



جامعة بنها



كلية الحاسبات والذكاء الاصطناعي



# Orthopedic classification.

Machine Learning Project.

*Team members:*

**Sherif Ashraf Ahmed Roshdy.**

Abdelrahman Mohamed Khalil.

Mohamed Hosny Mosad.

Yahia Zakria Ebrahim.

Sobeh Salah Sobeh.



# CONTENTS

1

## Introduction.

Domain of problem.

Software Details and Project Objective.

2

## Dataset.

Information about dataset.

Exploratory Dataset.

3

## The Algorithms used to classification

Data preprocessing for machine learning.

Voting many classifier (KNN, Naive Bayes classifiers, Decision tree classifier) to make best decision.

4

## Conclusion.

the decision of solving problem.

Save model.

Load model and test it with example.

# Introduction

Domain of problem.

- Orthopedic relating to the branch of medicine dealing with the correction of deformities of bones or muscles.



# Introduction

Conn.

- If you suffer from pain in the bones, the bone examination will show whether this pain is normal or abnormal.
- This is important because, in the event of abnormal pain, you will have to go to a specialist. After all, it may be very dangerous, such as Joints Dislocation or Fractures.
- Bone, normal or abnormal, is an important tool for detecting cancer spread in the bones from the site of the original tumor, such as breast or prostate cancer,
- It helps in repairing and preventing deformities in children at an early stage, and these deformities result from some factors that may affect the body while it is still at the embryo stage.

# Introduction

Software Details and Project Objective.



## Software Details:

Python (libraries) >> Pandas,  
SKLearn, Seaborn, Matplotlib,  
numpy.

Dataset >> Kaggle.



## Project Objective

Used machine learning  
algorithms to classify a  
patient's condition as normal  
or abnormal based on various  
orthopedic parameters

# Dataset.

[Biomechanical features of orthopedic patients | Kaggle](#)



# Dataset.

Information about dataset

- Biomechanical features of orthopedic patients

 UCI MACHINE LEARNING · UPDATED 5 YEARS AGO

▲ 201


New Notebook

Download (24 kB)

⋮

## Biomechanical features of orthopedic patients

Classifying patients based on six features



# Dataset.

Conn.

- The task consists in classifying patients as belonging to one out of two categories: Normal or Abnormal.
- Activity Overview of dataset

## Activity Overview

### ACTIVITY STATS

VIEWS

**73157**

DOWNLOADS

**15737**

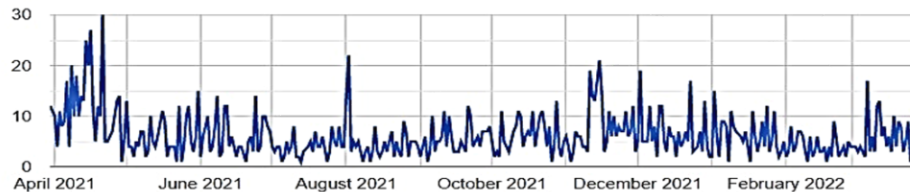
DOWNLOAD PER VIEW RATIO

**0.22**

TOTAL UNIQUE CONTRIBUTORS

**298**

Downloads ▾





# Dataset.

Conn.

Each patient is represented in the data set by six biomechanical attributes derived from the shape and orientation of the pelvis and lumbar spine (each one is a column)

Pelvic incidence حدوث الحوض	pelvic tilt إمالة الحوض	lumbar lordosis angle زاوية قعس قطني	sacral slope منحدر عجزي	pelvic radius نصف قطر الحوض	grade of spondylolisthesis درجة انزلاق الفقار
--------------------------------	----------------------------	---	----------------------------	--------------------------------	--

# Dataset.

Exploratory Dataset.

- Show the first 10 rows.

	pelvic_incidence	pelvic_tilt_numeric	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis	class
0	63.027817	22.552586	39.609117	40.475232	98.672917	-0.254400	Abnormal
1	39.056951	10.060991	25.015378	28.995960	114.405425	4.564259	Abnormal
2	68.832021	22.218482	50.092194	46.613539	105.985135	-3.530317	Abnormal
3	69.297008	24.652878	44.311238	44.644130	101.868495	11.211523	Abnormal
4	49.712859	9.652075	28.317406	40.060784	108.168725	7.918501	Abnormal
5	40.250200	13.921907	25.124950	26.328293	130.327871	2.230652	Abnormal
6	53.432928	15.864336	37.165934	37.568592	120.567523	5.988551	Abnormal
7	45.366754	10.755611	29.038349	34.611142	117.270067	-10.675871	Abnormal
8	43.790190	13.533753	42.690814	30.256437	125.002893	13.289018	Abnormal
9	36.686353	5.010884	41.948751	31.675469	84.241415	0.664437	Abnormal

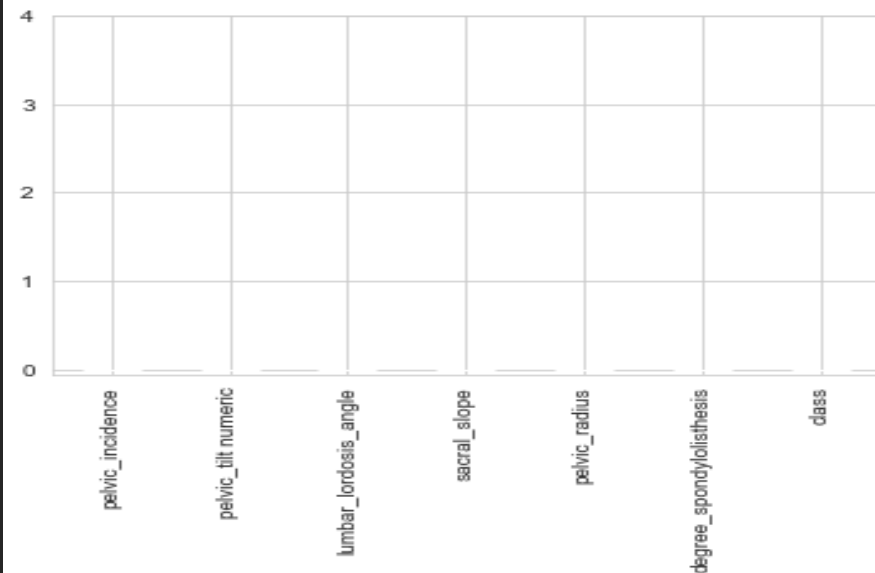
# Dataset.

Conn.

- Print a concise summary of a DataFrame.
- The shape >> (1180,7)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1180 entries, 0 to 1179
Data columns (total 7 columns):
#   Column                      Non-Null Count  Dtype
---  ---
0   pelvic_incidence             1180 non-null   float64
1   pelvic_tilt_numeric          1180 non-null   float64
2   lumbar_lordosis_angle        1180 non-null   float64
3   sacral_slope                 1180 non-null   float64
4   pelvic_radius                1180 non-null   float64
5   degree_spondylolisthesis     1180 non-null   float64
6   class                        1180 non-null   object
dtypes: float64(6), object(1)
memory usage: 64.7+ KB
```

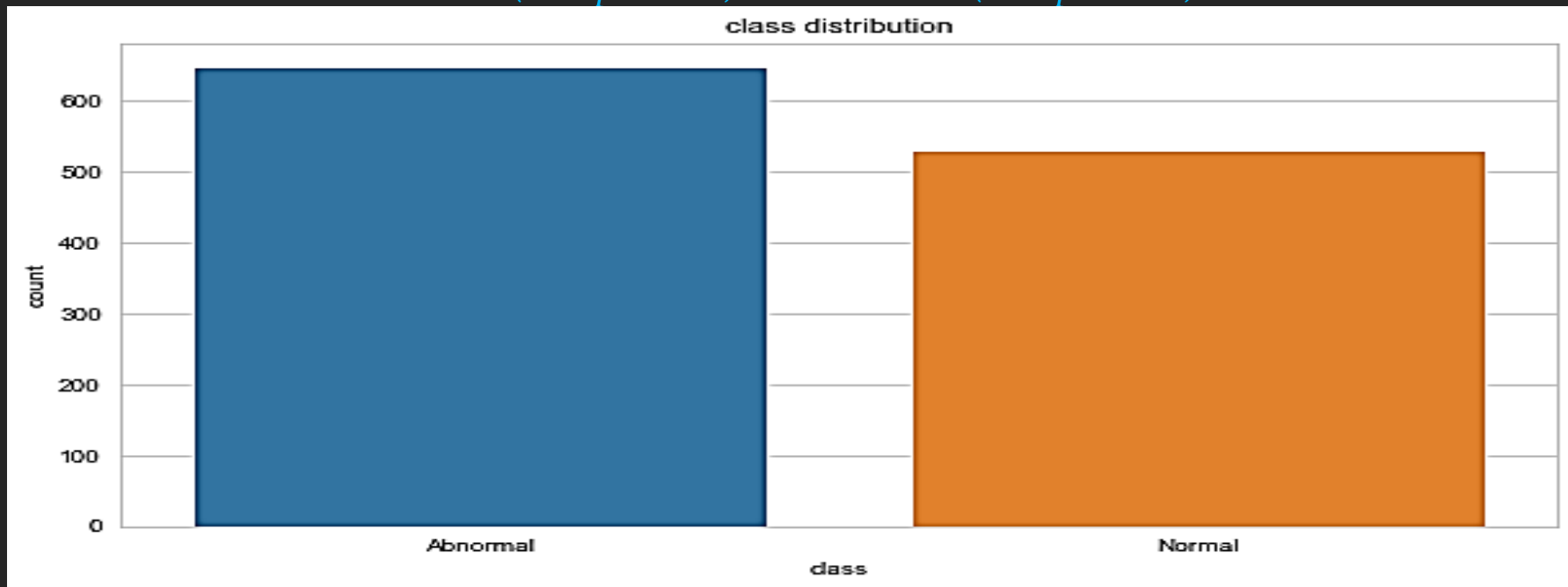
Check the non-null values



# Dataset.

Conn.

- Represent the number of output: (Checking for imbalanced output)  
*Normal (649 patients) or Abnormal (513 patients).*



# Dataset.

Conn.

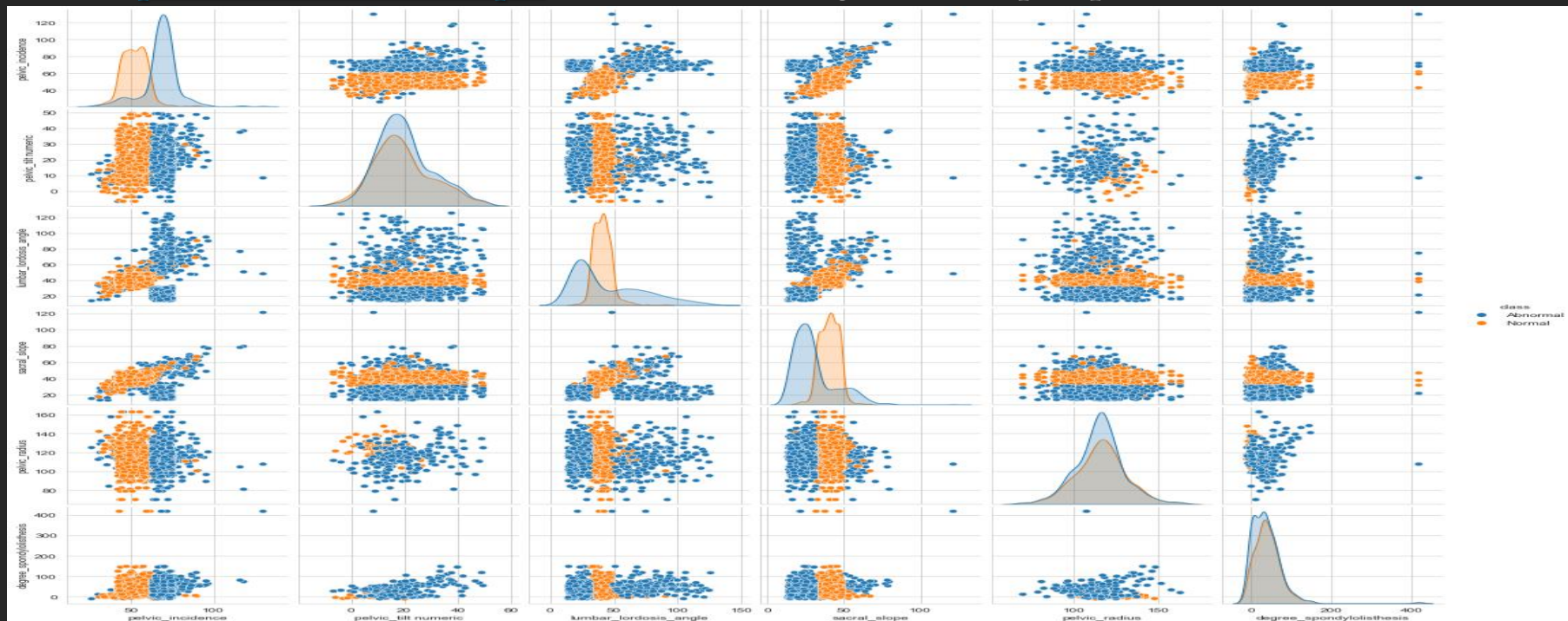
- Show [descriptive statistics](#) include those that summarize the central tendency, dispersion and shape of a dataset's distribution.

	pelvic_incidence	pelvic_tilt_numeric	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis
count	1180.000000	1180.000000	1180.000000	1180.000000	1180.000000	1180.000000
mean	60.628995	19.534941	42.966891	34.621027	115.654789	39.428121
std	12.212527	10.779453	21.270700	12.197810	14.427055	40.814087
min	26.147921	-6.554948	14.000000	13.366931	70.082575	-11.058179
25%	50.804924	12.537992	28.957819	25.065929	107.690466	11.400298
50%	61.542890	17.977784	38.926371	33.918037	116.250917	33.157646
75%	69.658921	24.822631	48.426306	43.163549	124.118877	55.995454
max	129.834041	49.431864	125.742385	121.429566	163.071041	418.543082

# Dataset.

Conn.

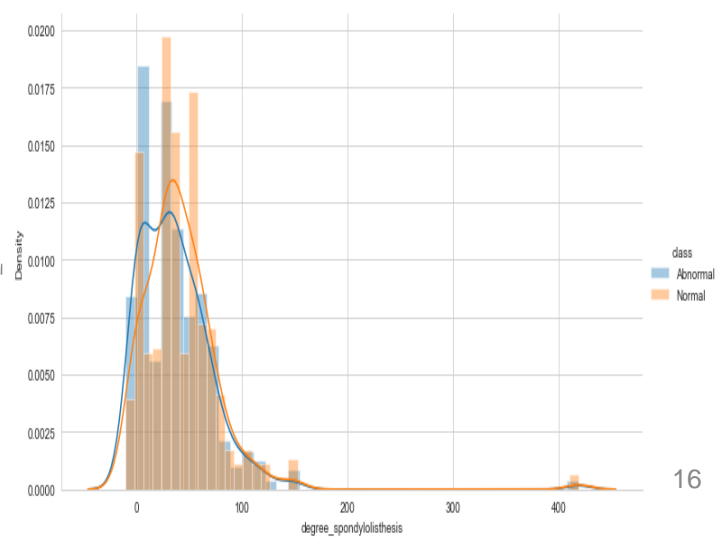
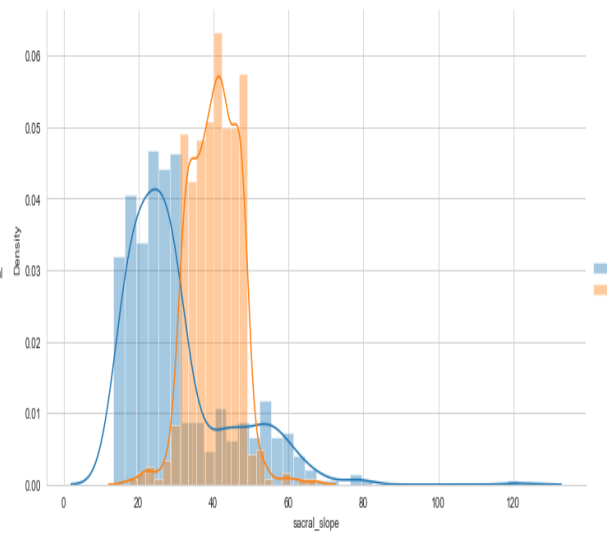
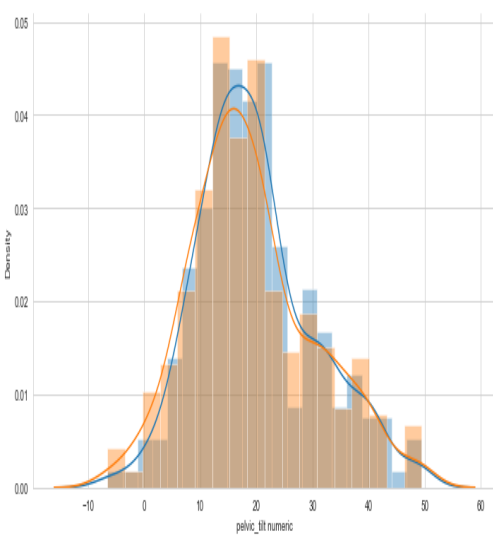
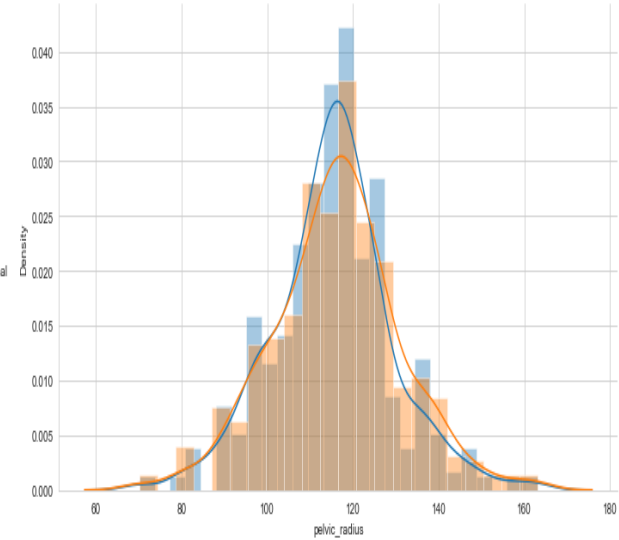
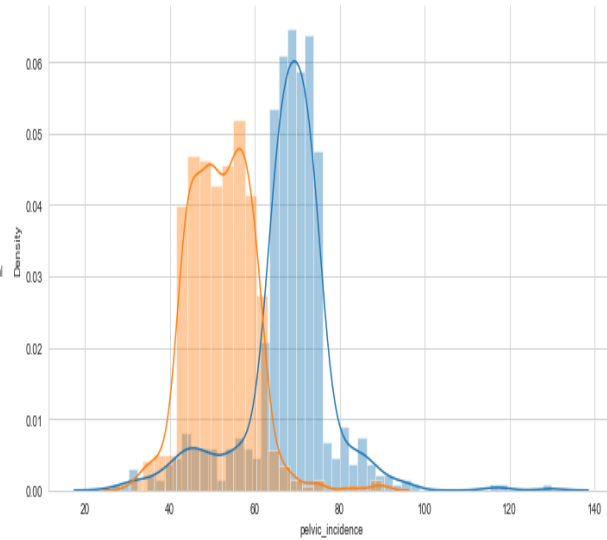
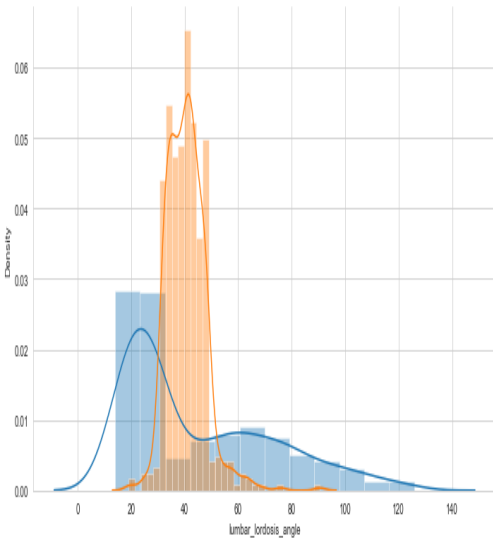
- Plot pairwise relationships in a dataset using `seaborn.pairplot` function.



# Dataset.

Conn.

- FacetGrid class helps in visualizing distribution of one variable as well as the relationship between multiple variables separately within subsets of your dataset using multiple panels.

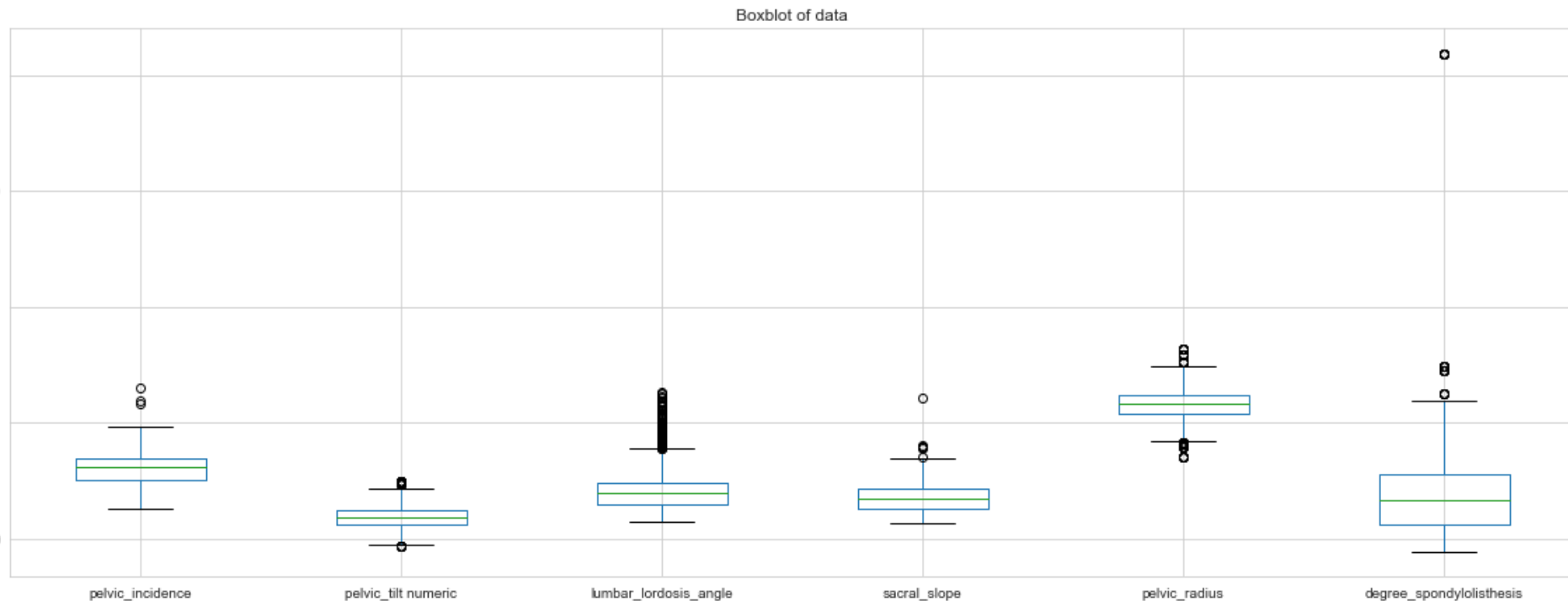




# Dataset.

Conn.

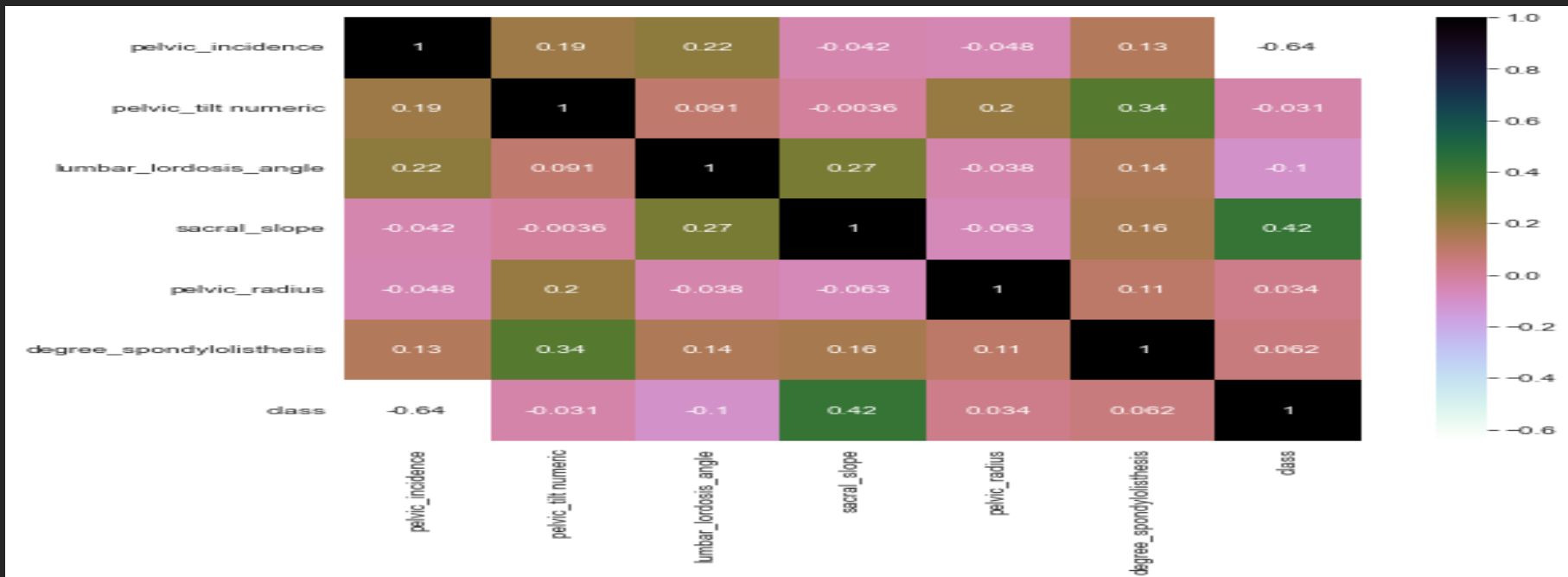
- Box plots are used to show distributions of numeric data values.



# Dataset.

Conn.

- Representing **the correlation** between features and some of them using heatmap function

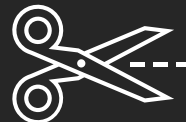
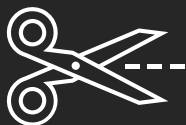
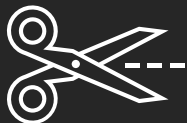


# Data preprocessing for machine learning

Splitting dataset

Input data.

	pelvic_incidence	pelvic_tilt_numeric	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis
0	63.027817	22.552586	39.609117	40.475232	98.672917	-0.254400
1	39.056951	10.060991	25.015378	28.995960	114.405425	4.564259
2	68.832021	22.218482	50.092194	46.613539	105.985135	-3.530317
3	69.297008	24.652878	44.311238	44.644130	101.868495	11.211523
4	49.712859	9.652075	28.317406	40.060784	108.168725	7.918501



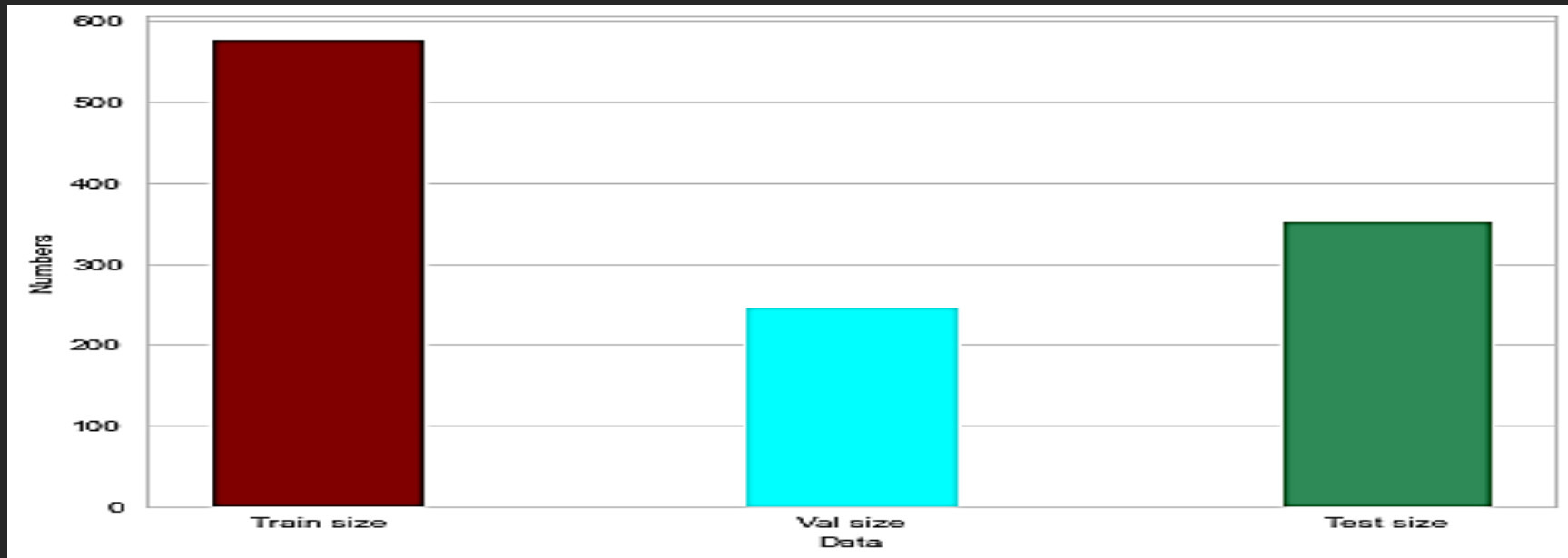
Output data

```
data_output.head()
0      0
1      0
2      0
3      0
4      0
Name: class, dtype: int64
```

# Data preprocessing for machine learning

## Splitting Datasets

- We need to split our dataset to 3 sets: Training, Validation, and Test data subsets.



```
X_train: (578, 6)  
y_train: (578,)
```



```
X_val: (248, 6)  
y_val: (248,)
```



```
X_test: (354, 6)  
y_test: (354,)
```

# The Algorithms used to Voting classification

(KNN, Naive Bayes, Decision Tree )



# The Algorithms used to classification

## Hyper-parameter tuning

- When we build a machine learning model ,we have a number of Hyper-parameters that will determine how our model looks like.
- Changing the values of these variables will affect the accuracy of our model.
- We need to find the best hyper-parameters for our model to achieve the best accuracy.



# KNN Algorithm.

# The Algorithms used to classification

KNN Algorithm.

Hyper-parameter in KNN



```
graph LR; A[Hyper-parameter in KNN] --- B[the number of neighbors]; A --- C[The weights]
```

A diagram illustrating the hyper-parameters of the KNN algorithm. A vertical blue box on the left is labeled "Hyper-parameter in KNN". A horizontal line extends from the right side of this box, which then splits into two horizontal branches. Each branch leads to a blue rectangular box. The top box contains the text "the number of neighbors" and the bottom box contains the text "The weights".

the number of neighbors

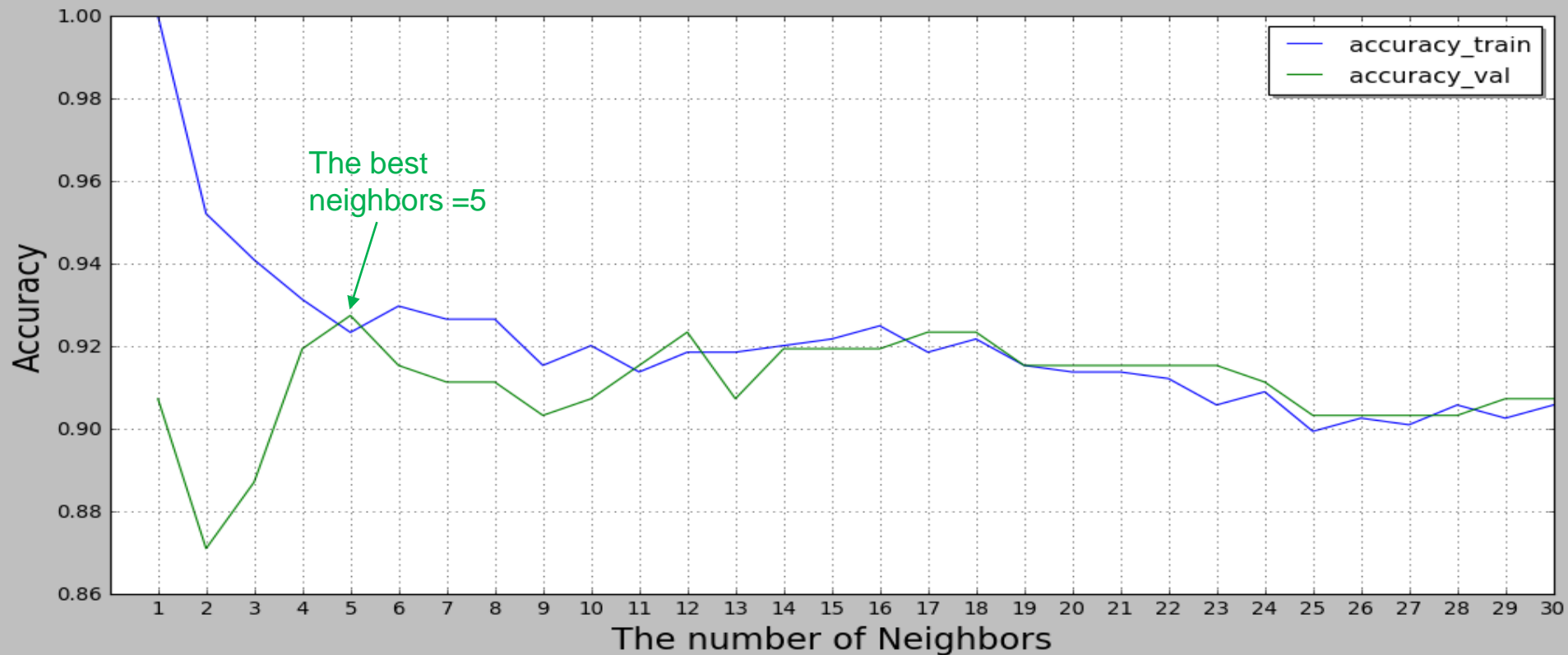
The weights



# The Algorithms used to classification

KNN Algorithm.

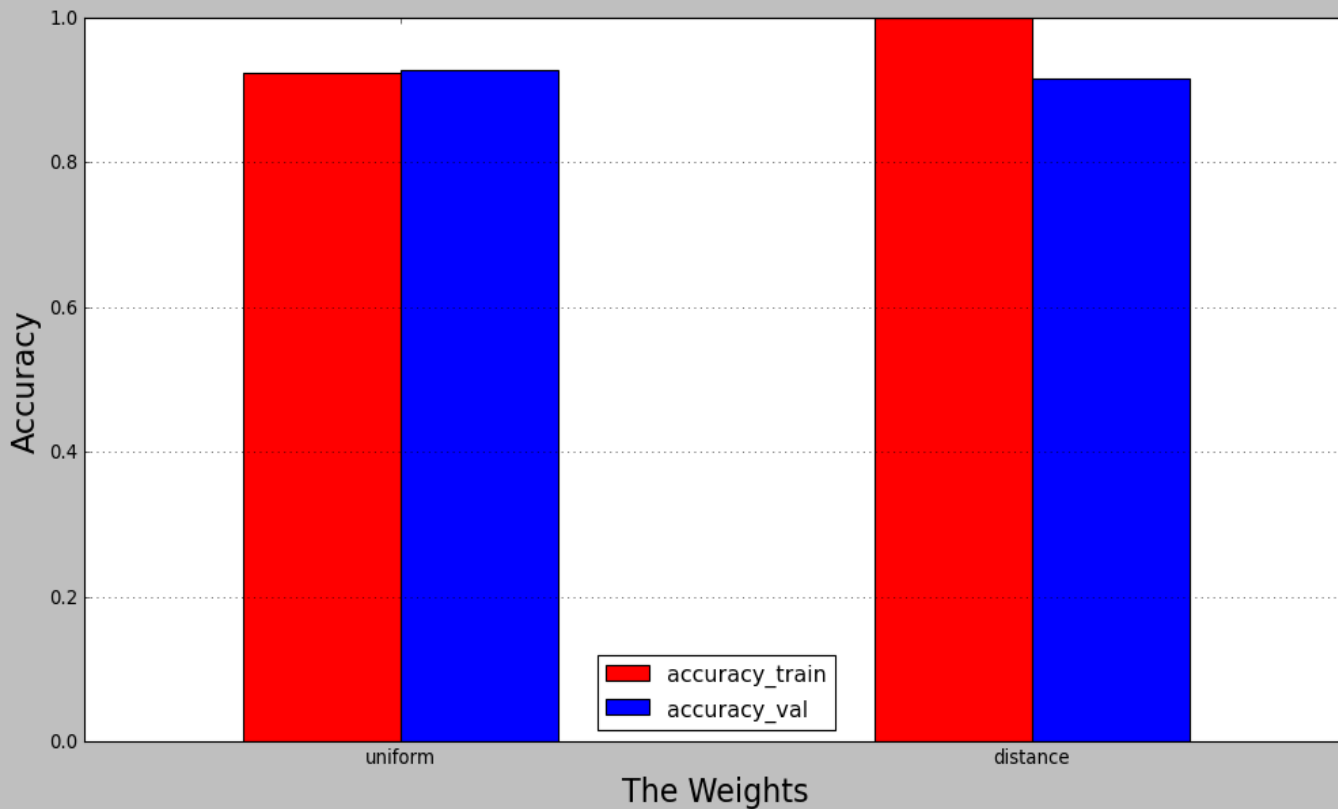
## Train "vs" Validation Accuracy of KNN



# The Algorithms used to classification

KNN Algorithm.

Test "vs" Validation Accuracy weight of KNN

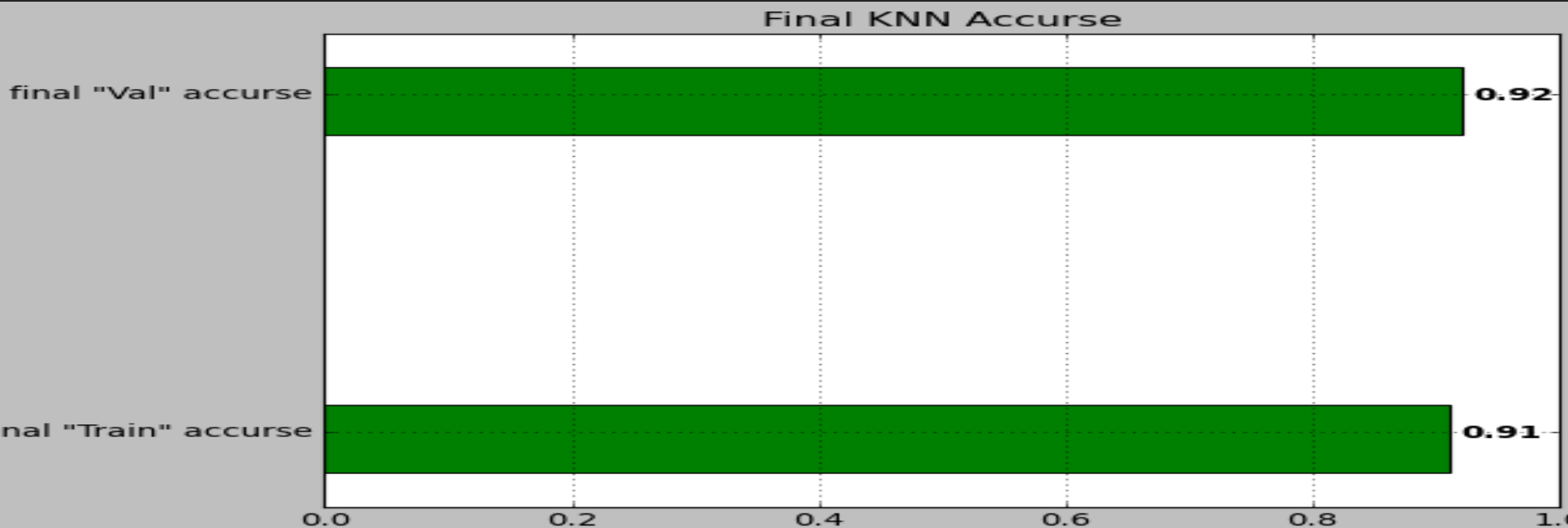


	weights	accuracy_train	accuracy_val
0	uniform	0.923323	0.927419
1	distance	1.000000	0.915323

# The Algorithms used to classification

KNN Algorithm.

- Then we applied the KNeighborsClassifier using  
the best  $K = (11)$  ,weight="uniform"



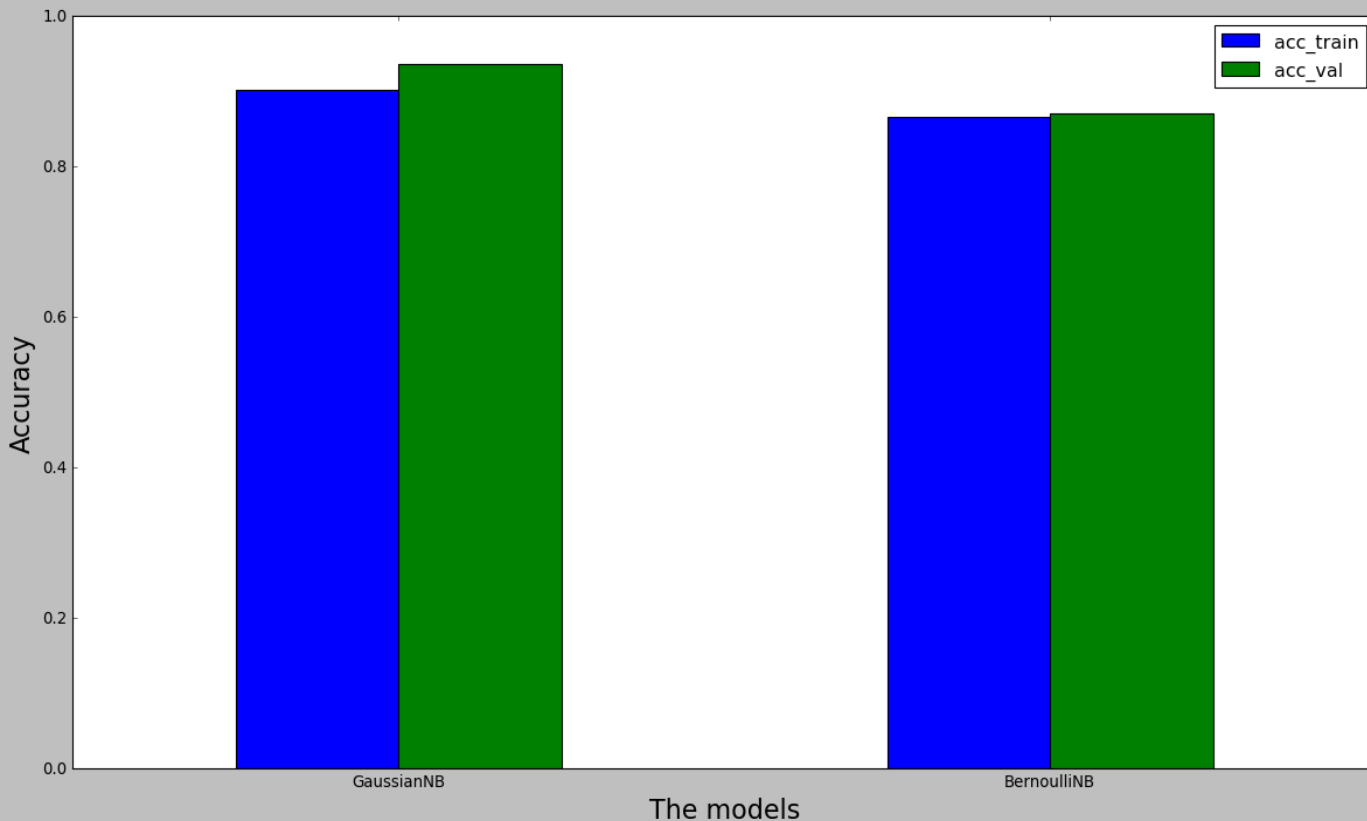


# Naive Bayes Algorithm.

# The Algorithms used to classification

## Naive Bayes Algorithm.

Test "vs" Validation Accuracy of NB



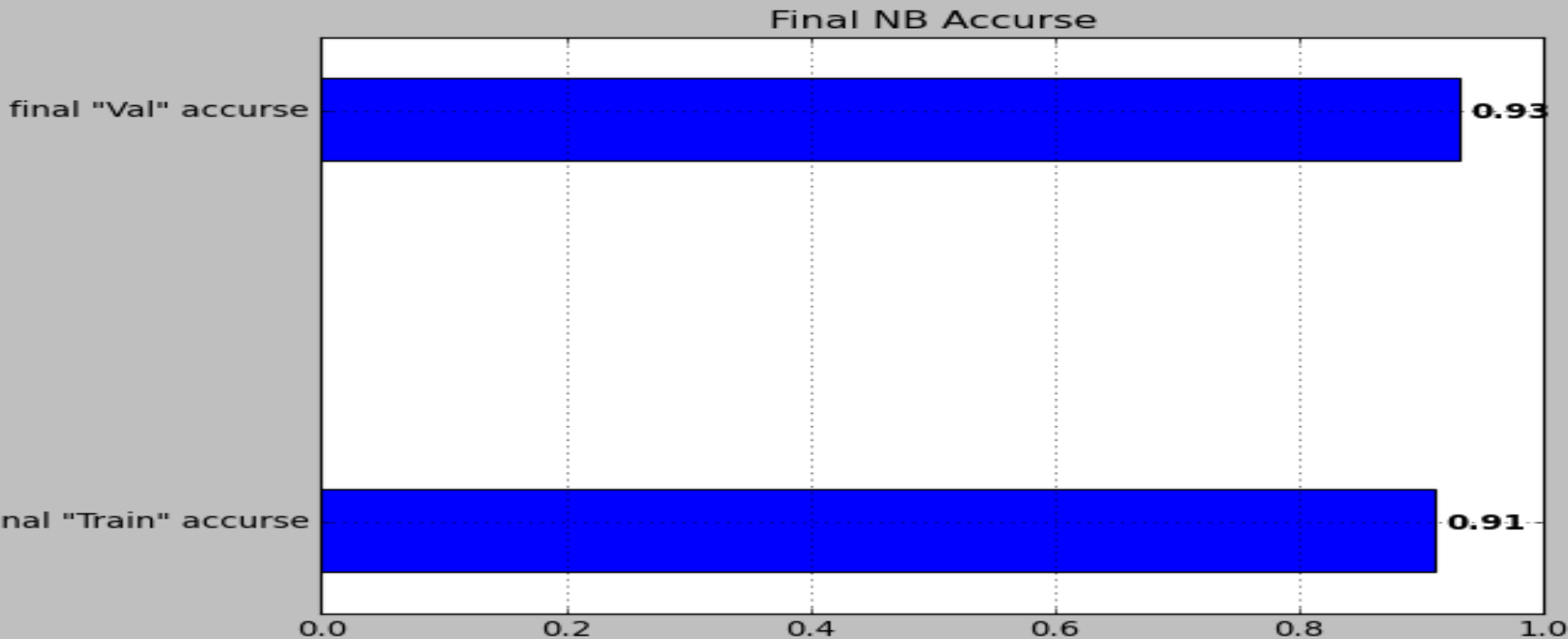
**GaussianNB  
model win**

	model	acc_train	acc_val
0	GaussianNB	0.901384	0.935484
1	BernoulliNB	0.865815	0.870968

# The Algorithms used to classification

Naive Bayes Algorithm.

- We applied `GaussianNB()` in the data





# Decision Tree Algorithm

# The Algorithms used to classification

DT Algorithm.

Hyper-parameter in DT

```
graph LR; A[Hyper-parameter in DT] --- B[The max depth of tree]; A --- C[The Criterion];
```

The diagram illustrates the hyper-parameters of a Decision Tree (DT) algorithm. A central vertical blue box labeled 'Hyper-parameter in DT' is connected by a horizontal line to a bracket. This bracket then branches into two horizontal blue boxes: 'The max depth of tree' and 'The Criterion'.

The max depth of tree

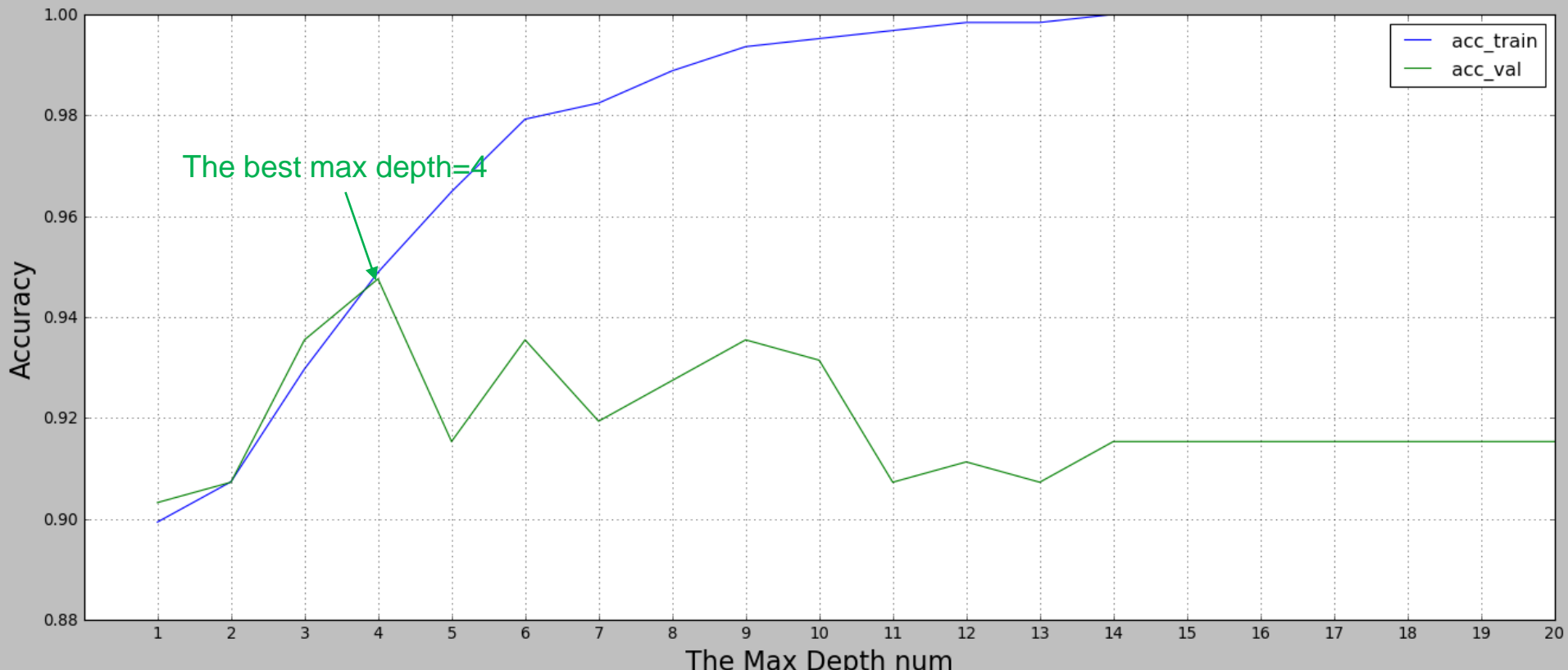
The Criterion



# The Algorithms used to classification

## Decision Tree Algorithm

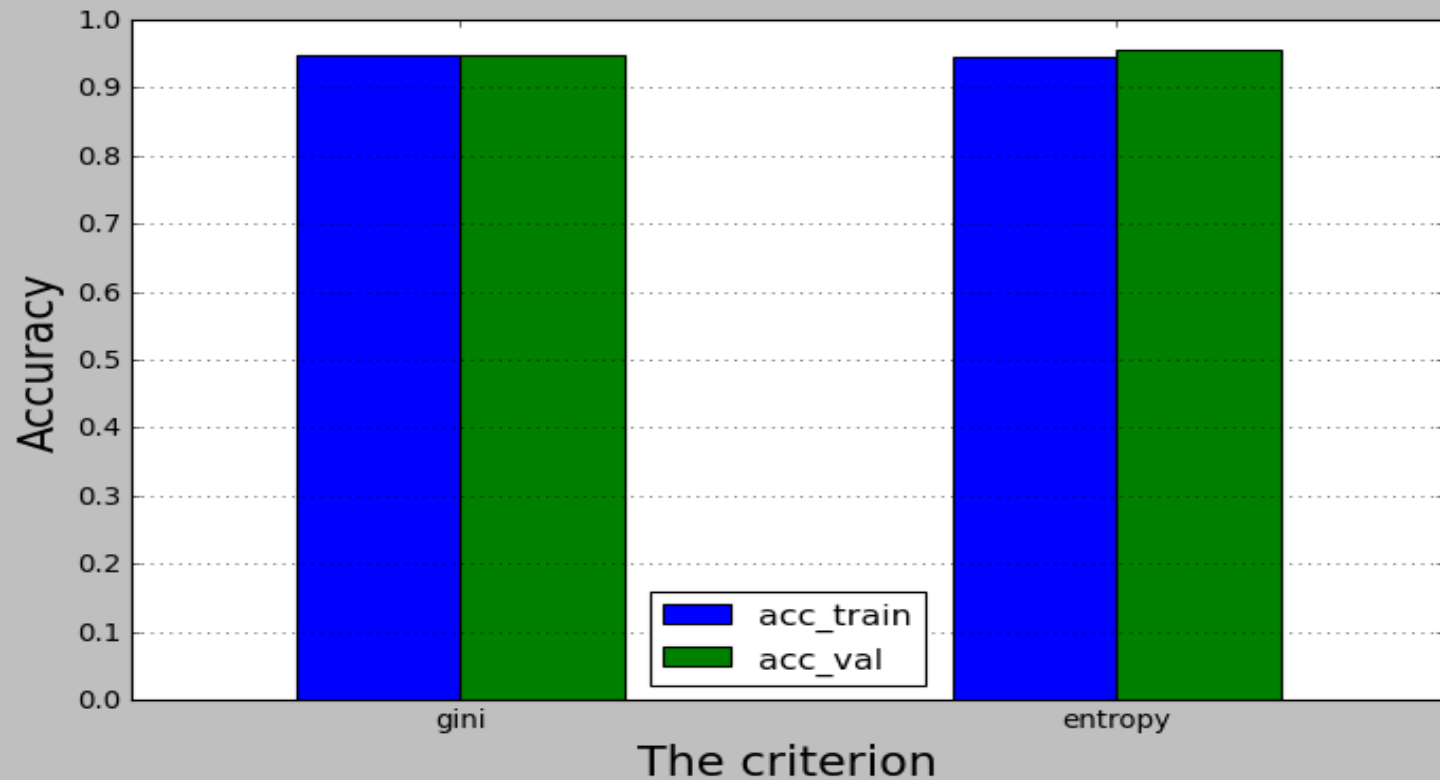
Train "vs" Validation Accuracy of DT "Max Depth"



# The Algorithms used to classification

## Decision Tree Algorithm

Test "vs" Validation Accuracy of DT

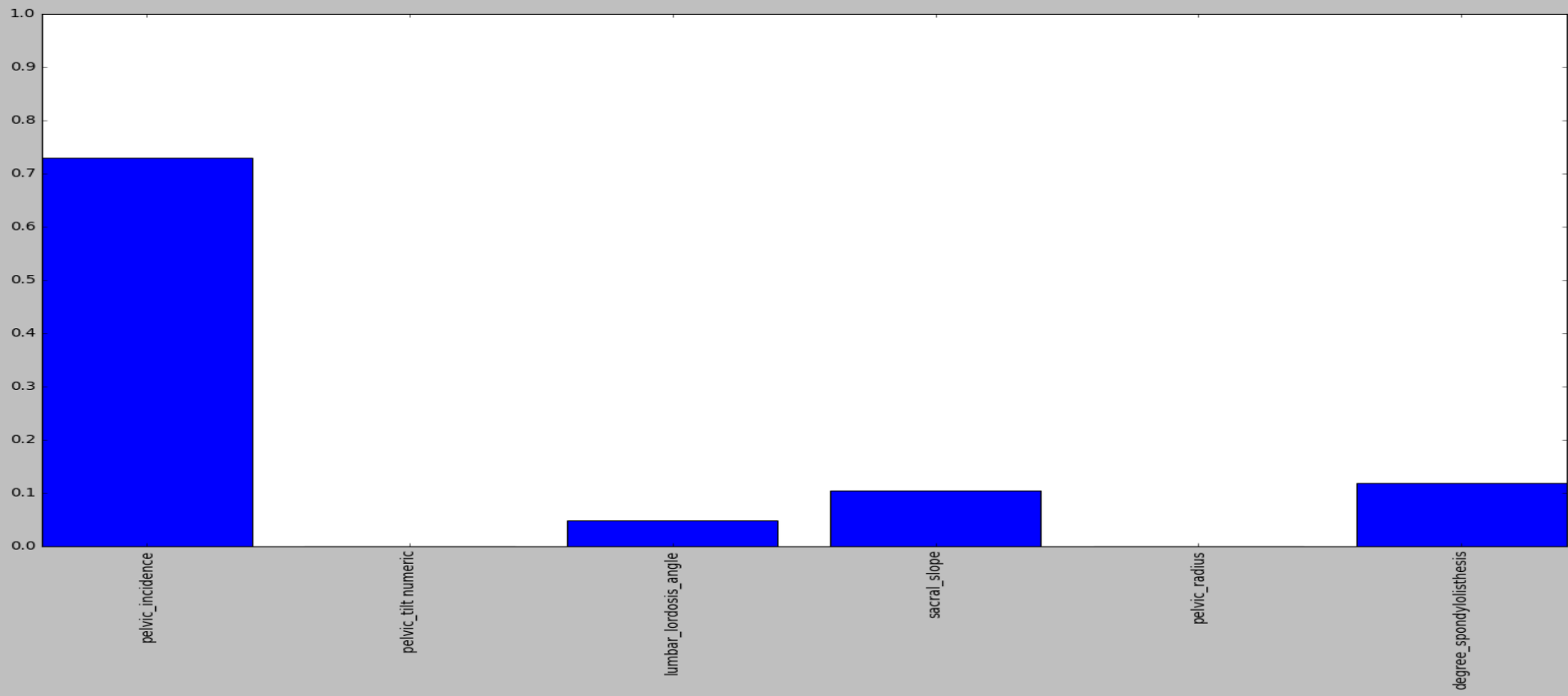


	criterion	acc_train	acc_val
0	gini	0.948882	0.947581
1	entropy	0.944089	0.955645

# The Algorithms used to classification

## Decision Tree Algorithm

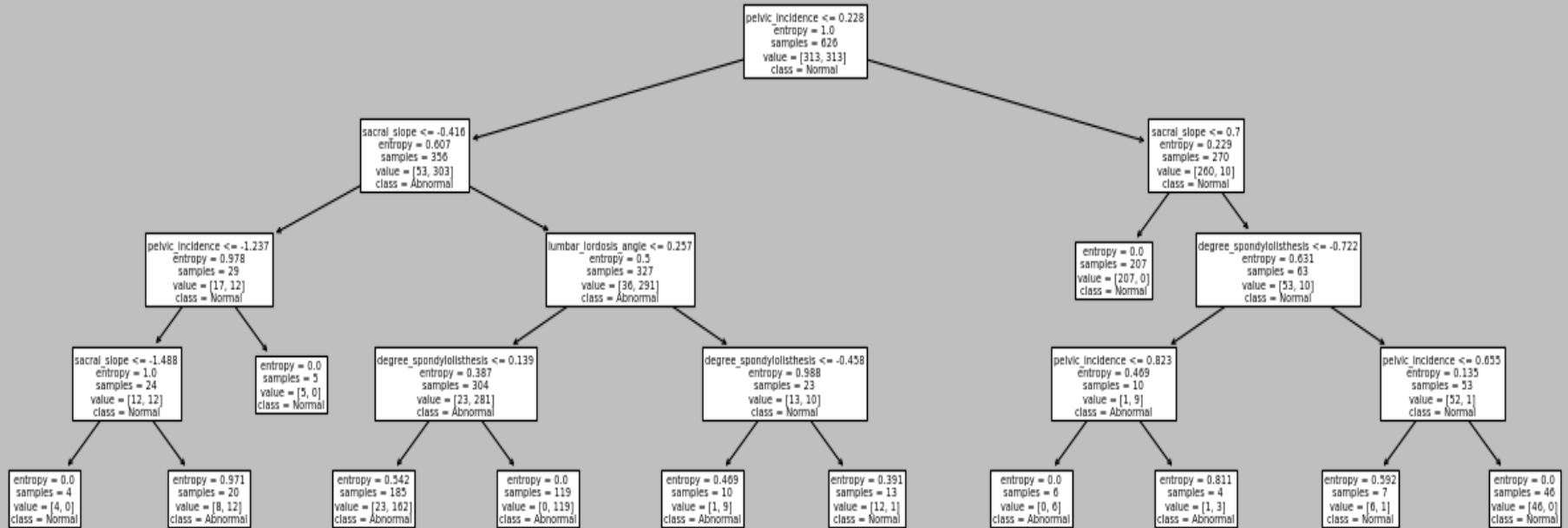
### Analysing feature importances



# The Algorithms used to classification

## Decision Tree Algorithm

## Visualizing the model

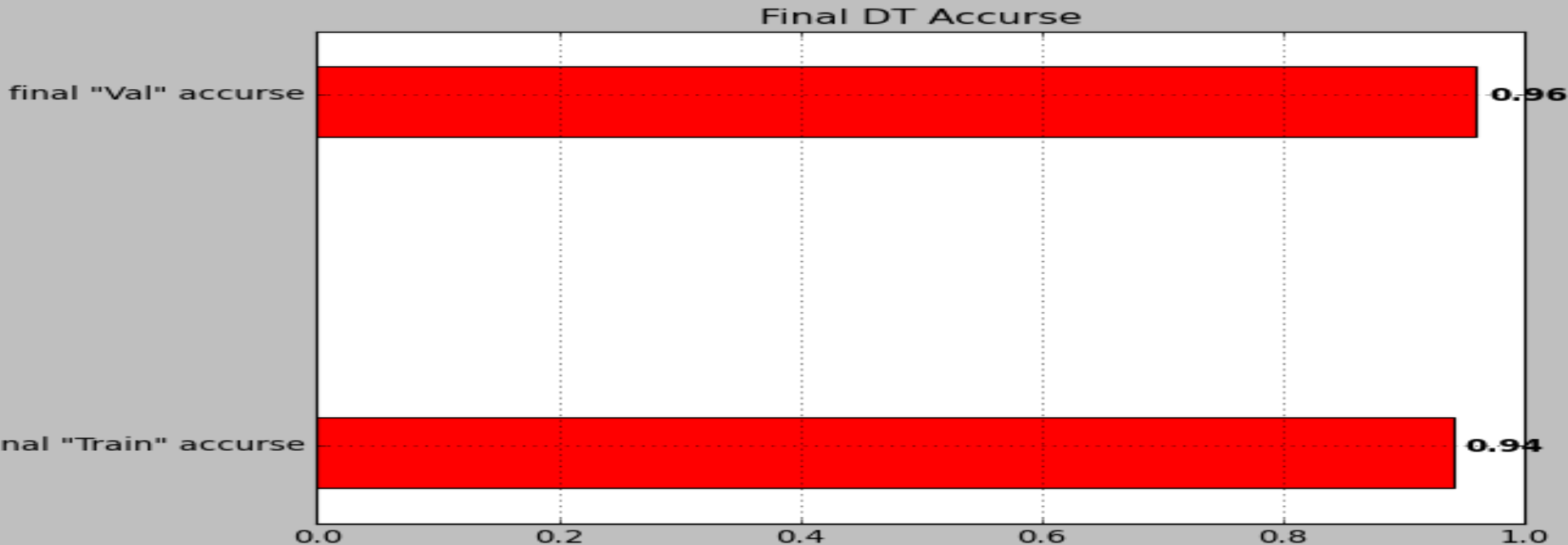


# The Algorithms used to classification

## Decision Tree Algorithm

- We applied the DecisionTreeClassifier

with max depth=4, criterion="entropy"

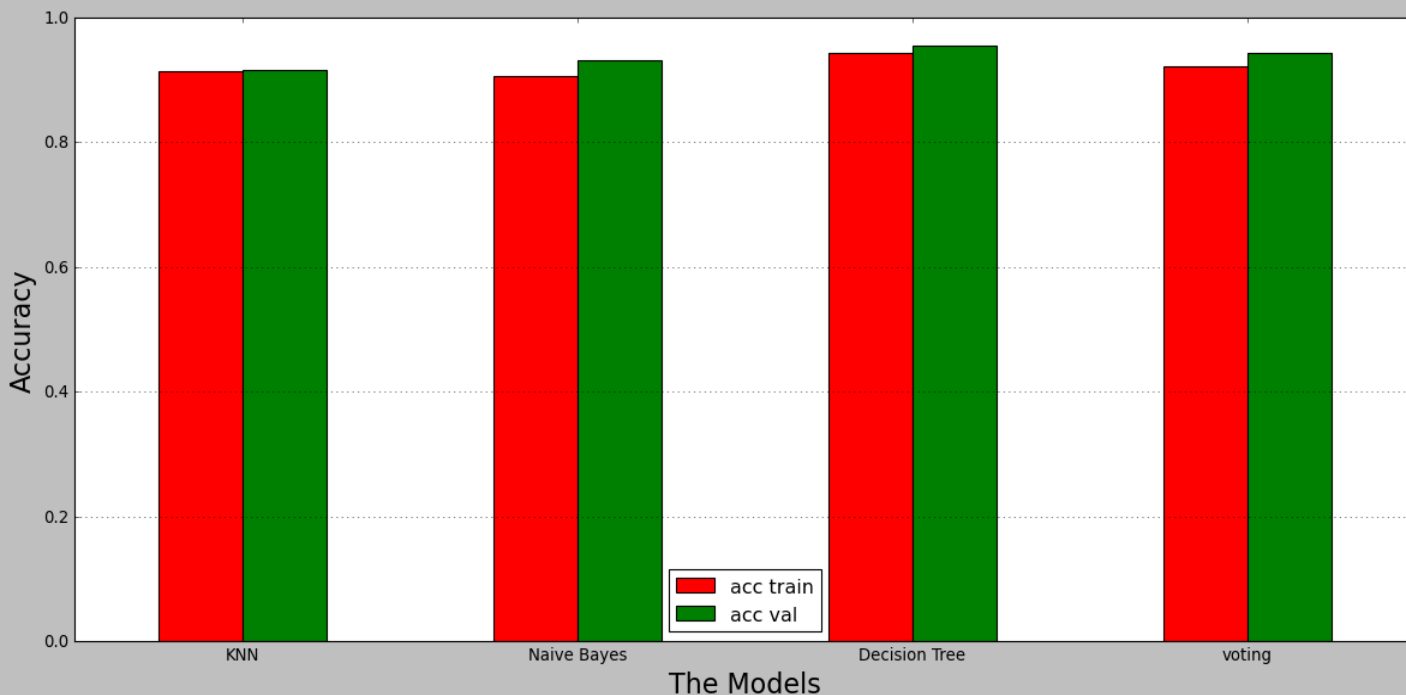


# The Algorithms used to classification

## Decision Tree Algorithm

- Voting to make decision

Final accurse for each model to make the decision

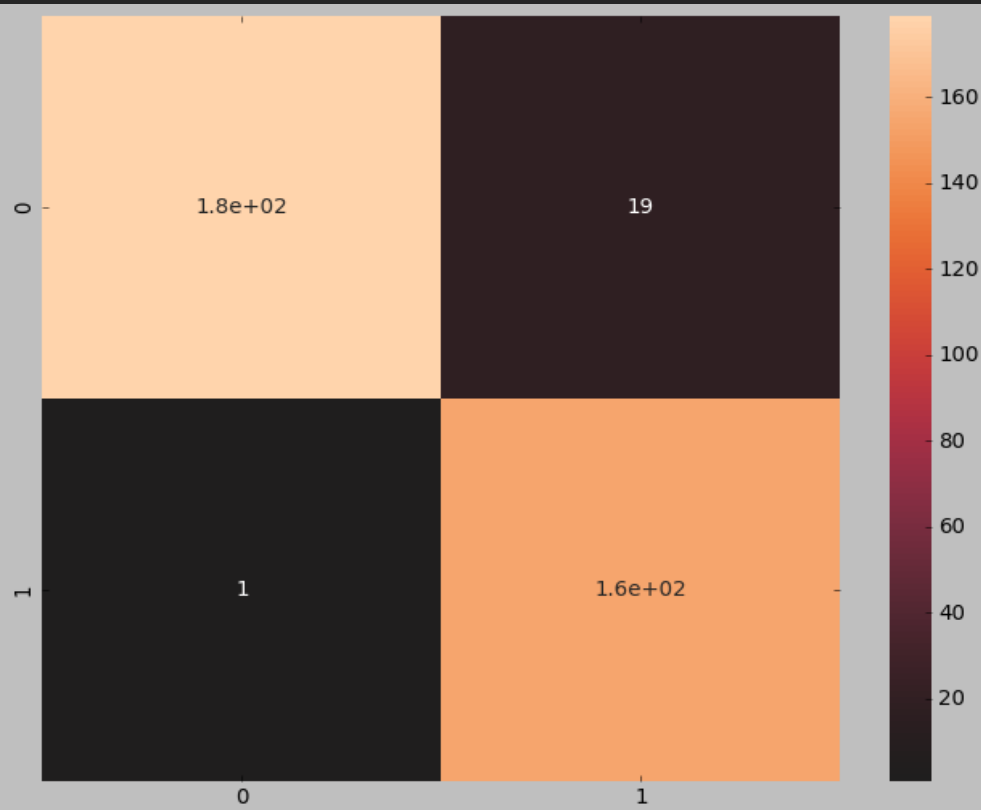


	model	acc train	acc val
0	Decision Tree	0.944089	0.955645
1	voting	0.921725	0.943548
2	Naive Bayes	0.905751	0.931452
3	KNN	0.913738	0.915323

# Final model test with test values.

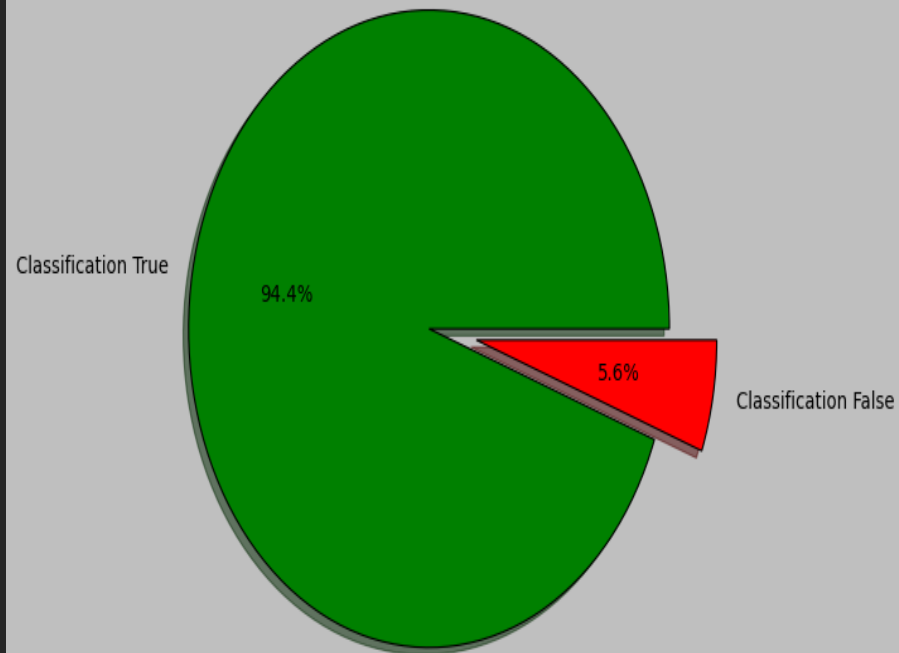
Final confuse matrix and accurse.

Confusion matrix



Final accurse.

final Classification Accuracy





**Conclusion.**



# Conclusion.

The Decision of solving problem.

- When a new orthopedic patient comes to the hospital, we will be able to largely classify if he is normal or abnormal by adding some new biomechanical features to the model.
- We can conclude that "Decision Tree Algorithm" works well for data because it has a high accuracy of 95.5%.

# Conclusion.

The Decision of solving problem.

- Save model to use it later.

```
import pickle

with open('saved-model.pickle', 'wb') as f:
    pickle.dump(final_model, f)
```

- Load model to use.

```
import pickle

with open('saved-model.pickle', 'rb') as f:
    my_model = pickle.load(f)

with open('scaler.pickle', 'rb') as f:
    my_scaler = pickle.load(f)
```

# Conclusion.

The Decision of solving problem.

- Test example.

```
... # example input
list_test=[67.2,10.6,66,30.1,109.3,55]
x = np.array(list_test)
```

```
... x_scaled = my_scaler.transform([x])
```

- Make decision.

```
... y = my_model.predict(x_scaled)
if y[0]=='1':
    print('Normal Case')
else:
    print("Abnormal Case")
```

Abnormal Case



THANK YOU!



Any Questions?