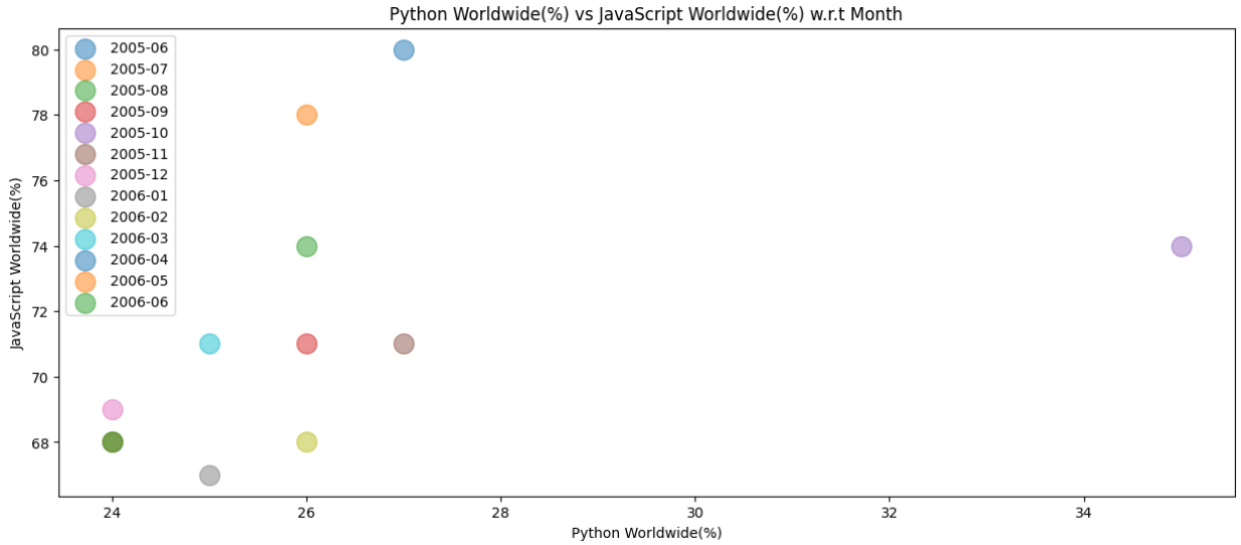


Рис. 3.7 – График построенный с помощью метода plt.scatter, учитывающий даты с 06.2005-06.2006 и популярность «Python Worldwide(%)», «JavaScript Worldwide(%)»

С помощью всех построенных графиков и гистограмм можно заметить несомненную зависимость популярности языка программирования от года и месяца, а также факт того, что мировая популярность некоторых языков затмила другие и их популярность пошла на спад (Например «Python Worldwide(%)» только повышает популярность, популярность «Java Worldwide(%)» идет на спад, а популярность «Matlab Worldwide(%)» пусть и скачет вверх и вниз, но держится на одном уровне). Следовательно языки можно разделить на группы, основанные на месяце года.

4. Изучить перечисленные ноутбуки, по примеру, попробовать применить метод KMeans к датасету

Применение метода KMeans к датасету:

```
X1 = df[['Python Worldwide(%)', 'JavaScript Worldwide(%)']].iloc[: , :].values
inertia = []
for n in range(1 , 11):
    algorithm = (KMeans(n_clusters = n, init='k-means++', n_init = 10, max_iter=300,
                        tol=0.0001,  random_state= 111, algorithm='lloyd') )
    algorithm.fit(X1)
    inertia.append(algorithm.inertia_)
```

```
plt.figure(1, figsize = (15, 6))
plt.plot(np.arange(1, 11), inertia, 'o')
plt.plot(np.arange(1, 11), inertia, '-', alpha = 0.5)
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.show()
```

```
algorithm = (KMeans(n_clusters = 6 ,init='k-means++', n_init = 10 ,max_iter=300,
                    tol=0.0001,  random_state= 111  , algorithm='elkan') )
algorithm.fit(X1)
labels1 = algorithm.labels_
centroids1 = algorithm.cluster_centers_

h = 0.02
x_min, x_max = X1[:, 0].min() - 1, X1[:, 0].max() + 1
y_min, y_max = X1[:, 1].min() - 1, X1[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z1 = algorithm.predict(np.c_[xx.ravel(), yy.ravel()])

plt.figure(1 , figsize = (15 , 7) )
plt.clf()
Z1 = Z1.reshape(xx.shape)
plt.imshow(Z1 , interpolation='nearest', extent=(xx.min(), xx.max(), yy.min(), yy.max()), cmap = plt.cm.Pastel2, aspect = 'auto', origin='lower')

plt.scatter( x = 'Python Worldwide(%)' ,y = 'JavaScript Worldwide(%)' , data = df , c = labels1 , s = 200 )
plt.scatter(x = centroids1[: , 0] , y =  centroids1[: , 1] , s = 300 , c = 'red' , alpha = 0.5)
plt.ylabel('JavaScript Worldwide(%)') , plt.xlabel('Python Worldwide(%)')
plt.show()
```

Рис. 4.1-2 – Пример кода для кластеризации

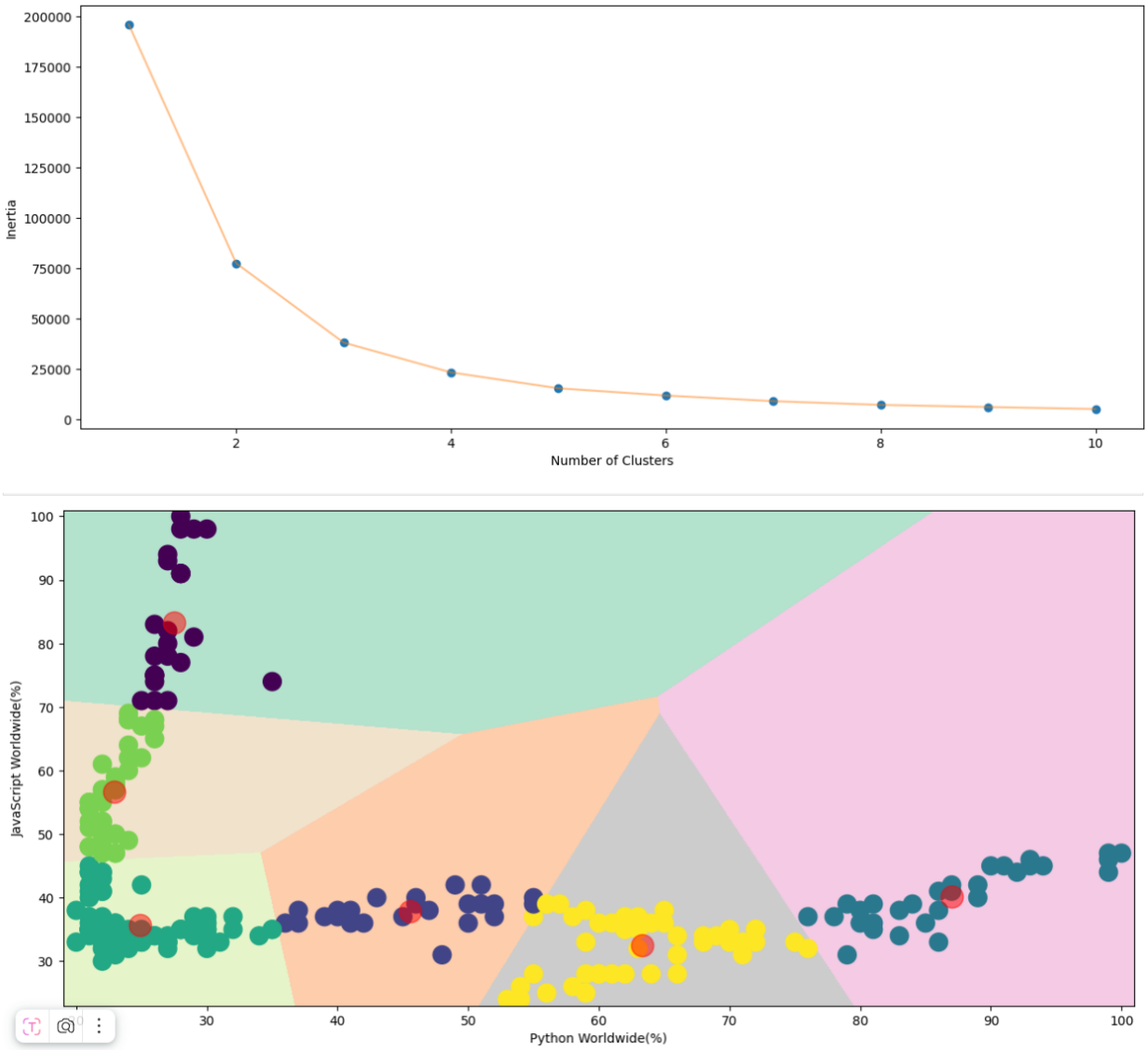


Рис. 4.3-4 – Кластеризация с использованием «Python Worldwide(%)» и «JavaScript Worldwide(%)»

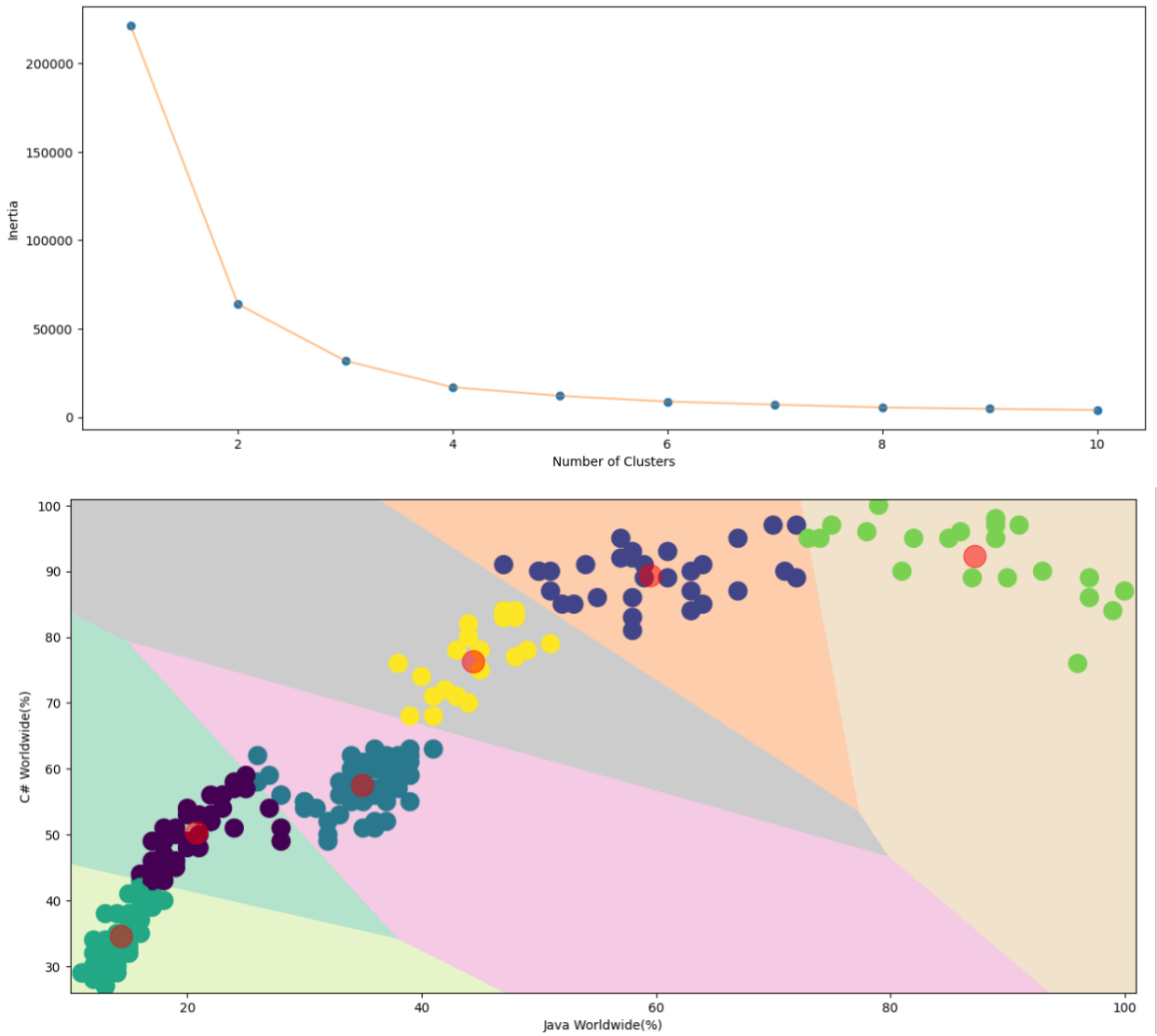


Рис. 4.5-6 – Кластеризация с использованием «C# Worldwide(%)» и «Java Worldwide(%)»

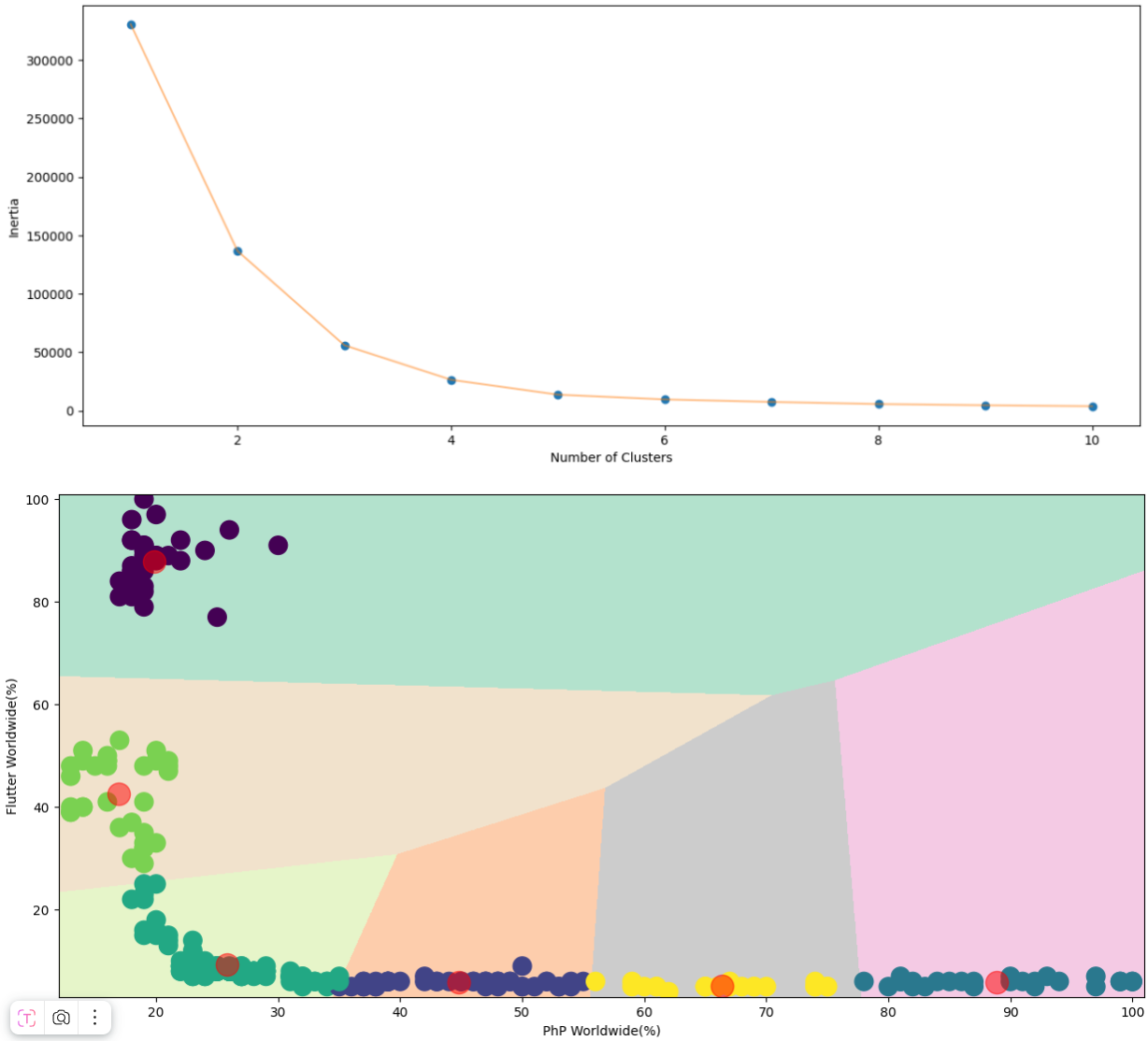
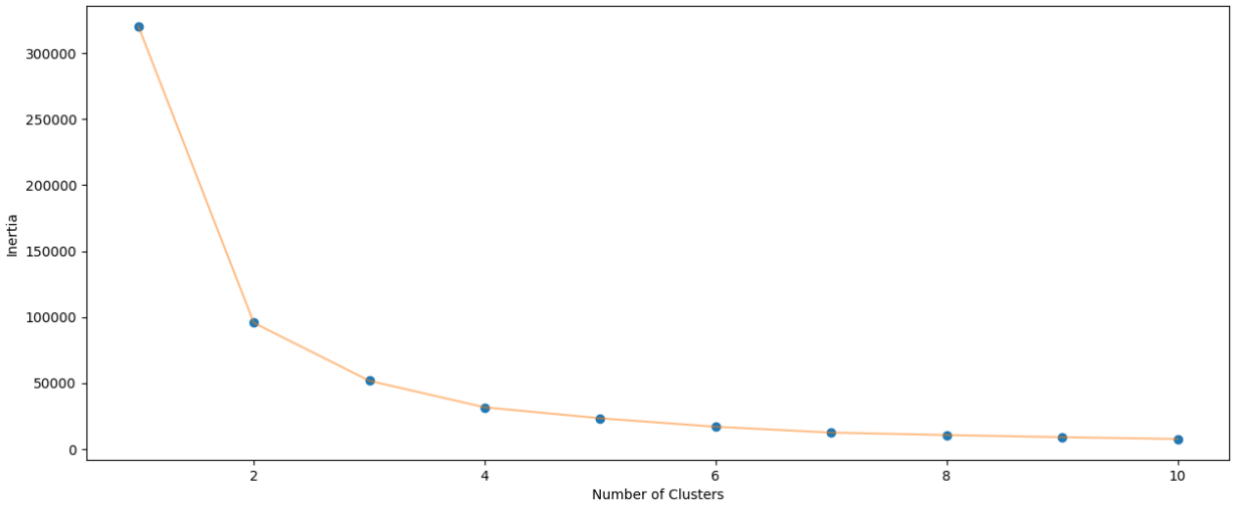


Рис. 4.7-8 – Кластеризация с использованием «Flutter Worldwide(%)» и «PhP Worldwide(%)»



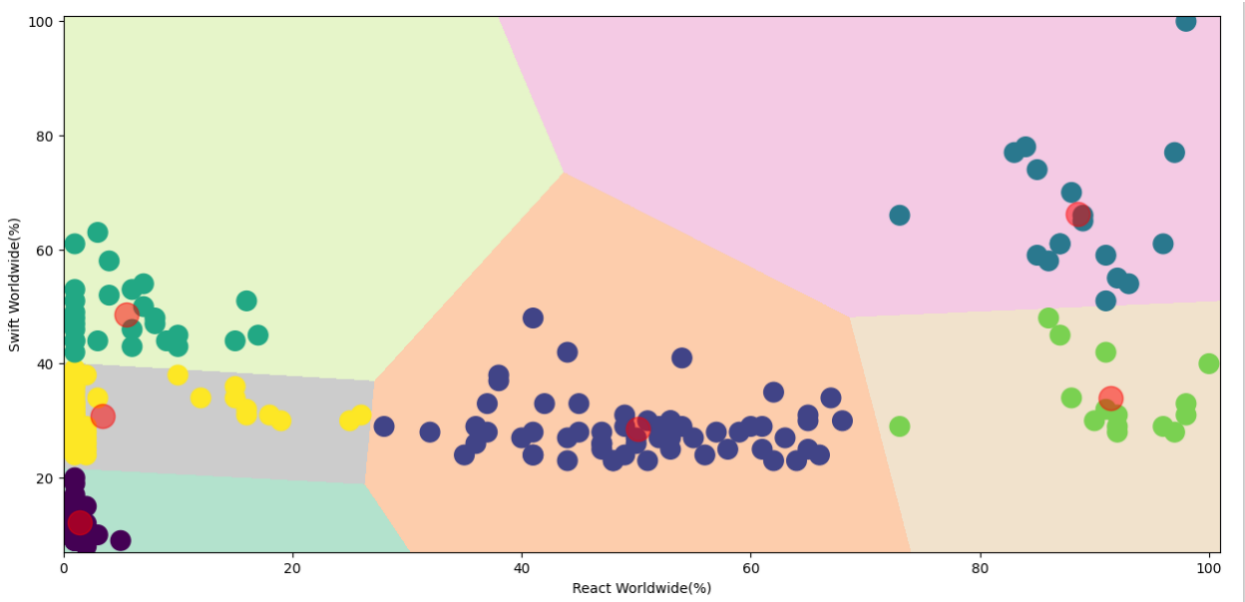


Рис. 4.9-10 – Кластеризация с использованием «Swift Worldwide(%)» и «React Worldwide(%)»

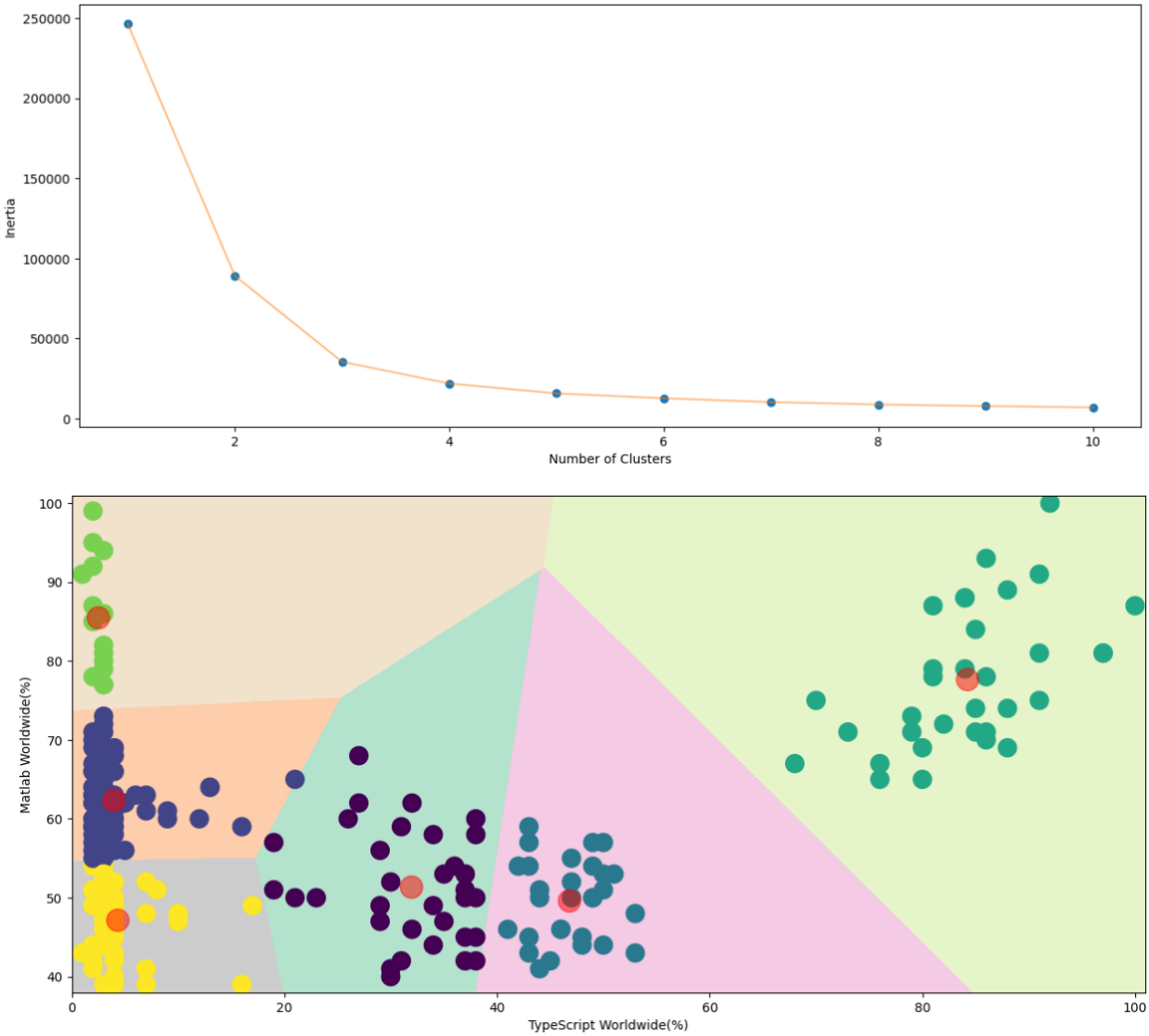


Рис. 4.11-12 – Кластеризация с использованием «Matlab Worldwide(%)» и «TypeScript Worldwide(%)»

ЗАКЛЮЧЕНИЕ

В данной работе были изучены:

- Наборы данных;
- Jupyter Notebook;
- Поиск и обработка пропущенных значений на Python;
- Поиск и обработка выбросов на Python;
- Поиск и обработка кривых значений на Python;
- Метод KMeans.

Были построены:

- Построение гистограмм на Python;
- Построение графиков на Python;
- Метод KMeans.

По построенным графикам, гистограммам были сделаны и записаны соответствующие выводы.

С помощью написания лабораторной работы стало ясно: как выявить, проанализировать, обработать значения атрибута. По выбранному датасету можно сказать об определенной степени зависимости популярности языков программирования, основанной на возможности совместного использования нескольких языков для одного проекта или насколько выбранный язык программирования подходит как для конкретной нишевой задачи, так и для разнообразных задач. Благодаря всем построенным графикам и гистограммам можно заметить несомненную зависимость популярности языка программирования от года и месяца, а также факт того, что мировая популярность некоторых языков затмила другие и их популярность пошла на спад (Например «Python Worldwide(%)» только повышает популярность, популярность «Java Worldwide(%)» идет на спад, а популярность «Matlab Worldwide(%)» пусть и скачет вверх и вниз, но держится на одном уровне).

Следовательно языки можно разделить на группы, основанные на месяце года.

Для выполнения практической работы была использована программа Jupyter-ноутбук; язык программирования Python; библиотеки: matplotlib, numpy, pandas, KMeans.

Были изучены ссылки, представленные в методических указаниях к лабораторной работе.

Написание практической работы помогло: усвоить информацию о наборах данных и взаимодействиями с ними, разобраться с необходимыми библиотеками для построения графиков, гистограмм на Python и методом KMeans.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. https://www.kaggle.com/kushal1996/customer-segmentation-k-means-analysis

2. https://www.kaggle.com/karthickaravindan/k-means-clustering-project

3. https://www.kaggle.com/hellbuoy/online-retail-k-means-hierarchical-clustering

4. https://www.kaggle.com/sirpunch/k-means-clustering

5. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html