November 27, 2024 Module 2

Plotting for Exploratory Data analysis (EDA)

→ EDA is a task of analysing data using simple tools from statistics, plotting tools and linear algebra.

→ Vector is a n- dimensional numerical array.

→ Dataset is whole given table

→ Feature / variable / input - variable / independent variable are sepal and petal length and width.

→ Label / #dependent variable / Output - variable / class / Response label is the species to which particular flower belongs.

Dataset Used

↳ Iris flower dataset

In [75]: # how many data - points and features?
print (iris. shape)
(150, 5)

In [76]: # what are column names in our dataset?
print (iris.columns)
Index(['sepal_length', 'sepal_width', 'petal_length',
'petal_width', 'species'],
dtype = 'object')

In [77]: # How many data points for each class are present
How many flowers for each species are present
iris ("species"). value_counts()

Out [77]: setosa 50
virginica 50
versicolor 50
Name: species, dtype: int64

[refer attached images]

(Pair-Plot)

```
In [11]: # pairwise scatter plot: Pair-Plot
         # Dis-advantages:
         ## can be used when number of features are
            high
         # cannot visualize higher dimensional
            Patterns in 3D and 4-D
         # only possible to view 2D patterns
         plt.close();
         sns.set_style("whitegrid");
         sns.pairplot(iris, hue="species", size=3);
         plt.show()
```

[refer given pair-plot images]

Observations

1. petal_length and petal_width are most useful features to identify various flower types.
2. while Setosa can be easily identified (linearly separable), Virginica and Versicolor have some overlap.

Limitation of pair-plot

1. Pair-plot are easy to understand when dimensionality of data is small but as soon as the dimensionality becomes greater than 6 pair-plot cannot help much.

Histogram, PDF, CDF

In[175]:
```
# what about 1-D scatter plot using one feature?
# 1-D scatter plot of petal length
import numpy as np
iris_setosa = iris.loc[iris["species"]=="setosa"];
iris_virginica= iris.loc[iris["species"]=="virginica"];
iris_versicolor= iris.loc[iris["species"]=="versicolor"];
plt.plot(iris_setosa["petal_length"], np.zeros_like(iris_setosa
                                        ['petal_length');
plt.plot(iris_versicolor["petal_length"], np.zeros_like(iris_
                                        versicolor['petal_length');
plt.plot(iris_virginica["petal_length"], np.zeros_like(iris_
                                        virginica['petal_length')
plt.show()
```

[refer to attached image]

In[72]:
```
sns.FacetGrid(iris, hue="species", size=5)
   .map(sns.distplot,"petal_length")
   add_legend();
plt.show();
```

[refer attached image]

Histogram on probability density function
(Univariant analysis)

CDF (Cumulative Distribution Function)

differentiating cdf → pdf
integration of pdf → cdf

In [12]:
```
# How to construct and read CDF?
# plot CDF of petal length
counts, bin_edges = np.histogram(iris_setosa
                                 ['petal-length'], bins=10, density=True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)

compute CDF
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:], pdf)
plt.plot(bin_edges[1:], cdf)

plt.show()
```

(refer attached image)

Mean, Variance and std-dev

```
In [164]: # Mean, variance, std-dev
          print("Means:")
          print(np.mean(iris_setosa["petal_length"]))
          # Mean with outlier
          print(np.mean(np.append(iris_setosa["petal_length"],
                                  150)));

          print(np.mean(iris_virginica["petal_length"]))
          print(np.mean(iris_versicolor["petal_length"]))

          print("\nst-dev:");
          print(np.std(iris_setosa["petal_length"]))
          print(np.std(iris_virginica["petal_length"]))
          print(np.std(iris_versicolor["petal_length"]))
```

$$\text{Means } (\mu) = \sum_{i=1}^{n} x_i \times \frac{1}{n}$$

$$\text{Var} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2$$

$$\text{(std-dev) } \sqrt{\text{Var}} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2}$$

Means:

1.464

2.4156862745098

5.552

4.26

Std-dev:

0.17176728442929

0.5463478745271

0.4651881339845

Std-dev ∝ spread

Median, Percentile, Quantile, IQR, MAD

```
In[167]: print("\nMedians:")
         print(np.median(iris_setosa["petal_length"]))
         # Median with outlier
         print(np.median(np.append(iris_setosa["petal_
                                   length"], 50)));
         print(np.median(iris_virginica["petal_length"]))
         print(np.median(iris_versicolor["petal_length"]))
```

Medians:

1.5
1.5
5.55
4.35

Median

$x = \{1, 1.2, 1.1, 2.1, 1.8, 1.6, 1.2\}$

① Sort them inc order

$\{1, 1.1, 1.2, 1.2, 1.6, 1.8, 2.1\}$

1 2 3 (4) 5 6 7

② pinck middle value $\left(\frac{n+1}{2}\right)$ odd $\left(\frac{7+1}{2}\right) = 4$

$\left(\frac{n}{2}\right)$ even $\left[\frac{n}{2}, \frac{n}{2}+1\right]$ average

Median of $x = 1.2$

```
In[168]: print("\n 90th Percentiles :")
         print(np.percentile(iris_setosa["petal_length"],90))
         print(np.percentile(iris_virginica["petal_length"],90))
         print(np.percentile(iris_versicolor["petal_length"],90))
```

90th Percentiles:

1.7

6.31

4.8

Percentile

$x_8 = \{1, 2, 3, \ldots\ldots, 100\}$

$n = 100$

$\text{Median}(x) = \dfrac{x_{50} + x_{51}}{2}$ (Mean $[x_{50}, x_{51}]$)

Median = 50th percentile value of $x = x_8[50]$

10th percentile value of $x = x_8[10]$

25th percentile, 50th, 75th, 100th = Quantile

In [167]:
```
print("\nQuantiles:")
print(np.percentile(iris_setosa["petal_length"],
                    np.arange(0,100,25)))
print(np.percentile(iris_virginica["petal_length"],
                    np.arange(0,100,25)))
print(np.percentile(iris_versicolor["petal_length"],
                    np.arange(0,100,25)))
from statsmodels import robust
print("\nMedian Absolute Deviation:")
print(robust.mad(iris_setosa["petal_length"]))
print(robust.mad(iris_virginica["petal_length"]))
print(robust.mad(iris_versicolor["petal_length"]))
```

Quantiles:

```
[1.    1.4   1.5   1.575]
[4.5   5.1   5.55  5.875]
[3.    4.    4.35  4.6]
```

Mean Absolute Deviation:

0.14826022185

0.667170998328

0.518910776477