Machine Learning 💻

K-Nearest Neighbor (KNN)

$BY \Longrightarrow PRINCE$ ❤️

Mounting Google Drive

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Importing Some Important Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv('/content/drive/MyDrive/Notes/Iris.csv')
df['Species'].value_counts()
```

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

```
# Checking Missing Values
df.isna().sum()
```

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

```
# Describe
df.describe()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

Visual Analysis

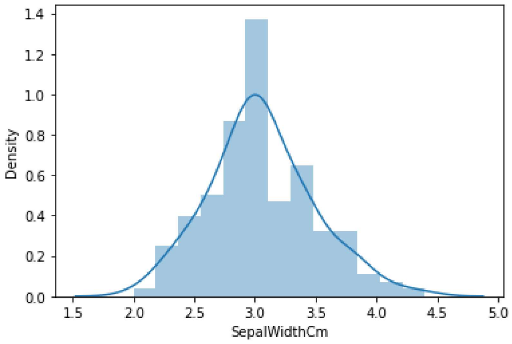
To analysis we don't need of 'Id', so we'll delete it.

```
# Visual Analysis
# To analysis we don't need of 'Id', so we'll delete it.
df.drop('Id', axis = 1, inplace = True)
```

SepalLengthCm has mean = 3.054 and median = 3.000 almost same(equal), So it has **Normalized Bell Curve**.

Normalized Bell Curve → Equally Distributed Data → Skewness closer to Zero

```
sns.distplot(df['SepalWidthCm'])
plt.show()
```

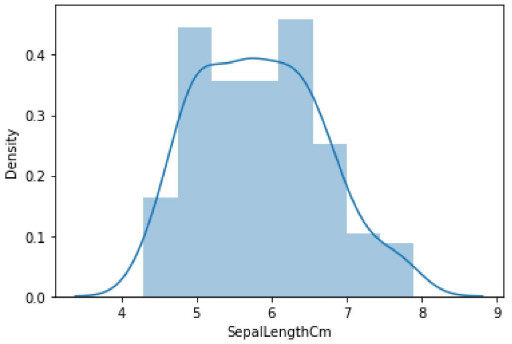



```
# Skewness
df.skew()
```

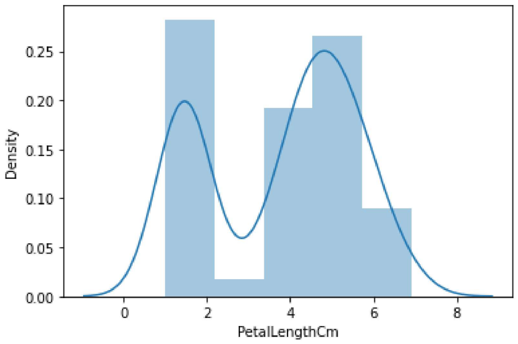
```
SepalLengthCm     0.314911
SepalWidthCm      0.334053
PetalLengthCm    -0.274464
PetalWidthCm     -0.104997
dtype: float64
```

```
# Skewness is positive and close to Zero
```

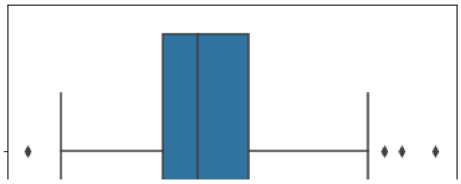
```
sns.distplot(df['SepalLengthCm'])
plt.show()
```



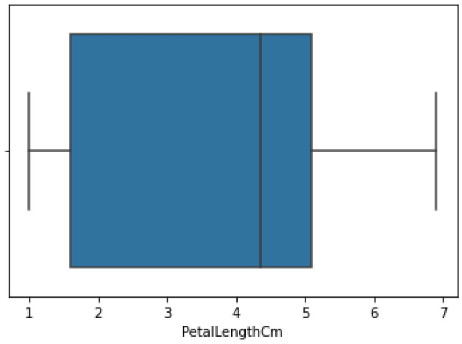

```
# PetalLengthCm has very poor data distribution
sns.distplot(df['PetalLengthCm'])
plt.show()
```



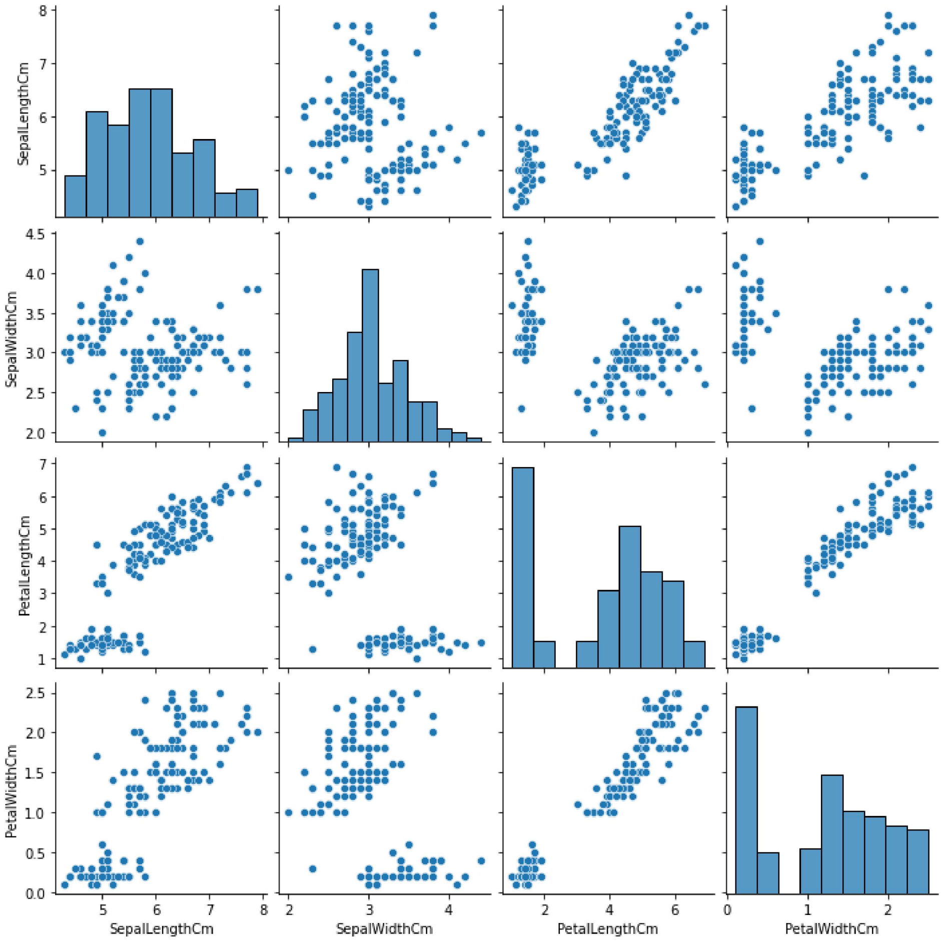

```
# Boxplot
sns.boxplot(df['SepalWidthCm'])
plt.show()
```



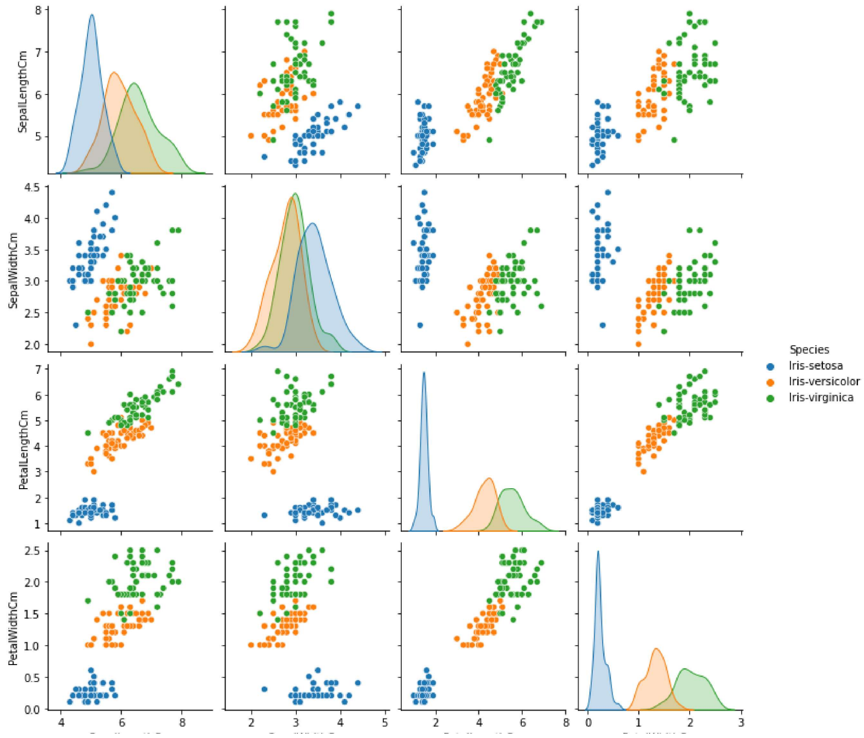
```
sns.boxplot(df['PetalLengthCm'])
plt.show()
```




```
# Pairplot
sns.pairplot(df)
plt.show()
```




```
# Pairplot w.r.t Species
sns.pairplot(df, hue = 'Species')
plt.show()
```



Preprocessing

Correlation

Predictor Variable ⟶ Species

Species ⟶ Object

So, we have to convert Species object to numbered categorical data.

```
# Correlation with Species
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Species'] = le.fit_transform(df['Species'])
df['Species'].value_counts()
```

```
0    50
1    50
2    50
Name: Species, dtype: int64
```

```
corr = df.corr()
corr
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954	0.782561
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544	-0.419446
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757	0.949043
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000	0.956464
Species	0.782561	-0.419446	0.949043	0.956464	1.000000

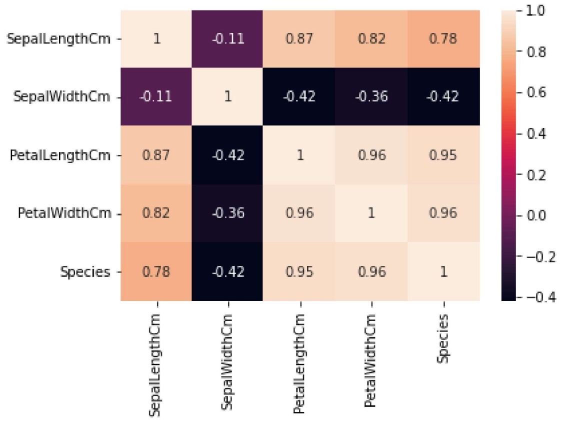
```
corr[['Species']]
```

	Species
SepalLengthCm	0.782561
SepalWidthCm	-0.419446
PetalLengthCm	0.949043
PetalWidthCm	0.956464
Species	1.000000

```
# SepalWidthCm has negative correlation with Species and
# except SepalWidthCm we've enough(four other) data for machine learning model
# So, We will drop 'SepalWidthCm'
df.drop('SepalWidthCm', axis = 1, inplace = True)
```

```
# SepalWidthCm is dropped from Iris Dataset
```

```
# Plot heatmap of correlation w.r.t 'Species'
sns.heatmap(corr, annot = True)
plt.show()
```

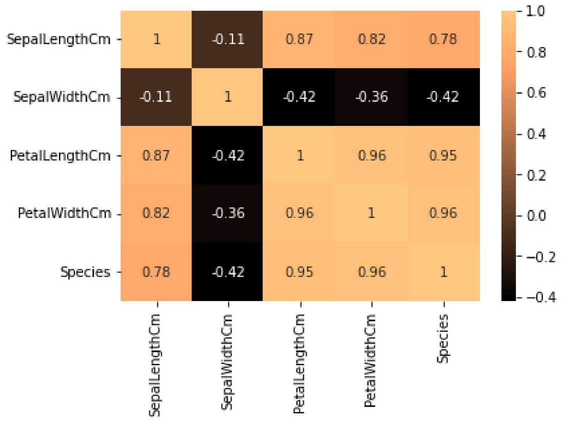


From the above graph we can see that ⟶

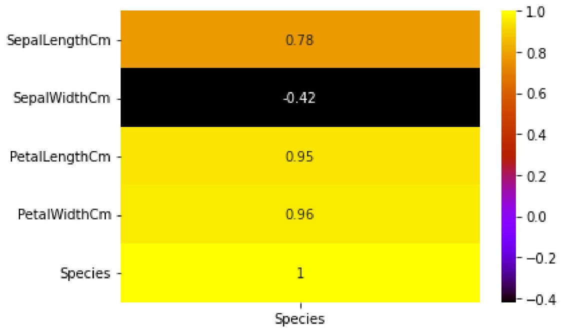
- PetalWidthCm and PetalLengthCm have high correlations.
- PetalLengthC and SepalWidthCm have good coorelations.
- PetalWidthCm and SepalLength have good correlations.

Heatmap without SepalWidthCm

```
# Heatmap without SepalWidthCm
sns.heatmap(corr, annot =True,cmap = 'copper')
# cmap = Westia, inferno, icefire, hot, gnuplot, flare, coolwarm, copper, autumn, bone, binary, bwr, cividis
plt.show()
```




```
# Correlation with Species
sns.heatmap(corr[['Species']], annot =True,cmap = 'gnuplot')
plt.show()
```



Define the Independent Variable

Note : We don't keep the dependent variable in independent (target) variable.

```
# Define the independent variable
x = df.drop('Species', axis = 1)
# We donm't keep dependent variable in independent (target) variable
x.head()
```

	SepalLengthCm	PetalLengthCm	PetalWidthCm
0	5.1	1.4	0.2
1	4.9	1.4	0.2
2	4.7	1.3	0.2
3	4.6	1.5	0.2
4	5.0	1.4	0.2

Define the Dependent (Target) Variable

```
y = df[['Species']]    # Take as 2-D or DataFrame
y.head()
```

	Species
0	0
1	0
2	0
3	0
4	0

Train Test Split in 80:20 ratio

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 1)
```

```
x_train.shape, x_test.shape
```

```
((120, 3), (30, 3))
```

```
y_train.shape, y_test.shape
```

```
((120, 1), (30, 1))
```

Second Machine Learning Algorithm

```
from sklearn.neighbors import KNeighborsClassifier
```

Created the object of the class of the model KNN.

n_neighbors is the value of K.

```
model = KNeighborsClassifier(n_neighbors = 3)
```

Train the data

```
model.fit(x_train, y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
y_pred = model.predict(x_test)
y_pred
```

```
array([0, 1, 1, 0, 2, 1, 2, 0, 0, 2, 1, 0, 2, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 0, 2, 1, 0, 0, 1, 2])
```

Accuracy

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)               # R2_Score
# Accuracy
accuracy_score(y_test, y_pred) * 100
```

```
100.0
```

Another method to find accuracy

```
r2 = model.score(x_test, y_test)
r2 * 100
```

```
100.0
```

0s completed at 3:59 PM