In [1]:
```
# import required packages
import pandas as pd
```

In [2]:
```
data = pd.read_csv("https://raw.githubusercontent.com/AP-State-Skill-Develop
```

In [3]:
```
data.head()
```

Out[3]:

	survived	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked
0	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [4]:
```
data.shape
```

Out[4]: (891, 11)

In [5]:
```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 columns):
survived    891 non-null int64
pclass      891 non-null int64
name        891 non-null object
sex         891 non-null object
age         714 non-null float64
sibsp       891 non-null int64
parch       891 non-null int64
ticket      891 non-null object
fare        891 non-null float64
cabin       204 non-null object
embarked    889 non-null object
dtypes: float64(2), int64(4), object(5)
memory usage: 76.6+ KB
```

In [6]:
```
1 data.isnull().sum()
```

Out[6]:
```
survived      0
pclass        0
name          0
sex           0
age         177
sibsp         0
parch         0
ticket        0
fare          0
cabin       687
embarked      2
dtype: int64
```

In [7]:
```
1 data.shape
```

Out[7]:
```
(891, 11)
```

In [8]:
```
1 891-687
```

Out[8]:
```
204
```

In [9]:
```
1 data.drop("cabin",axis=1,inplace=True)
```

In [11]:
```
1 data.columns
```

Out[11]:
```
Index(['survived', 'pclass', 'name', 'sex', 'age', 'sibsp', 'parch', 'ticket',
       'fare', 'embarked'],
      dtype='object')
```

In [12]:
```
1 data["age"].dtype
```

Out[12]:
```
dtype('float64')
```

In [13]:
```
1 data["embarked"].dtype
```

Out[13]:
```
dtype('O')
```

In [14]:
```
1 data["age"].mean()
```

Out[14]:
```
29.69911764705882
```

In [15]:
```
1 round(data["age"].mean())
```

Out[15]:
```
30
```

In [16]:
```
1 data["age"] = data["age"].fillna(round(data["age"].mean()))
```

In [17]:
```
1 data.isnull().sum()
```

Out[17]:
```
survived    0
pclass      0
name        0
sex         0
age         0
sibsp       0
parch       0
ticket      0
fare        0
embarked    2
dtype: int64
```

In [18]:
```
1 data["embarked"].value_counts()
```

Out[18]:
```
S    644
C    168
Q     77
Name: embarked, dtype: int64
```

In [19]:
```
1 data["embarked"] = data["embarked"].fillna("S")
```

In [20]:
```
1 data.isnull().sum()
```

Out[20]:
```
survived    0
pclass      0
name        0
sex         0
age         0
sibsp       0
parch       0
ticket      0
fare        0
embarked    0
dtype: int64
```

In [21]:
```
1 data.head()
```

Out[21]:

	survived	pclass	name	sex	age	sibsp	parch	ticket	fare	embarked
0	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
2	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	S
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S
4	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S

In [22]:
```
1 data.drop("name",axis=1,inplace=True)
```

In [23]:
```
1 data.columns
```

Out[23]:
```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'ticket', 'fare',
       'embarked'],
      dtype='object')
```

In [24]:
```
1 data.drop("ticket",axis=1,inplace=True)
```

In [25]:
```
1 data.columns
```

Out[25]:
```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked'],
      dtype='object')
```

In [26]:
```
1 data["sex"].value_counts()
```

Out[26]:
```
male      577
female    314
Name: sex, dtype: int64
```

In [28]:
```
1 data["embarked"].value_counts()
```

Out[28]:
```
S    646
C    168
Q     77
Name: embarked, dtype: int64
```

In [29]:
```
1 from sklearn.preprocessing import LabelEncoder
```

In [30]:
```
1 lab = LabelEncoder()
```

In [32]:
```
1 data["sex"] = lab.fit_transform(data["sex"])
```

In [33]:
```
1 data["sex"]
```

Out[33]:
```
0      1
1      0
2      0
3      0
4      1
5      1
6      1
7      1
8      0
9      0
10     0
11     0
12     1
13     1
14     0
15     0
16     1
17     1
18     0
19     0
20     1
21     1
22     0
23     1
24     0
25     0
26     1
27     1
28     0
29     1
      ..
861    1
862    0
863    0
864    1
865    0
866    0
867    1
868    1
869    1
870    1
871    0
872    1
873    1
874    0
875    0
876    1
877    1
878    1
879    0
880    0
881    1
882    0
883    1
```

```
884    1
885    0
886    1
887    0
888    0
889    1
890    1
Name: sex, Length: 891, dtype: int32
```

In [35]:
```
data["embarked"] = lab.fit_transform(data["embarked"])
```

In [36]:
```
data["embarked"].head()
```

Out[36]:
```
0    2
1    0
2    2
3    2
4    2
Name: embarked, dtype: int32
```

In [37]:
```
data.head()
```

Out[37]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked
0	0	3	1	22.0	1	0	7.2500	2
1	1	1	0	38.0	1	0	71.2833	0
2	1	3	0	26.0	0	0	7.9250	2
3	1	1	0	35.0	1	0	53.1000	2
4	0	3	1	35.0	0	0	8.0500	2

In [38]:
```
# select the features and target
input1 = data.drop("survived",axis=1)
input1.head()
```

Out[38]:

	pclass	sex	age	sibsp	parch	fare	embarked
0	3	1	22.0	1	0	7.2500	2
1	1	0	38.0	1	0	71.2833	0
2	3	0	26.0	0	0	7.9250	2
3	1	0	35.0	1	0	53.1000	2
4	3	1	35.0	0	0	8.0500	2

In [39]:
```
output1 = data["survived"]
```

In [40]:
```
1 data["survived"].value_counts()
```

Out[40]:
```
0    549
1    342
Name: survived, dtype: int64
```

In [41]:
```
1 # split the data for training and testing
```

In [42]:
```
1 from sklearn.model_selection import train_test_split
```

In [43]:
```
1 x_train,x_test,y_train,y_test = train_test_split(input1,output1,test_size=0.
```

In [45]:
```
1 # select the model
2 from sklearn.svm import SVC
```

In [46]:
```
1 help(SVC)
```

```
Help on class SVC in module sklearn.svm.classes:

class SVC(sklearn.svm.base.BaseSVC)
 |  SVC(C=1.0, kernel='rbf', degree=3, gamma='auto_deprecated', coef0=0.0, shri
nking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, ve
rbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None)
 |
 |  C-Support Vector Classification.
 |
 |  The implementation is based on libsvm. The fit time complexity
 |  is more than quadratic with the number of samples which makes it hard
 |  to scale to dataset with more than a couple of 10000 samples.
 |
 |  The multiclass support is handled according to a one-vs-one scheme.
 |
 |  For details on the precise mathematical formulation of the provided
 |  kernel functions and how `gamma`, `coef0` and `degree` affect each
 |  other, see the corresponding section in the narrative documentation:
 |  :ref:`svm_kernels`.
 |
 |  Read more in the :ref:`User Guide <svm_classification>`.
 |
 |  Parameters
 |  ----------
 |  C : float, optional (default=1.0)
 |      Penalty parameter C of the error term.
 |
 |  kernel : string, optional (default='rbf')
 |      Specifies the kernel type to be used in the algorithm.
 |      It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or
 |      a callable.
 |      If none is given, 'rbf' will be used. If a callable is given it is
 |      used to pre-compute the kernel matrix from data matrices; that matrix
 |      should be an array of shape ``(n_samples, n_samples)``.
 |
 |  degree : int, optional (default=3)
 |      Degree of the polynomial kernel function ('poly').
 |      Ignored by all other kernels.
 |
 |  gamma : float, optional (default='auto')
 |      Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.
 |
 |      Current default is 'auto' which uses 1 / n_features,
 |      if ``gamma='scale'`` is passed then it uses 1 / (n_features * X.var())
 |      as value of gamma. The current default of gamma, 'auto', will change
 |      to 'scale' in version 0.22. 'auto_deprecated', a deprecated version of
 |      'auto' is used as a default indicating that no explicit value of gamma
 |      was passed.
 |
 |  coef0 : float, optional (default=0.0)
 |      Independent term in kernel function.
 |      It is only significant in 'poly' and 'sigmoid'.
 |
 |  shrinking : boolean, optional (default=True)
```

```
|      Whether to use the shrinking heuristic.
|
|  probability : boolean, optional (default=False)
|      Whether to enable probability estimates. This must be enabled prior
|      to calling `fit`, and will slow down that method.
|
|  tol : float, optional (default=1e-3)
|      Tolerance for stopping criterion.
|
|  cache_size : float, optional
|      Specify the size of the kernel cache (in MB).
|
|  class_weight : {dict, 'balanced'}, optional
|      Set the parameter C of class i to class_weight[i]*C for
|      SVC. If not given, all classes are supposed to have
|      weight one.
|      The "balanced" mode uses the values of y to automatically adjust
|      weights inversely proportional to class frequencies in the input data
|      as ``n_samples / (n_classes * np.bincount(y))``
|
|  verbose : bool, default: False
|      Enable verbose output. Note that this setting takes advantage of a
|      per-process runtime setting in libsvm that, if enabled, may not work
|      properly in a multithreaded context.
|
|  max_iter : int, optional (default=-1)
|      Hard limit on iterations within solver, or -1 for no limit.
|
|  decision_function_shape : 'ovo', 'ovr', default='ovr'
|      Whether to return a one-vs-rest ('ovr') decision function of shape
|      (n_samples, n_classes) as all other classifiers, or the original
|      one-vs-one ('ovo') decision function of libsvm which has shape
|      (n_samples, n_classes * (n_classes - 1) / 2). However, one-vs-one
|      ('ovo') is always used as multi-class strategy.
|
|      .. versionchanged:: 0.19
|          decision_function_shape is 'ovr' by default.
|
|      .. versionadded:: 0.17
|         *decision_function_shape='ovr'* is recommended.
|
|      .. versionchanged:: 0.17
|         Deprecated *decision_function_shape='ovo' and None*.
|
|  random_state : int, RandomState instance or None, optional (default=None)
|      The seed of the pseudo random number generator used when shuffling
|      the data for probability estimates. If int, random_state is the
|      seed used by the random number generator; If RandomState instance,
|      random_state is the random number generator; If None, the random
|      number generator is the RandomState instance used by `np.random`.
|
|  Attributes
|  ----------
|  support_ : array-like, shape = [n_SV]
|      Indices of support vectors.
|
|  support_vectors_ : array-like, shape = [n_SV, n_features]
```

```
|      Support vectors.
|
|  n_support_ : array-like, dtype=int32, shape = [n_class]
|      Number of support vectors for each class.
|
|  dual_coef_ : array, shape = [n_class-1, n_SV]
|      Coefficients of the support vector in the decision function.
|      For multiclass, coefficient for all 1-vs-1 classifiers.
|      The layout of the coefficients in the multiclass case is somewhat
|      non-trivial. See the section about multi-class classification in the
|      SVM section of the User Guide for details.
|
|  coef_ : array, shape = [n_class * (n_class-1) / 2, n_features]
|      Weights assigned to the features (coefficients in the primal
|      problem). This is only available in the case of a linear kernel.
|
|      `coef_` is a readonly property derived from `dual_coef_` and
|      `support_vectors_`.
|
|  intercept_ : array, shape = [n_class * (n_class-1) / 2]
|      Constants in decision function.
|
|  fit_status_ : int
|      0 if correctly fitted, 1 otherwise (will raise warning)
|
|  probA_ : array, shape = [n_class * (n_class-1) / 2]
|  probB_ : array, shape = [n_class * (n_class-1) / 2]
|      If probability=True, the parameters learned in Platt scaling to
|      produce probability estimates from decision values. If
|      probability=False, an empty array. Platt scaling uses the logistic
|      function
|      ``1 / (1 + exp(decision_value * probA_ + probB_))``
|      where ``probA_`` and ``probB_`` are learned from the dataset [2]_. For
|      more information on the multiclass case and training procedure see
|      section 8 of [1]_.
|
|  Examples
|  --------
|  >>> import numpy as np
|  >>> X = np.array([[-1, -1], [-2, -1], [1, 1], [2, 1]])
|  >>> y = np.array([1, 1, 2, 2])
|  >>> from sklearn.svm import SVC
|  >>> clf = SVC(gamma='auto')
|  >>> clf.fit(X, y) #doctest: +NORMALIZE_WHITESPACE
|  SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
|      decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
|      max_iter=-1, probability=False, random_state=None, shrinking=True,
|      tol=0.001, verbose=False)
|  >>> print(clf.predict([[-0.8, -1]]))
|  [1]
|
|  See also
|  --------
|  SVR
|      Support Vector Machine for Regression implemented using libsvm.
|
|  LinearSVC
```

```
|      Scalable Linear Support Vector Machine for classification
|      implemented using liblinear. Check the See also section of
|      LinearSVC for more comparison element.
|
|  References
|  ----------
|  .. [1] `LIBSVM: A Library for Support Vector Machines
|      <http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf>`_
|
|  .. [2] `Platt, John (1999). "Probabilistic outputs for support vector
|      machines and comparison to regularizedlikelihood methods."
|      <http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.1639>`_
|
|  Method resolution order:
|      SVC
|      sklearn.svm.base.BaseSVC
|      abc.NewBase
|      sklearn.svm.base.BaseLibSVM
|      abc.NewBase
|      sklearn.base.BaseEstimator
|      sklearn.base.ClassifierMixin
|      builtins.object
|
|  Methods defined here:
|
|  __init__(self, C=1.0, kernel='rbf', degree=3, gamma='auto_deprecated', coef
0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weig
ht=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_stat
e=None)
|      Initialize self.  See help(type(self)) for accurate signature.
|
|  ----------------------------------------------------------------------
|  Data and other attributes defined here:
|
|  __abstractmethods__ = frozenset()
|
|  ----------------------------------------------------------------------
|  Methods inherited from sklearn.svm.base.BaseSVC:
|
|  decision_function(self, X)
|      Evaluates the decision function for the samples in X.
|
|      Parameters
|      ----------
|      X : array-like, shape (n_samples, n_features)
|
|      Returns
|      -------
|      X : array-like, shape (n_samples, n_classes * (n_classes-1) / 2)
|          Returns the decision function of the sample for each class
|          in the model.
|          If decision_function_shape='ovr', the shape is (n_samples,
|          n_classes).
|
|      Notes
|      ------
|      If decision_function_shape='ovo', the function values are proportional
```

```
|      to the distance of the samples X to the separating hyperplane. If the
|      exact distances are required, divide the function values by the norm of
|      the weight vector (``coef_``). See also `this question
|      <https://stats.stackexchange.com/questions/14876/
|      interpreting-distance-from-hyperplane-in-svm>`_ for further details.
|
|  predict(self, X)
|      Perform classification on samples in X.
|
|      For an one-class model, +1 or -1 is returned.
|
|      Parameters
|      ----------
|      X : {array-like, sparse matrix}, shape (n_samples, n_features)
|          For kernel="precomputed", the expected shape of X is
|          [n_samples_test, n_samples_train]
|
|      Returns
|      -------
|      y_pred : array, shape (n_samples,)
|          Class labels for samples in X.
|
|  ----------------------------------------------------------------------
|  Data descriptors inherited from sklearn.svm.base.BaseSVC:
|
|  predict_log_proba
|      Compute log probabilities of possible outcomes for samples in X.
|
|      The model need to have probability information computed at training
|      time: fit with attribute `probability` set to True.
|
|      Parameters
|      ----------
|      X : array-like, shape (n_samples, n_features)
|          For kernel="precomputed", the expected shape of X is
|          [n_samples_test, n_samples_train]
|
|      Returns
|      -------
|      T : array-like, shape (n_samples, n_classes)
|          Returns the log-probabilities of the sample for each class in
|          the model. The columns correspond to the classes in sorted
|          order, as they appear in the attribute `classes_`.
|
|      Notes
|      -----
|      The probability model is created using cross validation, so
|      the results can be slightly different than those obtained by
|      predict. Also, it will produce meaningless results on very small
|      datasets.
|
|  predict_proba
|      Compute probabilities of possible outcomes for samples in X.
|
|      The model need to have probability information computed at training
|      time: fit with attribute `probability` set to True.
|
```

```
|      Parameters
|      ----------
|      X : array-like, shape (n_samples, n_features)
|          For kernel="precomputed", the expected shape of X is
|          [n_samples_test, n_samples_train]
|
|      Returns
|      -------
|      T : array-like, shape (n_samples, n_classes)
|          Returns the probability of the sample for each class in
|          the model. The columns correspond to the classes in sorted
|          order, as they appear in the attribute `classes_`.
|
|      Notes
|      -----
|      The probability model is created using cross validation, so
|      the results can be slightly different than those obtained by
|      predict. Also, it will produce meaningless results on very small
|      datasets.
|
|  ----------------------------------------------------------------------
|  Methods inherited from sklearn.svm.base.BaseLibSVM:
|
|  fit(self, X, y, sample_weight=None)
|      Fit the SVM model according to the given training data.
|
|      Parameters
|      ----------
|      X : {array-like, sparse matrix}, shape (n_samples, n_features)
|          Training vectors, where n_samples is the number of samples
|          and n_features is the number of features.
|          For kernel="precomputed", the expected shape of X is
|          (n_samples, n_samples).
|
|      y : array-like, shape (n_samples,)
|          Target values (class labels in classification, real numbers in
|          regression)
|
|      sample_weight : array-like, shape (n_samples,)
|          Per-sample weights. Rescale C per sample. Higher weights
|          force the classifier to put more emphasis on these points.
|
|      Returns
|      -------
|      self : object
|
|      Notes
|      ------
|      If X and y are not C-ordered and contiguous arrays of np.float64 and
|      X is not a scipy.sparse.csr_matrix, X and/or y may be copied.
|
|      If X is a dense array, then the other methods will not support sparse
|      matrices as input.
|
|  ----------------------------------------------------------------------
|  Data descriptors inherited from sklearn.svm.base.BaseLibSVM:
|
```

```
|  coef_
|
|  ----------------------------------------------------------------------
|  Methods inherited from sklearn.base.BaseEstimator:
|
|  __getstate__(self)
|
|  __repr__(self)
|      Return repr(self).
|
|  __setstate__(self, state)
|
|  get_params(self, deep=True)
|      Get parameters for this estimator.
|
|      Parameters
|      ----------
|      deep : boolean, optional
|          If True, will return the parameters for this estimator and
|          contained subobjects that are estimators.
|
|      Returns
|      -------
|      params : mapping of string to any
|          Parameter names mapped to their values.
|
|  set_params(self, **params)
|      Set the parameters of this estimator.
|
|      The method works on simple estimators as well as on nested objects
|      (such as pipelines). The latter have parameters of the form
|      ``<component>__<parameter>`` so that it's possible to update each
|      component of a nested object.
|
|      Returns
|      -------
|      self
|
|  ----------------------------------------------------------------------
|  Data descriptors inherited from sklearn.base.BaseEstimator:
|
|  __dict__
|      dictionary for instance variables (if defined)
|
|  __weakref__
|      list of weak references to the object (if defined)
|
|  ----------------------------------------------------------------------
|  Methods inherited from sklearn.base.ClassifierMixin:
|
|  score(self, X, y, sample_weight=None)
|      Returns the mean accuracy on the given test data and labels.
|
|      In multi-label classification, this is the subset accuracy
|      which is a harsh metric since you require for each sample that
|      each label set be correctly predicted.
|
```

```
|       Parameters
|       ----------
|       X : array-like, shape = (n_samples, n_features)
|           Test samples.
|
|       y : array-like, shape = (n_samples) or (n_samples, n_outputs)
|           True labels for X.
|
|       sample_weight : array-like, shape = [n_samples], optional
|           Sample weights.
|
|       Returns
|       -------
|       score : float
|           Mean accuracy of self.predict(X) wrt. y.
```

In [48]:
```
1 sv = SVC(kernel ="linear")
```

In [49]:
```
1 sv.fit(x_train,y_train)
```

Out[49]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='linear', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
```

In [56]:
```
1 p = sv.predict(x_test)
```

In [57]:
```
1 p
```

Out[57]:
```
array([0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0,
       0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 1], dtype=int64)
```

In [58]:
```
1 from sklearn.metrics import accuracy_score,confusion_matrix,classification_r
```

In [60]:
```
1 accuracy_score(y_test,p)
```

Out[60]:
```
0.7686567164179104
```

In [61]:
```
1 confusion_matrix(y_test,p)
```

Out[61]:
```
array([[139,  21],
       [ 41,  67]], dtype=int64)
```

In [62]:
```
1 print(classification_report(y_test,p))
```

```
              precision    recall  f1-score   support

           0       0.77      0.87      0.82       160
           1       0.76      0.62      0.68       108

   micro avg       0.77      0.77      0.77       268
   macro avg       0.77      0.74      0.75       268
weighted avg       0.77      0.77      0.76       268
```

Decision tree Classifier

In [63]:
```
1 import pandas as pd
```

In [64]:
```
1 from sklearn.datasets import load_iris
```

In [65]:
```
1 iris = load_iris()
```

In [66]:
```
1 iris
```

Out[66]:
```
{'data': array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
       [4.7, 3.2, 1.6, 0.2],
       [4.8, 3.1, 1.6, 0.2],
       [5.4, 3.4, 1.5, 0.4],
       [5.2, 4.1, 1.5, 0.1],
       [5.5, 4.2, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 3.5, 1.3, 0.2],
       [4.9, 3.6, 1.4, 0.1],
       [4.4, 3. , 1.3, 0.2],
       [5.1, 3.4, 1.5, 0.2],
       [5. , 3.5, 1.3, 0.3],
       [4.5, 2.3, 1.3, 0.3],
       [4.4, 3.2, 1.3, 0.2],
       [5. , 3.5, 1.6, 0.6],
       [5.1, 3.8, 1.9, 0.4],
       [4.8, 3. , 1.4, 0.3],
       [5.1, 3.8, 1.6, 0.2],
       [4.6, 3.2, 1.4, 0.2],
       [5.3, 3.7, 1.5, 0.2],
       [5. , 3.3, 1.4, 0.2],
       [7. , 3.2, 4.7, 1.4],
       [6.4, 3.2, 4.5, 1.5],
       [6.9, 3.1, 4.9, 1.5],
       [5.5, 2.3, 4. , 1.3],
```

```
[6.5, 2.8, 4.6, 1.5],
[5.7, 2.8, 4.5, 1.3],
[6.3, 3.3, 4.7, 1.6],
[4.9, 2.4, 3.3, 1. ],
[6.6, 2.9, 4.6, 1.3],
[5.2, 2.7, 3.9, 1.4],
[5. , 2. , 3.5, 1. ],
[5.9, 3. , 4.2, 1.5],
[6. , 2.2, 4. , 1. ],
[6.1, 2.9, 4.7, 1.4],
[5.6, 2.9, 3.6, 1.3],
[6.7, 3.1, 4.4, 1.4],
[5.6, 3. , 4.5, 1.5],
[5.8, 2.7, 4.1, 1. ],
[6.2, 2.2, 4.5, 1.5],
[5.6, 2.5, 3.9, 1.1],
[5.9, 3.2, 4.8, 1.8],
[6.1, 2.8, 4. , 1.3],
[6.3, 2.5, 4.9, 1.5],
[6.1, 2.8, 4.7, 1.2],
[6.4, 2.9, 4.3, 1.3],
[6.6, 3. , 4.4, 1.4],
[6.8, 2.8, 4.8, 1.4],
[6.7, 3. , 5. , 1.7],
[6. , 2.9, 4.5, 1.5],
[5.7, 2.6, 3.5, 1. ],
[5.5, 2.4, 3.8, 1.1],
[5.5, 2.4, 3.7, 1. ],
[5.8, 2.7, 3.9, 1.2],
[6. , 2.7, 5.1, 1.6],
[5.4, 3. , 4.5, 1.5],
[6. , 3.4, 4.5, 1.6],
[6.7, 3.1, 4.7, 1.5],
[6.3, 2.3, 4.4, 1.3],
[5.6, 3. , 4.1, 1.3],
[5.5, 2.5, 4. , 1.3],
[5.5, 2.6, 4.4, 1.2],
[6.1, 3. , 4.6, 1.4],
[5.8, 2.6, 4. , 1.2],
[5. , 2.3, 3.3, 1. ],
[5.6, 2.7, 4.2, 1.3],
[5.7, 3. , 4.2, 1.2],
[5.7, 2.9, 4.2, 1.3],
[6.2, 2.9, 4.3, 1.3],
[5.1, 2.5, 3. , 1.1],
[5.7, 2.8, 4.1, 1.3],
[6.3, 3.3, 6. , 2.5],
[5.8, 2.7, 5.1, 1.9],
[7.1, 3. , 5.9, 2.1],
[6.3, 2.9, 5.6, 1.8],
[6.5, 3. , 5.8, 2.2],
[7.6, 3. , 6.6, 2.1],
[4.9, 2.5, 4.5, 1.7],
[7.3, 2.9, 6.3, 1.8],
[6.7, 2.5, 5.8, 1.8],
[7.2, 3.6, 6.1, 2.5],
[6.5, 3.2, 5.1, 2. ],
```

```
        [6.4, 2.7, 5.3, 1.9],
        [6.8, 3. , 5.5, 2.1],
        [5.7, 2.5, 5. , 2. ],
        [5.8, 2.8, 5.1, 2.4],
        [6.4, 3.2, 5.3, 2.3],
        [6.5, 3. , 5.5, 1.8],
        [7.7, 3.8, 6.7, 2.2],
        [7.7, 2.6, 6.9, 2.3],
        [6. , 2.2, 5. , 1.5],
        [6.9, 3.2, 5.7, 2.3],
        [5.6, 2.8, 4.9, 2. ],
        [7.7, 2.8, 6.7, 2. ],
        [6.3, 2.7, 4.9, 1.8],
        [6.7, 3.3, 5.7, 2.1],
        [7.2, 3.2, 6. , 1.8],
        [6.2, 2.8, 4.8, 1.8],
        [6.1, 3. , 4.9, 1.8],
        [6.4, 2.8, 5.6, 2.1],
        [7.2, 3. , 5.8, 1.6],
        [7.4, 2.8, 6.1, 1.9],
        [7.9, 3.8, 6.4, 2. ],
        [6.4, 2.8, 5.6, 2.2],
        [6.3, 2.8, 5.1, 1.5],
        [6.1, 2.6, 5.6, 1.4],
        [7.7, 3. , 6.1, 2.3],
        [6.3, 3.4, 5.6, 2.4],
        [6.4, 3.1, 5.5, 1.8],
        [6. , 3. , 4.8, 1.8],
        [6.9, 3.1, 5.4, 2.1],
        [6.7, 3.1, 5.6, 2.4],
        [6.9, 3.1, 5.1, 2.3],
        [5.8, 2.7, 5.1, 1.9],
        [6.8, 3.2, 5.9, 2.3],
        [6.7, 3.3, 5.7, 2.5],
        [6.7, 3. , 5.2, 2.3],
        [6.3, 2.5, 5. , 1.9],
        [6.5, 3. , 5.2, 2. ],
        [6.2, 3.4, 5.4, 2.3],
        [5.9, 3. , 5.1, 1.8]]),
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]),
 'target_names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'),
 'DESCR': '.. _iris_dataset:\n\nIris plants dataset\n--------------------\n\n**
Data Set Characteristics:**\n\n    :Number of Instances: 150 (50 in each of thr
ee classes)\n    :Number of Attributes: 4 numeric, predictive attributes and th
e class\n    :Attribute Information:\n        - sepal length in cm\n        - s
epal width in cm\n        - petal length in cm\n        - petal width in cm\n
- class:\n                - Iris-Setosa\n                - Iris-Versicolour\n
- Iris-Virginica\n                \n    :Summary Statistics:\n\n    ===========
=== ==== ==== ======= ===== ====================\n                    Min  Max
Mean    SD   Class Correlation\n    ============== ==== ==== ======= ===== ====
```

```
==========\n    sepal length:   4.3  7.9   5.84   0.83    0.7826\n    sep
al width:    2.0  4.4   3.05   0.43   -0.4194\n    petal length:   1.0  6.9
3.76   1.76    0.9490  (high!)\n    petal width:    0.1  2.5   1.20   0.76
0.9565  (high!)\n    ============== ==== ==== ======= ===== ====================
=\n\n    :Missing Attribute Values: None\n    :Class Distribution: 33.3% for ea
ch of 3 classes.\n    :Creator: R.A. Fisher\n    :Donor: Michael Marshall (MARS
HALL%PLU@io.arc.nasa.gov)\n    :Date: July, 1988\n\nThe famous Iris database, f
irst used by Sir R.A. Fisher. The dataset is taken\nfrom Fisher\'s paper. Note
that it\'s the same as in R, but not as in the UCI\nMachine Learning Repositor
y, which has two wrong data points.\n\nThis is perhaps the best known database
to be found in the\npattern recognition literature.  Fisher\'s paper is a class
ic in the field and\nis referenced frequently to this day.  (See Duda & Hart, f
or example.)  The\ndata set contains 3 classes of 50 instances each, where each
class refers to a\ntype of iris plant.  One class is linearly separable from th
e other 2; the\nlatter are NOT linearly separable from each other.\n\n.. topi
c:: References\n\n   - Fisher, R.A. "The use of multiple measurements in taxono
mic problems"\n     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contr
ibutions to\n     Mathematical Statistics" (John Wiley, NY, 1950).\n   - Duda,
R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.\n     (Q32
7.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.\n   - Dasarathy,
B.V. (1980) "Nosing Around the Neighborhood: A New System\n     Structure and C
lassification Rule for Recognition in Partially Exposed\n     Environments".  I
EEE Transactions on Pattern Analysis and Machine\n     Intelligence, Vol. PAMI-
2, No. 1, 67-71.\n   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".
IEEE Transactions\n     on Information Theory, May 1972, 431-433.\n   - See als
o: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II\n     conceptua
l clustering system finds 3 classes in the data.\n   - Many, many more ...',
 'feature_names': ['sepal length (cm)',
  'sepal width (cm)',
  'petal length (cm)',
  'petal width (cm)'],
 'filename': 'C:\\Users\\Alekhya\\Anaconda3\\lib\\site-packages\\sklearn\\datas
ets\\data\\iris.csv'}
```


In [67]:
```
input_data = pd.DataFrame(iris.data,columns=['sepal length (cm)','sepal widt
                                             'petal length (cm)','petal widt
```

In [68]:
```
input_data.head()
```

Out[68]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [69]:
```
output_data = pd.DataFrame(iris.target,columns = ["target"])
```

In [70]:
```
1 output_data.head()
```

Out[70]:

	target
0	0
1	0
2	0
3	0
4	0

In [71]:
```
1 input_data.shape
```

Out[71]:
```
(150, 4)
```

In [72]:
```
1 output_data.shape
```

Out[72]:
```
(150, 1)
```

In [73]:
```
1 input_data.isnull().sum()
```

Out[73]:
```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
dtype: int64
```

In [75]:
```
1 output_data.isnull().sum()
```

Out[75]:
```
target    0
dtype: int64
```

In [76]:
```
1 input_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
sepal length (cm)    150 non-null float64
sepal width (cm)     150 non-null float64
petal length (cm)    150 non-null float64
petal width (cm)     150 non-null float64
dtypes: float64(4)
memory usage: 4.8 KB
```

In [77]:
```
output_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 1 columns):
target    150 non-null int32
dtypes: int32(1)
memory usage: 680.0 bytes
```

In [78]:
```
# spliting the data for training and testing
```

In [79]:
```
from sklearn.model_selection import train_test_split
```

In [80]:
```
x_train,x_test,y_train,y_test = train_test_split(input_data,output_data,
                                                 test_size=0.3,random_state=
```

In [81]:
```
# select the model
```

In [82]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [83]:
```
dtc = DecisionTreeClassifier()
```

In [84]:
```
dtc.fit(x_train,y_train)
```

Out[84]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

In [85]:
```
#predict the data
```

In [86]:
```
pred = dtc.predict(x_test)
```

In [87]:
```
pred
```

Out[87]:
```
array([0, 0, 0, 0, 0, 2, 1, 0, 2, 1, 1, 0, 1, 1, 2, 0, 1, 2, 2, 0, 2, 2,
       2, 1, 0, 2, 2, 1, 1, 1, 0, 0, 2, 1, 0, 0, 2, 0, 2, 1, 2, 1, 0, 0,
       2])
```

In [88]:
```
from sklearn.metrics import accuracy_score,classification_report,confusion_m
```

In [89]:
```
accuracy_score(y_test,pred)
```

Out[89]:
```
0.9777777777777777
```

In [90]:
```
1 confusion_matrix(y_test,pred)
```

Out[90]:
```
array([[17,  0,  0],
       [ 0, 13,  1],
       [ 0,  0, 14]], dtype=int64)
```

In [92]:
```
1 print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       1.00      0.93      0.96        14
           2       0.93      1.00      0.97        14

   micro avg       0.98      0.98      0.98        45
   macro avg       0.98      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

Graphviz

Pydotplus

- conda install graphviz
- conda install pydotplus

In [93]:
```
1 conda install graphviz
```

```
Collecting package metadata (repodata.json): ...working... done
Solving environment: ...working... done

## Package Plan ##

  environment location: C:\Users\Alekhya\Anaconda3

  added / updated specs:
    - graphviz


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    ca-certificates-2021.7.5   |       haa95532_1         149 KB
    certifi-2021.5.30          |   py37haa95532_0         142 KB
    conda-4.10.3               |   py37haa95532_0         3.1 MB
    ------------------------------------------------------------
                                           Total:         3.4 MB

The following packages will be UPDATED:

  ca-certificates                     2021.5.25-haa95532_1 --> 2021.7.5-haa955
32_1
  certifi                         2020.12.5-py37haa95532_0 --> 2021.5.30-py37h
aa95532_0
  conda                             4.10.1-py37haa95532_1 --> 4.10.3-py37haa9
5532_0



Downloading and Extracting Packages

ca-certificates-2021 | 149 KB    |            |   0%
ca-certificates-2021 | 149 KB    | #          |  11%
ca-certificates-2021 | 149 KB    | ####3      |  43%
ca-certificates-2021 | 149 KB    | #######5   |  75%
ca-certificates-2021 | 149 KB    | ########## | 100%
ca-certificates-2021 | 149 KB    | ########## | 100%

conda-4.10.3         | 3.1 MB    |            |   0%
conda-4.10.3         | 3.1 MB    |            |   1%
conda-4.10.3         | 3.1 MB    | 2          |   2%
conda-4.10.3         | 3.1 MB    | 3          |   4%
conda-4.10.3         | 3.1 MB    | 5          |   6%
conda-4.10.3         | 3.1 MB    | 8          |   8%
conda-4.10.3         | 3.1 MB    | #          |  10%
conda-4.10.3         | 3.1 MB    | #2         |  12%
conda-4.10.3         | 3.1 MB    | #4         |  15%
conda-4.10.3         | 3.1 MB    | #7         |  17%
conda-4.10.3         | 3.1 MB    | #9         |  20%
conda-4.10.3         | 3.1 MB    | ##2        |  22%
conda-4.10.3         | 3.1 MB    | ##4        |  25%
```

```
conda-4.10.3          | 3.1 MB    | ##7         |  27%
conda-4.10.3          | 3.1 MB    | ##9         |  30%
conda-4.10.3          | 3.1 MB    | ###1        |  32%
conda-4.10.3          | 3.1 MB    | ###3        |  34%
conda-4.10.3          | 3.1 MB    | ###6        |  36%
conda-4.10.3          | 3.1 MB    | ###8        |  38%
conda-4.10.3          | 3.1 MB    | ####        |  40%
conda-4.10.3          | 3.1 MB    | ####2       |  42%
conda-4.10.3          | 3.1 MB    | ####4       |  45%
conda-4.10.3          | 3.1 MB    | ####8       |  49%
conda-4.10.3          | 3.1 MB    | #####1      |  51%
conda-4.10.3          | 3.1 MB    | #####3      |  54%
conda-4.10.3          | 3.1 MB    | #####5      |  56%
conda-4.10.3          | 3.1 MB    | #####7      |  58%
conda-4.10.3          | 3.1 MB    | ######      |  60%
conda-4.10.3          | 3.1 MB    | ######2     |  63%
conda-4.10.3          | 3.1 MB    | ######4     |  65%
conda-4.10.3          | 3.1 MB    | ######7     |  67%
conda-4.10.3          | 3.1 MB    | #######     |  70%
conda-4.10.3          | 3.1 MB    | #######4    |  74%
conda-4.10.3          | 3.1 MB    | #######7    |  77%
conda-4.10.3          | 3.1 MB    | #######9    |  80%
conda-4.10.3          | 3.1 MB    | ########2   |  82%
conda-4.10.3          | 3.1 MB    | ########4   |  85%
conda-4.10.3          | 3.1 MB    | ########7   |  87%
conda-4.10.3          | 3.1 MB    | #########   |  91%
conda-4.10.3          | 3.1 MB    | #########3  |  94%
conda-4.10.3          | 3.1 MB    | #########7  |  97%
conda-4.10.3          | 3.1 MB    | ########## | 100%
conda-4.10.3          | 3.1 MB    | ########## | 100%

certifi-2021.5.30     | 142 KB    |             |   0%
certifi-2021.5.30     | 142 KB    | #1          |  11%
certifi-2021.5.30     | 142 KB    | #####6      |  56%
certifi-2021.5.30     | 142 KB    | ########## | 100%
certifi-2021.5.30     | 142 KB    | ########## | 100%
Preparing transaction: ...working... done
Verifying transaction: ...working... done
Executing transaction: ...working... done

Note: you may need to restart the kernel to use updated packages.
```

In [94]:
```
1 conda install pydotplus
```

```
Collecting package metadata (repodata.json): ...working... done
Solving environment: ...working... done

# All requested packages already installed.


Note: you may need to restart the kernel to use updated packages.
```

In [95]:
```
1 from sklearn.tree import export_graphviz
```

In [96]:
```
1 import pydotplus
```

In [97]:
```
1 from IPython.display import Image
```

In [98]:
```
1 from sklearn.externals.six import StringIO
```

In [101]:
```
1 dot_data = StringIO()
2 export_graphviz(dtc,out_file = "iris.dot",feature_names = iris.feature_names
3                 class_names = iris.target_names,
4                 rounded = True,filled = True)
```

In [100]:
```
1 help(export_graphviz)
```

```
Help on function export_graphviz in module sklearn.tree.export:

export_graphviz(decision_tree, out_file=None, max_depth=None, feature_names=Non
e, class_names=None, label='all', filled=False, leaves_parallel=False, impurity
=True, node_ids=False, proportion=False, rotate=False, rounded=False, special_c
haracters=False, precision=3)
    Export a decision tree in DOT format.

    This function generates a GraphViz representation of the decision tree,
    which is then written into `out_file`. Once exported, graphical renderings
    can be generated using, for example::

        $ dot -Tps tree.dot -o tree.ps      (PostScript format)
        $ dot -Tpng tree.dot -o tree.png    (PNG format)

    The sample counts that are shown are weighted with any sample_weights that
    might be present.

    Read more in the :ref:`User Guide <tree>`.

    Parameters
    ----------
    decision_tree : decision tree regressor or classifier
        The decision tree to be exported to GraphViz.

    out_file : file object or string, optional (default=None)
        Handle or name of the output file. If ``None``, the result is
        returned as a string.

        .. versionchanged:: 0.20
            Default of out_file changed from "tree.dot" to None.

    max_depth : int, optional (default=None)
        The maximum depth of the representation. If None, the tree is fully
        generated.

    feature_names : list of strings, optional (default=None)
        Names of each of the features.

    class_names : list of strings, bool or None, optional (default=None)
        Names of each of the target classes in ascending numerical order.
        Only relevant for classification and not supported for multi-output.
        If ``True``, shows a symbolic representation of the class name.

    label : {'all', 'root', 'none'}, optional (default='all')
        Whether to show informative labels for impurity, etc.
        Options include 'all' to show at every node, 'root' to show only at
        the top root node, or 'none' to not show at any node.

    filled : bool, optional (default=False)
        When set to ``True``, paint nodes to indicate majority class for
        classification, extremity of values for regression, or purity of node
        for multi-output.
```

```
leaves_parallel : bool, optional (default=False)
    When set to ``True``, draw all leaf nodes at the bottom of the tree.

impurity : bool, optional (default=True)
    When set to ``True``, show the impurity at each node.

node_ids : bool, optional (default=False)
    When set to ``True``, show the ID number on each node.

proportion : bool, optional (default=False)
    When set to ``True``, change the display of 'values' and/or 'samples'
    to be proportions and percentages respectively.

rotate : bool, optional (default=False)
    When set to ``True``, orient tree left to right rather than top-down.

rounded : bool, optional (default=False)
    When set to ``True``, draw node boxes with rounded corners and use
    Helvetica fonts instead of Times-Roman.

special_characters : bool, optional (default=False)
    When set to ``False``, ignore special characters for PostScript
    compatibility.

precision : int, optional (default=3)
    Number of digits of precision for floating point in the values of
    impurity, threshold and value attributes of each node.

Returns
-------
dot_data : string
    String representation of the input tree in GraphViz dot format.
    Only returned if ``out_file`` is None.

    .. versionadded:: 0.18

Examples
--------
>>> from sklearn.datasets import load_iris
>>> from sklearn import tree

>>> clf = tree.DecisionTreeClassifier()
>>> iris = load_iris()

>>> clf = clf.fit(iris.data, iris.target)
>>> tree.export_graphviz(clf,
...     out_file='tree.dot')                # doctest: +SKIP
```

In [102]:

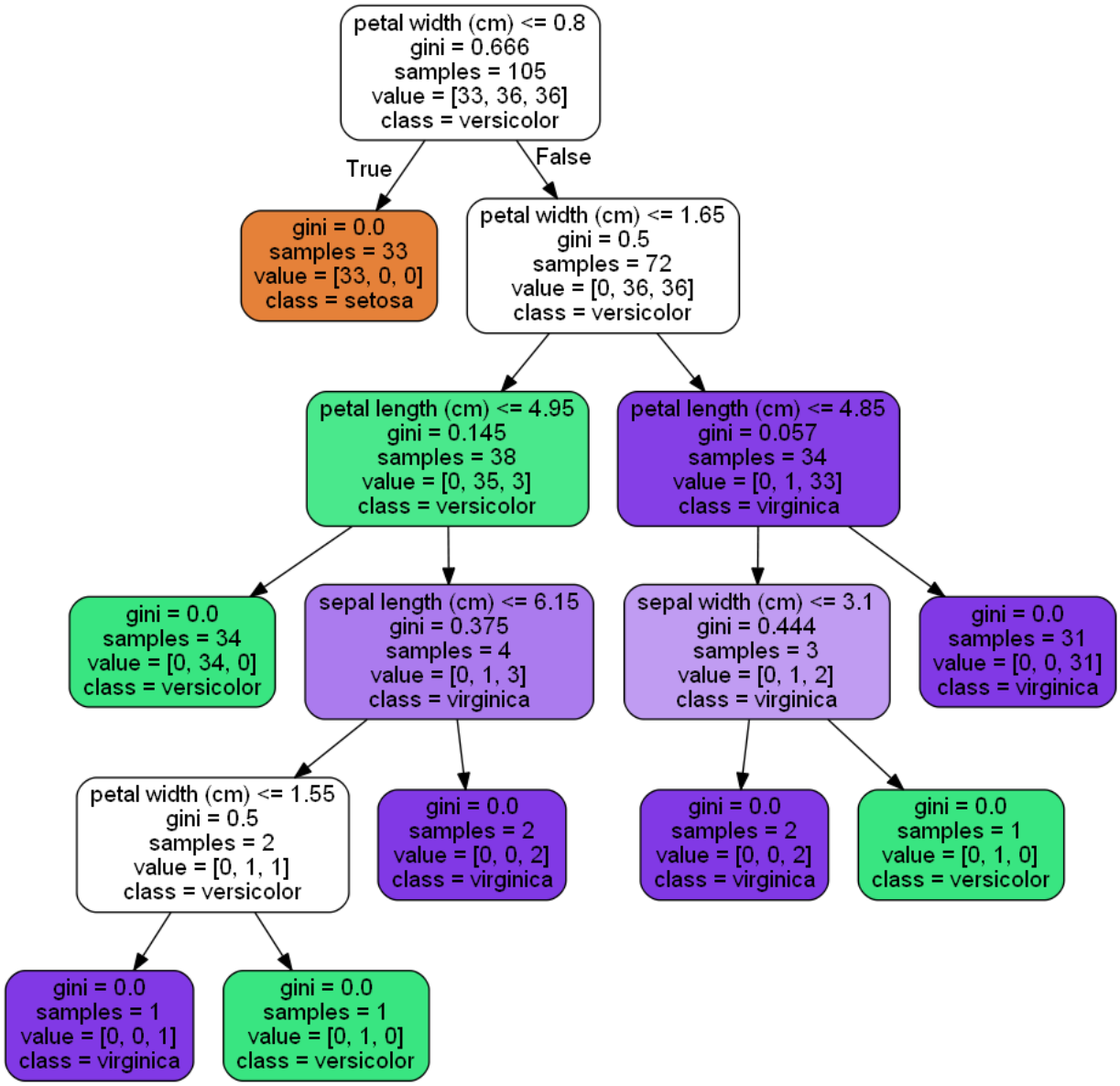
```
dot_data1 = StringIO()
export_graphviz(dtc,out_file =dot_data1,feature_names = iris.feature_names,
                class_names = iris.target_names,
                rounded = True,filled = True)
```

In [106]:
```
puplot = pydotplus.graph_from_dot_data(dot_data1.getvalue())
print(puplot)
```

```
<pydotplus.graphviz.Dot object at 0x000001BEFE6E2470>
```

In [104]:
```
Image(puplot.create_png())
```

Out[104]:



In [ ]:
```

```