Московский авиационный институт

(Национальный исследовательский университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»
Дисциплина: «Искусственный интеллект»

Лабораторная работа № 1

Студент: Дюсекеев А.

Группа: М80-304Б

Москва, 2020

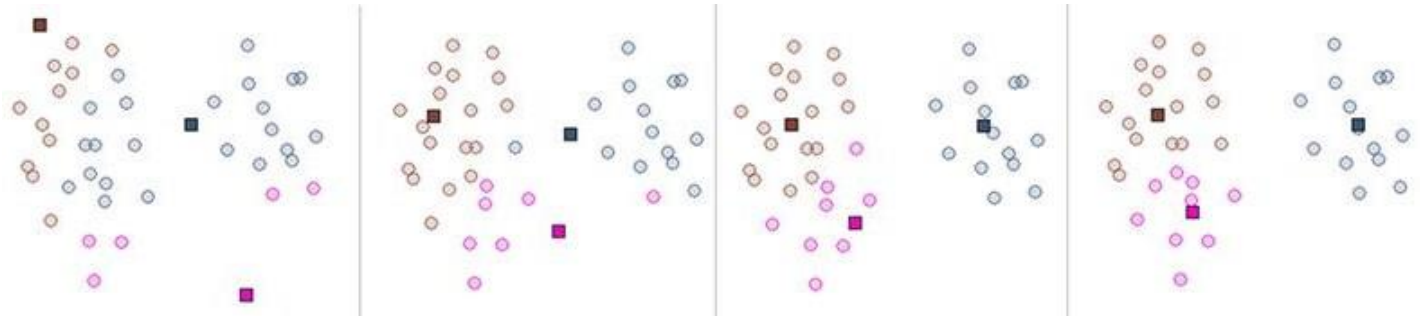
Постановка задачи

Требуется реализовать класс на выбранном языке программирования, который реализует один из алгоритмов машинного обучения. Обязательным является наличия в классе двух методов fit, predict. Необходимо проверить работу вашего алгоритма на ваших данных (на таблице и на текстовых данных), произведя необходимую подготовку данных. Также необходимо реализовать алгоритм полиномиальной регрессии, для предсказания значений в таблице. Сравнить результаты с стандартной реализацией sklearn, определить в чем сходство и различие ваших алгоритмов. Замерить время работы алгоритмов.

Метод К-средних

Метод k-средних (англ. k-means) - наиболее популярный метод кластеризации.
Алгоритму широко отдается предпочтение из-за его простоты реализации, большой скорости (а это очень важно при работе с видео).
Действие алгоритма таково, что он стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров. В простонародье говоря, это итеративный алгоритм, который делит данное множество пикселей на k кластеров точки, которых являются максимально приближенными к их центрам, а сама кластеризация происходит за счет смещения этих же центров. Такой себе принцип разделяй и властвуй.
Также следует оговорить то, что метод к-средних очень чувствительный к шуму, который может существенно исказить результаты кластеризации. Так что в идеале, перед кластеризацией, нужно прогнать кадры через фильтры предназначенные для его уменьшения. Вот собственно сам принцип простейшей кластеризации методом к-средних:

1. Надо выбрать из множества k пикселей те пиксели, которые будут центроидами соответствующих k кластеров.
   Выборка начальных центроидов может быть как рандомной так и по определенному алгоритму.
2. Входим в цикл, который продолжается до тех пор, пока центроиды кластеров не перестанут изменять свое положение.
3. Обходим каждый пиксель и смотрим, к какому центроиду какого кластера он является близлежащим.
4. Нашли близлежащий центроид? Привязываем пиксель к кластеру этого центроида.
5. Перебрали все пиксели? Теперь нужно высчитать новые координаты центроидов k кластеров.
6. Теперь проверяем координаты новых центроидов. Если они соответственно равны предыдущим центроидам — выходим из цикла, если нет возвращаемся к пункту 3.



Вот картинка, которая приблизительно демонстрируют работу алгоритма:
Источник: https://habr.com/ru/post/165087/

Листинг программы

```
import numpy as np
import pandas as pd
import pylab as plt
import sklearn.cluster as km


class MyKMeans:
    """ Класс реализующий алгоритм К-Средних """
    def init (self, n_clusters=2):
        self.n_clusters = n_clusters
        self.labels = None
        self.cluster_centers = None
        self.tol = 0.0001

    def fit(self, X):
        # Случайные центры
        self.cluster_centers = X[np.random.choice(range(X.shape[0]), self.n_clusters)].copy()

        samples = X.shape[0]
        self.labels = np.zeros(shape=(samples), dtype=np.uint8)
        min_dist = np.zeros(shape=(samples), dtype=np.float64)

        while True:
            # Перераспределение точек по кластерам
            for i in range(samples):
                min_dist[i] = np.linalg.norm(X[i] - self.cluster_centers[0])
                self.labels[i] = 0

            for clust in range(1, self.n_clusters):
                for i in range(samples):
                    dist = np.linalg.norm(X[i] - self.cluster_centers[clust])
                    if dist < min_dist[i]:
                        min_dist[i], self.labels[i] = dist, clust
```

```
            # Пересчет центров
            new_centers = np.array([X[self.labels == i].sum(axis=0) / X[self.labels == i].shape[0] for i
in range(self.n_clusters)])

            if (np.abs(new_centers - self.cluster_centers) < self.tol).all():
                break

            # print(min_dist, self.labels, self.cluster_centers, new_centers, sep='\n')
            # print("---------------------------------------------------------- \n")
            self.cluster_centers = new_centers.copy()

        return self

    def predict(self, X):
        samples = X.shape[0]
        min_dist = np.zeros(shape=(samples), dtype=np.float64)
        labels = np.zeros(shape=(samples), dtype=np.uint8)

        for i in range(samples):
            min_dist[i] = np.linalg.norm(X[i] - self.cluster_centers[0])
            labels[i] = 0

        for clust in range(1, self.n_clusters):
            for i in range(samples):
                dist = np.linalg.norm(X[i] - self.cluster_centers[clust])
                if dist < min_dist[i]:
                    min_dist[i], labels[i] = dist, clust

        return labels


if__name__== '_main_':

    data = pd.read_csv('MSFT.csv', index_col=0)
    # Нормировка Volume
    data['Volume'] = (data['Volume'] - data['Volume'].mean()) / data['Volume'].std()
    print("Первые 10 записей MSFT.csv: ")
    print(data[:10])

    sklearn_kmeans = km.KMeans(3).fit(data.values)
    print("Центры, которые вычислила библиотека sklearn:")
    print(sklearn_kmeans.cluster_centers_)

    my_kmeans = MyKMeans(3).fit(data.values)
    print("Центры, которые вычислила MyKMeans:")
    print(my_kmeans.cluster_centers)

    plt.grid()
    plt.xlabel('High')
    plt.ylabel('Low')
    plt.scatter(data['High'],data['Low'], c=my_kmeans.labels)
    plt.show()
```

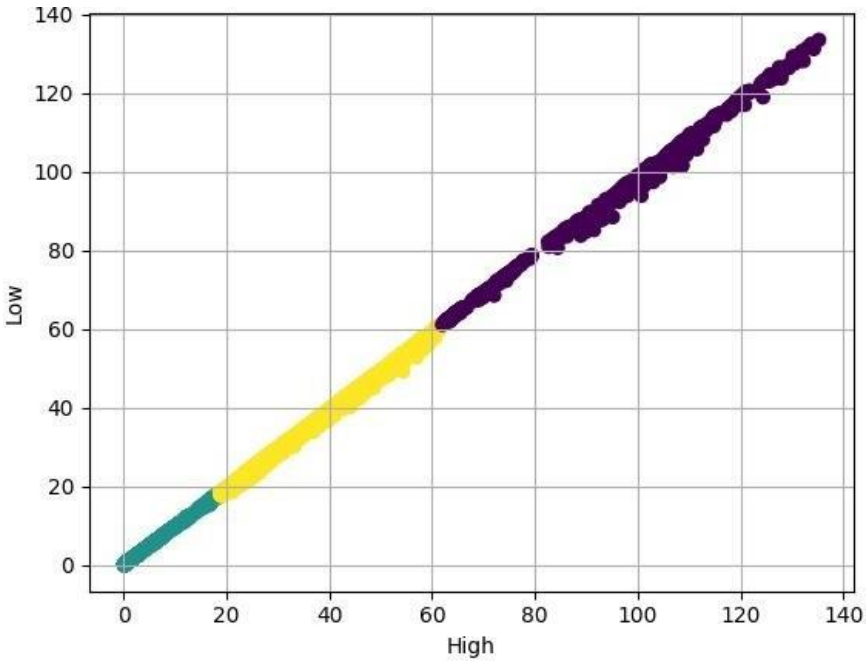
```
print("Случайные 20 записей для предсказания: ")
X = data.sample(20).values
print(X)

print("Предикт от sklearn: ")
print(sklearn_kmeans.predict(X))

print("Предикт от MyKMeans: ")
print(my_kmeans.predict(X))
```

Результат работы программы

```
~/KMeans python KMeans.py
Первые 10 записей MSFT.csv:
                Open      High       Low     Close  Adj Close     Volume
Date
1986-03-13  0.088542  0.101563  0.088542  0.097222   0.069996  24.939082
1986-03-14  0.097222  0.102431  0.097222  0.100694   0.072496   6.347915
1986-03-17  0.100694  0.103299  0.100694  0.102431   0.073746   1.852176
1986-03-18  0.102431  0.103299  0.098958  0.099826   0.071871   0.171824
1986-03-19  0.099826  0.100694  0.097222  0.098090   0.070621  -0.338719
1986-03-20  0.098090  0.098090  0.094618  0.095486   0.068746  -0.067909
1986-03-21  0.095486  0.097222  0.091146  0.092882   0.066871  -0.027954
1986-03-24  0.092882  0.092882  0.089410  0.090278   0.064996   0.108191
1986-03-25  0.090278  0.092014  0.089410  0.092014   0.066246  -0.744934
1986-03-26  0.092014  0.095486  0.091146  0.094618   0.068121  -0.984667
Центры, которые вычислила библиотека sklearn:
[[ 3.21599312e+01  3.25340788e+01  3.17933535e+01  3.21666684e+01
   2.56165179e+01 -1.94275354e-02]
 [ 3.56319662e+00  3.61240324e+00  3.51360561e+00  3.56513661e+00
   2.57575395e+00  2.14448616e-01]
 [ 9.15199212e+01  9.22694313e+01  9.06908690e+01  9.15349291e+01
   8.97672454e+01 -8.80844759e-01]]
Центры, которые вычислила MyKMeans:
[[ 9.15199212e+01  9.22694313e+01  9.06908690e+01  9.15349291e+01
   8.97672454e+01 -8.80844759e-01]
 [ 3.56319662e+00  3.61240324e+00  3.51360561e+00  3.56513661e+00
   2.57575395e+00  2.14448616e-01]
 [ 3.21599312e+01  3.25340788e+01  3.17933535e+01  3.21666684e+01
   2.56165179e+01 -1.94275354e-02]]
```

```
Случайные 20 записей для пресказания:
[[ 9.72220000e-02  1.02431000e-01  9.72220000e-02  1.00694000e-01
   7.24960000e-02  6.34791520e+00]
 [ 2.77999990e+01  2.83400000e+01  2.77900010e+01  2.82000010e+01
   2.28112960e+01  2.03026264e-01]
 [ 2.55300010e+01  2.58600010e+01  2.53700010e+01  2.58400000e+01
   2.12972050e+01 -5.70745029e-01]
 [ 7.74218800e+00  7.82031300e+00  7.73437500e+00  7.80468800e+00
   5.61903900e+00 -2.32417753e-01]
 [ 2.57800010e+01  2.57800010e+01  2.52199990e+01  2.53099990e+01
   2.02405410e+01  9.96174967e-03]
 [ 4.19399990e+01  4.22099990e+01  4.13600010e+01  4.18400000e+01
   3.78155400e+01 -5.00052078e-01]
 [ 2.51299990e+01  2.53400000e+01  2.46500000e+01  2.51000000e+01
   2.05575520e+01  9.52809171e-02]
 [ 2.89843800e+00  2.91406300e+00  2.78906300e+00  2.85156300e+00
   2.05300200e+00  1.51270577e-01]
 [ 2.56250000e+00  2.62500000e+00  2.48437500e+00  2.50781300e+00
   1.80551700e+00  7.78802829e-01]
 [ 4.04500010e+01  4.26899990e+01  3.97200010e+01  4.16800000e+01
   3.84373860e+01  7.11020734e-01]
 [ 8.54167000e-01  8.61111000e-01  8.29861000e-01  8.29861000e-01
   5.97464000e-01 -2.21812268e-01]
 [ 2.78200000e+01  2.81000000e+01  2.77900010e+01  2.80100000e+01
   2.07087730e+01 -4.81944652e-01]
 [ 3.50937500e+01  3.60625000e+01  3.50625000e+01  3.56562500e+01
   2.56709670e+01 -1.19641994e-01]
 [ 1.19860001e+02  1.20430000e+02  1.19150002e+02  1.19970001e+02
   1.19527557e+02 -9.81874535e-01]
 [ 3.12199990e+01  3.14000000e+01  3.09699990e+01  3.11000000e+01
   2.59955060e+01 -7.71314054e-01]
 [ 2.85499990e+01  2.91900010e+01  2.83300000e+01  2.89300000e+01
   2.21599480e+01  2.46470567e-02]
 [ 5.47300000e+01  5.50900000e+01  5.40000000e+01  5.50900000e+01
   5.11465570e+01  5.78914630e-01]
 [ 3.40250020e+01  3.49949990e+01  3.38750000e+01  3.43950000e+01
   2.47629260e+01  1.08966067e+00]
 [ 5.30937500e+01  5.43125000e+01  5.12500000e+01  5.16875000e+01
   3.72127880e+01  7.96424524e-02]
 [ 3.32187500e+01  3.35625000e+01  3.24375000e+01  3.27968750e+01
   2.36123410e+01  4.24084325e-01]]
Предикт от sklearn:
[1 0 0 1 0 0 0 1 1 0 1 0 0 2 0 0 0 0 0 0]
Предикт от MyKMeans:
[1 0 0 1 0 0 0 1 1 0 1 0 0 2 0 0 0 0 0 0]
```

Результат кластеризации



Вывод

Благодаря проделанной работе, я познакомился с алгоритмом k-means и реализовал его.
Была проверена работа моего алгоритма на данных.
Результаты работы обоих алгоритмов (собственной реализации и из библиотеки sklearn) совпадают с точностью до нумерации кластеров.