NNFS Lab -3

Building model and predicting accuracy of every algorithm used (KNN, Regression and Bayes classifier)

AIM: - Building model and predicting accuracy of every algorithm used (KNN, Regression and Bayes classifier).

PLATFORM & TOOLS USED:

1. Google colab
2. Python
3. Sklearn

THEORY:

Every machine learning project begins by understanding what the data and drawing the objectives. While applying machine learning algorithms to your data set, you are understanding, building and analyzing the data as to get the end result.

Following are the steps involved in creating a well-defined ML project:

1. Understand and define the problem
2. Prepare the data
3. Explore and Analyse the data
4. Apply the algorithms
5. Reduce the errors
6. Predict the result

To understand various machine learning algorithms let us use the Iris data set, one of the most famous datasets available.

PROBLEM STATEMENT

This data set consists of the physical parameters of three species of flower — Versicolor, Setosa and Virginica. The numeric parameters which the dataset contains are Sepal width, Sepal length, Petal width and Petal length. In this data we will be predicting the classes of the flowers based on these parameters.The data consists of continuous numeric values which describe the dimensions of the respective features. We will be training the model based on these features.

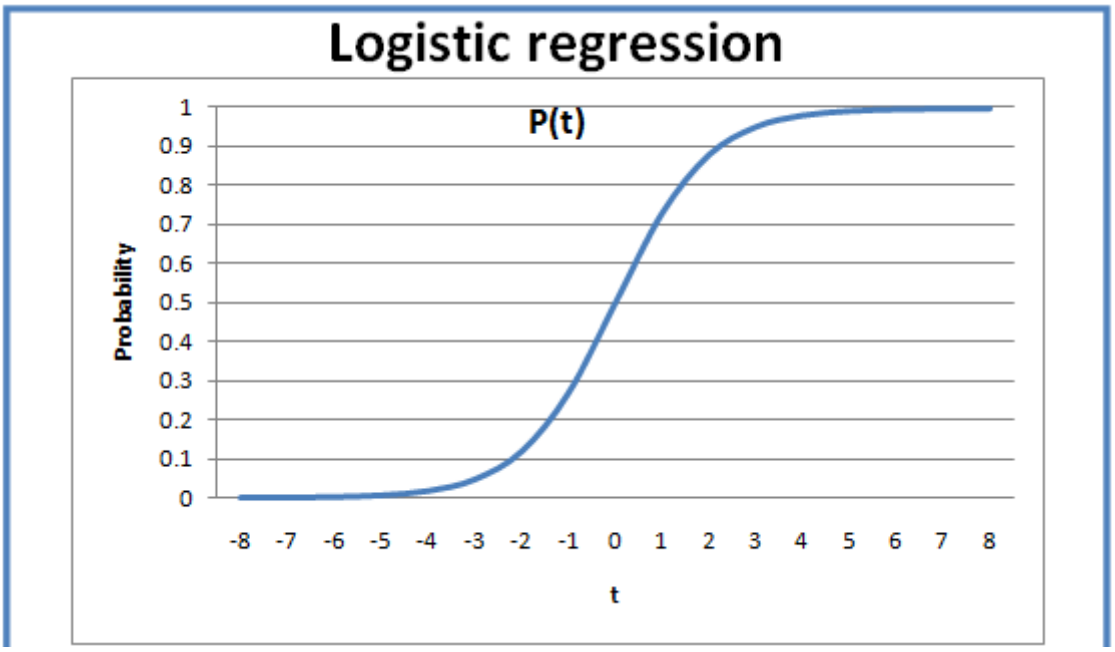
Our problem belongs to the classification category.

1. **Logistic Regression:**
   Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable (or output), y, can take only discrete values for given set of features (or inputs), X. The model builds a regression model to predict the probability that a given data entry belongs to the category numbered as “1”. Logistic regression models the data using the sigmoid function.

$$g(z) = \frac{1}{1+e^{-z}}$$

Logistic regression becomes a classification technique only when a decision threshold is brought into the picture. The setting of the threshold value is a very important aspect of Logistic regression and is dependent on the classification problem itself. For our example, we have set the appropriate thresholds in the criteria section.

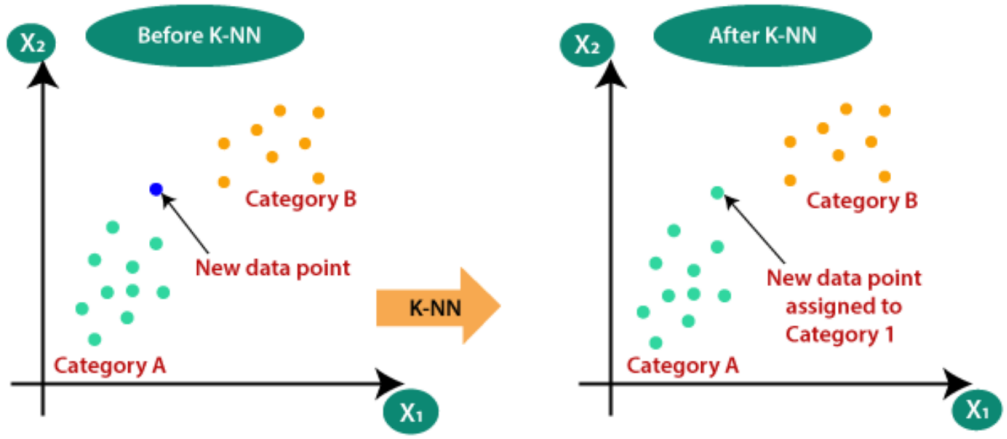
Binomial Logistic Regression: target variable can have only 2 possible types: "0" or "1" which may represent "win" vs. "loss", "pass" vs. "fail", "dead" vs. "alive", etc.



2. **K- Nearest Neighbours (KNN) Model:**
KNN is supervised machine learning algorithm which can be used for both classification and regression problems. In the case of classification K - Nearest Neighbour can be used for both binary and multi-class classifications. K-nearest neighbour or K-NN algorithm basically creates an imaginary boundary to classify the data. When new data points come in, the algorithm will try to predict that to the nearest of the boundary line. Therefore, larger k value means smother curves of separation resulting in less complex models. Whereas, smaller k value tends to over fit the data and resulting in complex models. Note: It's very important to have the right k-value when analysing the dataset to avoid over-fitting and under-fitting of the dataset. Using the k-nearest neighbour algorithm we fit the historical data (or train the model) and predict the future.

Why do we need KNN algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset.



To implement KNN algorithm you need to follow following steps.

Step-1: Select the number K of the neighbours

Step-2: Calculate the Euclidean distance of K number of neighbours

Step-3: Take the K nearest neighbours as per the calculated Euclidean distance.

Step-4: Among these k neighbours, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbour is maximum.

Step-6: Our model is ready.

3. Naïve Bayes Classifier

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

Naïve: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet

fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.

Bayes: It is called Bayes because it depends on the principle of Bayes Theorem.

Bayes' Theorem:

Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

The formula for Bayes' theorem is given as:

$$P(A|B)= \frac{P(B|A)P(A)}{P(B)}$$

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability**:** Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability**:** Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability**:** Probability of Evidence.

Evaluation of Models using Confusion matrix:

A confusion matrix is table which is used in every classification problem to describe the performance of a model on a test data.

Confusion Matrix gives a comparison between Actual and predicted values.

The confusion matrix is a N x N matrix, where N is the number of classes or outputs.

For 2 class, we get 2 x 2 confusion matrix.

For 3 class, we get 3 X 3 confusion matrix.

Confusion Matrix has 4 terms to understand True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN).

To evaluate our model we use the confusion matrix

	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

The accuracy is given by: **overall accuracy = (TP+TN)/ (TP+TN+FP+FN)**

CODE:

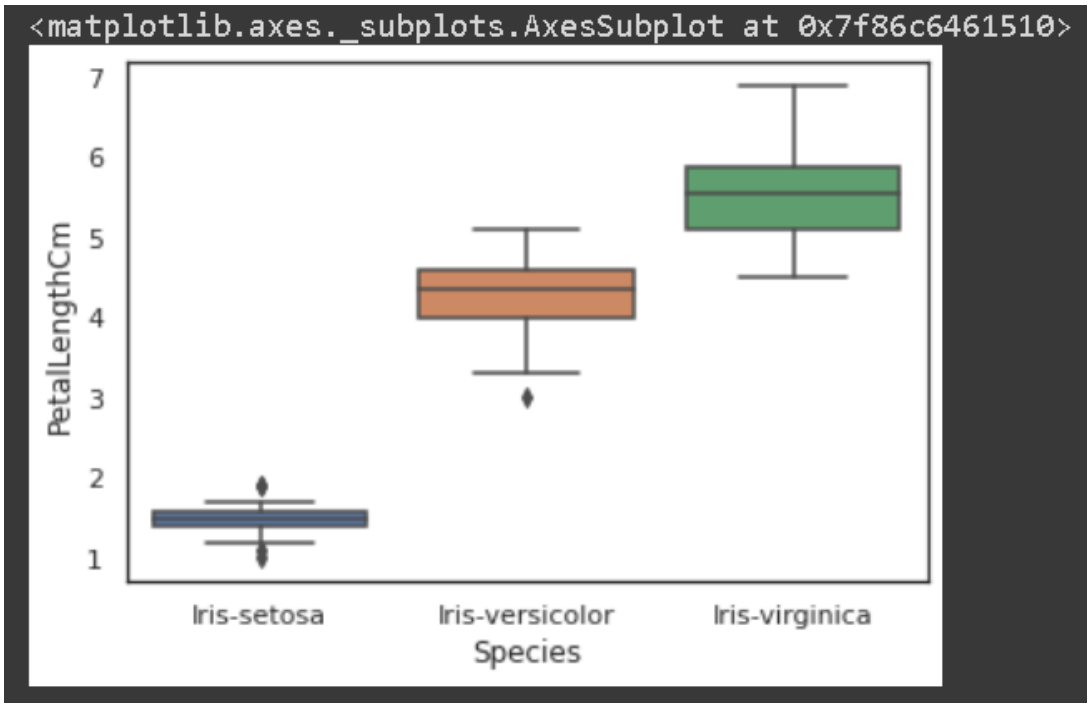
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="white", color_codes=True)

iris = pd.read_csv("Iris.csv")
iris.head(10)
iris.describe()

# We can look at an individual feature in Seaborn through a boxplot
sns.boxplot(x="Species", y="PetalLengthCm", data=iris)

# From the pairplot, we'll see that the Iris-
setosa species is separataed from the other
# two across all feature combinations
sns.pairplot(iris.drop("Id", axis=1), hue="Species", height=2)

# Andrews Curves involve using attributes of samples as coefficients fo
r Fourier series
# and then plotting these
from pandas.plotting import andrews_curves
andrews_curves(iris.drop("Id", axis=1), "Species")


from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split

# Seperating the data into dependent and independent variables
X = iris.iloc[:, :-1].values
y = iris.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0
.2, random_state = 0)

# LogisticRegression
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(max_iter=1000)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

# Summary of the predictions made by the classifier
# print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
# Accuracy score
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(y_pred,y_test))


# K-Nearest Neighbours
from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n_neighbors=8)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

# Summary of the predictions made by the classifier
# print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
# Accuracy score
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(y_pred,y_test))


# Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

# Summary of the predictions made by the classifier
# print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
# Accuracy score
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(y_pred,y_test))
```

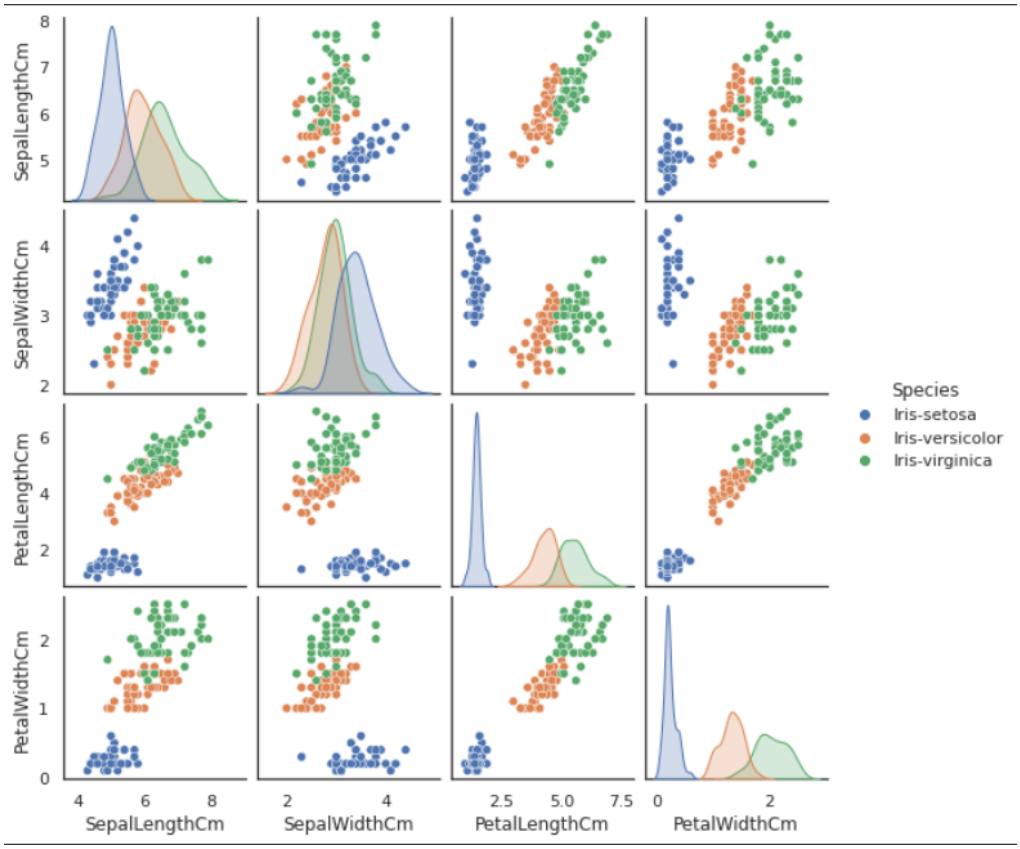
RESULT:
Description of Data

	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

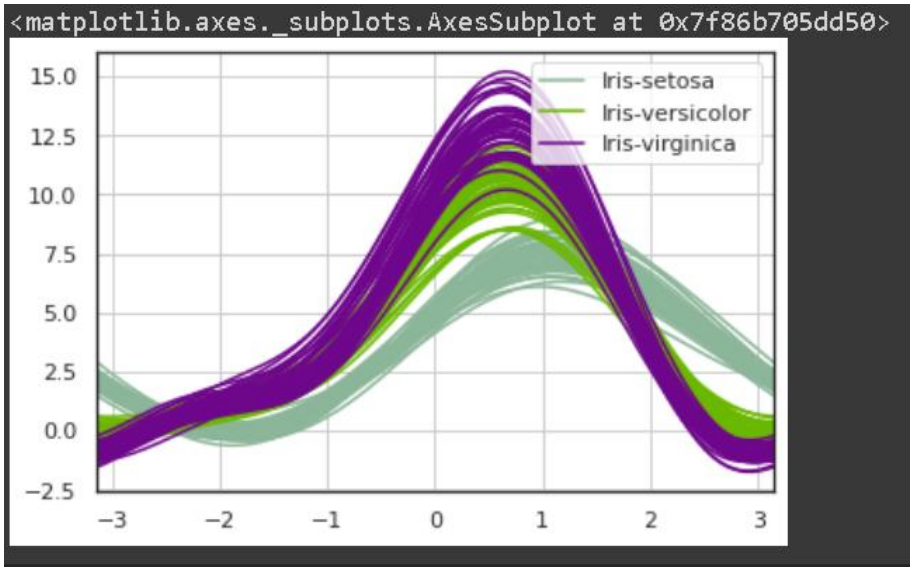
The box plot of the dataset, which shows us the visual representation of how our data is scattered over the plane. Box plot is a percentile-based graph, which divides the data into four quartiles of 25% each. This method is used in statistical analysis to understand various measures such as mean, median and deviation.



Another useful plot is a hybrid plot called pairplot, which shows the bivariate relation between each pair of features.



More sophisticated technique pandas has available is called Andrews Curves



Results of using the machine learning techniques.

1. Logistic Regression

```
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
accuracy is 1.0
```

2. KNN algorithm

```
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
accuracy is 1.0
```

3. Naïve Bayes Classifier

```
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
accuracy is 1.0
```

CONCLUSION:

1. Different visualisation techniques used in this program help to analyse the data and can be used for feature engineering if needed.
2. The accuracy of all the algorithm is 100% because of the small size of the data and since there is no learning involved, the algorithms can easily classify the test data set into respective classes.