# INTRODUCTION

This is the machine learning assignment; in this assignment we have to do a hands-on project related to machine learning techniques on real-world data using python.

Machine learning is the subfield of computer science which is mostly related to the data. Manually organizing and extracting information from the large amount of data is impossible. On the other hand, based on this large amount of data we can’t predict the unseen data’s result. So, classification technique in machine learning helps to predict with high accuracy. The classification methods we have studied in the module are polynomial regression, splines, tree-based methods, and deep learning neural network.

We have selected three machine learning model i.e., Support Vector Machine-SVM, K-Nearest Neighbor-KNN and logistic regression models to compare their accuracy on the same data set.

The dataset we chose was the IRIS- dataset, which is a multivariable dataset of three types of Iris flower. The data-set has 50 samples of data for each species and for every type there is a data of sepal length, width and petal length and width

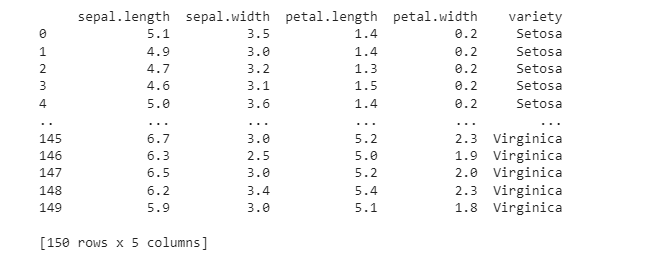


Figure 1: IRIS dataset inside.

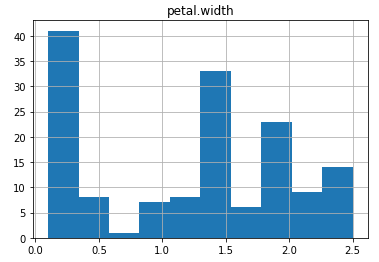
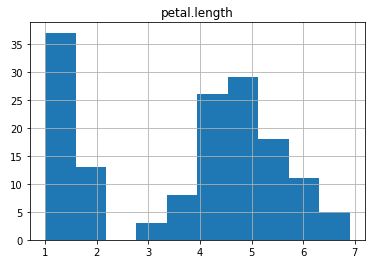
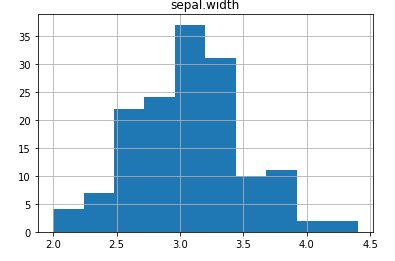
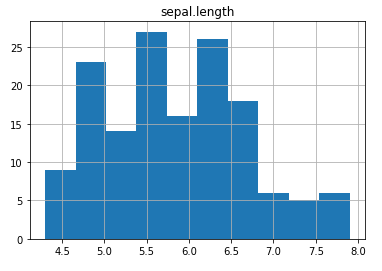


Figure 2: Dataset Visualization

# analysis and modeling

we have chosen different machine learning-ML models (SVM, naïve bayes, logistic regression, KNN, random Forest and decision tress) to compare their accuracy and identify which classification models has best accuracy.

1. **SVM:**

SVM stands for “support vector machine”. It is the machine learning model which needs data-set with labels, and it can be applied for two purposes: classification and regression problems. However, this model is helpful mostly for the classification problems.

Every point in a dataset in this model is drawn on n-dimensional plan (n denotes the number of features), each value represents the coordinate value. Finally, the hyper-planes are generated to separate the dataset.



Figure : SVM hyper-plan

1. **Naïve bayes:**

The nave bayes algorithm is composed of two words: nave and bayes, which are defined as follows:

a). **Naïve:** it is termed as Nave, because the features of one element is not related to the features of another element. For example, the red, spherical and sweet taste of a fruit is identified as apple due to its shape, color and taste. Therefore, according to the above features, every fruit having the same features is identified as apple.

b). **Bayes:** It is called Bayes because it depends on the principle of Bayes’ Theorem.

Bayes’ theorem is the formula for calculating the hypothesis Bayes' theorem, often known as Bayes' rule, is a mathematical formula for calculating the probability of a hypothesis based on past knowledge.

The formula for Bayes’ theorem is given as:

Naïve Bayes Classifier Algorithm

Due to the nature of naïve bayes, it assumes the all the features are independent, so can’t find the relation between the features.

1. **Logistic regression:**

Logistic regression is again supervised machine learning algorithm, which is able to predict the target similarity. The target variable varies between the o and 1 due to the nature of the logistic regression. It is the simplest machine learning model which is used to classification problems.

Logistic regression is used to classify the observation depending on the available dataset and it can quickly classify variable efficiently. The logistic function is depicted in the diagram below.



Figure : Logistic Regression

1. **KNN:**

K-Nearest Neighbor is the most basic supervised learning machine learning model. The K-NN model generate similar group of data together, and whenever new data is predicted then the model checks whether the new data is grouped to which group of data, that is most similar to the existing categories.

The K-NN approach saves all available data and categorizes new data points depending on how similar they are to the current data. This means that utilising the K-NN approach, fresh data can be swiftly sorted into a suitable category.

The K-NN technique can be used for both regression and classification, while classification is the most popular application.



Figure : KNN

1. **Decision tree:**

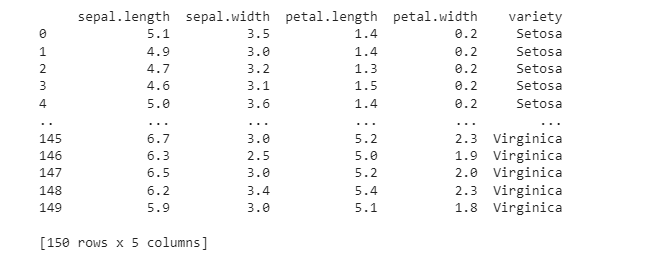
The decision tree is a supervised learning technique that may be applied to both classification and regression problems, however it is most commonly employed to tackle classification concerns. The "decision tree classifier" is a tree-structured classifier with core nodes corresponding to dataset attributes, branches corresponding to decision rules, and leaf nodes indicating outcomes.



Figure : Decision tree model

**LOADING:**

We have IRIS dataset having 150 samples of different species of iris flower that is setosa, versicolor, virginica. Then downloaded dataset is loaded to python environment using pandas library. We used “pd.read\_csv()” function to import from computer to python environment. The dataset looks like the figure,



This dataset is divided into train and test dataset. Test data size is set of 30% and remaining 70% data is kept for training the model.

We used 5 models to train and then test for our model evaluation purpose, we used sci-kit learn library for model creation.

The accuracy of **SVM** model is 100%

The accuracy of **Naïve** **Bayes** is also 100%

The accuracy of **logistic** **regression** is 97.3%

The accuracy of **KNN** is also 97.3%

The accuracy of **decision** **tress** is 94%

# discussion and future work

In this assignment we have applied 5 different machine learning model on a same dataset to evaluate their accuracy.

We used SVM, Naïve Bayes, Logistic regression, KNN and decision trees algorithms and have compare the accuracy results. We have found that SVM classification is more effective than others. The table shows the accuracy table of different machine learning models.

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Machine learning model** | **Accuracy** |
| **1** | SVM | 100% |
| **2** | Naïve Bayes | 100% |
| **3** | Logistic Regression | 97.3% |
| **4** | KNN | 97.3% |
| **5** | Decision trees | 94% |

# exective summary

To handle a large number of data and to predict according to the available data, it is very difficult and impossible to predict and summarize the data. To easily handle and predict the data, we have different machine learning models to predict the unknown data.

Basically, machine learning model is like a black box, initially we have to train this box according to the available data, once the box is trained, and this box totally recognize all the data and their relation, then this box can be used to predict unknow data to predict the result.

There are different types of machine learning model/ black boxes, to train and to predict values. We used five different machine learning model, trained them with a single dataset, and found their accuracy.

Then all the models accuracy is compared to find out which model is having higher accuracy, and how can we improve the model accuracy further.

# appndix

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from warnings import filterwarnings

filterwarnings(action='ignore')

iris=pd.read\_csv("/content/iris.csv")

print(iris)

print(iris.shape)

print(iris.describe())

#Checking for null values

print(iris.isna().sum())

print(iris.describe())

iris.head(150)

fig = plt.figure()

ax = fig.add\_axes([0,0,1,1])

ax.axis('equal')

l = ['Versicolor', 'Setosa', 'Virginica']

s = [50,50,50]

ax.pie(s, labels = l,autopct='%1.2f%%')

plt.show()

iris.hist('sepal.length')

iris.hist('sepal.width')

iris.hist('petal.length')

iris.hist('petal.width')

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn import svm

from sklearn import metrics

from sklearn.tree import DecisionTreeClassifier

train, test = train\_test\_split(iris, test\_size = 0.25)

print(train.shape)

print(test.shape)

train\_X = train[['sepal.length', 'sepal.length', 'petal.length','petal.width']]

train\_y = train.variety

test\_X = test[['sepal.length', 'sepal.length', 'petal.length','petal.width']]

test\_y = test.variety

train\_X.head()

#Using LogisticRegression

model = LogisticRegression()

model.fit(train\_X, train\_y)

prediction = model.predict(test\_X)

accuracy1=metrics.accuracy\_score(prediction,test\_y)

print(accuracy1)

print('Accuracy:',metrics.accuracy\_score(prediction,test\_y))

#Confusion matrix

from sklearn.metrics import confusion\_matrix,classification\_report

confusion\_mat = confusion\_matrix(test\_y,prediction)

print("Confusion matrix: \n",confusion\_mat)

print(classification\_report(test\_y,prediction))

#Using Support Vector

from sklearn.svm import SVC

model1 = SVC()

model1.fit(train\_X,train\_y)

pred\_y = model1.predict(test\_X)

from sklearn.metrics import accuracy\_score

accuracy2=accuracy\_score(test\_y,pred\_y)

print (accuracy2)

print("Acc=",accuracy\_score(test\_y,pred\_y))

#Using KNN Neighbors

from sklearn.neighbors import KNeighborsClassifier

model2 = KNeighborsClassifier(n\_neighbors=5)

model2.fit(train\_X,train\_y)

y\_pred2 = model2.predict(test\_X)

from sklearn.metrics import accuracy\_score

accuracy3=accuracy\_score(test\_y,y\_pred2)

print(accuracy3)

print("Accuracy Score:",accuracy\_score(test\_y,y\_pred2))

#Using GaussianNB

from sklearn.naive\_bayes import GaussianNB

model3 = GaussianNB()

model3.fit(train\_X,train\_y)

y\_pred3 = model3.predict(test\_X)

from sklearn.metrics import accuracy\_score

accuracy4=accuracy\_score(test\_y,y\_pred3)

print(accuracy4)

print("Accuracy Score:",accuracy\_score(test\_y,y\_pred3))

#Using Decision Tree

from sklearn.tree import DecisionTreeClassifier

model4 = DecisionTreeClassifier(criterion='entropy',random\_state=7)

model4.fit(train\_X,train\_y)

y\_pred4 = model4.predict(test\_X)

from sklearn.metrics import accuracy\_score

accuracy5=accuracy\_score(test\_y,y\_pred4)

print(accuracy5)

print("Accuracy Score:",accuracy\_score(test\_y,y\_pred4))

#USILNG random forest classifier

from sklearn.ensemble import RandomForestClassifier

clfr= RandomForestClassifier(random\_state = 100)

# Performing training

clfr.fit(train\_X, train\_y)

#making prediction

Y\_pred=clfr.predict(test\_X)

accuracy6= metrics.accuracy\_score(test\_y, Y\_pred)

print("Accuracy:",metrics.accuracy\_score(test\_y, Y\_pred))

results = pd.DataFrame({'Model': ['Logistic Regression','Support Vector Machines', 'Naive Bayes','KNN' ,

'Decision Tree'],'Score': [accuracy1,accuracy2, accuracy3,

accuracy4, accuracy5]})

result\_df = results.sort\_values(by='Score', ascending=False)

result\_df = result\_df.set\_index('Score')

result\_df.head(9)