# classification_naive_bayes

February 4, 2024

## 1 Import required libraries and load dataset

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

[ ]:
```
df = pd.read_csv("datasets/iris/Iris.csv")
```

## 2 Explore the dataset

[ ]:
```
df.head()
```

[ ]:

| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|---|---|
| 0 | 1 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| 1 | 2 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| 2 | 3 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 3 | 4 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| 4 | 5 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |

[ ]:
```
df.shape
```

[ ]:
```
(150, 6)
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
```

```
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

[ ]:
```
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 |

## 3 Null value check

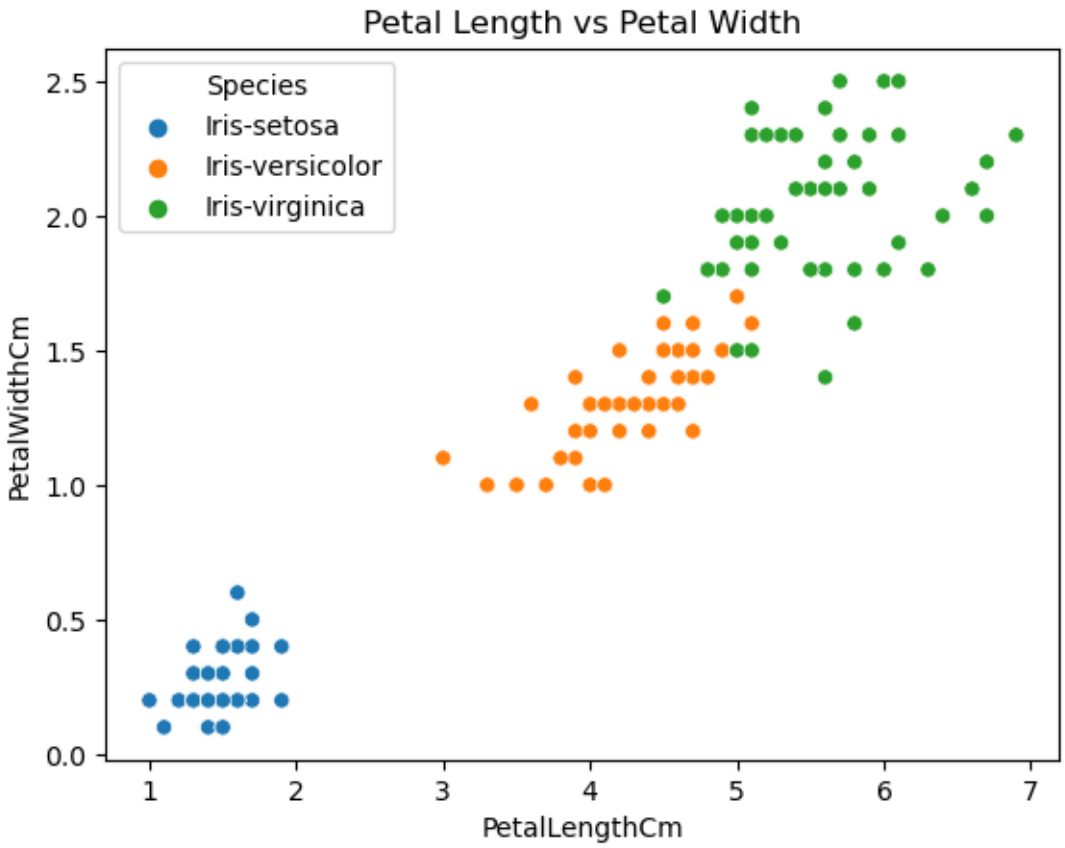
[ ]:
```
df.isnull().sum()
```

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

## 4 Some plot to visualize the data

[ ]:
```
sns.scatterplot(x = 'SepalLengthCm', y = 'PetalWidthCm', data = df ,hue␣
↪='Species')
plt.title('Petal Length vs Petal Width')
plt.show()
```


[ ]:
```
sns.scatterplot(x = 'PetalLengthCm', y = 'PetalWidthCm', data = df ,hue␣
↪='Species')
plt.title('Petal Length vs Petal Width')
plt.show()
```




## 5 lets label encode all the categorical problems

[ ]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df["Species"] = le.fit_transform(df["Species"])
df.head()
```

[ ]:

| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|---|---|
| 0 | 1 | 5.1 | 3.5 | 1.4 | 0.2 | 0 |
| 1 | 2 | 4.9 | 3.0 | 1.4 | 0.2 | 0 |
| 2 | 3 | 4.7 | 3.2 | 1.3 | 0.2 | 0 |
| 3 | 4 | 4.6 | 3.1 | 1.5 | 0.2 | 0 |
| 4 | 5 | 5.0 | 3.6 | 1.4 | 0.2 | 0 |

[ ]:
```
X = df.drop(columns=["Species","Id"])
y = df.Species
```

## 6 Do the test train split

[ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
↪80,random_state=0)
```

## 7 Train the model

[ ]:
```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train,y_train)
```

[ ]:
```
GaussianNB()
```

## 8 Make some predictions

[ ]:
```
y_pred = model.predict(X_test)
y_pred
```

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

## 9 Evaluate the model

[ ]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

[ ]:
```
0.9666666666666667
```

[ ]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       0.93      1.00      0.96        13
           2       1.00      0.83      0.91         6

    accuracy                           0.97        30
   macro avg       0.98      0.94      0.96        30
weighted avg       0.97      0.97      0.97        30
```

**10 Conclusion: As the data is clustered as seen the plot's above that makes it clear to the conclusion that why knn can perform beeter on 0(Iris-setosa) and as both the other classes are kinda merging together at the end that would explain why there is some drop in recall ,f1-score and precision, an accuracy of 96.6 and good scores in confusion matrix.**