بخش پیاده‌سازی

ابتدا به کمک تکه کد زیر، داده‌های مورد نظر را خوانده و در متغیری به نام df ذخیره می‌کنیم.

Start) Loading the given data using pandas library.

```python
import pandas as pd

dataset_path = "./iris.data"
df = pd.read_csv(dataset_path, names=['sepal_length','sepal_width','petal_length','petal_width','target'])
```

سوال) ابتدا به دنبال داده های NaN در مجموعه داده بگردید و ذکر کنید که از هر ویژگی چند سطر فاقد داده هستند. برای اینکار از تابع ()isna استفاده کنید.

به کمک ()isna می‌توان مشخص کرد آیا در این مجموعه داده مورد نظر مقدار NaN وجود دارد یا نه. می‌توان به صورت کلی روی df چک کردی یا به صورت دستی و تکی تمام ستون ها را چک کرد.

ستون sepal_length دارای دو داده‌ی دارای NaN است.

```python
sepal_length_NaN = df[df['sepal_length'].isna()]
print(f"The number of missing data in sepal_length is {len(sepal_length_NaN)}.\n")
print(sepal_length_NaN)
```

```
The number of missing data in sepal_length is 2.

     sepal_length  sepal_width  petal_length  petal_width          target
73            NaN          2.2           4.5          1.5  Iris-versicolor
143           NaN          3.0           6.1          2.3   Iris-virginica
```

ستون sepal_width هیچ داده ای که مقدار NaN داشته باشه، ندارد.

```python
sepal_width_NaN = df[df['sepal_width'].isna()]
print(f"The number of missing data in sepal_width is {len(sepal_width_NaN)}.\n")
print(sepal_width_NaN)
```

```
The number of missing data in sepal_width is 0.

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

ستون petal_length دارای دو داده‌ی دارای NaN است.

```
petal_length_NaN = df[df['petal_length'].isna()]
print(f"The number of missing data in petal_length is {len(petal_length_NaN)}.\n")
print(petal_length_NaN)
```

```
The number of missing data in petal_length is 2.

    sepal_length  sepal_width  petal_length  petal_width          target
20           5.4          3.9           NaN          1.7     Iris-setosa
87           5.5          2.4           NaN          NaN  Iris-versicolor
```

ستون petal_width دارای سه داده‌ی دارای NaN است.

```
petal_width_NaN = df[df['petal_width'].isna()]
print(f"The number of missing data in petal_width is {len(petal_width_NaN)}.\n")
print(petal_width_NaN)
```

```
The number of missing data in petal_width is 3.

    sepal_length  sepal_width  petal_length  petal_width          target
6            5.4          3.9           1.7          NaN     Iris-setosa
27           5.0          3.0           1.6          NaN     Iris-setosa
87           5.5          2.4           NaN          NaN  Iris-versicolor
```

ستون target دارای سه داده‌ی دارای NaN است.

```
target_NaN = df[df['target'].isna()]
print(f"The number of missing data in target is {len(target_NaN)}.\n")
print(target_NaN)
```

```
The number of missing data in target is 3.

     sepal_length  sepal_width  petal_length  petal_width target
60            6.3          3.3           4.7          1.6    NaN
139           6.2          3.4           5.4          2.0    NaN
156           6.5          3.0           5.2          2.0    NaN
```

سوال) داده های از دست رفته در مجموعه داده را با استفاده از dropna حذف کنید.

به کمک دستور dropna می‌توان سطرهایی که دارای داده‌ی NaN هستند را حذف کرد. این دستور را روی کل df اعمال می‌کنیم. سپس به دنبال داده‌های NaN در هر سطر گشته و تعداد آن‌ها را پرینت می‌کنیم. تمام ستون ها را چک می‌کنیم.

```
df = df.dropna()
print(f"The number of missing data in sepal_length is {len(df[df['sepal_length'].isna()])}")
print(f"The number of missing data in sepal_width is {len(df[df['sepal_width'].isna()])}")
print(f"The number of missing data in petal_length is {len(df[df['petal_length'].isna()])}")
print(f"The number of missing data in petal_width is {len(df[df['petal_width'].isna()])}")
print(f"The number of missing data in target is {len(df[df['target'].isna()])}")
```

```
The number of missing data in sepal_length is 0
The number of missing data in sepal_width is 0
The number of missing data in petal_length is 0
The number of missing data in petal_width is 0
The number of missing data in target is 0
```

همانطور که از نتیجه مشخص است، تمام داده‌های NaN حذف شده‌اند.

سوال) با استفاده از Label Encoder در ستون target ، Iris-setosa را به ۰ ، Iris-versicolor را به ۱ و Iris-virginica را به ۲ تبدیل کنید. ایرادی که ممکن است این روش داشته باشد را بیان کنید.

یک label encoder به نام le تعریف کردیم و با کمک آن داده‌های ستون target را که یکی از سه مقدار Iris-setosa، Iris-versicolor و یا Iris- virginica هستند را به به ترتیب به مقادیر ۰، ۱ و ۲ به اصطلاح transform می‌کند.

```
from sklearn import preprocessing

le = preprocessing.LabelEncoder()
le.fit(["Iris-setosa", "Iris-versicolor", "Iris-virginica"])

label_encoded_df = df.copy()

label_encoded_df['target'] = le.transform(df['target'].values)
```

برای تست کردن این موضوع که ببینیم آیا داده‌ها تغییر کردن، به طور تصادفی سه سطر را پرینت می‌کنیم:

```
print(f"Before using label encoding: {df['target'][0]}")
print(f"After using label encoding: {label_encoded_df['target'][0]}\n")

print(f"Before using label encoding: {df['target'][71]}")
print(f"After using label encoding: {label_encoded_df['target'][71]}\n")

print(f"Before using label encoding: {df['target'][111]}")
print(f"After using label encoding: {label_encoded_df['target'][111]}\n")
```

```
Before using label encoding: Iris-setosa
After using label encoding: 0

Before using label encoding: Iris-versicolor
After using label encoding: 1

Before using label encoding: Iris-virginica
After using label encoding: 2
```

این روش مشکلی نیز دارد. مشکل این روش در این است که بعد از تبدیل داده‌ها به عدد، این اعداد صرفا نماینده اسم‌های آنها هستند و به معنی بزرگتر و کوچکتر بودن نیست. اما ممکن است این داده ها به اشتباه به عنوان عدد در نظر گرفته شوند و روابط عددی (بزرگتر، کوچکتر) روی آن‌ها صورت گیرد.

سوال) در رابطه با این رو توضیح دهید و یک مثال برای درک بهتر بیان کنید.

OneHotEncoder را می توان به عنوان فرآیند ضروری تبدیل متغیرهای داده طبقه بندی شده برای استفاده در الگوریتم های یادگیری ماشینی و عمیق تعریف کرد. که به نوبه خود پیش بینی ها و همچنین دقت طبقه بندی یک مدل را بهبود می بخشد. OneHotEncoder نمایشی از متغیرهای طبقه بندی شده به عنوان بردارهای باینری است. این ابتدا مستلزم آن است که مقادیر طبقه بندی به مقادیر integer نگاشت شوند. سپس، هر عدد int به عنوان یک بردار باینری نشان داده می شود که همه مقادیر آن صفر است به جز شاخص آن عدد صحیح که با ۱ مشخص شده است.

به عنوان مثال فرض می‌کنیم یک مجموعه از رنگ‌ها داریم که سه ستون قرمز، سبز و آبی دارد. زمانی که از OneHotEncoder استفاده کنیم. در هر ستون به جای همه‌ی رنگ‌ها صفر می‌گذارد و تنها رنگ آن ستون را با عدد یک نمایش می‌دهد.

	Red	Green	Blue
۱	۱	۰	۰
۲	۰	۱	۰
۳	۰	۰	۱

سوال) با استفاده از StandardScaler در sklearn.preprocessing اقدام به نرمالسازی داده ها کنید. مقدار واریانس و میانگین هر ستون را قبل از نرمالسازی و پس از آن ذکر کنید.

می‌خواهیم داده‌ها را نرمال‌سازی کنیم. برای اینکار از کتابخانه‌ی sklearn.preprocessing استفاده می‌کنیم. با تعریف متغیر مورد نظر، آن را روی داده‌های عددی fit می‌کنیم و با دستور transform نرمال‌سازی انجام می‌شود. نباید این نرمال سازی روی ستون آخر (target) انجام شود، برای همین این ستون را در هر مرحله در نظر نمی‌گیریم.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(df.loc[:,df.columns != 'target'])

normalized_df = label_encoded_df.copy()

normalized_df.loc[:, normalized_df.columns != 'target'] = scaler.fit_transform(df.loc[:,df.columns != 'target'])

print(df)
print(normalized_df)
```

نتیجه به صورت زیر است:

داده‌های اولیه

	sepal_length	sepal_width	petal_length	petal_width	target
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
..	...	...	...	...	...
153	6.7	3.0	5.2	2.3	Iris-virginica
154	6.3	2.5	5.0	1.9	Iris-virginica
155	6.5	3.0	5.2	2.0	Iris-virginica
157	6.2	3.4	5.4	2.3	Iris-virginica
158	5.9	3.0	5.1	1.8	Iris-virginica

[150 rows x 5 columns]

داده‌های نرمال‌شده

	sepal_length	sepal_width	petal_length	petal_width	target
0	-0.900681	1.032057	-1.341272	-1.312977	0
1	-1.143017	-0.124958	-1.341272	-1.312977	0
2	-1.385353	0.337848	-1.398138	-1.312977	0
3	-1.506521	0.106445	-1.284407	-1.312977	0
4	-1.021849	1.263460	-1.341272	-1.312977	0
..	...	...	...	...	...
153	1.038005	-0.124958	0.819624	1.447956	2
154	0.553333	-1.281972	0.705893	0.922064	2
155	0.795669	-0.124958	0.819624	1.053537	2
157	0.432165	0.800654	0.933356	1.447956	2
158	0.068662	-0.124958	0.762759	0.790591	2

[150 rows x 5 columns]

حال میانگین و واریانس قدیم و جدید را با هم مقایسه می‌کنیم:

میانگین

```
Average before normalization:
sepal_length    5.843333
sepal_width     3.054000
petal_length    3.758667
petal_width     1.198667
dtype: float64

Average after normalization:
sepal_length   -4.736952e-16
sepal_width    -6.631732e-16
petal_length    3.315866e-16
petal_width    -2.842171e-16
dtype: float64
```

واریانس

```
Variance before normalization:
sepal_length    0.685694
sepal_width     0.188004
petal_length    3.113179
petal_width     0.582414
dtype: float64

Variance after normalization:
sepal_length    1.006711
sepal_width     1.006711
petal_length    1.006711
petal_width     1.006711
dtype: float64
```

سوال) با استفاده از PCA در sklrean.decomposition مولفه های اصلی داده ها را حساب کنید و بردار ویژگی ها از یک فضای ۴ بعدی به ۲ بعدی کاهش دهید.

می‌خواهیم بعدها را کاهش دهیم برای اینکار از کتاب خانه‌ی sklrean.decomposition استفاده می‌کنیم. برای اینکار باید تعداد component ها را مشخص کنیم، یعنی به چه تعداد بعد می‌خواهیم کاهش دهیم؛ که در اینجا دو است. سپس با دستور fit_transform بر روی داده‌ها fit میشود و اینکار صورت می‌گیرد. در این مرحله نیز ستون آخر (target) را جدا می‌کنیم. دو ستون جدیدی که بدست می‌آید را به نام First_Dimention و Second_Dimention نام‌گذاری می‌کنیم. سپس این دو ستون و ستون target را کنار هم قرار داده و نمایش می‌دهیم.

```
from sklearn.decomposition import PCA

reduced_features = pd.DataFrame(data = PCA(n_components=2).fit_transform(normalized_df.iloc[:, :4])  ,
 columns = ['First_Dimention', 'Second_Dimention'])
reduced_dfa = reduced_features.join(normalized_df[['target']], lsuffix="_left", rsuffix="_right")

reduced_dfa
```

	First_Dimention	Second_Dimention	target
0	-2.264542	0.505704	0.0
1	-2.086426	-0.655405	0.0
2	-2.367950	-0.318477	0.0
3	-2.304197	-0.575368	0.0
4	-2.388777	0.674767	0.0
...	...	...	...
145	1.870522	0.382822	2.0
146	1.558492	-0.905314	2.0
147	1.520845	0.266795	2.0
148	1.376391	1.016362	2.0
149	0.959299	-0.022284	2.0

150 rows × 3 columns

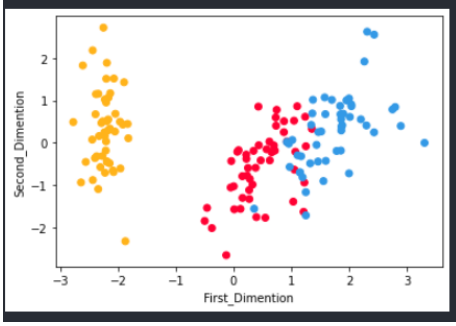
سوال) با استفاده از کتابخانه ی matplotlib داده های مجموعه داده را رسم کنید. دقت کنید برای ویژگی ها از ویژگی های حاصل از PCA استفاده کنید. و برای هر کلاس رنگ متفاوتی استفاده کنید.

داده‌های نرمال شده‌ای که کاهش ابعاد روی آنها صورت گرفته را که در قسمت‌های قبل بدست آوردیم را plot می‌کنیم. برای اینکه نقاط به هم نچسبند و هرکدام را جدا نمایش دهد از نمودار scatter استفاده می‌کنیم. ستون اول و دوم را به آن می‌دهیم و رنگ‌های آن را بر اساس target می‌دهیم. که سه رنگ داریم. رنگ نارنجی نشان دهنده ی ۰.۰، رنگ صورتی نشان دهنده ی ۱.۰ و رنگ آبی نشان دهنده ی ۲.۰ است.

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

color1=(1.0, 0.7, 0.1, 1.0)
color2=(1.0, 0.0, 0.2, 1.0)
color3=(0.2, 0.6, 0.9, 1.0)
colormap = np.array([color1, color2, color3])

plt.scatter(reduced_features.values[:, 0], reduced_features.values[:, 1], c=colormap[normalized_df[['target']]])
plt.xlabel('First_Dimention')
plt.ylabel('Second_Dimention')
plt.show()
```



سوال) برای هر چهار ویژگی ارائه شده در مجموعه داده، نمودار box plot را رسوم کنید.

داده‌های نرمال شده که داده‌های NaN از آنها حذف شده است را که در قسمت‌های قبل بدست آوردیم به صورت نمودار جعبه‌ای نمایش می‌دهیم.

```
box = plt.boxplot(normalized_df.loc[:,df.columns != 'target'], patch_artist=True)

colors = ['green', 'lightGreen', 'lightBlue', 'blue']

for patch, color in zip(box['boxes'], colors):
    patch.set_facecolor(color)

plt.xticks([1, 2, 3, 4], ['sepal_length','sepal_width','petal_length','petal_width'])
plt.show()
```