به نام خدا

گزارش HW۱ داده کاوی

علیرضا شایان

شماره دانشجویی: ٤٠٣٣١٥٦١

گرایش: علوم داده

فهرست

فایل ها ٤

run.py ٤

کتابخانه های مورد استفاده: ٤

توضیحات: ٤

USS.py ٤

کتابخانه های مورد استفاده: ٤

توضیحات: ٥

CA.py ٧

کتابخانه های مورد استفاده: ٧

توضیحات: ٧

Visualization.py ٨

کتابخانه های مورد استفاده: ٨

توضیحات: ٨

Distribution of Labels ٨

خروجی Distribution of Labels: ٩

Histograms ٩

خروجی Histograms: ١٠

٣D Histogram ١٠

خروجی ٣D Histogram: ١١

Box Plots ١٢

خروجی Box Plots: ١٢

٢D boxplot ١٣

خروجی ٢D Boxplot: ١٥

Quantile Plots ۱۵
خروجی Quantile plot: ۱۶
Scatter Plots ۱۶
خروجی Scatterplot: ۱۷
۳D scatter plot ۱۷
خروجی ۳D Scatter Plot: ۱۸
Probability Distributions ۱۸
خروجی Probability Distributions: ۱۹
تفاسیر ۲۰
Correlation Analysis ۲۰
بررسی ماتریس Correlation ۲۰
تفسیر ضرایب همبستگی ۲۰
تفسیر هر جفت ویژگی ۲۰
Box plot and ۲D Box plot ۲۱
Quantile plot ۲۲
تفسیر نمودار sepal width: ۲۲
تفسیر نمودار petal length: ۲۳
Probability Distributions ۲٤
تفسیر نمودار توزیع احتمالاتی هر نوع (species) بر اساس petal length: ۲٤

فایل ها

۱. Run.py
۲. USS.py
۳. CA.py
٤. Visualization.py

run.py

کتابخانه های مورد استفاده:

۱. subprocess

توضیحات:

```python
files = ["CA.py", "USS.py", "Visualization.py"]
# Run each file
for script in files:
    subprocess.run(["python", script])
```

با اجرا گرفتن از این فایل بقیه فایل ها نیز به ترتیب اجرا گرفته میشوند و نیاز به اجرا گرفتن جداگانه نیست.

ابتدا نام فایل هایی که قصد اجرا داریم به یک لیست بنام files داده شده سپس در یک لوپ با تابع run از کتابخانه subprocess فایل ها اجرا میشوند.

USS.py

نام این فایل مخفف Univariate Summary Statistics است. دستورات مربوط به بخش اول هوم ورک را اجرا می کند.

کتابخانه های مورد استفاده:

۱. pandas as pd
۲. sklearn.datasets import load_iris
۳. numpy as np
٤. os

توضیحات:

پس از افزودن کتابخانه ها ، توسط کد زیر دیتاست را لود میکنیم و دیتا فریمی از داده ها به همراه تارگت یا لیبل با دستور np.c_ می سازیم.

```
iris = load_iris()
df = pd.DataFrame(data=np.c_[iris['data'], iris['target']],
                  columns=iris['feature_names'] + ['target'])
```

بعد از آن با دستورات زیر، نام تارگت هارا از پیش فرض اعداد ۰ ، ۱ و ۲ به نام واقعی گونه ها تغییر می‌دهیم.

```
df['target'] = df['target'].replace({0: 'Setosa', 1: 'Versicolor', 2: 'Virginica'})
```

سپس با تابع groupby ، دیتافریم ساخته شده از تارگت هارا دسته بندی می‌کنیم، این امر به محاسبات آماری ما روی داده های گروه بندی شده کمک می‌کند.

```
grouped_df = df.groupby('target')
```

بعد از آن لیستی بنام statistics برای ذخیره مقادیر محاسبه شده می‌سازیم و سپس در دیتافریم گروه بندی شده لوپ میزنیم و برای هر گروه مقادیر آماری گفته شده را محاسبه میکنیم.

```
for name, group in grouped_df:
```

Sepal width را از گروه مورد نظر برداشت میکنیم و سپس missing values را از این گروه ذخیره می کنیم.

```
sepal_width = group['sepal width (cm)'].dropna()
missing_values = group['sepal width (cm)'].isnull().sum()
```

بعد از آن به سراغ محاسبات آماری میرویم که از توابع پیشفرض کتابخانه های numpy و pandas استفاده کرده ایم:

```
sepal_width = group['sepal width (cm)'].dropna()
missing_values = group['sepal width (cm)'].isnull().sum()
min_value = sepal_width.min()
q1 = sepal_width.quantile(0.25)
median = sepal_width.median()
q3 = sepal_width.quantile(0.75)
p95 = sepal_width.quantile(0.95)
max_value = sepal_width.max()
mean = sepal_width.mean()
range_value = max_value - min_value
iqr = q3 - q1
std = sepal_width.std()
std_pop = sepal_width.std(ddof=0)
```

در ادامه برای محاسبه mad بصورت زیر عمل کرده ایم:

```
mad = np.median(np.abs(sepal_width - median))
```

در نهایت داده های محاسبه شده را در لیست از پیش ساخته statistics اضافه می‌کنیم:

```
statistics.append([name, missing_values, min_value, q1, median, q3, p95, max_value, mean, range_value, iqr, std, std_pop, mad])
```

برای نام گذاری هدر در فایل csv ای که قرار است ذخیره کنیم بصورت زیر لیست محاسبه شده به همراه نام هدر ها به ترتیب با استفاده از تابع Dataframe به دیتا فریمی بنام statistics_df تبدیل میکنیم. تبدیل این داده های به دیتافریم به ما کمک می‌کند به نحو بهتری آنها را به csv تبدیل کنیم.

```
statistics_df = pd.DataFrame(statistics, columns=['label', 'missing', 'min', 'q1',
                                                  'med', 'q3', 'p95', 'max', 'mean',
                                                  'range', 'iqr', 'std', 'std_pop', 'mad'])
```

برای آدرس دهی Relative جهت ذخیره سازی روی هر نوع حافظه ای به صورت زیر عمل میکنیم و با استفاده از کتابخانه os و دستور path و نام فولدر های تودر تو(اگر قرار باشد به داخلی ترین ها آدرس دهیم) و نام فایل اصلی جهت ذخیره، مسیر یا path نهایی را میسازیم:

```
output_path = os.path.join("..", "dist", "statistics.csv")
```

و در آخر با دستور to_csv و با دادن آدرسی که با دستور قبلی ساختیم، فایل یا ذخیره میکنیم:

```
statistics_df.to_csv(output_path, header=True, index=False)
```

CA.py

نام این فایل مخفف Correlation Analysis است. دستورات مربوط به بخش دوم هوم ورک را اجرا می کند.

کتابخانه های مورد استفاده:

۱. pandas as pd
۲. sklearn.datasets import load_iris
۳. numpy as np
۴. os

توضیحات:

مراحل لود دیتاست ، ساخت دیتافریم و بازنویسی نام تارگت های مثل بخش قبل انجام میشود. سپس با استفاده از کد زیر، از دیتاست فقط داده های عددی را به دیتافریم numerical_df تخصیص میدهیم:

```
numerical_df = df.drop( labels: 'target', axis=1)
```

بعد از آن با استفاده از تابع corr بر روی این دیتافریم ، ماتریس همبستگی یا correlation_matrix را ایجاد میکنیم:

```
correlation_matrix = numerical_df.corr()
```

سپس مثل مرحله قبل برای ذخیره فایل csv به همانگونه عمل میکنیم.

بعد از آن برای محاسبه حداقل و حداکثر همبستگی مطلق ، ماتریس همبستگی را باید به جفت عناصر مرتب کنیم که با دستور correlation_matrix.stack() صورت می گیرد، برای محاسبه حداقل یا مینیمم همبستگی باید به دستور گفته شده nsmallest(۱). را اضافه کرد که کمترین مقدار را برمیگرداند، در ادامه این دستور برای دریافت نام تارگت یا feature ، index[۰]. را به دستور اصافه میکنیم. برای محاسبه حداکثر همبستگی نیز به همین صورت عمل میکنیم:

```
min_corr_pair = correlation_matrix.stack().nsmallest(1).index[0]
max_corr_pair = correlation_matrix.stack().nlargest(1).index[0]
```

در نهایت خروجی هارا در کنسول چاپ میکنیم که دستورات به شکل زیر:

```
print("Pair of features with minimum absolute correlation:", min_corr_pair)
print("Pair of features with maximum absolute correlation:", max_corr_pair)
```

و خروجی به شکل زیر است:

```
Pair of features with minimum absolute correlation: ('sepal width (cm)', 'petal length (cm)')
Pair of features with maximum absolute correlation: ('sepal length (cm)', 'sepal length (cm)')
```

Visualization.py

این فایل مربوط به بخش سوم هوم ورک و پیاده سازی نمودار ها است.

کتابخانه های مورد استفاده:

۱. matplotlib.pyplot *as* plt
۲. pandas *as* pd
۳. numpy *as* np
٤. seaborn *as* sns
٥. sklearn.datasets *import* load_iris
٦. scipy.stats *import* gaussian_kde
۷. matplotlib.patches *import* Rectangle

توضیحات:

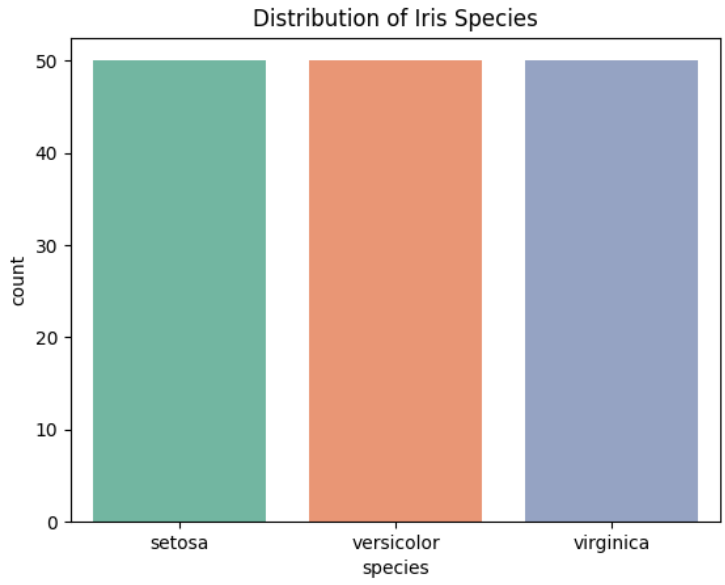
Distribution of Labels

در این بخش برای نشان داده توزیع لیبل ها از نمودار میله یا countplot ا زکتابخانه seaborn استفاده میکنیم:

```
sns.countplot(x='species', data=df, palette='Set2', hue='species', legend=False)
```

برا نام گذاری هر نمودار یا پلات از دستور plt.title() استفاده میکنیم که درون پرانتر به همراه دابل کوتیشن نام نمودار را مینویسیم. همچنین در آخر برای نشان دادن نمودار از دستور plt.show() استفاده میکنیم. در ادامه نیز با همین روش بقیه نمودار هارا نشان میدهیم و از گفتن از توضیحات پرهیز می کنیم.

خروجی *Distribution of Labels*:



با توجه به نمودار بالا میتوان به راحتی فهمید که هر نوع یا species به تعداد مساوی در دیتاست رکورد شده اند.

Histograms

در این بخش برای رسم هیستوگرام ها تابعی تعریف کردیم که دو ورودی df که دیتافریم مورد نظر است و features که لیستی از فیچر هایی است که میخواهیم رسم کینم.

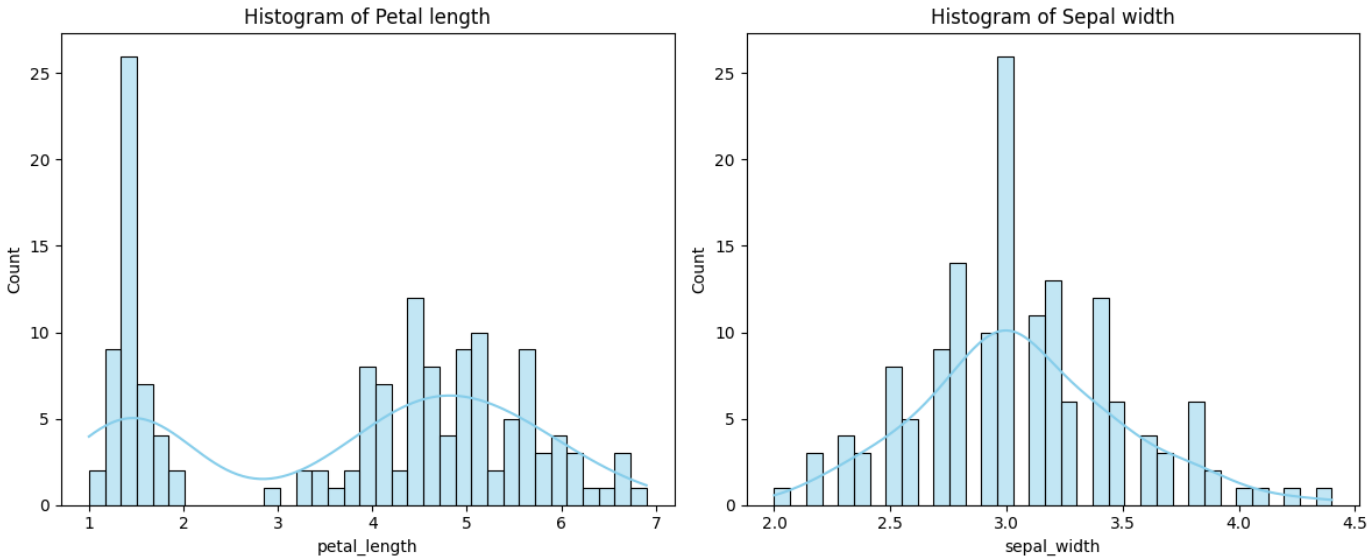
در این تابع ابتدا سایز شکل مشخص شده و سپس در یک لوپ که درون فیچر ها پیمایش می کند ، ابتدا ساب پلات یا نمودار اول از شکل مان را تعیین میکنیم و بعد از آن با دستور histplot از کتابخانه seaborn نمودار اول را چاپ میکنیم. دستور histplot ویژگی های قابل تنظیمی دارد که ما از bins برای دقیق تر نشان دادن تعداد میله ها و از kde برای نشان داده curve توزیع استفاده کرده ایم:

```
def plot_histograms(df, features):  1 usage
    plt.figure(figsize=(12, 5))
    for i, feature in enumerate(features, start=1):
        plt.subplot( *args: 1, 2, i)
        sns.histplot(df[feature], bins=35, kde=True, color='skyblue')
        plt.title(f'Histogram of {feature.replace("_", " ").capitalize()}')
    plt.tight_layout()
    plt.show()
```

در ادامه از این تابه به صورت زیر استفاده کرده ایم و با دادن دیتافریم تعریف شده از پیش df و لیستی از فیچر هایی که قصد رسم کردنشان را داریم، پلات را رسم میکنیم:

```
plot_histograms(df, features: ['petal_length', 'sepal_width'])
```

خروجی Histograms:



هیستوگرام در petal length بیان از این دارد که ما در این ویژگی دو گروه در دو بازه متفاوت داریم.

۳D Histogram

برای رسم این هیستوگرام ابتداعاً با دستورات زیر شکل را رسم می کنیم و یک فضای نمودار سه بعدی در آن قرار میدهیم:

```
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
```

در ادامه یک هیستوگرام دو بعدی با محور های sepal width و petal length ایجاد میکنیم و یک آرایه دو بعدی (hist) را برای ذخیره سازی این مقادیر تعریف می کنیم. همچنین مقادیر petal length و sepal width را به محور های x و y تخصیص می دهیم:

```
hist, xedges, yedges = np.histogram2d(df['petal_length'], df['sepal_length'], bins=25)
```

سپس در دو خط زیر با دستور meshgrid از کتابخانه numpy یک شبکه ایجاد میکنیم برای میله های هیستوگرام سه بعدی:

در ادامه ابعاد هر میله را مشخص میکنیم و با تابع ravel ارتفاع هر نقطع را به او اختصاص میدهیم.

```
xpos, ypos = np.meshgrid( *xi: xedges[:-1] + 0.25, yedges[:-1] + 0.25, indexing="ij")
xpos, ypos, zpos = xpos.ravel(), ypos.ravel(), 0
```

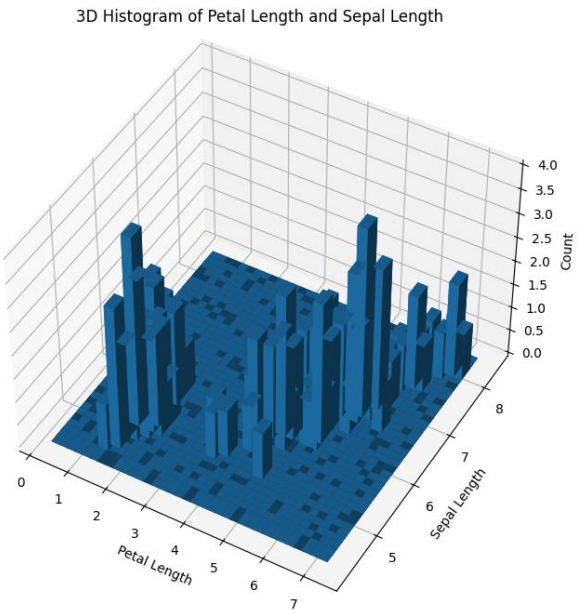
سپس با دستور زیر میله های سه بعدی را پلات میکنیم:

```
ax.bar3d(xpos, ypos, zpos, dx, dy, dz, zsort='average')
```

با دستور زیر نیز زاویه دید را نسبت به نمودار ست میکنیم:

```
ax.view_init(elev=45)
```

خروجی ۳D Histogram:



با توجه به خروجی ۳ بعدی هیستوگرام میتوان پی برد که داده ها در دو محدوده تجمیع دارند که حاصل از دو گروه شدن petal length است ، اما از دید sepal width اورلپ داریم.

Box Plots

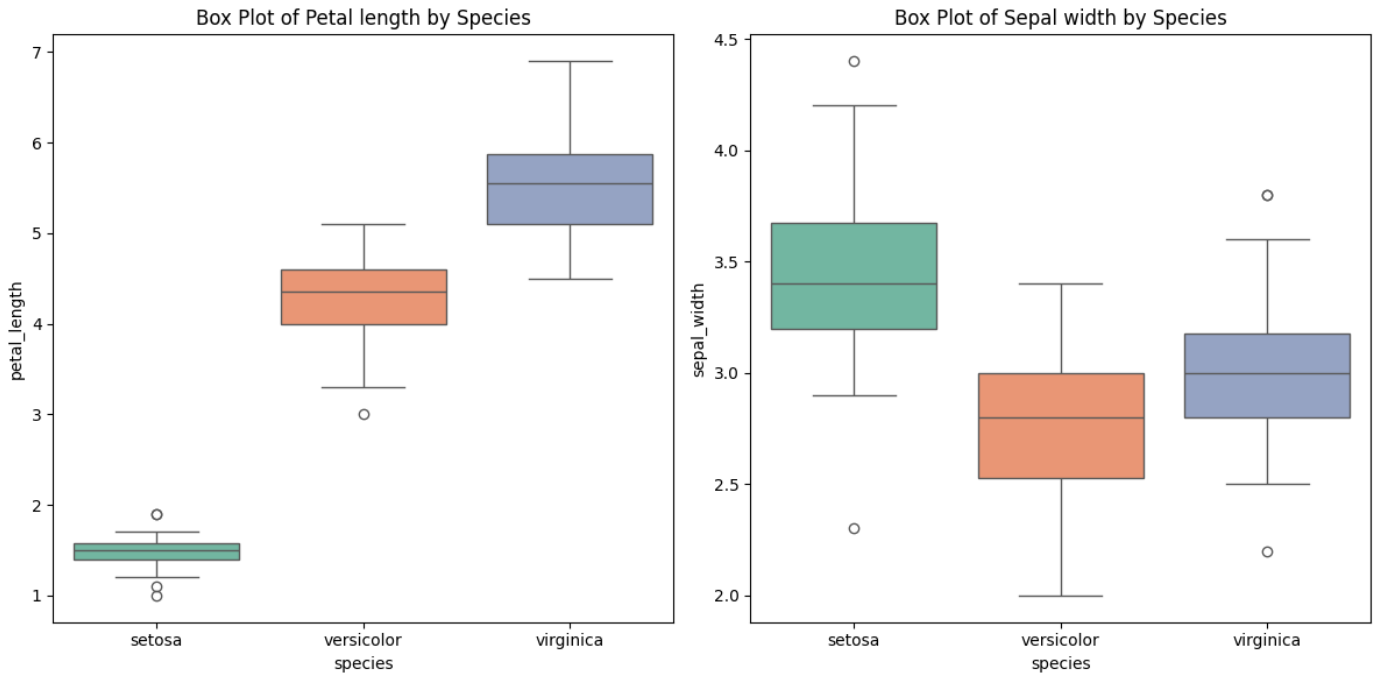
در این بخش برای رسم باکس پلات ها، تابعی تعریف کردیم که ورودی های آن دیتافریم و ویژگی هایی از دیتاست است که قصد رسم باکس پلات آنها را داریم. در این تابع یک لوپ داریم که درون ویژگی ها پیمایش میکند و به ترتیب باکس پلات آنرا با دستور boxplot از کتابخانه seaborn رسم می کنیم:

```
def plot_boxplots(df, features):  1 usage
    plt.figure(figsize=(12, 6))
    for i, feature in enumerate(features, start=1):
        plt.subplot( *args: 1, 2, i)
        sns.boxplot(x='species', y=feature, data=df, palette='Set2', hue='species', legend=False)
        plt.title(f'Box Plot of {feature.replace("_", " ").capitalize()} by Species')
```

سپس با دستور زیر تابع را فراخوانی کرده و باکس پلات هارا رسم می کنیم:

```
plot_boxplots(df, features: ['petal_length', 'sepal_width'])
```

خروجی Box Plots:



با دقت به ویژگی petal length میبینیم که نوع setosa دارای نقاط داده پرت بیشتری است و همچنین بازه بسیار محدود تری نسبت به دو نوع دیگر دارد. از طرفی با توجه به ویژگی sepal width متوجه پراکندگی تقریبا یکسان تمام گونه ها میشویم.

۲D boxplot

در این بخش تابعی تعریف کرده ایم که دیتافریم و ویژگی های مربوط به محور xها و yها به عنوان ورودی های آن هستند.

درون این تابع دو تابع دیگر تعریف شده که تابع اول وظیفه دارد نقاط داده پرت یا outlier هارا پیدا کند. به این صورت که با دادن ویژگی مورد نظر، چارک اول و سوم را پیدا می کنیم سپس شاخص IQR را محاسبه می کنیم و در نهایت با مقایسه داده ها با ۱.۵ × IQR (در حد بالا این مقدار بعلاوه چارک سوم می شود و در حد پایین این مقدار را از چارک اول کم می کنیم):

```
def find_outliers(series):
    q1, q3 = series.quantile([0.25, 0.75])
    iqr = q3 - q1
    return series[(series < (q1 - 1.5 * iqr)) | (series > (q3 + 1.5 * iqr))]
```

تابع دوم برای پیدا کردن مینیمم و ماکسیمم باکس پلات تعریف شده و دو ورودی ویژگی و متغیر بولین مربوط به مینیمم دارد. متغیر بولین اینگونه اختصاص داده می شود که اگر میخواهیم بر روی ویژگی مینیمم بگیریم باید این مقدار را True کنیم:

```
def calc_min_max(series, is_min=True):
    q1, q3 = series.quantile([0.25, 0.75])
    iqr = q3 - q1
    return max(q3 - 1.5 * iqr, min(series)) if is_min else min(q3 + 1.5 * iqr, max(series))
```

در خط زیر با استفاده از تابع describe مقدار شاخص های آماری را به ستون مقادیر هر ویژگی تخصیص میدهیم.

```
x_stats, y_stats = df[x_feature].describe(), df[y_feature].describe()
```

دستور describe می تواند مقادیر زیر را محاسبه و ذخیره کند:

- count - The number of not-empty values.
- mean - The average (mean) value.
- std - The standard deviation.
- min - the minimum value.
- percentile*.۲۵٪ - The ۲۵٪
- percentile*.۵۰٪ - The ۵۰٪
- percentile*.۷۵٪ - The ۷۵٪
- max - the maximum value.

سپس به شکل زیر نقاط داده پرت را با فراخوانی تابعی که قبلا تعریف کردیم بدست می آوریم:

```
outliers = df[(df[x_feature].isin(find_outliers(df[x_feature]))) |
              (df[y_feature].isin(find_outliers(df[y_feature])))]
```

در ادامه به دلیل اینکه تابع describe مینیمم وماکسیمم را طبق قوانین Boxplot محاسبه نمی کند، به صورت زیر مینیمم و ماکسیمم هر محور را با فراخوانی تابعی که قبلا تعریف کردیم بدست می آوریم و دوباره در مقادیر مینیمم و ماکسیمم میریزیم:

```
x_stats['min'], x_stats['max'] = calc_min_max(df[x_feature], is_min: True), calc_min_max(df[x_feature], is_min: False)
y_stats['min'], y_stats['max'] = calc_min_max(df[y_feature], is_min: True), calc_min_max(df[y_feature], is_min: False)
```

سپس باید بخش های باکس پلات ۲ بعدی را مرحله به مرحله رسم کنیم. در مرحله اول با استفاده از کد زیر بدنه باکس پلات را رسم میکنیم. تابع add_patch(Rectangle(یک مستطیل را میسازد که در ادامه با دادن مقادیر چارک های هر محور، اضلاع آن مشخص میشوند. (x_stats['۲۵٪'], y_stats['۲۵٪']) که پس از دستور خط بالا می آید، گوشه پایین سمت چپ این مستطیل را مشخص می کند و x_stats['۷۵٪'] - x_stats['۲۵٪'] در ادامه مشخص کننده بازه عرض مستطیل است و در آخر y_stats['۷۵٪'] - y_stats['۲۵٪'] نشان دهنده میزان ارتفاع آن است:

```
plt.gca().add_patch(Rectangle( xy: (x_stats['25%'], y_stats['25%']),
                               x_stats['75%'] - x_stats['25%'],
                               y_stats['75%'] - y_stats['25%'],
                               fill=True, color=color, alpha=0.5))
```

در مرحله بعدی از رسم با استفاده از دستورات زیر whisker ها و خطوط انتهایی عمود بر whisker ها که نشان دهنده مینیمم و ماکسیمم هستند را پلات می کنیم:

```
plt.plot( *args: [x_stats['min'], x_stats['max']], [y_stats['50%'], y_stats['50%']], color=color)
plt.plot( *args: [x_stats['50%'], x_stats['50%']], [y_stats['min'], y_stats['max']], color=color)
plt.plot( *args: [x_stats['min'], x_stats['min']], [y_stats['25%'], y_stats['75%']], color=color)
plt.plot( *args: [x_stats['max'], x_stats['max']], [y_stats['25%'], y_stats['75%']], color=color)
plt.plot( *args: [x_stats['25%'], x_stats['75%']], [y_stats['min'], y_stats['min']], color=color)
plt.plot( *args: [x_stats['25%'], x_stats['75%']], [y_stats['max'], y_stats['max']], color=color)
```

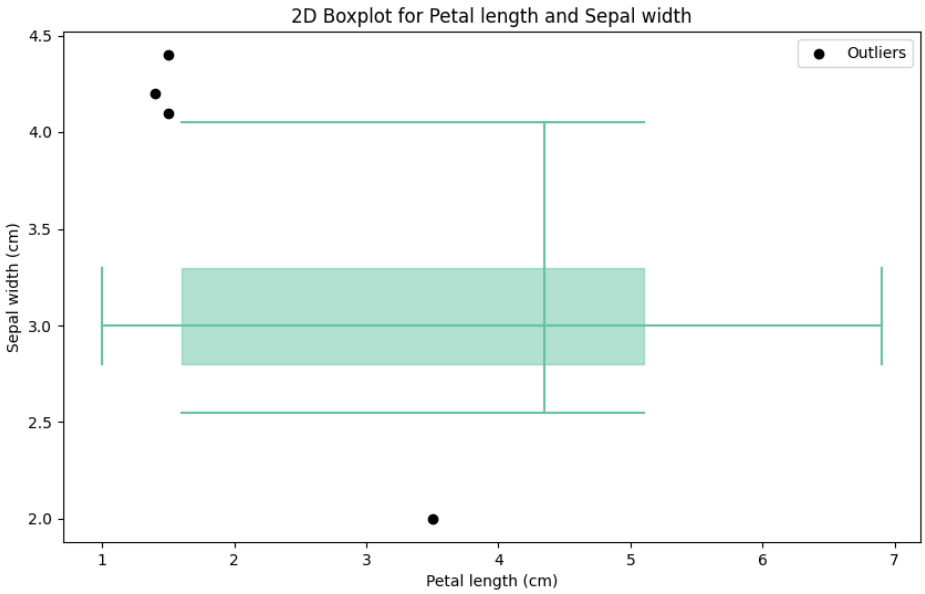
سپس با تابع scatter نقاط داده پرت را رسم میکنیم:

```
plt.scatter(outliers[x_feature], outliers[y_feature], color='black', marker='o', label='Outliers')
```

در انتهای تابع ساخته شده برای رسم این باکس پلات دو بعدی را بصورت زیر فراخوانی می کنیم:

```
plot_2d_box_with_outliers(df, x_feature: 'petal_length', y_feature: 'sepal_width')
```

خروجی ۲D Boxplot:



Quantile Plots

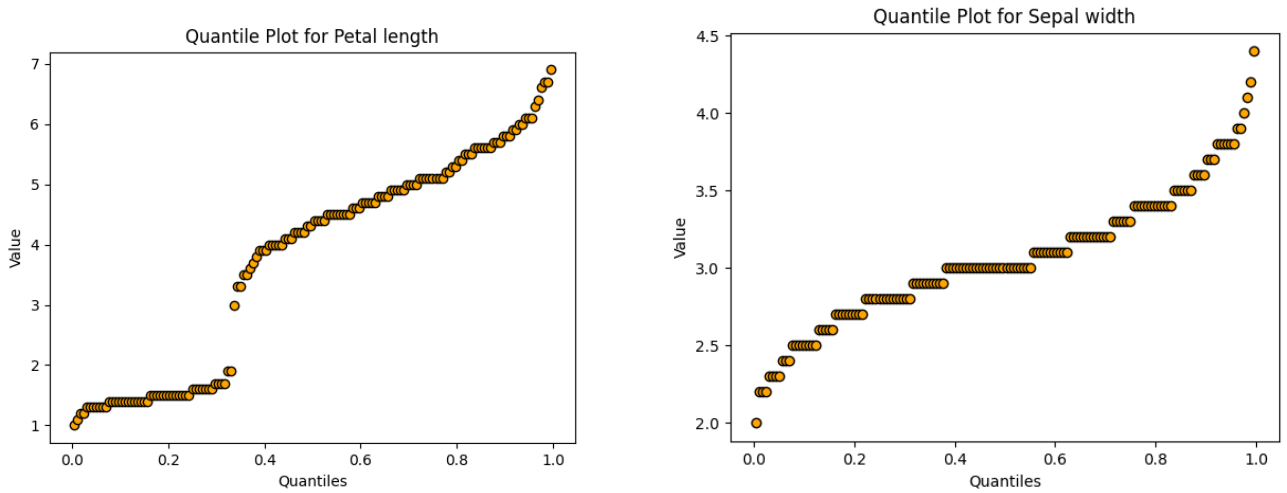
در این قسمت تابعی تعریف کردیم که با استفاده از آن q plot را رسم کنیم. این تابع ورودی ای از جنس دیتافریم میگیرد و در ادامه داده هارا مرتب می کند و نقاط را تابعی از توزیع نرمال رسم میکند:

```
def quantile_plot(series):  2 usages
    sorted_data = series.sort_values().reset_index(drop=True)
    N = len(sorted_data)
    quantiles = [(i - 0.5) / N for i in range(1, N + 1)]
    plt.scatter(quantiles, sorted_data, color='orange', edgecolor='black')
```

در نهایت تابع بالا به صورت زیر فراخوانی شده است:

```
quantile_plot(df['sepal_width'])
quantile_plot(df['petal_length'])
```

خروجی Quantile plot:



تفسیر این خروجی ها در بخش آخر گزارش آمده است.

Scatter Plots

برای رسم اسکتر پلات نیز طبق روال های قبلی تابعی تعریف کرده ایم که ورودی اش دیتافریم از پیش ساخته شده و فیچر هایی است که قصد پلات کردن آنها را داریم. در خط زیر لیستی از تاپل هایی که حاوی جفت مقادیر متمایزی از ویژگی مورد بررسی هستند را ذخیره میکنیم:

```
pair_indices = [(i, j) for i in range(len(features)) for j in range(i + 1, len(features))]
```

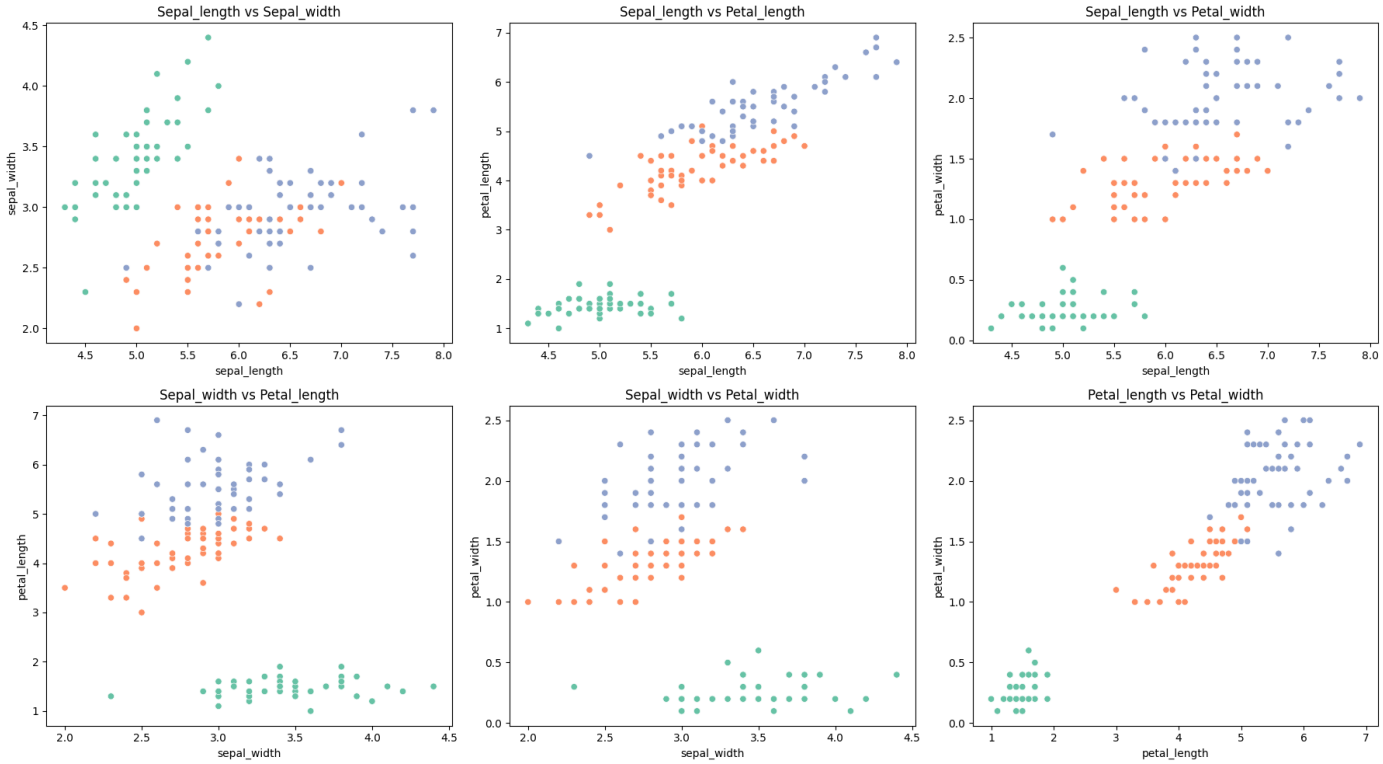
سپس در این لیست یک لوپ میزنیم تا تک تک نقاط را در صفحه رسم کنیم:

```
plt.figure(figsize=(18, 10))
for idx, (i, j) in enumerate(pair_indices, start=1):
    plt.subplot( *args: 2, 3, idx)
    sns.scatterplot(x=features[i], y=features[j], hue='species', data=df, palette='Set2', legend=False)
    plt.title(f'{features[i].capitalize()} vs {features[j].capitalize()}')
```

فراخوانی تابع بصورت زیر است:

```
plot_feature_pairs(df, features: ['sepal_length', 'sepal_width', 'petal_length', 'petal_width'])
```

خروجی *Scatterplot*:



۳D scatter plot

در این بخش برای رسم این نمودار سه بعدی ابتداعاً دو نمودار سه بعدی ax۱ و ax۲ را به شکلی که میسازیم اضافه میکنیم و با اتریبیوت projection='۳d' آنها را تبدیل به یک پلات سه بعدی میکنیم. سپس در یک لوپ برای هر رنگ اختصاص داده شده به هر نوع یا species دیستاست را به زیر مجموعه یا subset های مختلف موجود فیلتر میکنیم. پس از آن در یک لوپ روی هر نمودار ، زیرمجموعه های مرحله قبل را با دستور scatter پلات میکنیم:

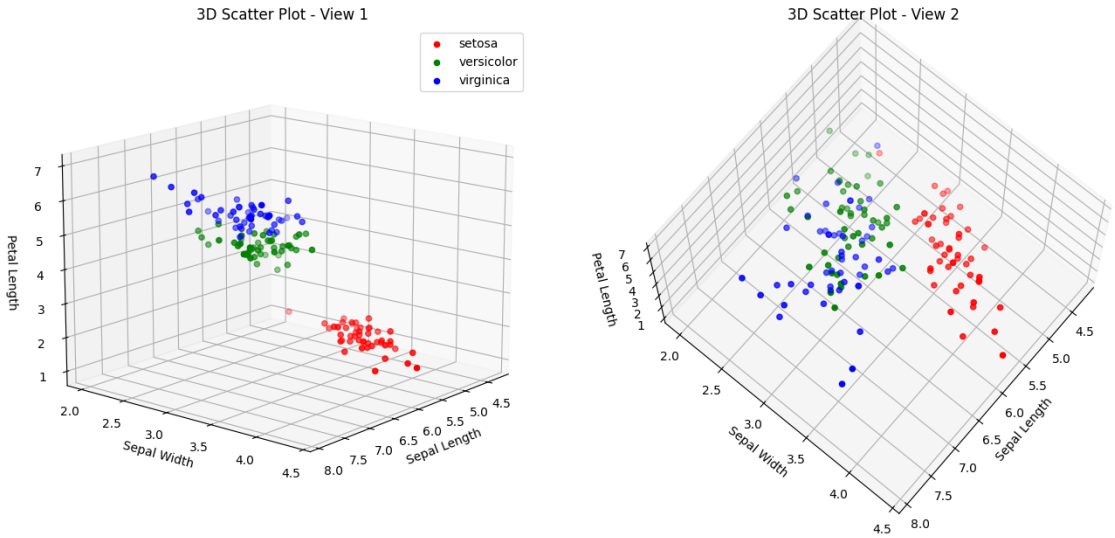
```
fig = plt.figure(figsize=(16, 8))
colors = {'setosa': 'r', 'versicolor': 'g', 'virginica': 'b'}
ax1, ax2 = fig.add_subplot(121, projection='3d'), fig.add_subplot(122, projection='3d')

for species, color in colors.items():
    subset = df[df['species'] == species]
    for ax in [ax1, ax2]:
        ax.scatter(subset['sepal_length'], subset['sepal_width'], subset['petal_length'], color=color, label=species)
```

خط دیگری که قابل توضیح است خط زیر است که با دستور view_init ویو یا دید نمودار را میتوانیم آنگونه که میخواهیم تنظیم کنیم. elev در راستای عمود دید را تغییر میدهد و azim در راستای افق:

```
ax1.view_init(elev=15, azim=40)
```

خروجی ۳D Scatter Plot:



برای دید و تفسیر بهتر دو ویو از این پلات رسم شده.

Probability Distributions

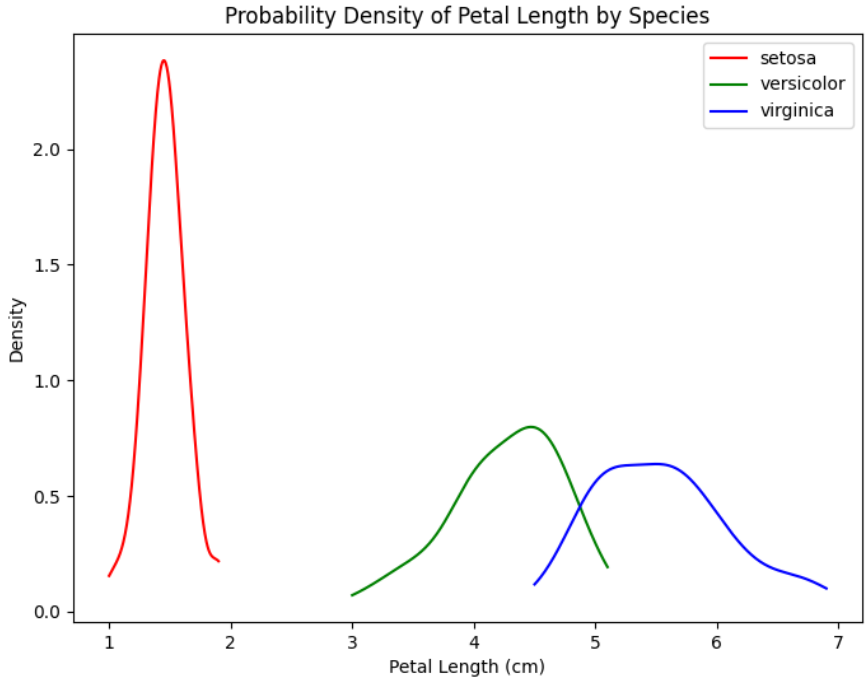
برای رسم این نمودار کافیست یک لوپ ایجاد کنیم که برای هر ویژگی و رنگی که درون دیکشنری colors است ، زیر مجموعه ای از مقادیر هر ویژگی موجود در دیتاست بسازد. سپس با تابع gaussian_kde مقدار چگالی هسته (Kernel Density Estimate) که مورد نیاز است برا کشیدن نمودار های توزیع احتمالاتی را برای species حساب میکنیم:

```
for species, color in colors.items():
    subset = df[df['species'] == species]
    density = gaussian_kde(subset['petal_length'])
```

در خط بعدی از این لوپ یک آرایه بنام xs ساخته ایم که متشکل از ۲۰۰ نقطه با فواصل یکسان از حد پایین و بالایی است که به آن میدهیم که در اینجا حد پایین subset['petal_length'].min() است و حد بالا subset['petal_length'].max(). این کار به ما کمک میکند که در ادامه با استفاده از اعمال kde محاسبه شده در خطوط قبل، روی این آرایه ، بتوانیم منحنی را شکل دهیم که دستور آن به صورت زیر است:

```
xs = np.linspace(subset['petal_length'].min(), subset['petal_length'].max(), num: 200)
plt.plot( *args: xs, density(xs), label=species, color=color)
```

خروجی Probability Distributions:



تفسیر این پلات در بخش آخر گزارش آورده شده است.

تفاسیر

Correlation Analysis

1	-0.11757	0.871754	0.817941
-0.11757	1	-0.42844	-0.36613
0.871754	-0.42844	1	0.962865
0.817941	-0.36613	0.962865	1

بررسی ماتریس Correlation

ماتریس همبستگی متقارن است و مقادیر آن از ۱ تا ۱ متغیر است. مقادیر روی قطر همه ۱ هستند، چون هر ویژگی کاملاً با خودش همبستگی دارد. مقادیر دیگر قدرت و جهت رابطه خطی بین جفت ویژگی ها را نشان می دهد.

تفسیر ضرایب همبستگی

- **مقادیر نزدیک به ۱:** همبستگی مثبت قوی، به این معنی که با افزایش یک ویژگی، ویژگی دیگر نیز تمایل به افزایش دارد.
- **مقادیر نزدیک به ۱-** : همبستگی منفی قوی، نشان دهنده یک رابطه معکوس. با افزایش یک ویژگی، ویژگی دیگر کاهش می یابد.
- **مقادیر نزدیک به ۰:** رابطه خطی ای بین ویژگی ها وجود ندارد.

تفسیر هر جفت ویژگی

ویژگی سطر ۱ و ستون ۳ (0.871754):

این همبستگی مثبت بالا (نزدیک به ۱) نشان دهنده یک رابطه خطی قوی است. با افزایش ویژگی سطر ۱، ویژگی ستون ۳ نیز تمایل به افزایش دارد.

ویژگی سطر ۱ و ستون ٤ (0.817941):

این همبستگی مثبت قوی دیگری است که نشان می دهد با افزایش ویژگی ۱، ویژگی ٤ نیز تمایل به افزایش دارد. این جفت ممکن است شامل ابعاد گلبرگ نیز باشد که اغلب رابطه مثبتی دارند.

ویژگی سطر ۲ و ستون ۳ (-0.42844):

این همبستگی منفی متوسط نشان می دهد که با افزایش ویژگی ۲، ویژگی ۳ تمایل به کاهش دارد. این جفت ممکن است شامل روابط بین ابعاد کاسبرگ و گلبرگ باشد که گاهی اوقات می تواند روند معکوس را در گونه های گیاهی خاص نشان دهد.

ویژگی سطر۲ و ستون٤ (-0.36613):

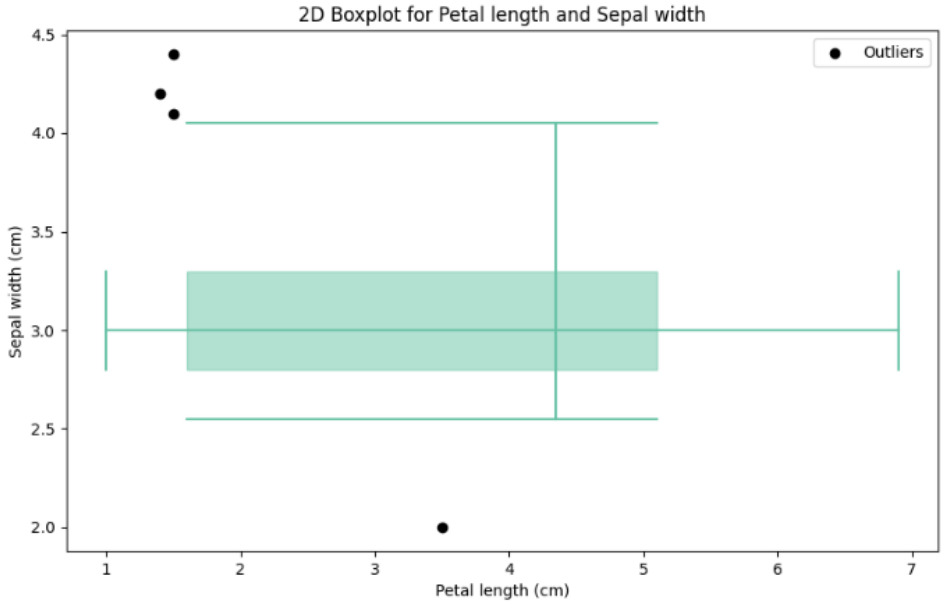
یک همبستگی منفی متوسط در اینجا مشاهده می شود. این بدان معنی است که با افزایش ویژگی ۲، ویژگی ٤ تمایل به کاهش دارد.

جفت ویژگی های دیگر (همبستگی های پایین تر):

جفت‌های باقی‌مانده همبستگی‌های کمتری را نشان می‌دهند (به عنوان مثال، ۰.۱۱۷۵۷)، که نشان‌دهنده روابط خطی ضعیف‌تر بین این ویژگی‌ها است.

در دیتاست IRIS ، این همبستگی ها نشان می دهد که ابعاد petal به شدت با یکدیگر همبستگی دارند، در حالی که ابعاد petal و sepal ممکن است یک رابطه ضعیف تر و گاهی معکوس داشته باشند.

Box plot and ۲D Box plot

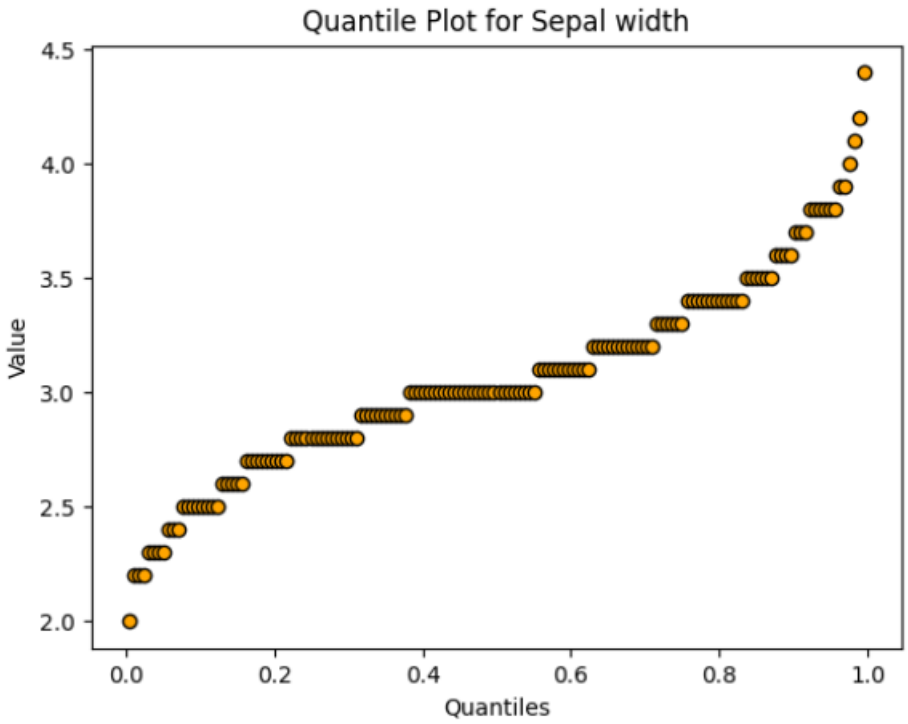


Whiskers: خطوطی هستند که از محاسبه جمع و تفریق چارک اول و سوم با ۱.۵ برابر شاخص IQR که مساوی است با اختلاف چارک اول و سوم به دست می آیند و حد بالای آنها مشخص می شود.

مشخص کردن نقاط داده پرت: داده هایی که بیشتر از مقدار محاسبه شده whisker ها برا داده های بزرگ تر از چارک سوم یا به عبارتی بیشتر از $Q3 + IQR \times ۱.۵$ هستند به عنوان داده های پرت حد بالا مشخص میشوند و داده هایی که کمتر از $Q1 - IQR \times ۱.۵$ باشند به عنوان داده های پرت حد پایین تشخیص داده می شوند.

Quantile plot

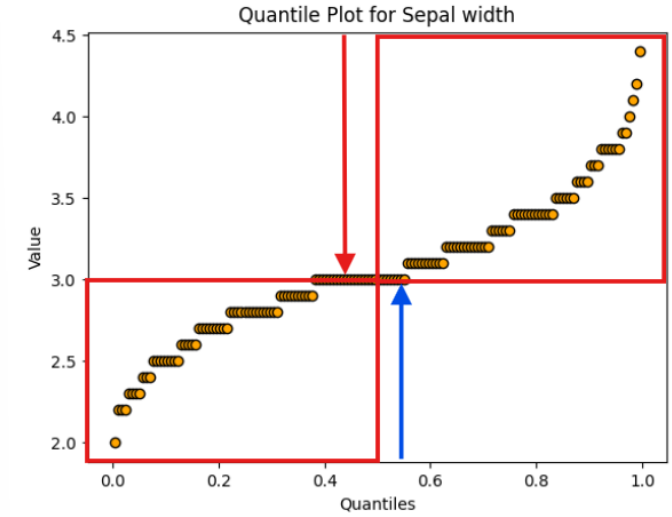
تفسیر نمودار sepal width:



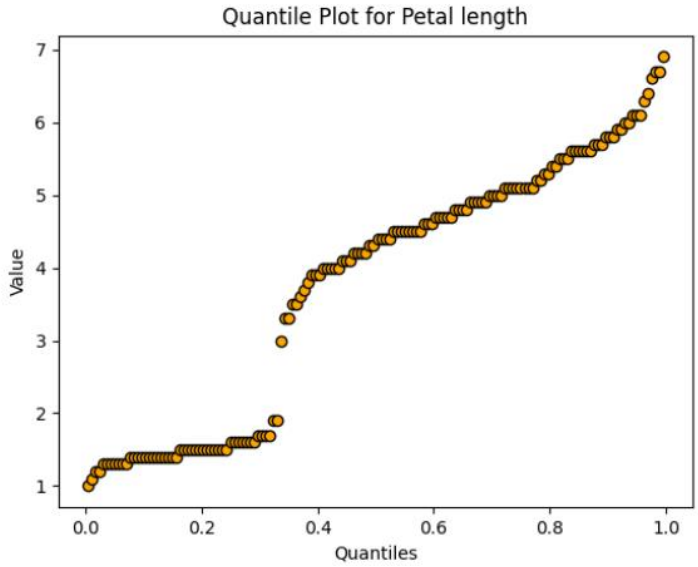
با توجه به نمودار میتوان به نکات زیر پی برد:

- ✓ نمودار یک روند به صورت یکنواخت افزایشی را از چارک های پایین (چپ) به چارک های بالاتر (راست) نشان می دهد، که نشان می دهد مقادیر sepal width به طور مداوم در محدوده آنها توزیع می شود.
- ✓ شیب نمودار از چپ به راست تغییر می کند، که نشان می دهد توزیع کاملا یکنواخت یا نرمال نیست.
- ✓ کم شدن تراکم در چندک های بالاتر (سمت راست) نشان دهنده وجود مقادیر بالاتر (داده های پرت) برای sepal width است.

- ✓ اگر از quantile = ۰٫۵ نمودار را به دو قسمت چپ و راست تقسیم کنیم، متوجه میشویم که قسمت راست بزرگتر است(اندازه فلش قرمز به شکل مشهودی از آبی بیشتر است) و داده ها پراکندگی بیشتری نسبت به هم در مقایسه با داده های سمت چپ دارند که این امر ممکن است توزیع را به سمت راست متمایل کرده باشد.



تفسیر نمودار petal length:

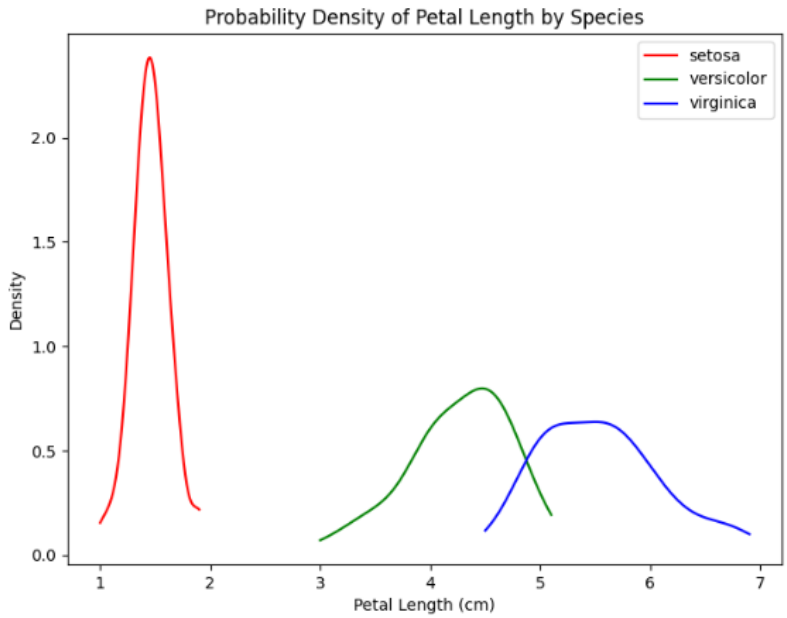


با توجه به نمودار میتوان به نکات زیر پی برد:

- ✓ نمودار دارای یک جهت مشهود حدود ۰.۳ دارد که فاصله زیادی بین نقاط داده ایجاد شده که این میتواند نشان دهنده توزیع bimodal یا دو وجهی باشد که میتوان پی برد احتمالا دو گروه مجزا از داده های iris وجود دارد.
- ✓ گروه اول دارای petal length در بازه ۱ تا ۲ است درحالی که گروه دوم از حدود ۳ تا ۷ است که هم نشان از طول بیشتر آنها دارد هم نشان از تعدد و فراوانی آنها در طول های مختلف است.
- ✓ توزیع در هر دو گروه با توجه به شکل پیوسته و به صورت تدریجی افزایشی است.

Probability Distributions

تفسیر نمودار توزیع احتمالاتی هر نوع (species) بر اساس petal length:



منحنی setosa:

- با توجه به فرم منحنی و باریک بودن آن میتوان تشخیص داد که تنوع و فراوانی طول های مختلف در این نوع کم است و همه در بازه ۱ تا ۲ قرار دارند.
- این نوع دارای کمترین طول petal در بین همه انواع است.
- پرتکرارترین طولی که در این توع قرار دارد ۱.۵ است.

منحنی versicolor:

- این منحنی نسبت به منحنی قبل دارای بازه بیشتری یعنی بین۳ تا حدود ۵ است که حکم از تنوع بیشترطول ها در این نوع دارد.
- این نوع دارای چولگی چپ است و بیشتر طول ها به سمت راست بازه تمایل دارند.
- پرتکرارترین طولی که در این توع قرار دارد ٤.٥ است.

منحنی virginica:

- این منحنی تقریبا تنوع طولی بیشتری از هر دو منحنی قبل دارد و بازه طولی آن از ٤.٥ تا ۷ است که بیشترین است که نشان از پراکندگی بیشتر طول ها در این نوع است.

- این نوع دارای چولگی راست است و بیشتر طول ها به چپ راست بازه تمایل دارند.
- پرتکرارترین طولی که در این توع قرار دارد تقریبا در بازه ۵.۲ تا ۵.٦ است.

جمع بندی:

- ✓ سه گونه iris دارای محدوده مشخصی از طول گلبرگ هستند، با حداقل همپوشانی.
- ✓ Setosa دارای طول گلبرگ بسیار کوتاه و ثابت، Versicolorدارای طول های متوسط با مقداری متغیر، و Virginica دارای طولانی ترین گلبرگ ها با بیشترین تنوع است.