Exploratory-Data-Analysis(EDA)

EDA is an approach to analyse the main chracteristics of data with some statistical, linear algebra and visulation tools to find out the hidden pattern, detect outliers and important features in data.

In [136…

```
#importing dataset
import pandas as pd
iris_dataset = pd.read_csv(r"C:\Users\abdul\Downloads\iris.csv")
```

In [137…

```
print("Number of rows/datapoints are ",iris_dataset.shape[0])
print("Number of columns/features are ",iris_dataset.shape[1])
print("columns names =",iris_dataset.columns.values)
```

```
Number of rows/datapoints are  150
Number of columns/features are  6
columns names = ['Id' 'SepalLengthCm' 'SepalWidthCm' 'PetalLengthCm' 'PetalWidthCm'
 'Species']
```

In [138…

```
iris_dataset = iris_dataset.loc[:, 'SepalLengthCm':] #removing Id column
print("after removing Id column, remaining columns names =",iris_dataset.columns.values)
```

```
after removing Id column, remaining columns names = ['SepalLengthCm' 'SepalWidthCm' 'PetalLengthCm' 'PetalWidthCm
' 'Species']
```

In [139…

```
# iris_dataset.rename(columns={"SepalLengthCm":"sepal_length", }, inplace=True)
print(iris_dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
 4   Species        150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
```

In [140…

```
print(iris_dataset["Species"].value_counts())
```

```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

observations:

```
a.)dataset doesn't have null values so preprosessing not required.
b.) Species is a categorical feature, it will be class label/feature.
c.) others features contain float values.
d.) Data is balanced because number of data points for every class is same.
```

A. Univariant Analysis

Analysis with single feature.

1. MAX, MIN, MEAN, MEDIAN, MODE, STD, PERCENTILE, QUANTILE, IQR

1.1 Max

In [141...
```
iris_dataset.max(axis=0, skipna=True,numeric_only=True)
```

Out[141...
```
SepalLengthCm    7.9
SepalWidthCm     4.4
PetalLengthCm    6.9
PetalWidthCm     2.5
dtype: float64
```

1.2 Min

In [142...
```
iris_dataset.min(axis=0, skipna=True, numeric_only=True)
```

Out[142...
```
SepalLengthCm    4.3
SepalWidthCm     2.0
PetalLengthCm    1.0
PetalWidthCm     0.1
dtype: float64
```

1.3 Mean

In [143...
```
iris_dataset.mean(axis=0, skipna=True, numeric_only=True)
```

Out[143...
```
SepalLengthCm    5.843333
SepalWidthCm     3.054000
PetalLengthCm    3.758667
PetalWidthCm     1.198667
dtype: float64
```

1.3.1 Mean after adding single outlier

In [144...
```
df = pd.DataFrame({"SepalLengthCm":[50], "SepalWidthCm":[60], "PetalLengthCm":[70], "PetalWidthCm":[80]})
iris_dataset1 = iris_dataset.append(df)
print("Mean of the fetaures after adding single outlier")
iris_dataset1.mean(axis=0, skipna=True, numeric_only=True)
```

```
Mean of the fetaures after adding single outlier
```

Out[144...
```
SepalLengthCm    6.135762
SepalWidthCm     3.431126
PetalLengthCm    4.197351
PetalWidthCm     1.720530
dtype: float64
```

1.3.2 Mean after adding multiple outliers

In [145...
```
df = pd.DataFrame({"SepalLengthCm":[50,60,70,80], "SepalWidthCm":[60,70,80,90], "PetalLengthCm":[70,60,50,80], "P
iris_dataset2 = iris_dataset.append(df)
print("mean of the fetaures after adding multiple outlier")
iris_dataset2.mean(axis=0, skipna=True, numeric_only=True)
```

```
mean of the fetaures after adding multiple outlier
```

Out[145...
```
SepalLengthCm    7.379870
SepalWidthCm     4.922727
PetalLengthCm    5.349351
PetalWidthCm     3.115584
dtype: float64
```

1.4 Median

In [146...
```
iris_dataset.median(axis=0, skipna=True, numeric_only=True)
```

```
iris_dataset.median(axis=0, skipna=True, numeric_only=True)
```

Out[146…

```
SepalLengthCm    5.80
SepalWidthCm     3.00
PetalLengthCm    4.35
PetalWidthCm     1.30
dtype: float64
```

1.4.1 Median after adding single outlier

In [147…

```
print("Median of the fetaures after adding single outlier")
iris_dataset1.median(axis=0, skipna=True, numeric_only=True)
```

```
Median of the fetaures after adding single outlier
```

Out[147…

```
SepalLengthCm    5.8
SepalWidthCm     3.0
PetalLengthCm    4.4
PetalWidthCm     1.3
dtype: float64
```

1.4.2 median after adding multiple outliers

In [148…

```
print("Median of the fetaures after adding multiple outlier")
iris_dataset2.median(axis=0, skipna=True, numeric_only=True)
```

```
Median of the fetaures after adding multiple outlier
```

Out[148…

```
SepalLengthCm    5.8
SepalWidthCm     3.0
PetalLengthCm    4.4
PetalWidthCm     1.3
dtype: float64
```

1.5 std

In [149…

```
iris_dataset.std(axis=0, skipna=True,numeric_only=True)
```

Out[149…

```
SepalLengthCm    0.828066
SepalWidthCm     0.433594
PetalLengthCm    1.764420
PetalWidthCm     0.763161
dtype: float64
```

1.5.1 std after adding single outliers

In [150…

```
print("Std of the fetaures after adding single outlier")
iris_dataset1.std(axis=0, skipna=True, numeric_only=True)
```

```
Std of the fetaures after adding single outlier
```

Out[150…

```
SepalLengthCm    3.686974
SepalWidthCm     4.654305
PetalLengthCm    5.670226
PetalWidthCm     6.457712
dtype: float64
```

1.5.2 std after adding multiple outliers

In [151…

```
print("Std of the fetaures after adding multiple outlier")
iris_dataset2.std(axis=0, skipna=True, numeric_only=True)
```

```
Std of the fetaures after adding multiple outlier
```

Out[151…
```
SepalLengthCm     9.646235
SepalWidthCm     11.630232
PetalLengthCm    10.089859
PetalWidthCm     11.938689
dtype: float64
```

1.6 Percentile

In [152…
```
iris_dataset.quantile([.20], axis=0, numeric_only=True)
```

Out[152…

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0.2	5.0	2.7	1.5	0.2

1.7 Quantile

In [153…
```
iris_dataset.quantile([.25, .5, .75, 1.0], axis=0, numeric_only=True)
```

Out[153…

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0.25	5.1	2.8	1.60	0.3
0.50	5.8	3.0	4.35	1.3
0.75	6.4	3.3	5.10	1.8
1.00	7.9	4.4	6.90	2.5

Observations:

```
a. from the mean of feature we can observe that features values will be more or less same with mean but also we can
   observe that single outlier may courrpt our mean significantly.
b. median doesnt currpoted with signle outlier but may be currpoted with many outliers. we can use median where
   all values are not eually important.
c. from the std we can understand the spread/distribution of our data from the mean. if it is low then most of the
   values will be closer to mean and if it is large then values will be far away from mean. it uses mean to compute
   the spread so it can also be currpoted with signle outlier and also std value is not much interpretable because
   it can be anything from 0 to infinity.
d. we can use mode for categorical variables.
e. percentile tells us that what percentage of values will be lesser or greater than the given percentile value. from
   the above example we can understand that 20% values of SepalLengthCm will be less than 5.0 and 80% values will
   be greater than 5.
f. quantiles basically are the [.25, .50, .75, 1.0] percentiles where .25 is Q1, .50 is Q2, .75 is Q3 and 1.0 is Q4.
g. Inter quantile range (IQR) is Q3-Q1 means between 50% values comes in IQR. Q2 is median of feature.
```

from above observations we can predict feature values. feature values will be near to the mean, median, mode, std but these can be currpted by outliers.

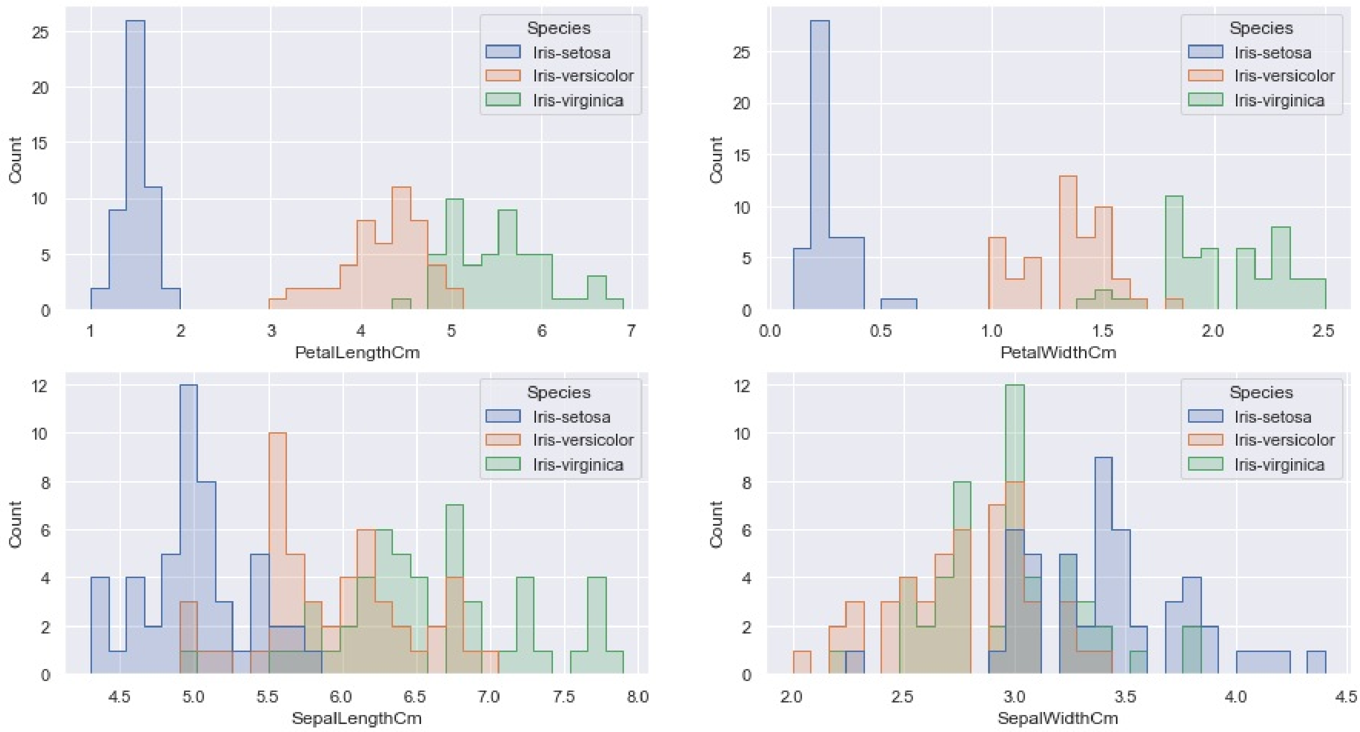
2. histogram, pdf, cdf

2.1 histogram

count the numbers of observations/datapoints are coming in a perticular bin.

In [154…
```
import seaborn as sns
```

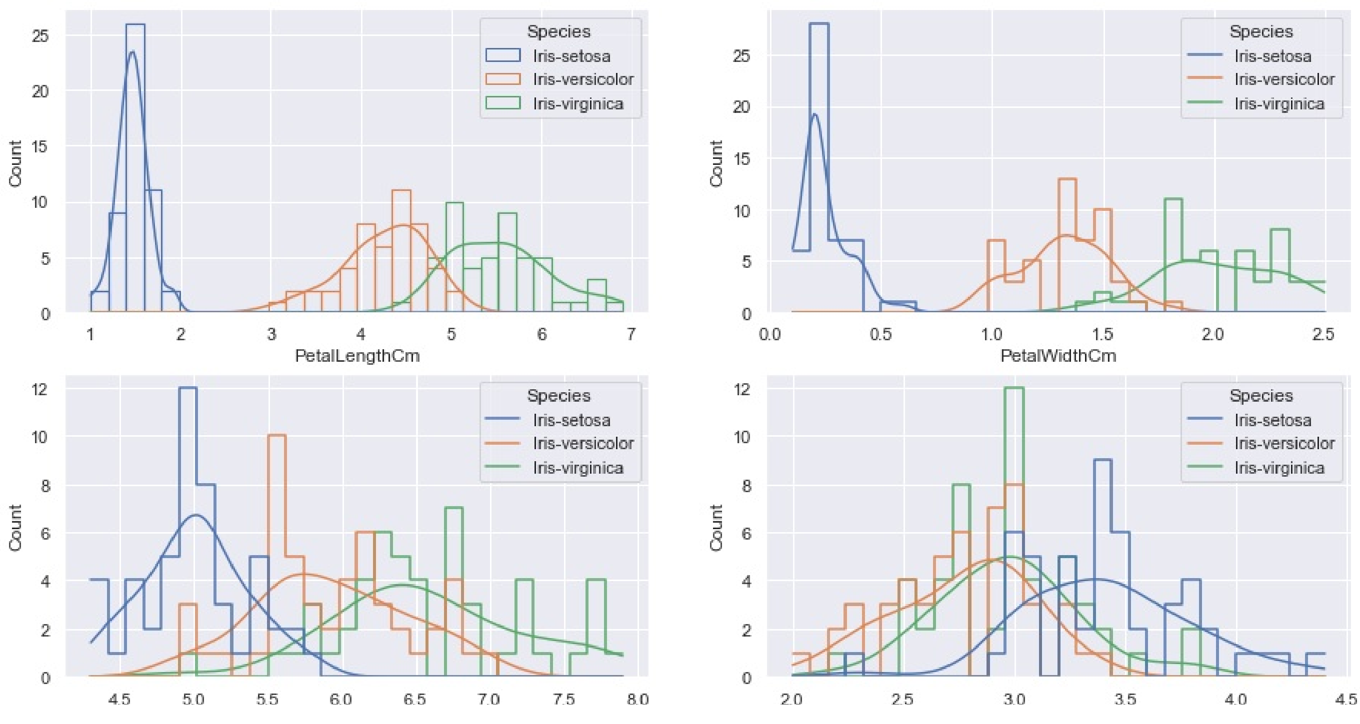
```python
import matplotlib.pyplot as plt
plt.figure(1)
plt.subplot(2,2,1)
sns.histplot(iris_dataset, x="PetalLengthCm",bins=30,hue="Species", element='step', fill=True)
plt.subplot(2,2,2)
sns.histplot(iris_dataset, x="PetalWidthCm",bins=30,hue="Species", element='step', fill=True)
plt.subplot(2,2,3)
sns.histplot(iris_dataset, x="SepalLengthCm",bins=30,hue="Species", element='step', fill=True)
plt.subplot(2,2,4)
sns.histplot(iris_dataset, x="SepalWidthCm",bins=30,hue="Species", element='step', fill=True)
plt.show()
```



2.2 pdf (smooth histogram by kde (kernal density estimation))

In [155…

```python
plt.figure(1)
plt.subplot(2,2,1)
sns.histplot(iris_dataset, x="PetalLengthCm",bins=30,hue="Species", fill=False, kde=True)
plt.subplot(2,2,2)
sns.histplot(iris_dataset, x="PetalWidthCm",bins=30,hue="Species", element='step', fill=False, kde=True)
plt.subplot(2,2,3)
sns.histplot(iris_dataset, x="SepalLengthCm",bins=30,hue="Species", element='step', fill=False, kde=True)
plt.subplot(2,2,4)
sns.histplot(iris_dataset, x="SepalWidthCm",bins=30,hue="Species", element='step', fill=False, kde=True)
plt.show()
```



In [156…

```
# sns.displot(iris_dataset, x="PetalLengthCm",hue="Species", kind="kde")
# plt.show()
# sns.displot(iris_dataset, x="PetalWidthCm",hue="Species", kind="kde")
# plt.show()
# sns.displot(iris_dataset, x="SepalLengthCm",hue="Species", kind="kde")
# plt.show()
# sns.displot(iris_dataset, x="SepalWidthCm",hue="Species", kind="kde")
# plt.show()
```

2.3 CDF

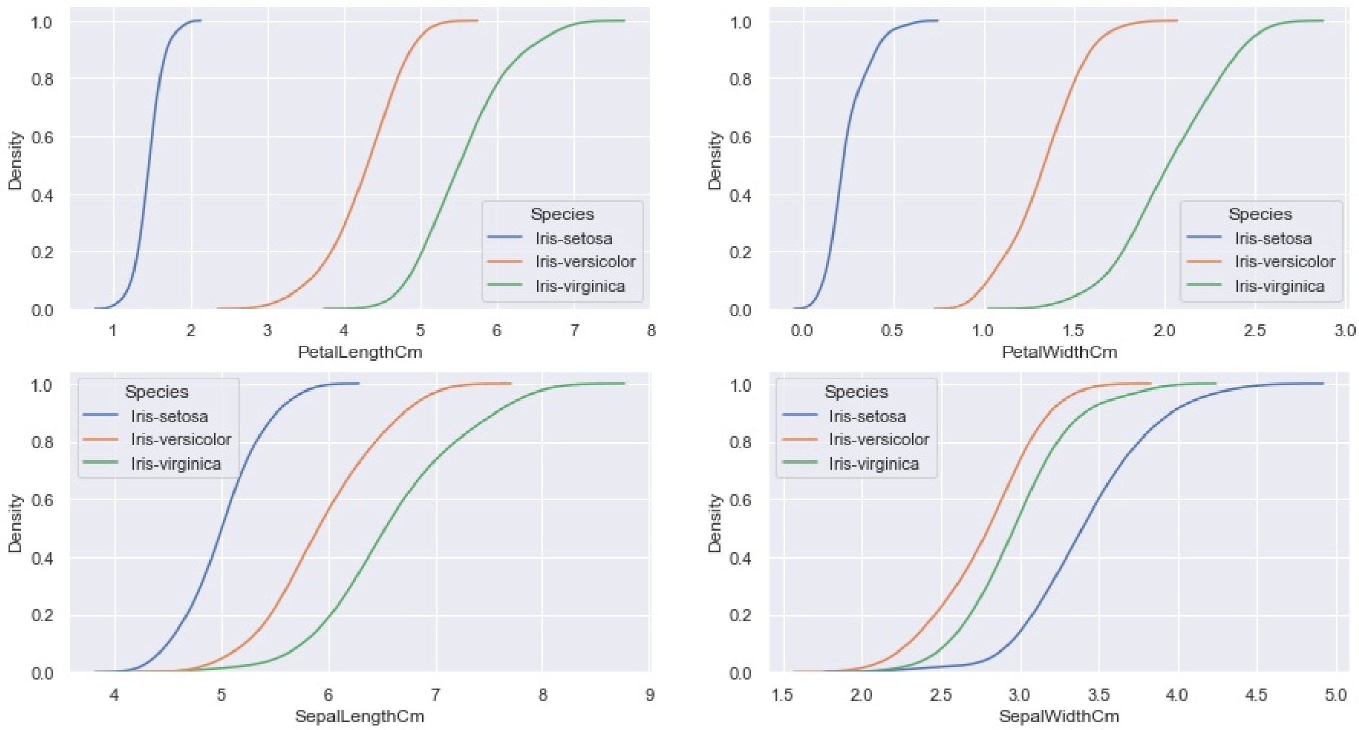
The cumulative distribution function (CDF) FX(x) describes the probability that a random variable X with a given probability distribution will be found at a value less than or equal to x.

In [157…

```
# sns.displot(iris_dataset, x="PetalLengthCm",hue="Species", kind="ecdf")
# plt.show()
# sns.displot(iris_dataset, x="PetalWidthCm",hue="Species", kind="ecdf")
# plt.show()
# sns.displot(iris_dataset, x="SepalLengthCm",hue="Species", kind="ecdf")
# plt.show()
# sns.displot(iris_dataset, x="SepalWidthCm",hue="Species", kind="ecdf")
# plt.show()
```

In [158…

```
plt.figure(1)
plt.subplot(2,2,1)
sns.kdeplot(data=iris_dataset, x="PetalLengthCm", hue="Species",cumulative=True, common_norm=False)
plt.subplot(2,2,2)
sns.kdeplot(data=iris_dataset, x="PetalWidthCm", hue="Species",cumulative=True, common_norm=False)
plt.subplot(2,2,3)
sns.kdeplot(data=iris_dataset, x="SepalLengthCm", hue="Species",cumulative=True, common_norm=False)
plt.subplot(2,2,4)
sns.kdeplot(data=iris_dataset, x="SepalWidthCm", hue="Species",cumulative=True, common_norm=False)
plt.show()
```

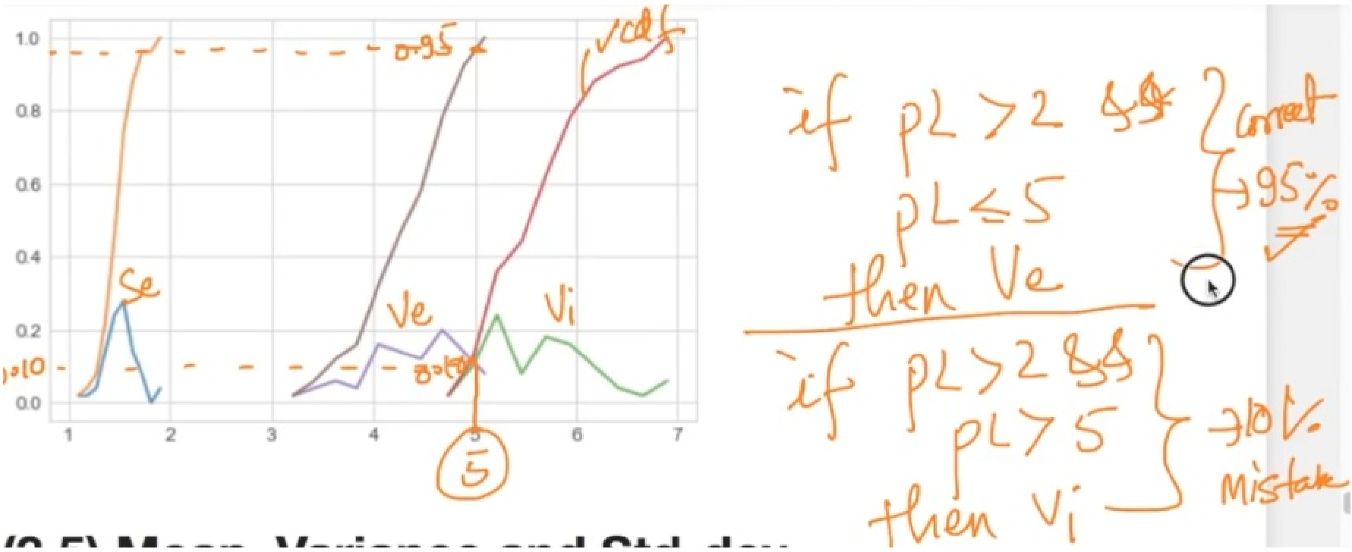


Observations:

```
a. histogram basically counts the number of observations in a prticular bin. it helps us to
observe the distribution
   of data.
b. pdfs are nothing but smooth histogram with kde that provide us the actual shape of our
distribution. just by looking
   the pdfs, we can observe that petal length of setosa flower is much smaller than virginica and
versicolor and its
   ditribution is well seprated from both. we can create a roughly model like
```

```
if petal_length <2:
    flower=setosa
```

c. from the CDFs we can observe that how much (%) data is overlapping and can create rough model.



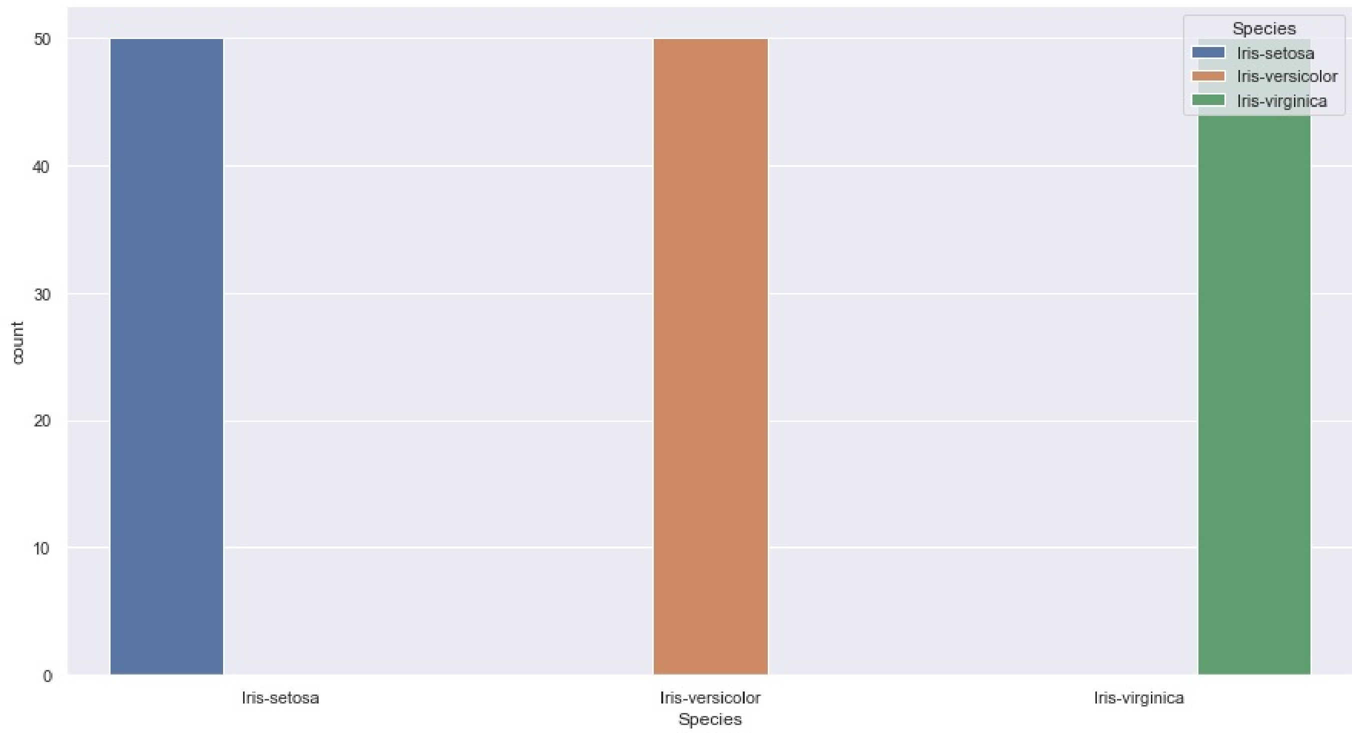
3. countplot, swarmplot, boxplot, violenplot

3.1 countplot

Show the counts of observations in each categorical bin using bars. A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable.

In [159...

```
sns.countplot(data=iris_dataset, x = "Species", hue="Species")
plt.show()
```

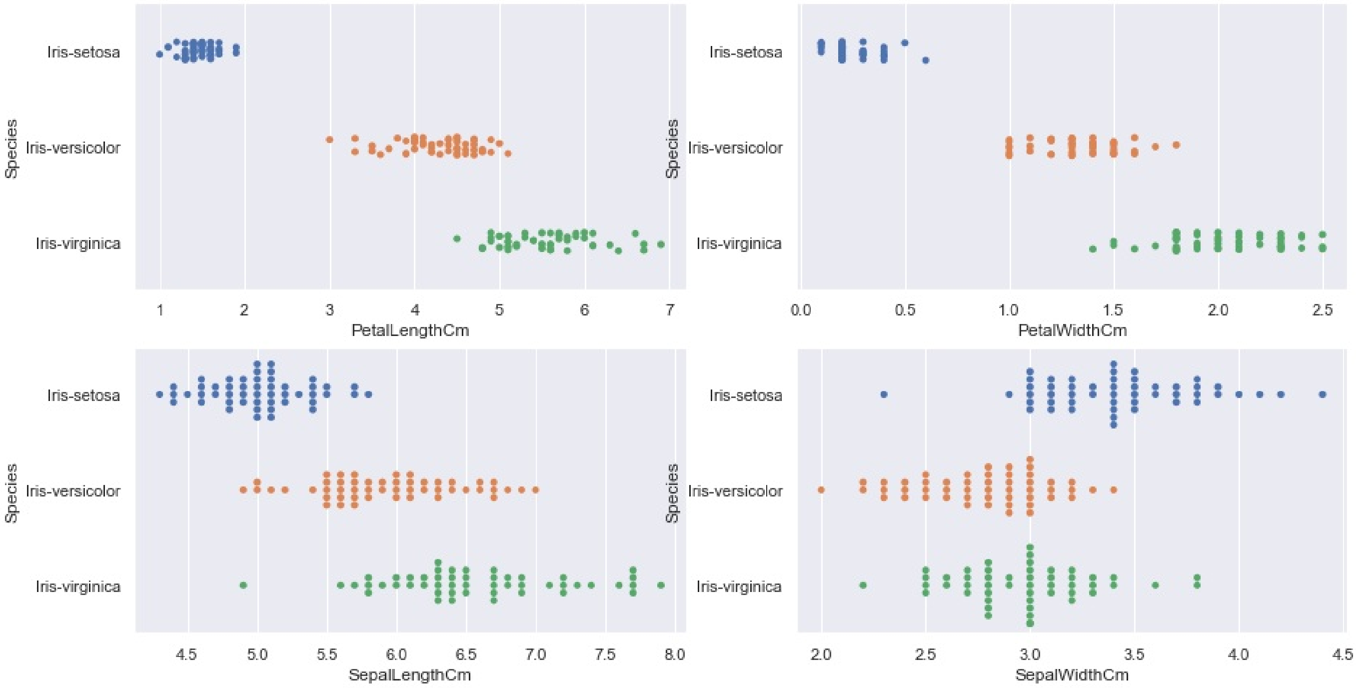


3.2 swarmplot

Draw a categorical scatterplot with non-overlapping points. This gives a better representation of the distribution of values, but it does not scale well to large numbers of observations.

In [160…

```python
plt.figure(1)
plt.subplot(2,2,1)
sns.stripplot(data = iris_dataset, x="PetalLengthCm", y="Species")
plt.subplot(2,2,2)
sns.stripplot(data = iris_dataset, x="PetalWidthCm", y="Species")
plt.subplot(2,2,3)
sns.swarmplot(data = iris_dataset, x="SepalLengthCm", y="Species")
plt.subplot(2,2,4)
sns.swarmplot(data = iris_dataset, x="SepalWidthCm", y="Species")
plt.show()
```

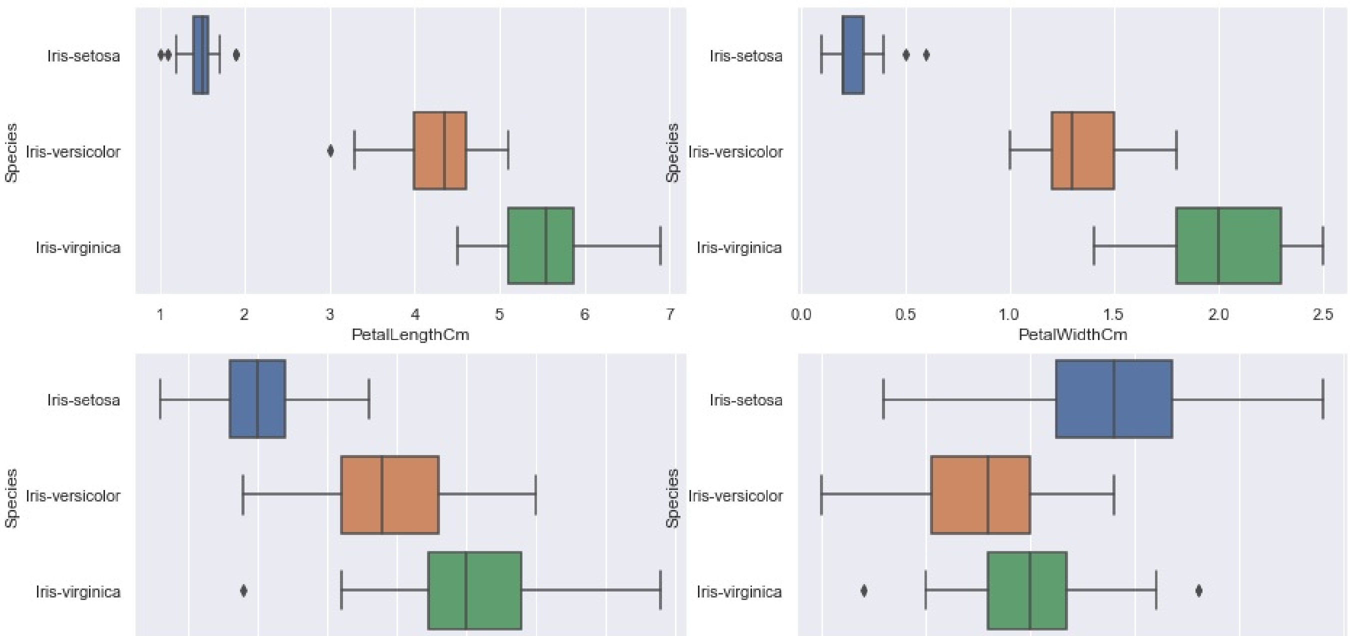


3.3 box plot

A box plot (or box-and-whisker plot) shows the distribution of quantitative data in a way that facilitates comparisons between variables or across levels of a categorical variable. The box shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution, except for points that are determined to be “outliers” using a method that is a function of the inter-quartile range.

In [161…

```python
plt.figure(1)
plt.subplot(2,2,1)
sns.boxplot(data = iris_dataset, x="PetalLengthCm", y="Species")
plt.subplot(2,2,2)
sns.boxplot(data = iris_dataset, x="PetalWidthCm", y="Species")
plt.subplot(2,2,3)
sns.boxplot(data = iris_dataset, x="SepalLengthCm", y="Species")
plt.subplot(2,2,4)
sns.boxplot(data = iris_dataset, x="SepalWidthCm", y="Species")
plt.show()
```



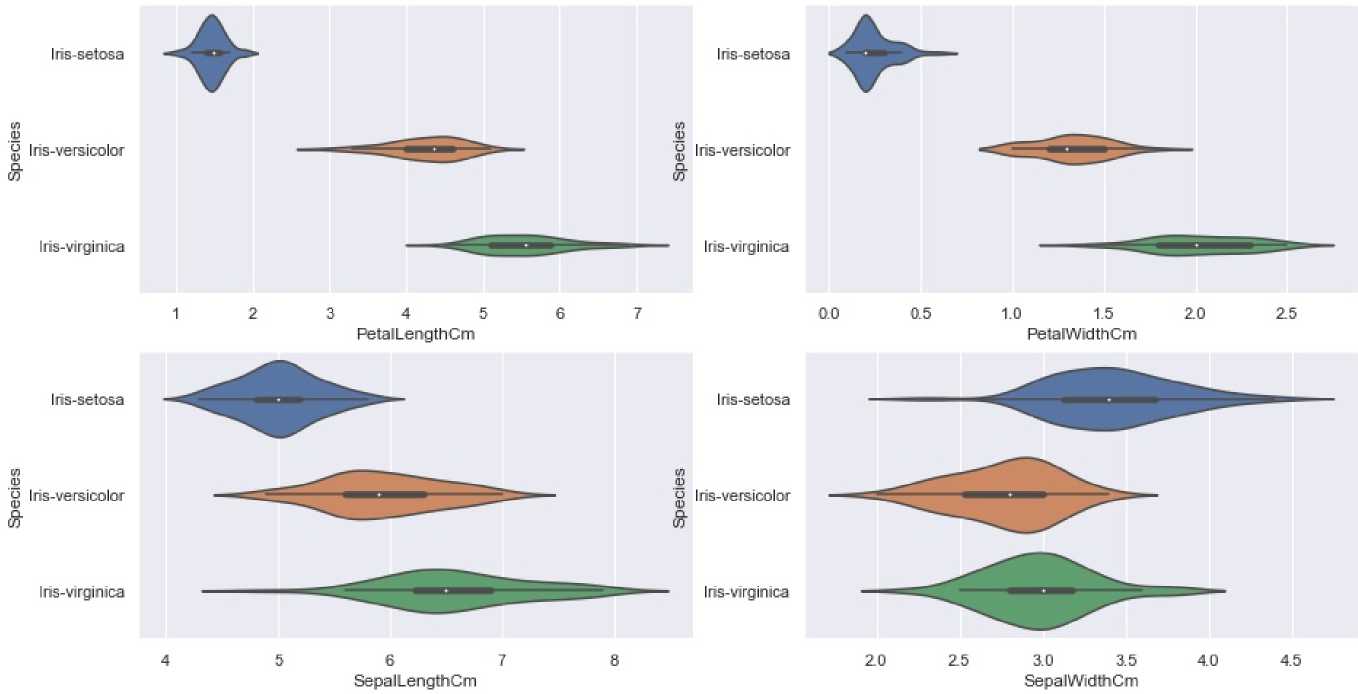
3.4 violinplot

Draw a combination of boxplot and kernel density estimate.

A violin plot plays a similar role as a box and whisker plot. It shows the distribution of quantitative data across several levels of one (or more) categorical variables such that those distributions can be compared. Unlike a box plot, in which all of the plot components correspond to actual datapoints, the violin plot features a kernel density estimation of the underlying distribution.

In [162…

```
plt.figure(1)
plt.subplot(2,2,1)
sns.violinplot(data = iris_dataset, x="PetalLengthCm", y="Species")
plt.subplot(2,2,2)
sns.violinplot(data = iris_dataset, x="PetalWidthCm", y="Species")
plt.subplot(2,2,3)
sns.violinplot(data = iris_dataset, x="SepalLengthCm", y="Species")
plt.subplot(2,2,4)
sns.violinplot(data = iris_dataset, x="SepalWidthCm", y="Species")
plt.show()
```



Obervations:

```
a. just by looking the countplot we can say our data is balanced because every class has equal
distribution.
b. from the swarmplot we can observe how many points are actually overlapping with other class.
c. boxplot help us to detect the outliers and also shows how much data is overlapping.
d. violinplot is combination of pdf(kde) and boxplot where it shows the distribution of data and
also class overlapping.
```

B. Bivariant Analysis

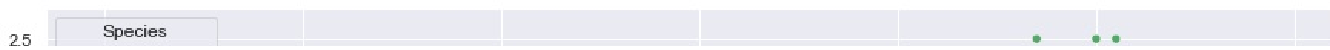
analysis with 2 features

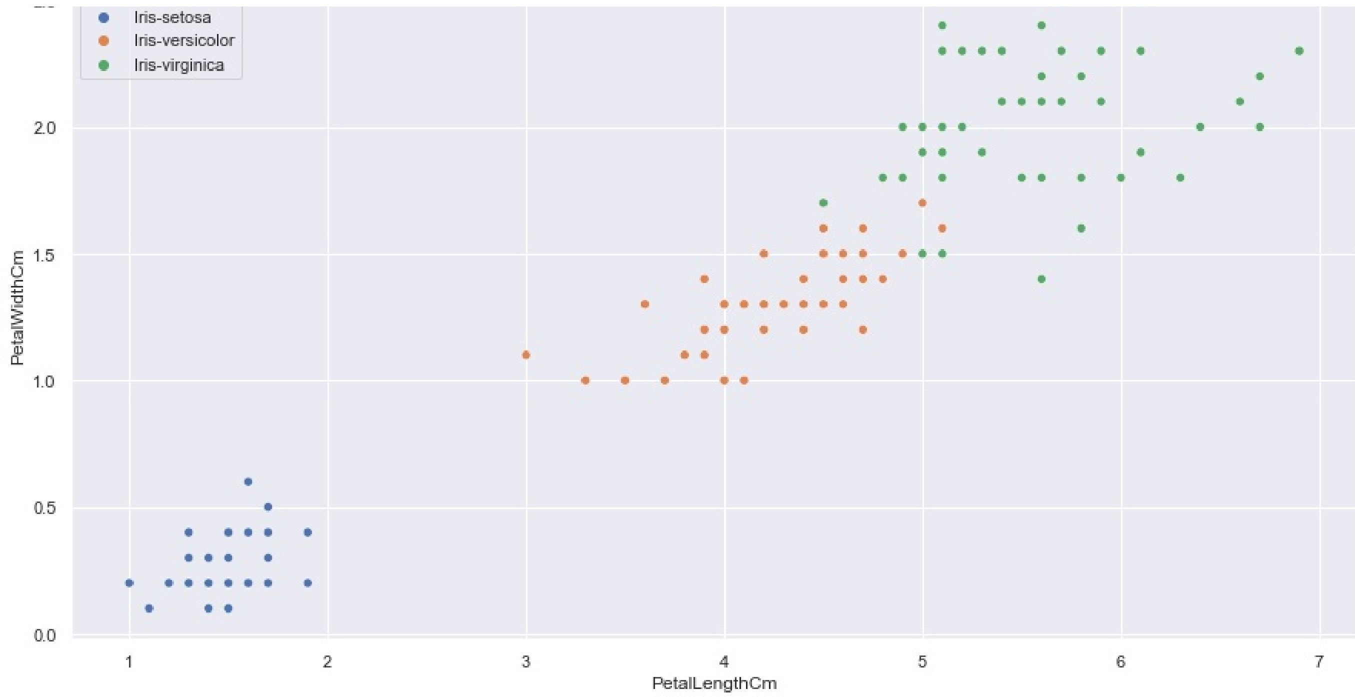
1. scatterplot

Draw a scatter plot with possibility of several semantic groupings.

In [163…

```
sns.scatterplot(data=iris_dataset, x="PetalLengthCm", y="PetalWidthCm", hue="Species")
plt.show()
```



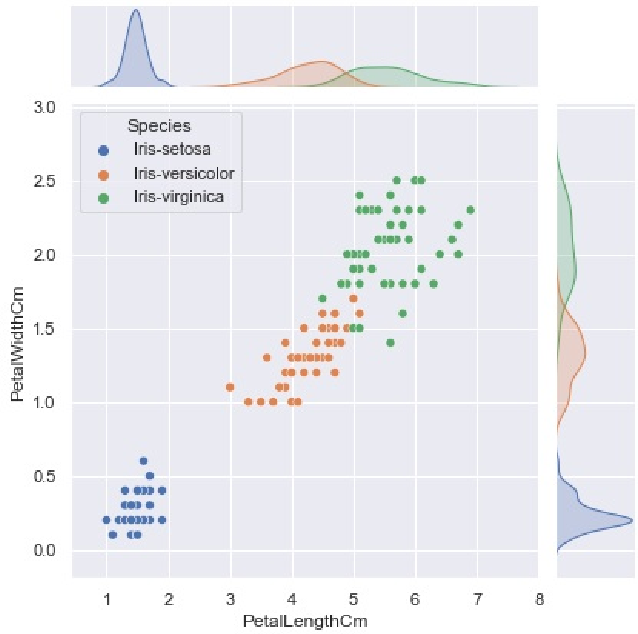


2. Jointplot

Draw a plot of two variables with bivariate and univariate graphs.

In [164…

```
sns.jointplot(data=iris_dataset, x="PetalLengthCm", y="PetalWidthCm", hue="Species")
plt.show()
```



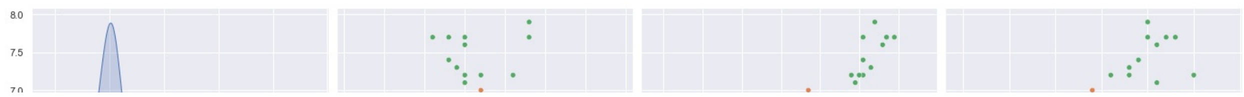
3. pairplot

Plot pairwise relationships in a dataset. useful when number of features are less.

By default, this function will create a grid of Axes such that each numeric variable in data will by shared across the y-axes across a single row and the x-axes across a single column. The diagonal plots are treated differently: a univariate distribution plot is drawn to show the marginal distribution of the data in each column.

In [165…

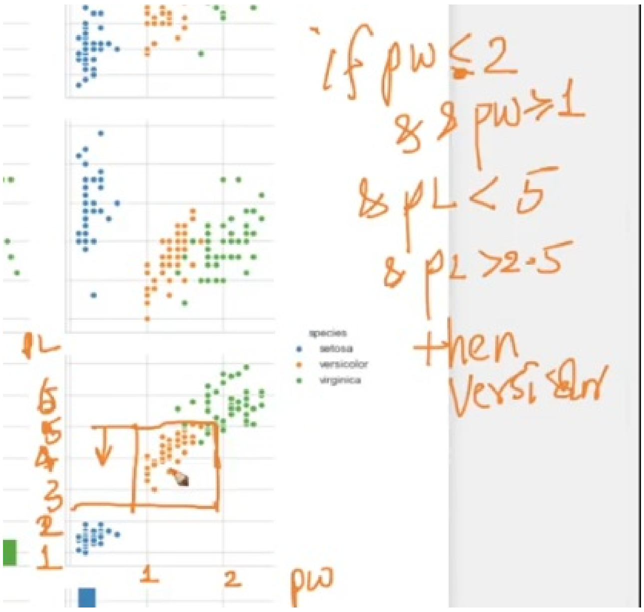
```
sns.pairplot(data=iris_dataset, hue="Species", height=5)
plt.show()
```



Observations:

```
a. scatterplot take 2 features at a time and plot the graph, help us to find the relationship
between features.
b. jointplot combination of scatterplot and pdf(kde) of features.
c. pairplot plot the every possible combination of 2 features in dataset. it is hepful when data
dim is low.
   it contains scatter plot and joinplot analysis.
```

from all of these we can make good prediction than univariant analysis.



In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js