ML from Scratch with IRIS!!

Python notebook using data from Iris Species · 22,738 views · 2y ago

Hello Kagglers!!

This is a very basic tutorial to Machine Learning for complete Beginners using the Iris Dataset. You can learn how to implement a machine learning to a given dataset by following this notebook. I have explained everything related to the implementation in detail . Hope you find it useful.

For a more advanced notebook that covers some more detailed concepts, have a look at this notebook

If this notebook to be useful, **Please Upvote!!!**

In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load in

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
# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list the files in the
 input directory

from subprocess import check_output
print(check_output(["ls", "../input"]).decode("utf8"))

# Any results you write to the current directory are saved as output.
```

```
Iris.csv
database.sqlite
```

In [2]:
```
iris = pd.read_csv("../input/Iris.csv") #load the dataset
```

In [3]:
```
iris.head(2) #show the first 2 rows from the dataset
```

Out[3]:

	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

In [4]:
```
iris.info()  #checking if there is any inconsistency in the dataset
#as we see there are no null values in the dataset, so the data can be processed
```

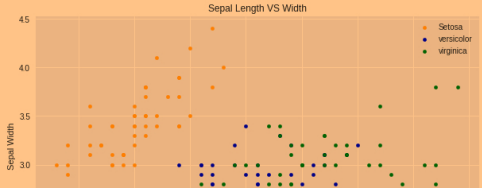
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
Id              150 non-null int64
SepalLengthCm   150 non-null float64
SepalWidthCm    150 non-null float64
PetalLengthCm   150 non-null float64
PetalWidthCm    150 non-null float64
Species         150 non-null object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.1+ KB
```

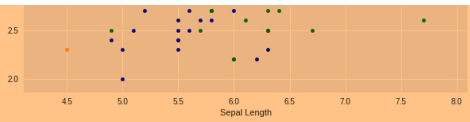
Removing the unneeded column

In [5]:
```
iris.drop('Id',axis=1,inplace=True) #dropping the Id column as it is unecessary, axis=1 specifie
s that it should be column wise, inplace =1 means the changes should be reflected into the datafr
ame
```

Some Exploratory Data Analysis With Iris

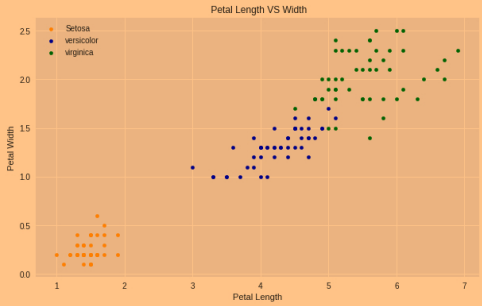
In [6]:
```
fig = iris[iris.Species=='Iris-setosa'].plot(kind='scatter',x='SepalLengthCm',y='SepalWidthCm',
color='orange', label='Setosa')
iris[iris.Species=='Iris-versicolor'].plot(kind='scatter',x='SepalLengthCm',y='SepalWidthCm',co
lor='blue', label='versicolor',ax=fig)
iris[iris.Species=='Iris-virginica'].plot(kind='scatter',x='SepalLengthCm',y='SepalWidthCm',col
or='green', label='virginica', ax=fig)
fig.set_xlabel("Sepal Length")
fig.set_ylabel("Sepal Width")
fig.set_title("Sepal Length VS Width")
fig=plt.gcf()
fig.set_size_inches(10,6)
plt.show()
```





The above graph shows relationship between the sepal length and width. Now we will check relationship between the petal length and width.

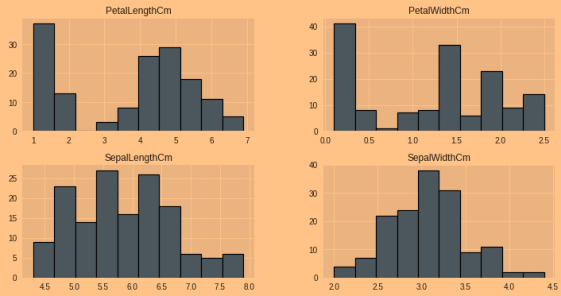
In [7]:
```
fig = iris[iris.Species=='Iris-setosa'].plot.scatter(x='PetalLengthCm',y='PetalWidthCm',color=
'orange', label='Setosa')
iris[iris.Species=='Iris-versicolor'].plot.scatter(x='PetalLengthCm',y='PetalWidthCm',color='bl
ue', label='versicolor',ax=fig)
iris[iris.Species=='Iris-virginica'].plot.scatter(x='PetalLengthCm',y='PetalWidthCm',color='gre
en', label='virginica', ax=fig)
fig.set_xlabel("Petal Length")
fig.set_ylabel("Petal Width")
fig.set_title(" Petal Length VS Width")
fig=plt.gcf()
fig.set_size_inches(10,6)
plt.show()
```



As we can see that the Petal Features are giving a better cluster division compared to the Sepal features. This is an indication that the Petals can help in better and accurate Predictions over the Sepal. We will check that later.

Now let us see how are the length and width are distributed

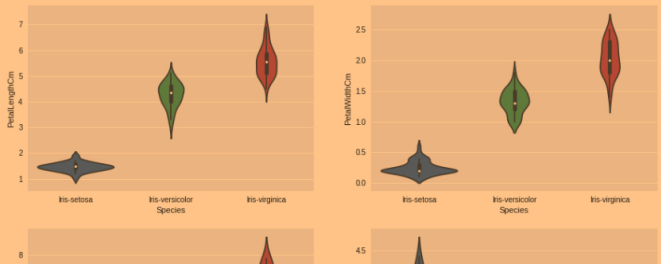
In [8]:
```
iris.hist(edgecolor='black', linewidth=1.2)
fig=plt.gcf()
fig.set_size_inches(12,6)
plt.show()
```

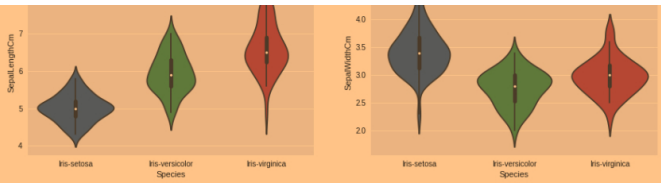


Now let us see how the length and width vary according to the species

In [9]:
```
plt.figure(fisgize=(15,10))
plt.subplot(2,2,1)
sns.violinplot(x='Species',y='PetalLengthCm',data=iris)
plt.subplot(2,2,2)
sns.violinplot(x='Species',y='PetalWidthCm',data=iris)
plt.subplot(2,2,3)
sns.violinplot(x='Species',y='SepalLengthCm',data=iris)
plt.subplot(2,2,4)
sns.violinplot(x='Species',y='SepalWidthCm',data=iris)
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f846388ee10>
```





The violinplot shows density of the length and width in the species. The thinner part denotes that there is less density whereas the fatter part conveys higher density

Now the given problem is a classification problem.. Thus we will be using the classification algorithms to build a model.

Classification: samples belong to two or more classes and we want to learn from already labeled data how to predict the class of unlabeled data

Regression: if the desired output consists of one or more continuous variables, then the task is called regression. An example of a regression problem would be the prediction of the length of a salmon as a function of its age and weight.

Before we start, we need to clear some ML notations.

attributes-->An attribute is a property of an instance that may be used to determine its classification. In the following dataset, the attributes are the petal and sepal length and width. It is also known as **Features**.

Target variable, in the machine learning context is the variable that is or should be the output. Here the target variables are the 3 flower species.

In [10]:
```
# importing alll the necessary packages to use the various classification algorithms
from sklearn.linear_model import LogisticRegression  # for Logistic Regression algorithm
from sklearn.cross_validation import train_test_split #to split the dataset for training and tes
ting
from sklearn.neighbors import KNeighborsClassifier  # for K nearest neighbours
from sklearn import svm  #for Support Vector Machine (SVM) Algorithm
from sklearn import metrics #for checking the model accuracy
from sklearn.tree import DecisionTreeClassifier #for using Decision Tree Algoithm
```

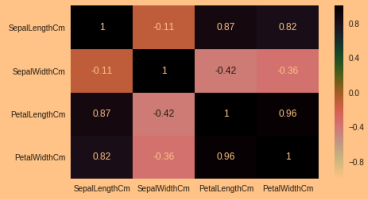
```
/opt/conda/lib/python3.6/site-packages/sklearn/cross_validation.py:41: DeprecationWarning: This
module was deprecated in version 0.18 in favor of the model_selection module into which all the
refactored classes and functions are moved. Also note that the interface of the new CV iterator
s are different from that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [11]:
```
iris.shape #get the shape of the dataset
```

Out[11]:
```
(150, 5)
```

Now, when we train any algorithm, the number of features and their correlation plays an important role. If there are features and many of the features are highly correlated, then training an algorithm with all the featues will reduce the accuracy. Thus features selection should be done carefully. This dataset has less featues but still we will see the correlation.

In [12]:
```
plt.figure(figsize=(7,4))
sns.heatmap(iris.corr(),annot=True,cmap='cubehelix_r') #draws  heatmap with input as the correla
tion matrix calculted by(iris.corr())
plt.show()
```



Observation--->

The Sepal Width and Length are not correlated The Petal Width and Length are highly correlated

We will use all the features for training the algorithm and check the accuracy.

Then we will use 1 Petal Feature and 1 Sepal Feature to check the accuracy of the algorithm as we are using only 2 features that are not correlated. Thus we can have a variance in the dataset which may help in better accuracy. We will check it later.

Steps To Be followed When Applying an Algorithm

1. Split the dataset into training and testing dataset. The testing dataset is generally smaller than training one as it will help in training the model better.
2. Select any algorithm based on the problem (classification or regression) whatever you feel may be good.
3. Then pass the training dataset to the algorithm to train it. We use the **.fit()** method
4. Then pass the testing data to the trained algorithm to predict the outcome. We use the **.predict()** method.
5. We then check the accuracy by **passing the predicted outcome and the actual output** to the model.

Splitting The Data into Training And Testing Dataset

In [13]:
```
train, test = train_test_split(iris, test_size = 0.3)# in this our main data is split into train
and test
# the attribute test_size=0.3 splits the data into 70% and 30% ratio. train=70% and test=30%
```

```
print(train.shape)
print(test.shape)
```

```
(105, 5)
(45, 5)
```

In [14]:
```
train_X = train[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']]# taking the tr
aining data features
train_y=train.Species# output of our training data
test_X= test[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']] # taking test dat
a features
test_y =test.Species   #output value of test data
```

Lets check the Train and Test Dataset

In [15]:
```
train_X.head(2)
```

Out[15]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
75	6.6	3.0	4.4	1.4
16	5.4	3.9	1.3	0.4

In [16]:
```
test_X.head(2)
```

Out[16]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
91	6.1	3.0	4.6	1.4
134	6.1	2.6	5.6	1.4

In [17]:
```
train_y.head()  ##output of the training data
```

Out[17]:
```
75     Iris-versicolor
16         Iris-setosa
25         Iris-setosa
101     Iris-virginica
125     Iris-virginica
Name: Species, dtype: object
```

Support Vector Machine (SVM)

In [18]:
```
model = svm.SVC() #select the algorithm
model.fit(train_X,train_y) # we train the algorithm with the training data and the training outp
ut
prediction=model.predict(test_X) #now we pass the testing data to the trained algorithm
print('The accuracy of the SVM is:',metrics.accuracy_score(prediction,test_y))#now we check the
accuracy of the algorithm.
#we pass the predicted output by the model and the actual output
```

```
The accuracy of the SVM is: 1.0
```

SVM is giving very good accuracy . We will continue to check the accuracy for different models.

Now we will follow the same steps as above for training various machine learning algorithms.

Logistic Regression

In [19]:
```
model = LogisticRegression()
model.fit(train_X,train_y)
prediction=model.predict(test_X)
print('The accuracy of the Logistic Regression is',metrics.accuracy_score(prediction,test_y))
```

```
The accuracy of the Logistic Regression is 0.977777777778
```

Decision Tree

In [20]:
```
model=DecisionTreeClassifier()
model.fit(train_X,train_y)
prediction=model.predict(test_X)
print('The accuracy of the Decision Tree is',metrics.accuracy_score(prediction,test_y))
```

```
The accuracy of the Decision Tree is 0.977777777778
```

K-Nearest Neighbours

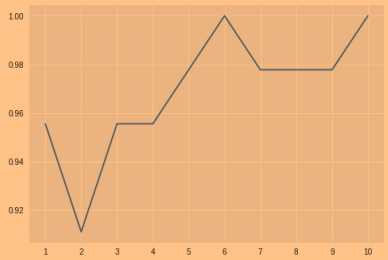
In [21]:
```
model=KNeighborsClassifier(n_neighbors=3) #this examines 3 neighbours for putting the new data i
nto a class
model.fit(train_X,train_y)
prediction=model.predict(test_X)
print('The accuracy of the KNN is',metrics.accuracy_score(prediction,test_y))
```

```
The accuracy of the KNN is 0.955555555556
```

Let's check the accuracy for various values of n for K-Nearest nerighbours

In [22]:
```
a_index=list(range(1,11))
a=pd.Series()
x=[1,2,3,4,5,6,7,8,9,10]
for i in list(range(1,11)):
    model=KNeighborsClassifier(n_neighbors=i)
    model.fit(train_X,train_y)
    prediction=model.predict(test_X)
    a=a.append(pd.Series(metrics.accuracy_score(prediction,test_y)))
plt.plot(a_index, a)
plt.xticks(x)
```

Out[22]:
```
([<matplotlib.axis.XTick at 0x7f8456e94668>,
  <matplotlib.axis.XTick at 0x7f8456f0a8d0>,
  <matplotlib.axis.XTick at 0x7f8456eb7ba8>,
  <matplotlib.axis.XTick at 0x7f8463954b70>,
  <matplotlib.axis.XTick at 0x7f8463b695f8>,
  <matplotlib.axis.XTick at 0x7f8463b69278>,
  <matplotlib.axis.XTick at 0x7f8463baa048>,
  <matplotlib.axis.XTick at 0x7f8463bb56d8>,
  <matplotlib.axis.XTick at 0x7f8463bb55c0>,
  <matplotlib.axis.XTick at 0x7f8463bfb908>],
 <a list of 10 Text xticklabel objects>)
```



Above is the graph showing the accuracy for the KNN models using different values of n.

We used all the features of iris in above models. Now we will use Petals and Sepals Seperately

Creating Petals And Sepals Training Data

In [23]:
```
petal=iris[['PetalLengthCm','PetalWidthCm','Species']]
sepal=iris[['SepalLengthCm','SepalWidthCm','Species']]
```

In [24]:
```
train_p,test_p=train_test_split(petal,test_size=0.3,random_state=0)  #petals
train_x_p=train_p[['PetalWidthCm','PetalLengthCm']]
train_y_p=train_p.Species
test_x_p=test_p[['PetalWidthCm','PetalLengthCm']]
test_y_p=test_p.Species


train_s,test_s=train_test_split(sepal,test_size=0.3,random_state=0)  #Sepal
train_x_s=train_s[['SepalWidthCm','SepalLengthCm']]
train_y_s=train_s.Species
test_x_s=test_s[['SepalWidthCm','SepalLengthCm']]
test_y_s=test_s.Species
```

SVM

In [25]:
```
model=svm.SVC()
model.fit(train_x_p,train_y_p)
prediction=model.predict(test_x_p)
print('The accuracy of the SVM using Petals is:',metrics.accuracy_score(prediction,test_y_p))

model=svm.SVC()
model.fit(train_x_s,train_y_s)
prediction=model.predict(test_x_s)
print('The accuracy of the SVM using Sepal is:',metrics.accuracy_score(prediction,test_y_s))
```

```
The accuracy of the SVM using Petals is: 0.977777777778
The accuracy of the SVM using Sepal is: 0.8
```

Logistic Regression

In [26]:
```
model = LogisticRegression()
model.fit(train_x_p,train_y_p)
prediction=model.predict(test_x_p)
print('The accuracy of the Logistic Regression using Petals is:',metrics.accuracy_score(predict
ion,test_y_p))

model.fit(train_x_s,train_y_s)
prediction=model.predict(test_x_s)
print('The accuracy of the Logistic Regression using Sepals is:',metrics.accuracy_score(predict
ion,test_y_s))
```

```
The accuracy of the Logistic Regression using Petals is: 0.688888888889
The accuracy of the Logistic Regression using Sepals is: 0.644444444444
```

Decision Tree

In [27]:
```
model=DecisionTreeClassifier()
model.fit(train_x_p,train_y_p)
prediction=model.predict(test_x_p)
print('The accuracy of the Decision Tree using Petals is:',metrics.accuracy_score(prediction,test_y_p))

model.fit(train_x_s,train_y_s)
prediction=model.predict(test_x_s)
print('The accuracy of the Decision Tree using Sepals is:',metrics.accuracy_score(prediction,test_y_s))
```

```
The accuracy of the Decision Tree using Petals is: 0.955555555556
The accuracy of the Decision Tree using Sepals is: 0.644444444444
```

K-Nearest Neighbours

In [28]:
```
model=KNeighborsClassifier(n_neighbors=3)
model.fit(train_x_p,train_y_p)
prediction=model.predict(test_x_p)
print('The accuracy of the KNN using Petals is:',metrics.accuracy_score(prediction,test_y_p))

model.fit(train_x_s,train_y_s)
prediction=model.predict(test_x_s)
print('The accuracy of the KNN using Sepals is:',metrics.accuracy_score(prediction,test_y_s))
```

```
The accuracy of the KNN using Petals is: 0.977777777778
The accuracy of the KNN using Sepals is: 0.733333333333
```

Observations:

- Using Petals over Sepal for training the data gives a much better accuracy.
- This was expected as we saw in the heatmap above that the correlation between the Sepal Width and Length was very low whereas the correlation between Petal Width and Length was very high.

Thus we have just implemented some of the common Machine Learning. Since the dataset is small with very few features, I didn't cover some concepts as they would be relevant when we have many features.

I have compiled a notebook covering some advanced ML concepts using a larger dataset. Have a look at that tooo.

I hope the notebook was useful to you to get started with Machine Learning.

If find this notebook, **Please Upvote**.

Thank You!!

In [29]:

Data

Data Sources

- Iris Species
  - Iris.csv 150 x 6
  - database.sqlite
    - Iris 150 x 6

Iris Species

Classify iris plants into three species in this classic dataset

Last Updated: 3 years ago (Version 2)

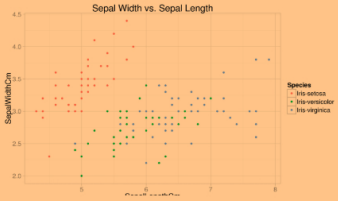
About this Dataset

The Iris dataset was used in R.A. Fisher's classic 1936 paper, The Use of Multiple Measurements in Taxonomic Problems, and can also be found on the UCI Machine Learning Repository.

It includes three iris species with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

The columns in this dataset are:

- Id
- SepalLengthCm
- SepalWidthCm
- PetalLengthCm
- PetalWidthCm
- Species



Comments (50)

Sort by: All Comments | Most Votes

Click here to comment...

MichaelSalam • Posted on Latest Version • 2 years ago • Options • Reply — 5

In line[:12] you said correlation leads to lower accuracy.
But in line[:28] you said due to higher correlation between Petal width and Petal length, there is higher accuracy.
Can you please explain me? I am a noobie in this field.

a year ago

This Comment was deleted.

GSD • Posted on Latest Version • a year ago • Options • Reply — 1

Thks I.Coder for such an interesting kernel on various classification algorithms ..I picked up few ideas from here for an NLP kernel of mine.Do check this out .Appreicate your thoughts..

Gutabaga Gilbert • Posted on Latest Version • a year ago • Options • Reply — 1

very nice but I thought for target variable we have to always change to binary number

Khan • Posted on Latest Version • a month ago • Options • Reply — 0

Hi guys i'm new in this web

Fanta Silue • Posted on Latest Version • a month ago • Options • Reply — 0

Nice for beginners!!!

Sten Ivan • Posted on Latest Version • a month ago • Options • Reply — 0

Thanks so much for creating this kernel, definitely very helpful and easy-to-follow!

Ritbik Bharti • Posted on Latest Version • 2 months ago • Options • Reply — 0

Very well explained for beginners....

2 months ago

This Comment was deleted.

Shailesh Z • Posted on Latest Version • 2 months ago • Options • Reply — 0

Nice Job, especially Data Visualization.

Bhavesh Kamble • Posted on Latest Version • 3 months ago • Options • Reply — 0

from sklearn.cross*validation import train*test_split is returning an error. Please Rectify the error

Nayan Lakhwani • Posted on Latest Version • 3 months ago • Options • Reply — 1

1. use sklearn.model_selection instead of crossvalidation
2. it is train _test _split and not traintest_split

vinhthanh • Posted on Latest Version • 3 months ago • Options • Reply — 0

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

Abdul Wahab • Posted on Latest Version • 4 months ago • Options • Reply — 0

Great Work!

sairamdgr8 • Posted on Latest Version • 4 months ago • Options • Reply — 0

Great sir...

Mateusz Bagiński • Posted on Latest Version • 5 months ago • Options • Reply — 0

I regret that I didn't see your kernel at the begining of learning ml. Easy explained on the code as well as on the graphs.

Keerthi • Posted on Latest Version • 5 months ago • Options • Reply — 0

Well explained. Thanks!

Kaushil Kundalia • Posted on Latest Version • 7 months ago • Options • Reply — 0

Great kernel for beginners like me ! Thanks a lot :D

Jie Zhang • Posted on Latest Version • 7 months ago • Options • Reply

0

There's a type in line [10]: # importing alll the necessary packages to use the various classification algorithms

Rosie Junghwa Yang • Posted on Latest Version • 8 months ago • Options • Reply

0

Hi Ashwini!!
When we evaluate predictions, don't we need to use 'accuracy_score(y_test, y_pred)'? (True result comes first, and predictions come next!) I checked out official docs as well. And of course I could be wrong as well, if so pls share with me! Thank you for great notebook :)

Shubhang Sharma • Posted on Latest Version • 8 months ago • Options • Reply

0

Great Explanation of how to achieve better accuracy and how to use correlation feature to boost performance of model. It was an interesting kernel to go through.

Pedram Salimi • Posted on Latest Version • 8 months ago • Options • Reply

0

Thank you, that was awesome!

ANN MARY SHAJU • Posted on Latest Version • 9 months ago • Options • Reply

0

Well explained . Thanks

10 months ago

This Comment was deleted.

Binu • Posted on Latest Version • a year ago • Options • Reply

0

Thanks for sharing

Boggler • Posted on Latest Version • 2 years ago • Options • Reply

0

Thanks!

SachinKumar • Posted on Latest Version • 2 years ago • Options • Reply

0

Liked your work very much. Thanks!

cat_yao • Posted on Latest Version • 2 years ago • Options • Reply

0

Nice job! I learn a lot from this post

Praveen Kumar Neelappa • Posted on Latest Version • 2 years ago • Options • Reply

0

Good job on explaining it so well.

Shyam Sundar B • Posted on Latest Version • 2 years ago • Options • Reply

0

Thanks, Ashwin. Your approach is quite easy to understand. Well done.

SametDumankaya • Posted on Latest Version • 2 years ago • Options • Reply

0

Very nice tutorial. Thanks.

pawan • Posted on Latest Version • 2 years ago • Options • Reply

0

Good job!

Prasanna Nadimpalli • Posted on Version 28 • 2 years ago • Options • Reply

0

Thanks. This was informative.

Suraj Jha • Posted on Version 27 • 2 years ago • Options • Reply

0

good job here!!

Shaurya Munshi • Posted on Version 26 • 2 years ago • Options • Reply

0

I've used pretty much the same model and the maximum accuracy I've been able to get is 0.99. I'm still not able to get an accuracy of 1 even after running exactly the same code as yours. Please help me with this.

Shaurya Munshi • Posted on Version 26 • 2 years ago • Options • Reply

0

Gughapriyaa • Posted on Version 26 • 2 years ago • Options • Reply

0

Thanks for your post. Helped me a lot !

Diego Barra • Posted on Version 25 • 2 years ago • Options • Reply

0

Thank you for doing this!

[Deleted User] • Posted on Version 25 • 2 years ago • Options • Reply 0

Thank you for sharing this. It helped a lot!

cnwarden • Posted on Version 25 • 2 years ago • Options • Reply 0

Thank you for sharing

Ashwini Swain Kernel Author • Posted on Version 25 • 2 years ago • Options • Reply 1

You are welcome...Do upvote if useful!!

Arun94 • Posted on Version 25 • 2 years ago • Options • Reply 0

well explained

DineshKumar • Posted on Version 24 • 2 years ago • Options • Reply 0

Thank you so much for your post.

Karthick Mohanraj • Posted on Version 18 • 2 years ago • Options • Reply 0

Good Job!

[Deleted User] • Posted on Version 13 • 2 years ago • Options • Reply 0

Thanks @ashwin

Ashwini Swain Kernel Author • Posted on Version 13 • 2 years ago • Options • Reply 0

You are welcome :)

Sadid A. Hasan • Posted on Version 13 • 2 years ago • Options • Reply 0

Nice job!

Ashwini Swain Kernel Author • Posted on Version 13 • 2 years ago • Options • Reply 0

Thanks a lot!!

[Deleted User] • Posted on Version 13 • 2 years ago • Options • Reply 0

Little elaborate at In [23] : ?

Ashwini Swain Kernel Author • Posted on Version 13 • 2 years ago • Options • Reply 3

Okay so it is the same thing I did earlier by taking all the attributes for training the model. In this step, we take the Sepals And Petals seperately for training the model. Let's see

petal=iris[['PetalLengthCm','PetalWidthCm','Species']]
sepal=iris[['SepalLengthCm','SepalWidthCm','Species']]

In the above steps, we separate the original dataframe by splitting it by columns [PetalLength , PetalWidth] for the petals and similarly for Sepal. Now what petal and sepal variables have is a dataframe with only 3 columns i.e ['PetalLengthCm','PetalWidthCm','Species'] for **(petal)** variable and ['SepalLengthCm','SepalWidthCm','Species'] for **sepal** variable.

train*p*,*test*p=train*test*split(petal,test_size=0.3)

In the above step we split the petal dataframe into test and train dataset as we had done earlier.

trainxp=train_p[['PetalWidthCm','PetalLengthCm']]

Now we take only the attributes or the features columns into seperate variable and not the output column. This is the training dataset.

trainyp=train_p.Species

Then we store the corresponding output of each row into another variable.This is the output for the training dataset. Now we will use trainxp and trainyp for training the model.

testxp=test*p[['PetalWidthCm','PetalLengthCm']] testyp=test*p.Species

These are similar to the previous steps that we did for training dataset.

Hope You understood...If not please ask again.