Interactive exercise: decision trees

Liping Wu 300-958-061   11/18/2020

In this exercise, we will do the following:

- Explore a dataset
- Split the data set
- Build a decision tree model using training data, visualize the tree
- Cross validate the model
- Test the model using testing data & generate a confusion matrix
- Prune the tree

Pre-requisites:

1- Install Anoconda
2- We will be using a lot of Public datasets these datasets are available at https://goo.gl/zjS4C6 under a folder named "Datasets for Predictive Modelling with Python", the datasets are organized in the order of the text book chapters: Python: Advanced Predictive Analytics, chapter # 8 files are required

Steps for exploring and building a logistic regression model:

1- Open your spyder IDE
2- Load the 'Iris.csv' file into a dataframe name *the dataframe data_firstname_i where first name is your first name* carry out the following activities:
   a. Display the column names
   b. Display the shape of the data frame i.e number of rows and number of columns
   c. Display the main statistics of the data
   d. Display the types of columns
   e. Display the first five records
   f. Find the unique values of the class

Following is the code, *make sure you update the path to the correct path where you placed the files and update the data frame name correctly*:

```
import pandas as pd
import os
path = "D:/CentennialWu/2020Fall/COMP309Data/Assignments/Lab08Wk10/"
filename = "iris.csv"
fullpath = os.path.join(path,filename)
print(fullpath)
data_liping_i = pd.read_csv(fullpath,sep=',')
print('Columns:',data_liping_i.columns.values)
print('Shape(rows and columns):',data_liping_i.shape)
```

print('Describe:\n',data_liping_i.describe())
print('DataType:\n',data_liping_i.dtypes)
print('First 5 rows:\n', data_liping_i.head(5))
print('Unique Species:\n',data_liping_i['Species'].unique())

```
In [18]: import pandas as pd
    ...: import os
    ...: path = "D:/CentennialWu/2020Fall/COMP309Data/Assignments/Lab08Wk10/"
    ...: filename = "iris.csv"
    ...: fullpath = os.path.join(path,filename)
    ...: print(fullpath)
    ...: data_liping_i = pd.read_csv(fullpath,sep=',')
    ...: print('Columns:',data_liping_i.columns.values)
    ...: print('Shape(rows and columns):',data_liping_i.shape)
    ...: print('Describe:\n',data_liping_i.describe())
    ...: print('DataType:\n',data_liping_i.dtypes)
    ...: print('First 5 rows:\n', data_liping_i.head(5))
    ...: print('Unique Species:\n',data_liping_i['Species'].unique())
D:/CentennialWu/2020Fall/COMP309Data/Assignments/Lab08Wk10/iris.csv
Columns: ['Sepal.Length' 'Sepal.Width' 'Petal.Length' 'Petal.Width' 'Species']
Shape(rows and columns): (150, 5)
Describe:
        Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.057333      3.758000     1.199333
std        0.828066     0.435866      1.765298     0.762238
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
DataType:
 Sepal.Length    float64
Sepal.Width     float64
Petal.Length    float64
Petal.Width     float64
Species          object
dtype: object
First 5 rows:
    Sepal.Length  Sepal.Width  Petal.Length  Petal.Width Species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
Unique Species:
 ['setosa' 'versicolor' 'virginica']
```

3- Separate the predictors from the target then split the dataset using numpy random function.

Following is the code, *make sure you update the the data frame name correctly*:

# change columns to list
colnames=data_liping_i.columns.values.tolist()
print(colnames)
# collect the first four columns as predictors
predictors=colnames[:4]
print(predictors)
#assign the 5th collomn as target(or predict results)

```
target=colnames[4]
print(target)

import numpy as np
# add one column 'is_train' to the dataframe
data_liping_i['is_train'] = np.random.uniform(0, 1, len(data_liping_i)) <= .75
print(data_liping_i.head(5))
# Create two new dataframes, one with the training rows, one with the test rows
train, test = data_liping_i[data_liping_i['is_train']==True],
data_liping_i[data_liping_i['is_train']==False]
print('dataframe train:\n',train)
print('dataframe test:\n',test)
print('Number of observations in the training data:', len(train))
print('Number of observations in the test data:',len(test))
```

```
In [30]:
    ...: colnames=data_liping_i.columns.values.tolist()
    ...: print(colnames)
    ...: # collect the first four columns as predictors
    ...: predictors=colnames[:4]
    ...: print(predictors)
    ...: #assign the 5th collomn as target(or predict results)
    ...: target=colnames[4]
    ...: print(target)
    ...:
    ...: import numpy as np
    ...: # add one column 'is_train' to the dataframe
    ...: data_liping_i['is_train'] = np.random.uniform(0, 1, len(data_liping_i)) <= .75
    ...: print(data_liping_i.head(5))
    ...: # Create two new dataframes, one with the training rows, one with the test rows
    ...: train, test = data_liping_i[data_liping_i['is_train']==True], data_liping_i[data_liping_i['is_train']==False]
    ...: print('dataframe train:\n',train)
    ...: print('dataframe test:\n',test)
    ...: print('Number of observations in the training data:', len(train))
    ...: print('Number of observations in the test data:',len(test))
['Sepal.Length', 'Sepal.Width', 'Petal.Length', 'Petal.Width', 'Species', 'is_train']
['Sepal.Length', 'Sepal.Width', 'Petal.Length', 'Petal.Width']
Species
   Sepal.Length  Sepal.Width  Petal.Length  Petal.Width Species  is_train
0           5.1          3.5           1.4          0.2  setosa     False
1           4.9          3.0           1.4          0.2  setosa      True
2           4.7          3.2           1.3          0.2  setosa      True
3           4.6          3.1           1.5          0.2  setosa      True
4           5.0          3.6           1.4          0.2  setosa      True
dataframe train:
      Sepal.Length  Sepal.Width  Petal.Length  Petal.Width    Species  is_train
1             4.9          3.0           1.4          0.2     setosa      True
2             4.7          3.2           1.3          0.2     setosa      True
3             4.6          3.1           1.5          0.2     setosa      True
4             5.0          3.6           1.4          0.2     setosa      True
5             5.4          3.9           1.7          0.4     setosa      True
..            ...          ...           ...          ...        ...       ...
145           6.7          3.0           5.2          2.3  virginica      True
146           6.3          2.5           5.0          1.9  virginica      True
147           6.5          3.0           5.2          2.0  virginica      True
148           6.2          3.4           5.4          2.3  virginica      True
149           5.9          3.0           5.1          1.8  virginica      True

[114 rows x 6 columns]
```

```
[114 rows x 6 columns]
dataframe test:
     Sepal.Length  Sepal.Width  Petal.Length  Petal.Width     Species  is_train
0             5.1          3.5           1.4          0.2      setosa     False
13            4.3          3.0           1.1          0.1      setosa     False
20            5.4          3.4           1.7          0.2      setosa     False
25            5.0          3.0           1.6          0.2      setosa     False
27            5.2          3.5           1.5          0.2      setosa     False
30            4.8          3.1           1.6          0.2      setosa     False
31            5.4          3.4           1.5          0.4      setosa     False
42            4.4          3.2           1.3          0.2      setosa     False
44            5.1          3.8           1.9          0.4      setosa     False
55            5.7          2.8           4.5          1.3  versicolor     False
56            6.3          3.3           4.7          1.6  versicolor     False
58            6.6          2.9           4.6          1.3  versicolor     False
59            5.2          2.7           3.9          1.4  versicolor     False
62            6.0          2.2           4.0          1.0  versicolor     False
68            6.2          2.2           4.5          1.5  versicolor     False
70            5.9          3.2           4.8          1.8  versicolor     False
71            6.1          2.8           4.0          1.3  versicolor     False
82            5.8          2.7           3.9          1.2  versicolor     False
84            5.4          3.0           4.5          1.5  versicolor     False
92            5.8          2.6           4.0          1.2  versicolor     False
96            5.7          2.9           4.2          1.3  versicolor     False
97            6.2          2.9           4.3          1.3  versicolor     False
98            5.1          2.5           3.0          1.1  versicolor     False
102           7.1          3.0           5.9          2.1   virginica     False
108           6.7          2.5           5.8          1.8   virginica     False
114           5.8          2.8           5.1          2.4   virginica     False
117           7.7          3.8           6.7          2.2   virginica     False
120           6.9          3.2           5.7          2.3   virginica     False
127           6.1          3.0           4.9          1.8   virginica     False
128           6.4          2.8           5.6          2.1   virginica     False
130           7.4          2.8           6.1          1.9   virginica     False
132           6.4          2.8           5.6          2.2   virginica     False
136           6.3          3.4           5.6          2.4   virginica     False
138           6.0          3.0           4.8          1.8   virginica     False
142           5.8          2.7           5.1          1.9   virginica     False
144           6.7          3.3           5.7          2.5   virginica     False
Number of observations in the training data: 114
Number of observations in the test data: 36
```

4- Build the decision tree using the training dataset. Name the model dt_firstname where firstname is your first name. Use enotrpy as a method for splitting, and split only when reaching 20 matches.

```
from sklearn.tree import DecisionTreeClassifier
dt_liping = DecisionTreeClassifier(criterion='entropy',min_samples_split=20, random_state=99)
dt_liping.fit(train[predictors], train[target])
```

5- Test the model using the testing dataset and calculate a confusion matrix this time using pandas

Following is the code, *make sure you update model name correctly*:

```
preds=dt_liping.predict(test[predictors])
pd.crosstab(test['Species'],preds,rownames=['Actual'],colnames=['Predictions'])
```

```
In [40]: '''
    ...: 4- Build the decision tree using the training dataset.
    ...:  Use enotrpy as a method for splitting, and split only when reaching  20 matches.
    ...: '''
    ...: from sklearn.tree import DecisionTreeClassifier
    ...: dt_liping = DecisionTreeClassifier(criterion='entropy',min_samples_split=20, random_state=99)
    ...: dt_liping.fit(train[predictors], train[target])
    ...:
    ...: '''
    ...: 5- Test the model using the testing dataset and calculate a confusion matrix this time using pandas
    ...: Following is the code, make sure you update model name correctly:
    ...:     '''
    ...: preds=dt_liping.predict(test[predictors])
    ...: pd.crosstab(test['Species'],preds,rownames=['Actual'],colnames=['Predictions'])
Out[40]:
Predictions  setosa  versicolor  virginica
Actual
setosa           12           0          0
versicolor        0          15          1
virginica         0           0          7
```

6- Generate a dot file and visualize the tree using the online viz graph editor and share (download) as picture.

```
path = "D:/CentennialWu/2020Fall/COMP309Data/Assignments/Lab08Wk10/"
from sklearn.tree import export_graphviz
with open(path+ 'dtree3.dot', 'w') as dotfile:
   export_graphviz(dt_liping, out_file = dotfile, feature_names = predictors)
dotfile.close()
```

After run the code above, dtree3.dot file produced under the working folder

Name	Ext	Size	Date	Attr
[..]		<DIR>	11/18/2020 10:43	----
wk10DecisionTree_Liping	py	561	11/16/2020 19:24	-a--
wine	csv	84,199	11/15/2020 16:16	-a--
iris	csv	3,867	11/15/2020 16:16	-a--
Interactive_exercise_clustering1	docx	183,953	11/15/2020 16:16	-a--
Interactive_exercise_clustering	docx	182,078	11/15/2020 16:16	-a--
Interactive exercise decision trees1_Liping	docx	218,196	11/18/2020 10:37	-a--
Interactive exercise decision trees1	docx	25,005	11/15/2020 16:16	-a--
dtree3	dot	890	11/18/2020 10:43	-a--
~WRL1293	tmp	25,067	11/16/2020 12:55	--h-
~$teractive exercise decision trees1_Liping	docx	162	11/18/2020 09:51	-ah-

Open it with notepad++ , the code shows as:

```
digraph Tree {
node [shape=box] ;
0 [label="Petal.Length <= 2.45\nentropy = 1.578\nsamples = 115\nvalue = [38, 34, 43]"] ;
1 [label="entropy = 0.0\nsamples = 38\nvalue = [38, 0, 0]"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="Petal.Width <= 1.65\nentropy = 0.99\nsamples = 77\nvalue = [0, 34, 43]"] ;
0 -> 2 [labeldistance=2.5, labelangle=-45, headlabel="False"] ;
3 [label="Petal.Length <= 4.95\nentropy = 0.316\nsamples = 35\nvalue = [0, 33, 2]"] ;
2 -> 3 ;
4 [label="entropy = 0.0\nsamples = 32\nvalue = [0, 32, 0]"] ;
3 -> 4 ;
5 [label="entropy = 0.918\nsamples = 3\nvalue = [0, 1, 2]"] ;
3 -> 5 ;
6 [label="Petal.Length <= 4.85\nentropy = 0.162\nsamples = 42\nvalue = [0, 1, 41]"] ;
2 -> 6 ;
7 [label="entropy = 0.811\nsamples = 4\nvalue = [0, 1, 3]"] ;
6 -> 7 ;
8 [label="entropy = 0.0\nsamples = 38\nvalue = [0, 0, 38]"] ;
6 -> 8 ;
}
```

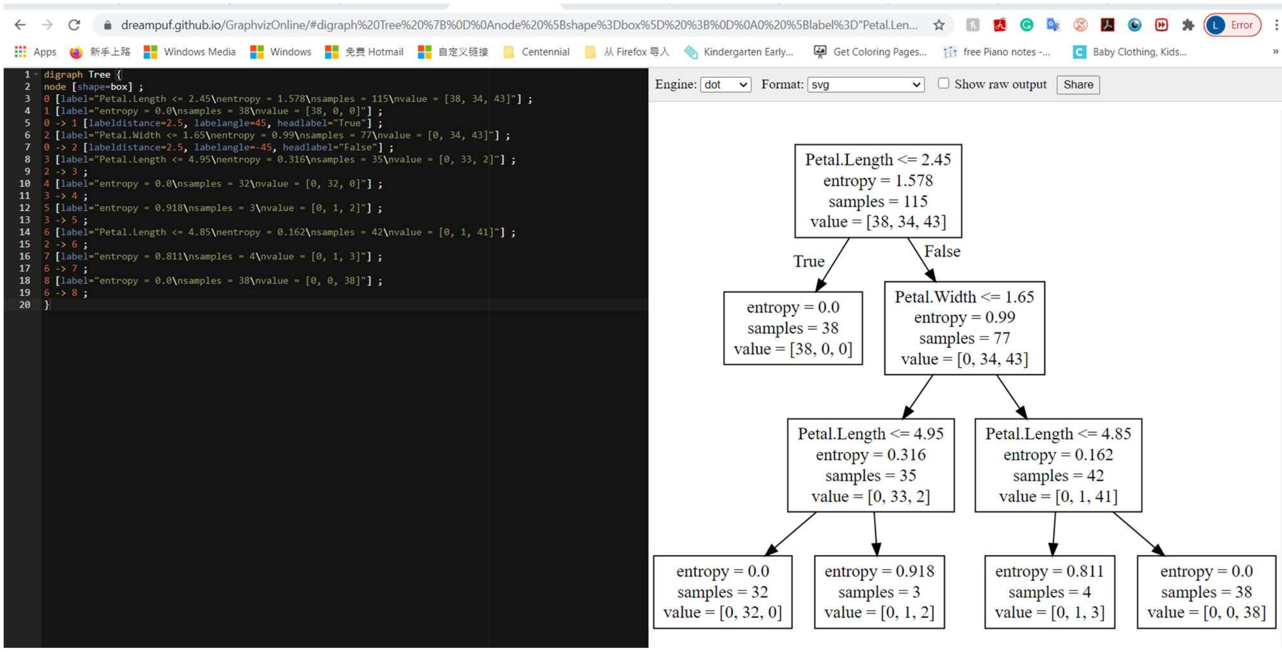
Google Graphviz Online

dreampuf.github.io › GraphvizOnline

Graphviz Online

node [style=filled,color=white];. a0 -> a1 -> a2 -> a3;. label = "process #1";. } subgraph cluster_1 {. **node** [style=filled];. b0 -> b1 -> b2 -> b3;. label = "process #2";.

Open the link and paste the code to the website



online graph editor links:
https://edotor.net/
https://dreampuf.github.io/GraphvizOnline/

Notice

The tree reads as follows:

- *If Petal Length<=2.45, then the flower species is setosa.*
- *If Petal Length>2.45, then check Petal Width. If Petal Length<=4.95, then the species is versicolor. If Petal Length > 4.95, then there is 1 versicolor and 2 virginica, and further classification is not possible.*
- *If Petal Length>2.45, then check Petal Width. If Petal Length<=4.85, then there is 1 versicolor and 2 virginica, and further classification is not possible. If Petal Length>4.85, then the species is virginica.*

Some other observations from the tree are as follows:

- *The maximum depth (the number of levels) of the tree is 3. In 3 leaves, the tree has been able to identify categories, which will make the dataset homogeneous.*
- *Sepal dimensions don't seem to be playing any role in the tree formation or, in other words, in the classification of these flowers into one of the species.*
- *There is a terminal node at the 1st level itself. If Petal Length<=2.45, one gets only the setosa species of flowers.*
- *The value in each node denotes the number of observations belonging to the three species (setosa, versicolor, and virginica in that order) at that node. Thus, the terminal node in the 1st level has 34 setosas, 0 versicolors, and 0 virginicas.*

7- Let us build the tree classifier again but this time let us split the data into 80% for training and 20% for testing:

X=data_liping_i[predictors]

Y=data_liping_i[target]

```
#split the data sklearn module
from sklearn.model_selection import train_test_split
trainX,testX,trainY,testY = train_test_split(X,Y, test_size = 0.2)
```

8- Let us now build the tree using the training as follows:
   a. Set the tree parameters
   b. Fit the training data
   c. Use the cross validation module and carry out a ten cross validation
   d. Use the sklearn metrics module to generate the score for the cross validation(i.e. build the model 10 times)
   e. Print the mean of the ten time runs

```
dt1_liping = DecisionTreeClassifier(criterion='entropy',max_depth=5, min_samples_split=20, random_state=99)
dt1_liping.fit(trainX,trainY)
# 10 fold cross validation using sklearn and all the data i.e validate the data
from sklearn.model_selection import KFold
#help(KFold)
crossvalidation = KFold(n_splits=10, shuffle=True, random_state=1)
from sklearn.model_selection import cross_val_score
score = np.mean(cross_val_score(dt1_liping, trainX, trainY, scoring='accuracy', cv=crossvalidation, n_jobs=1))
score
```

```
In [45]: from sklearn.model_selection import KFold
    ...: #help(KFold)
    ...: crossvalidation = KFold(n_splits=10, shuffle=True, random_state=1)
    ...: from sklearn.model_selection import cross_val_score
    ...: score = np.mean(cross_val_score(dt1_liping, trainX, trainY, scoring='accuracy', cv=crossvalidation, n_jobs=1))
    ...: score
Out[45]: 0.925
```

9- Now let us test the model using the testing data i.e. the 20%:
   a. Use the predict method and pass the 20% test data without labels i.e testX. This should generate the predicted data store it in testY_predict
   b. Use the metrics module from sklearn to calculate the score and the confusion matrix.

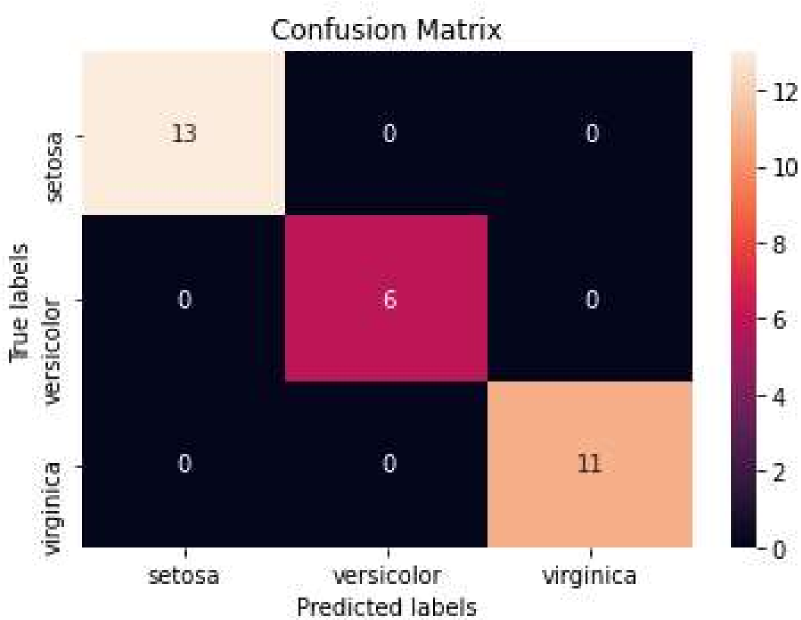
```
testY_predict = dt1_liping.predict(testX)
testY_predict.dtype
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
labels = Y.unique()
print(labels)
print("Accuracy:",metrics.accuracy_score(testY, testY_predict))
#Let us print the confusion matrix
from sklearn.metrics import confusion_matrix
print("Confusion matrix \n" , confusion_matrix(testY, testY_predict, labels))
```

```
In [46]:
    ...: testY_predict = dt1_liping.predict(testX)
    ...: testY_predict.dtype
    ...: #Import scikit-learn metrics module for accuracy calculation
    ...: from sklearn import metrics
    ...: labels = Y.unique()
    ...: print(labels)
    ...: print("Accuracy:",metrics.accuracy_score(testY, testY_predict))
    ...: #Let us print the confusion matrix
    ...: from sklearn.metrics import confusion_matrix
    ...: print("Confusion matrix \n" , confusion_matrix(testY, testY_predict, labels))
['setosa' 'versicolor' 'virginica']
Accuracy: 1.0
Confusion matrix
 [[13  0  0]
 [ 0  6  0]
 [ 0  0 11]]
C:\Users\foxpe\anaconda3\lib\site-packages\sklearn\utils\validation.py:68: FutureWarning: Pass labels=['setosa' 'versicolor' 'virginica']
as keyword args. From version 0.25 passing these as positional arguments will result in an error
  warnings.warn("Pass {} as keyword args. From version 0.25 "
```

10- Use Seaborn heatmaps to print the confusion matrix in a more clear and fancy way☺

```
import seaborn as sns
import matplotlib.pyplot as plt
cm = confusion_matrix(testY, testY_predict, labels)
ax= plt.subplot()
sns.heatmap(cm, annot=True, ax = ax); #annot=True to annotate cells


# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
ax.xaxis.set_ticklabels(['setosa', 'versicolor', 'virginica']); ax.yaxis.set_ticklabels(['setosa',
'versicolor', 'virginica']);
plt.show()
```



Exercises

1. Prune the tree exercise

Change the max depth and re-run the model 10 times using some of the above code (you need to decide which code) each time with a different value of max depth ranging from 1 to 10 record your results on a table. It would be nice if you could have a loop to automate. Show the results to your professor.

Code part please see attached python file.
Brief results please see the following table. More details please see attached pdf file.

Max_Depth	1	2	3	4	5	6	7	8	9	10
Score	0.575	0.93	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94
labels	['setosa' 'versicolor' 'virginica']									
Accuracy	0.63	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
Confusion Matrix	[[9 0 0] [0 0 11] [0 0 10]]	[[9 0 0] [0 10 1] [0 2 8]]	[[9 0 0] [0 10 1] [0 2 8]]	[[9 0 0] [0 10 1] [0 2 8]]	[[9 0 0] [0 10 1] [0 2 8]]	[[9 0 0] [0 10 1] [0 2 8]]	[[9 0 0] [0 10 1] [0 2 8]]	[[9 0 0] [0 10 1] [0 2 8]]	[[9 0 0] [0 10 1] [0 2 8]]	[[9 0 0] [0 10 1] [0 2 8]]

2. Do a feature importance test to determine which of the variables in the preceding dataset are actually important for the model. Share results with your professor.
   Code part please see attached python file.
   For whole dataframe:

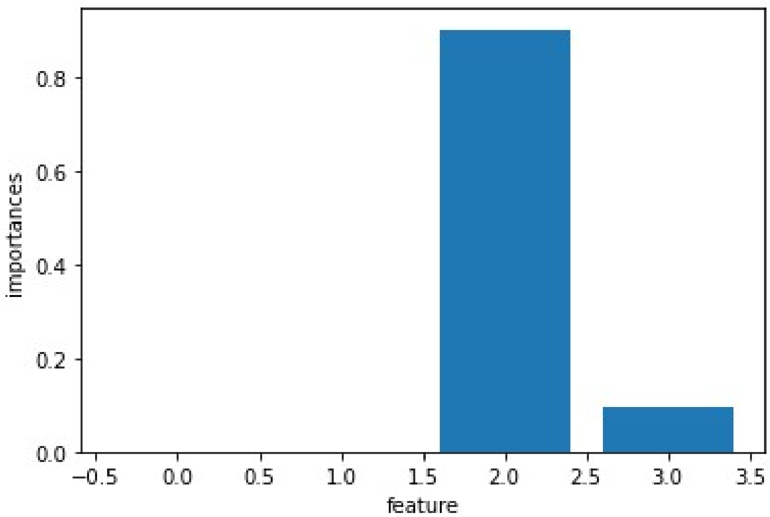
```
In [166]:
     ...: importance = dt_liping.feature_importances_
     ...:
     ...: # summarize feature importance
     ...: for i,v in enumerate(importance):
     ...:     print('Feature: %0d, Score: %.5f' % (i,v))
     ...: # plot feature importance
     ...: pyplot.bar([x for x in range(len(importance))], importance)
     ...: pyplot.xlabel('feature')
     ...: pyplot.ylabel('importances')
     ...: pyplot.show()
Feature: 0, Score: 0.00000
Feature: 1, Score: 0.00000
Feature: 2, Score: 0.90345
Feature: 3, Score: 0.09655

In [167]:
```

   For dataframe for different max-depth

```
***********Max_depth=1**************************
Feature: 0, Score: 0.00000
Feature: 1, Score: 0.00000
Feature: 2, Score: 1.00000
Feature: 3, Score: 0.00000
***********Max_depth=2**************************
Feature: 0, Score: 0.00000
Feature: 1, Score: 0.00000
Feature: 2, Score: 1.00000
Feature: 3, Score: 0.00000
***********Max_depth=3**************************
Feature: 0, Score: 0.00000
Feature: 1, Score: 0.00000
Feature: 2, Score: 0.95510
Feature: 3, Score: 0.04490
***********Max_depth=4**************************
Feature: 0, Score: 0.00000
Feature: 1, Score: 0.00000
Feature: 2, Score: 0.95606
Feature: 3, Score: 0.04394
***********Max_depth=5**************************
Feature: 0, Score: 0.00000
Feature: 1, Score: 0.00000
Feature: 2, Score: 0.95606
Feature: 3, Score: 0.04394
***********Max_depth=6**************************
Feature: 0, Score: 0.00000
Feature: 1, Score: 0.00000
Feature: 2, Score: 0.95606
Feature: 3, Score: 0.04394
***********Max_depth=7**************************
Feature: 0, Score: 0.00000
Feature: 1, Score: 0.00000
Feature: 2, Score: 0.95606
Feature: 3, Score: 0.04394
***********Max_depth=8**************************
Feature: 0, Score: 0.00000
Feature: 1, Score: 0.00000
Feature: 2, Score: 0.95606
Feature: 3, Score: 0.04394
***********Max_depth=9**************************
Feature: 0, Score: 0.00000
Feature: 1, Score: 0.00000
Feature: 2, Score: 0.95606
Feature: 3, Score: 0.04394
***********Max_depth=10**************************
Feature: 0, Score: 0.00000
Feature: 1, Score: 0.00000
Feature: 2, Score: 0.95606
Feature: 3, Score: 0.04394
```

All of the dataframes, the third feature - 'Petal.Length' is the most important feature in our study.


importances
feature
0.0
0.2
0.4
0.6
0.8
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5