**به نام خدا**

**محمدعلی کشت پرور ۹۷۳۴۰۲۲** **گزارش تمرین اول داده کاوی**

---

## ۱-اهمیت داده های از دست رفته :

### سوال اول :

```python
# 1-1) counts the number of NAN rows for each feature
def count_nan_rows():
    print(f'sepal_length :  {df["sepal_length"].isna().sum()}')
    print(f'sepal_width  :  {df["sepal_width"].isna().sum()}')
    print(f'petal_length :  {df["petal_length"].isna().sum()}')
    print(f'petal_width  :  {df["petal_width"].isna().sum()}')
    print(f'target       :  {df["target"].isna().sum()}\n')
```

```
Run: main
sepal_length :  2
sepal_width  :  0
petal_length :  2
petal_width  :  3
target       :  3
```

شمارش داده های NAN

### سوال دوم:

```python
# 1-2) removes each row that contains the NAN feature
def drop_nan_value(data_frame):
    return data_frame.dropna()
```

```
Run: main
sepal_length :  2
sepal_width  :  0
petal_length :  2
petal_width  :  3
target       :  3

sepal_length :  0
sepal_width  :  0
petal_length :  0
petal_width  :  0
target       :  0
```

حذف داده های NAN

۲-داده های غیرعددی

سوال اول:

```
# 2-1) label encoder for target feature
df['target'] = LabelEncoder().fit_transform(df.target.values)
```

استفاده از labelEncoder

| | sepal_length | sepal_width | petal_length | petal_width | target |
|---|---|---|---|---|---|
| 45 | 4.40000 | 3.20000 | 1.30000 | 0.20000 | 0 |
| 46 | 5.00000 | 3.50000 | 1.60000 | 0.60000 | 0 |
| 47 | 5.10000 | 3.80000 | 1.90000 | 0.40000 | 0 |
| 48 | 4.80000 | 3.00000 | 1.40000 | 0.30000 | 0 |
| 49 | 5.10000 | 3.80000 | 1.60000 | 0.20000 | 0 |
| 50 | 4.60000 | 3.20000 | 1.40000 | 0.20000 | 0 |
| 51 | 5.30000 | 3.70000 | 1.50000 | 0.20000 | 0 |
| 52 | 5.00000 | 3.30000 | 1.40000 | 0.20000 | 0 |
| 53 | 7.00000 | 3.20000 | 4.70000 | 1.40000 | 1 |
| 54 | 6.40000 | 3.20000 | 4.50000 | 1.50000 | 1 |
| 55 | 6.90000 | 3.10000 | 4.90000 | 1.50000 | 1 |
| 56 | 5.50000 | 2.30000 | 4.00000 | 1.30000 | 1 |
| 57 | 6.50000 | 2.80000 | 4.60000 | 1.50000 | 1 |
| 58 | 5.70000 | 2.80000 | 4.50000 | 1.30000 | 1 |
| 59 | 6.30000 | 3.30000 | 4.70000 | 1.60000 | 1 |

دیتافریم بعد از استفاده از labelEncoder

این یک جنبه بسیار مهم است که باید در نظر گرفته شود، که ارزشی را که یک ویژگی به یک مدل اضافه می کند، بررسی شود . ویژگی‌های ترتیبی ، که نوع بسیار رایجی از داده‌های طبقه‌بندی هستند که در آن دسته‌های مختلف آن نظم طبیعی را ارائه می‌دهند. یک مثال می تواند ویژگی دما باشد که دسته بندی های سرد، ملایم و گرم را در بر می گیرد. یک اشتباه نسبتاً رایج این است که فقط یک مقدار عددی منحصر به فرد را به هر دسته از یک ویژگی اختصاص می دهیم، بدون توجه به ترتیبی که ممکن است داشته باشد. و به طور مشابه با ویژگی های اسمی ممکن در این روش به صورت ترتیبی کدگذاری شوند و تبدیل به یک ویژگی ترتیبی شود در حالی که اصلا ترتیبی وجود ندارد. این مشکل در OneHotEncoder برطرف شده است .

**سوال دوم:**

در روش OneHotEncoder برای هر طبقه یک ستون جداگانه درنظر گرفته میشود و برای آن عدد ۰ یا ۱ قرار داده می شود که این روش مشکل ذکر شده در روش Encoding Label را برطرف می کند به جای یک ستون به تعداد ویژگی ها ستون نگه می داریم . به طور مثال :

| سبز | آبی | قرمز |
|---|---|---|
| ۰ | ۰ | ۱ |
| ۱ | ۰ | ۰ |

## ۳-نرمال سازی:

```
# 3-1) normalize features
df_numpy = StandardScaler().fit_transform(df[features])
df_numpy = np.append(df_numpy, df[['target']].to_numpy(), axis=1)
df = pd.DataFrame(df_numpy, columns=features + ['target'])
```

Run: main

| | sepal_length | sepal_width | petal_length | petal_width |
|---|---|---|---|---|
| count | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| mean | 5.843333 | 3.054000 | 3.758667 | 1.198667 |
| std | 0.828066 | 0.433594 | 1.764420 | 0.763161 |
| min | 4.300000 | 2.000000 | 1.000000 | 0.100000 |
| 25% | 5.100000 | 2.800000 | 1.600000 | 0.300000 |
| 50% | 5.800000 | 3.000000 | 4.350000 | 1.300000 |
| 75% | 6.400000 | 3.300000 | 5.100000 | 1.800000 |
| max | 7.900000 | 4.400000 | 6.900000 | 2.500000 |
| var | 0.685694 | 0.188004 | 3.113179 | 0.582414 |

| | sepal_length | sepal_width | petal_length | petal_width |
|---|---|---|---|---|
| count | 1.500000e+02 | 1.500000e+02 | 1.500000e+02 | 1.500000e+02 |
| mean | -4.736952e-16 | -6.631732e-16 | 3.315866e-16 | -2.842171e-16 |
| std | 1.003350e+00 | 1.003350e+00 | 1.003350e+00 | 1.003350e+00 |
| min | -1.870024e+00 | -2.438987e+00 | -1.568735e+00 | -1.444450e+00 |
| 25% | -9.006812e-01 | -5.877635e-01 | -1.227541e+00 | -1.181504e+00 |
| 50% | -5.250608e-02 | -1.249576e-01 | 3.362659e-01 | 1.332259e-01 |
| 75% | 6.745011e-01 | 5.692513e-01 | 7.627586e-01 | 7.905908e-01 |
| max | 2.492019e+00 | 3.114684e+00 | 1.786341e+00 | 1.710902e+00 |
| var | 1.006711 | 1.006711 | 1.006711 | 1.006711 |

اطلاعات آماری قبل و بعد از نرمال سازی

## ۴-تحلیل مولفه های اصلی:

```
# 4) PCA
x = df.loc[:, features].values
pca = PCA(n_components=2)
principal_components = pca.fit_transform(x)
principal_df = pd.DataFrame(data=principal_components, columns=['PC1', 'PC2'])
result_df = pd.concat([principal_df, df[['target']]], axis=1)
# show result
```

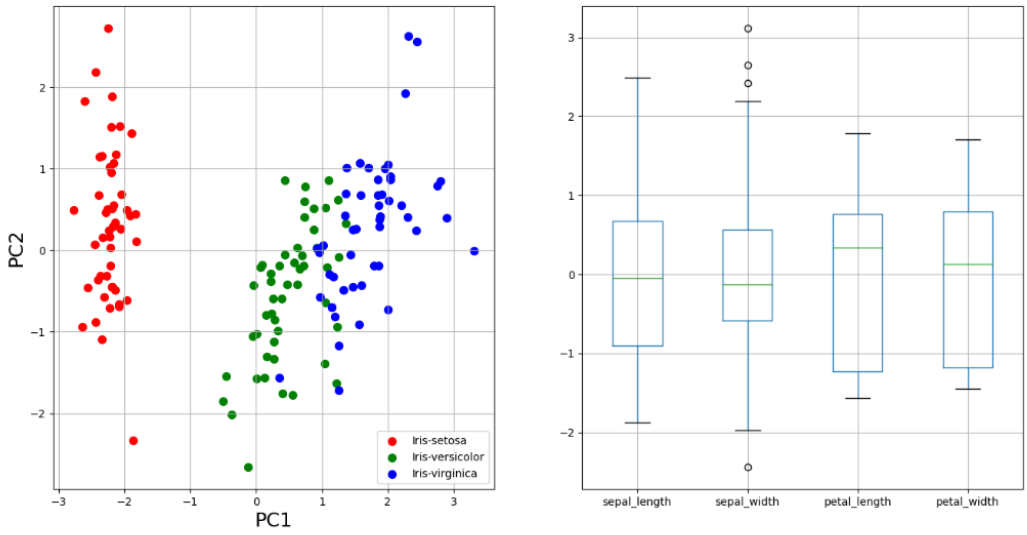
| | PC1 | PC2 | target |
|---|---|---|---|
| 0 | -2.26454 | 0.50570 | 0.00000 |
| 1 | -2.08643 | -0.65540 | 0.00000 |
| 2 | -2.36795 | -0.31848 | 0.00000 |
| 3 | -2.30420 | -0.57537 | 0.00000 |
| 4 | -2.38878 | 0.67477 | 0.00000 |
| 5 | -2.07054 | 1.51855 | 0.00000 |

دیتافریم پس از کاهش بعد

## ۵-مصورسازی:

```
# 5) visualization
f, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 8))
fig = plt.figure(figsize=(7, 7))
ax1.set_xlabel('PC1', fontsize=18)
ax1.set_ylabel('PC2', fontsize=18)
targets = [0, 1, 2]
colors = ['r', 'g', 'b']
for target, color in zip(targets, colors):
    is_equal = result_df['target'] == target
    ax1.scatter(result_df.loc[is_equal, 'PC1'],
                result_df.loc[is_equal, 'PC2'], c=color, s=50)
ax1.legend(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'])
ax1.grid()
boxplot = df.boxplot(column=features, ax=ax2)
plt.show()
```

---




---

نمودار ویژگی ها پس از کاهش بعد و نمودار جعبه ایی پس از حذف داده های NAN