



دانشگاه صنعتی امیرکبیر
(پلیتکنیک تهران)

درس تحلیل داده و اطلاعات پروژه نهایی

امیرمهدی زرین نژاد

۹۷۳۱۰۸۷

سوال ۱)

در سوال اول با توجه به اسلایدهای درس عمل می‌کنیم و فرمول و روابط را پیاده‌سازی می‌کنیم:

Normalization

- Min-max normalization:** to $[new_min_A, new_max_A]$

$$v' = \frac{v - min_A}{max_A - min_A} (new_max_A - new_min_A) + new_min_A$$
 - Ex. Let income range \$12,000 to \$98,000 normalized to [0.0, 1.0]. Then \$73,000 is mapped to $\frac{73,600 - 12,000}{98,000 - 12,000} (1.0 - 0) + 0 = 0.716$
- Z-score normalization** (μ : mean, σ : standard deviation):

$$v' = \frac{v - \mu_A}{\sigma_A}$$
 - Ex. Let $\mu = 54,000$, $\sigma = 16,000$. Then $\frac{73,600 - 54,000}{16,000} = 1.225$
- Normalization by decimal scaling**

$$v' = \frac{v}{10^j} \quad \text{Where } j \text{ is the smallest integer such that } \text{Max}(|v'|) < 1$$

28

در کد نیز نام‌گذاری‌ها و پرینت‌ها به خوبی و قابل درک انجام شده‌اند.

```

question_1.py question_2.py question_3.py
output_set = []
for item in input_set:
    new_item = (item - min(input_set)) / (max(input_set) - min(input_set)) * (NEW_MAX - NEW_MIN) + NEW_MIN
    new_item = float("{:.4f}".format(new_item))
    output_set.append(new_item)
return output_set

def decimal_scaling_normalization(input_set):
    output_set = []
    for item in input_set:
        new_item = item / 10 ** J
        new_item = float("{:.4f}".format(new_item))
        output_set.append(new_item)
    return output_set

if __name__ == '__main__':
    print("\n> Given Dataset:\n ",
          DATASET)
    print("\n> Decimal scaling normalization on interval [-1, 1]:\n ",
          decimal_scaling_normalization(input_set=DATASET))
    print("\n> Min-Max normalization on interval [0, 1]:\n ",
          min_max_normalization(input_set=DATASET))
    print("\n> Standard Deviation normalization:\n ",
          standard_deviation_normalization(input_set=DATASET))

/Users/apple/PycharmProjects/Term9/DA/Final Project/venv/bin/python "/Users/apple/PycharmProjects/Term9/DA/Final Project/question_1.py"
> Given Dataset:
[-5.0, 23.0, 17.6, 7.23, 1.11]
> Decimal scaling normalization on interval [-1, 1]:
[-0.05, 0.23, 0.176, 0.0723, 0.0111]
> Min-Max normalization on interval [0, 1]:
[0.0, 1.0, 0.8071, 0.4368, 0.2182]
> Standard Deviation normalization:
[-1.3378, 1.3789, 0.855, -0.1512, -0.745]

```

سوال ۲)

در سوال ۲ نیز با فرمول زیر عمل می‌کنیم:

$$\text{Upper limit} = Q3 + 1.5 * (Q3 - Q1) = \text{mean} + 2.7 * \text{stdev}$$

$$\text{Lower limit} = Q1 - 1.5 * (Q3 - Q1) = \text{mean} - 2.7 * \text{stdev}$$

در کد نیز نام‌گذاری‌ها و پرینت‌ها به خوبی و قابل درک انجام شده‌اند.

همانطور که در نتیجه می‌بینیم، با کاهش ترش‌هولد از ۳ به ۲، داده (عدد) ۱۵ در دسته داده‌های پرت قرار می‌گیرد.

```
question_1.py question_2.py question_3.py
if __name__ == '__main__':
    print("> Given Dataset:\n", DATASET)

    dataset = sorted(DATASET)
    print("\n> Sorted Dataset:\n", dataset)

    # q1, q2, q3 = find_quantiles(dataset)
    # print("\n> Q1: " + str(q1) + ', Median: ' + str(q2) + ', Q3: ' + str(q3))
    # iqr, lower_bound, upper_bound = find_bounds(q1, q3, threshold=1.5)
    # print_output(iqr, lower_bound, upper_bound, threshold=1.5)

    mean = mean(dataset)
    pstdev = standard_deviation(dataset, mean)

    lower_bound, upper_bound = find_bounds(mean, pstdev, threshold=3)
    print_output(lower_bound, upper_bound, threshold=3)
    outliers = find_outliers(dataset, lower_bound, upper_bound)
    print("> Outliers: " + str(outliers))

    lower_bound, upper_bound = find_bounds(mean, pstdev, threshold=2.7)
    print_output(lower_bound, upper_bound, threshold=2.7)
    outliers = find_outliers(dataset, lower_bound, upper_bound)
    print("> Outliers: " + str(outliers))

    lower_bound, upper_bound = find_bounds(mean, pstdev, threshold=2)

/Users/apple/PycharmProjects/Term9/DA/Final Project/venv/bin/python "/Users/apple/PycharmProjects/Term9/DA/Final Project/question_2.py"
> Given Dataset:
[3, 1, 0, 2, 7, 3, 6, 4, 2, 0, 0, 10, 15, 6]

> Sorted Dataset:
[0, 0, 0, 1, 2, 2, 3, 3, 4, 6, 6, 7, 10, 15]

> threshold: 3, Upper bound: 16.644704296234153, Lower bound: -8.216132867662722
> Outliers: []

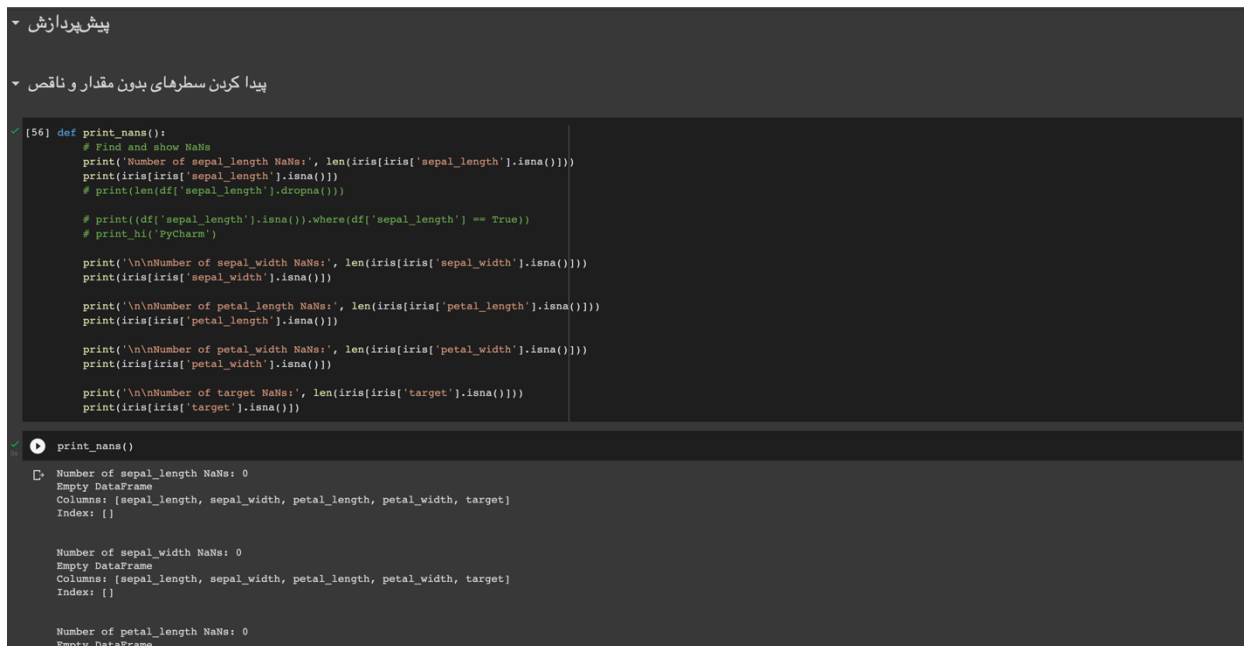
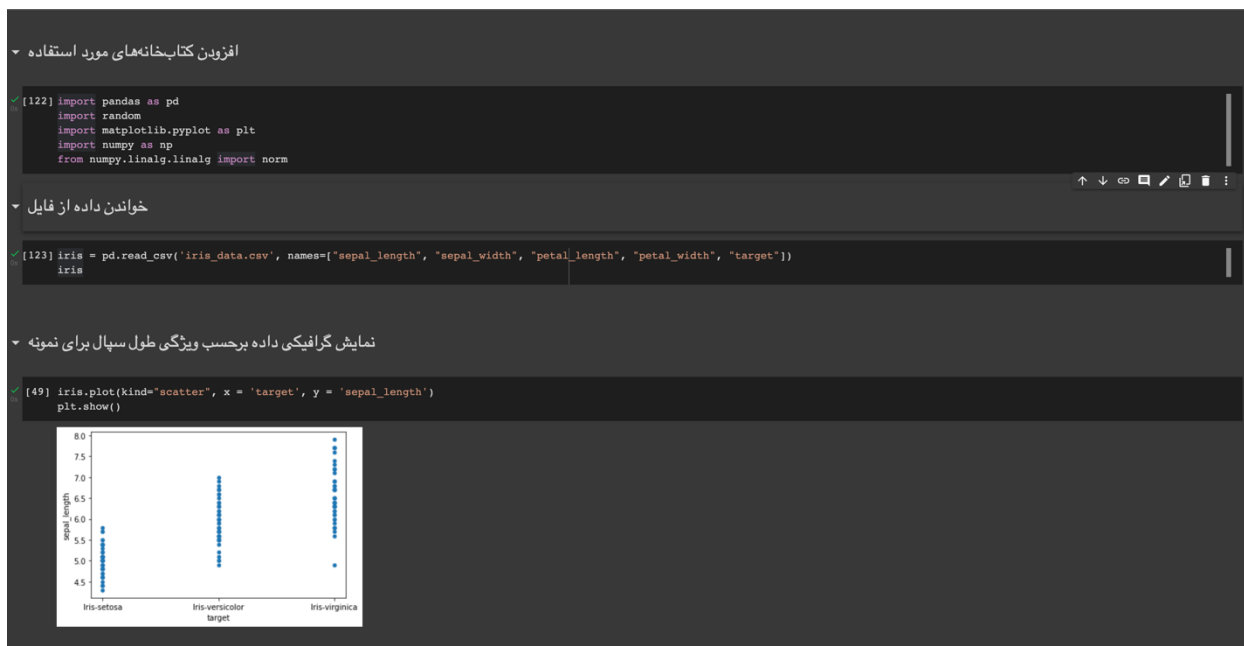
> threshold: 2.7, Upper bound: 15.40166243803931, Lower bound: -6.97309100946788
> Outliers: []

> threshold: 2, Upper bound: 12.501231435584671, Lower bound: -4.072660087013243
> Outliers: [15]
```

سوال ۳)

پروژه همراه با توضیحات در google colab پیاده‌سازی شده‌است که فایلش هم ضمیمه شده است.

عکس از پیاده‌سازی و گزارش پروژه در فضای colab:



```
Columns: [sepal_length, sepal_width, petal_length, petal_width, target]
[57] Index: []
```

```
Number of petal_length NaNs: 0
Empty DataFrame
Columns: [sepal_length, sepal_width, petal_length, petal_width, target]
Index: []
```

```
Number of petal_width NaNs: 0
Empty DataFrame
Columns: [sepal_length, sepal_width, petal_length, petal_width, target]
Index: []
```

```
Number of target NaNs: 0
Empty DataFrame
Columns: [sepal_length, sepal_width, petal_length, petal_width, target]
Index: []
```

همانطور که می‌بینیم، تمامی سطرها مقدار دارند و سطر بدون مقدار نداریم

حذف داده‌های ناقص و بدون مقدار

```
[54] def remove_nans():
      # print(df[df.isna()])
      global iris
      print('\n\nNumber of rows = ', len(iris))
      iris = iris.dropna()
      iris.reset_index(drop=True)
      print('NaNs were removed. Number of rows = ', len(iris))
```

```
remove_nans()
```

```
Number of rows = 150
NaNs were removed. Number of rows = 150
```

همانطور که می‌بینیم، هیچ سطر بی تغییر نکرد زیرا تمامی سطرها مقدار دارند و سطر بدون مقدار نداریم

پیاده‌سازی K-means

ابتدا باید به هر نمونه داده، یک کلاس را به صورت رندم اختصاص بدهیم

```
[63] def apply_clusters(nodes, number_of_clusters):
      nodes['cluster'] = nodes.apply(lambda x: random.randint(0, number_of_clusters-1), axis=1)
```

```
apply_clusters(iris, 3)
iris
```

	sepal_length	sepal_width	petal_length	petal_width	target	cluster
0	5.1	3.5	1.4	0.2	Iris-setosa	0
1	4.9	3.0	1.4	0.2	Iris-setosa	2
2	4.7	3.2	1.3	0.2	Iris-setosa	1
3	4.6	3.1	1.5	0.2	Iris-setosa	2
4	5.0	3.6	1.4	0.2	Iris-setosa	1
...
145	6.7	3.0	5.2	2.3	Iris-virginica	2
146	6.3	2.5	5.0	1.9	Iris-virginica	1
147	6.5	3.0	5.2	2.0	Iris-virginica	2
148	6.2	3.4	5.4	2.3	Iris-virginica	0
149	5.9	3.0	5.1	1.8	Iris-virginica	0

150 rows x 6 columns

با تابع زیر مراکز را آپدیت می‌کنیم. به این صورت که مرکز جدید هر دسته را برابر میانگین مقادیر آن دسته قرار می‌دهیم

```
[138] def calculate_centroids(nodes, number_of_clusters):
      means = {}
```

با تابع زیر مراکز را آپدیت می‌کنیم، به این صورت که مرکز جدید هر دسته را برابر میانگین مقادیر آن دسته قرار می‌دهیم

```
[138] def calculate_centroids(nodes, number_of_clusters):
      means = []
      for i in range(number_of_clusters):
          temp_nodes = nodes.loc[nodes['cluster'] == i]
          means.append(temp_nodes.mean().iloc[0:4])
      return means
```

با این تابع، خوشه مربوط به هر داده را با نزدیکترین مرکز به آن داده بروزرسانی می‌کنیم. برای محاسبه فاصله از تفاوت مقدار فیچرها به صورت نرم شده استفاده کرده‌ایم

```
[124] def update_clusters(nodes, centroids):
      changed = False
      for index, node in nodes.iterrows():
          distances = [norm((node.iloc[0:4] - centroid).to_numpy()) for centroid in centroids]
          new_cluster = distances.index(min(distances))
          if new_cluster != node['cluster']:
              changed = True
              nodes.at[index, 'cluster'] = new_cluster
      return changed
```

در یک حلقه تا جایی که مراکز دیگر آپدیت نشوند، فرایند بروزرسانی مراکز و خوشه‌ی هر داده را تکرار می‌کنیم

```
changed = True
number_of_clusters = 3
apply_clusters(iris, number_of_clusters)
iteration = 0

while changed:
    centroids = calculate_centroids(iris, number_of_clusters)
    changed = update_clusters(iris, centroids)
    iteration += 1

print(f'Centroids: {centroids}')
```

```
[124] nodes.at[index, 'cluster'] = new_cluster
      return changed
```

در یک حلقه تا جایی که مراکز دیگر آپدیت نشوند، فرایند بروزرسانی مراکز و خوشه‌ی هر داده را تکرار می‌کنیم

```
changed = True
number_of_clusters = 3
apply_clusters(iris, number_of_clusters)
iteration = 0

while changed:
    centroids = calculate_centroids(iris, number_of_clusters)
    changed = update_clusters(iris, centroids)
    iteration += 1

print(f'Centroids: {centroids}')
```

```
<ipython-input-138-875ab929879a>:5: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise Typ
means.append(temp_nodes.mean().iloc[0:4])
Centroids: [sepal_length    5.883607
sepal_width      2.746984
petal_length     4.388525
petal_width      1.434426
dtype: float64, sepal_length    6.853846
sepal_width      3.076923
petal_length     5.715385
petal_width      2.053846
dtype: float64, sepal_length    5.006
sepal_width      3.418
petal_length     1.464
petal_width      0.244
dtype: float64]
Number of Iterations: 13
```

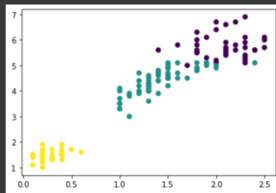
مراکز نهایی چاپ شده‌اند. هم چنین می‌بینی که در ۱۱ پیمایش به شرط توقف می‌روسیم و دیگر مراکز و خوشه‌ی داده‌ها تغییر نمی‌کند.

مصورسازی

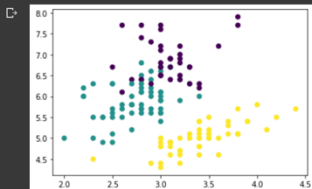
مصوری سازی

نمایش گرافیکی پراکندگی داده‌ها برحسب دو ویژگی که خوشه‌ها با ۳ رنگ نمایش داده شده‌اند. پس داده‌های هم‌رنگ مربوط به یک خوشه هستند

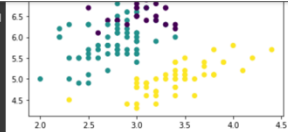
```
[133] plt.scatter(x = iris['petal_width'], y = iris['petal_length'], c=iris['cluster'])  
plt.show()
```



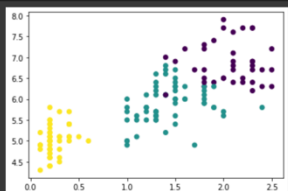
```
1 plt.scatter(x = iris['sepal_width'], y = iris['sepal_length'], c=iris['cluster'])  
plt.show()
```



```
[134] plt.scatter(x = iris['petal_width'], y = iris['sepal_length'], c=iris['cluster'])  
plt.show()
```



```
[135] plt.scatter(x = iris['petal_width'], y = iris['sepal_length'], c=iris['cluster'])  
plt.show()
```



```
1 plt.scatter(x = iris['sepal_width'], y = iris['petal_length'], c=iris['cluster'])  
plt.show()
```

