Task 2: Apply algorithm on breast cancer wisconsin dataset - One Hot Encoding of features: and Train test Division 50%-50%

```
from sklearn.datasets import load_breast_cancer
d = load_breast_cancer()
list(d.target_names)
```

```
['malignant', 'benign']
```

```
# print the names of the 13 features
print("Features: ", d.feature_names)

# print the label type of wine(class_0, class_1, class_2)
print("Labels: ", d.target_names)

# print data(feature)shape
print("\nData shape: ",d.data.shape)
#print data(target)shape
print("\nTraget shape: ",d.target.shape)

print("\nData type: ",type(d.data))
```

```
Features:  ['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
Labels:  ['malignant' 'benign']

Data shape:  (569, 30)

Traget shape:  (569,)

Data type:  <class 'numpy.ndarray'>
```

```
# all features are numeric, not categorial so directly splitting it
#import the necessary module
from sklearn.model_selection import train_test_split
#split data set into train and test sets
data_train, data_test, target_train, target_test = train_test_split(d.data,
                        d.target, test_size = 0.50, random_state = 142)
```

```
import numpy as np
gnb = GaussianNB()
```

```
gnb = GaussianNB()

#Train the model using the training sets
gnb.fit(data_train, target_train)

#Predict the response for test dataset
target_pred = gnb.predict(data_test)


#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(target_test, target_pred))
```

```
Accuracy: 0.9368421052631579
```

```
#Import confusion_matrix from scikit-learn metrics module for confusion_matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(target_test, target_pred)
```

```
array([[ 94,  13],
       [  5, 173]])
```

```
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score

precision = precision_score(target_test, target_pred)
recall = recall_score(target_test, target_pred)

print('precision: {}'.format(precision))
print('recall: {}'.format(recall))
```

```
precision: 0.9301075268817204
recall: 0.9719101123595506
```