# Chapter 8 - Exercise 2: Breast cancer

## Sử dụng tập dữ liệu ung thư, một vấn đề phân loại nhiều lớp rất nổi tiếng. Số liệu này được tính toán từ một hình ảnh số hóa của FNA về ung thư vú. Chúng mô tả các đặc điểm của nhân tế bào có trong hình ảnh.

## Dữ liệu này có hai loại ung thư: ác tính (có hại) và lành tính (không có hại). Ta có thể xây dựng một mô hình để phân loại loại ung thư.

## Cho dữ liệu breast_cancer nằm trong sklearn.datasets

Yêu cầu: đọc dữ liệu về, chuẩn hóa dữ liệu (nếu cần) và áp dụng thuật toán SVM để thực hiện việc dự đoán có bị ung thư hay không dựa trên thông tin được cung cấp

1. Tạo X_train, X_test, y_train, y_test từ dữ liệu đọc được với tỷ lệ dữ liệu test là 0.3
2. Áp dụng thuật toán SVM
3. Tìm kết quả
4. Kiểm tra độ chính xác
5. So sánh hiệu suất của các thuật toán classification: RandomForestClassifier, SVC, GaussianNB, LogisticRegression
6. Trực quan hóa kết quả. Trong các thuật toán trên, thuật toán nào phù hợp nhất với yêu cầu phân loại trên?

In [1]:
```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn import svm
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
```

In [2]:
```
cancer = datasets.load_breast_cancer()
```

In [3]:
```
# print the names of the features
print("Features: ", cancer.feature_names)
```

```
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']
```

In [4]:
```
# print the label type of cancer('malignant' 'benign') # ac tinh, lanh tinh
print("Labels: ", cancer.target_names)
```

```
Labels:  ['malignant' 'benign']
```

In [5]:
```
cancer.data.shape
```

Out[5]:
```
(569, 30)
```

In [6]:
```
print(cancer.target[[0, 1, 2]])
print(list(cancer.target_names))
```

```
[0 0 0]
['malignant', 'benign']
```

In [7]:
```
# print the cancer data features (top 5 records)
print(cancer.data[0:5])
```

```
[[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
  1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
  6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
  1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
  4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 1.326e+03 8.474e-02 7.864e-02 8.690e-02
  7.017e-02 1.812e-01 5.667e-02 5.435e-01 7.339e-01 3.398e+00 7.408e+01
  5.225e-03 1.308e-02 1.860e-02 1.340e-02 1.389e-02 3.532e-03 2.499e+01
  2.341e+01 1.588e+02 1.956e+03 1.238e-01 1.866e-01 2.416e-01 1.860e-01
  2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 1.203e+03 1.096e-01 1.599e-01 1.974e-01
  1.279e-01 2.069e-01 5.999e-02 7.456e-01 7.869e-01 4.585e+00 9.403e+01
  6.150e-03 4.006e-02 3.832e-02 2.058e-02 2.250e-02 4.571e-03 2.357e+01
  2.553e+01 1.525e+02 1.709e+03 1.444e-01 4.245e-01 4.504e-01 2.430e-01
  3.613e-01 8.758e-02]
 [1.142e+01 2.038e+01 7.758e+01 3.861e+02 1.425e-01 2.839e-01 2.414e-01
  1.052e-01 2.597e-01 9.744e-02 4.956e-01 1.156e+00 3.445e+00 2.723e+01
  9.110e-03 7.458e-02 5.661e-02 1.867e-02 5.963e-02 9.208e-03 1.491e+01
  2.650e+01 9.887e+01 5.677e+02 2.098e-01 8.663e-01 6.869e-01 2.575e-01
  6.638e-01 1.730e-01]
 [2.029e+01 1.434e+01 1.351e+02 1.297e+03 1.003e-01 1.328e-01 1.980e-01
  1.043e-01 1.809e-01 5.883e-02 7.572e-01 7.813e-01 5.438e+00 9.444e+01
  1.149e-02 2.461e-02 5.688e-02 1.885e-02 1.756e-02 5.115e-03 2.254e+01
  1.667e+01 1.522e+02 1.575e+03 1.374e-01 2.050e-01 4.000e-01 1.625e-01
  2.364e-01 7.678e-02]]
```

In [8]:
```
X = cancer.data
```

In [9]:
```
X[:, :5]
```

Out[9]:
```
array([[1.799e+01, 1.038e+01, 1.228e+02, 1.001e+03, 1.184e-01],
       [2.057e+01, 1.777e+01, 1.329e+02, 1.326e+03, 8.474e-02],
       [1.969e+01, 2.125e+01, 1.300e+02, 1.203e+03, 1.096e-01],
       ...,
       [1.660e+01, 2.808e+01, 1.083e+02, 8.581e+02, 8.455e-02],
       [2.060e+01, 2.933e+01, 1.401e+02, 1.265e+03, 1.178e-01],
       [7.760e+00, 2.454e+01, 4.792e+01, 1.810e+02, 5.263e-02]])
```

In [10]:
```
y = cancer.target
y[:5]
```

Out[10]:
```
array([0, 0, 0, 0, 0])
```

In [11]:
```
# print the cancer labels (0:malignant, 1:benign) =>
# malignant (ác tính), benign (lành tính)
print("malignant:",y[y==0].size)
print("benign:",y[y==1].size)
```

```
malignant: 212
benign: 357
```

In [12]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3,
                                                    random_state=1)
```

In [13]:
```
clf = svm.SVC(kernel='linear')
# kernel='linear', C=100, gamma=0.01,
clf.fit(X_train, y_train)
```

Out[13]:
```
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 [14]:
```
y_pred = clf.predict(X_test)
```

In [15]:
```
#y_pred
```

In [16]:
```
from sklearn.metrics import accuracy_score
print("Accuracy is ", accuracy_score(y_test,y_pred)*100,"%")
```

```
Accuracy is  95.32163742690058 %
```

In [17]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[ 57   6]
 [  2 106]]
              precision    recall  f1-score   support

           0       0.97      0.90      0.93        63
           1       0.95      0.98      0.96       108

    accuracy                           0.95       171
   macro avg       0.96      0.94      0.95       171
weighted avg       0.95      0.95      0.95       171
```

In [18]:
```
from sklearn import metrics
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Precision: 0.9464285714285714
Recall: 0.9814814814814815
```

In [19]:
```
# Score of traing and testing data
```

In [20]:
```
print("Training R^2 Score", clf.score(X_train, y_train))
print("Testing R^2 Score", clf.score(X_test, y_test))
```

```
Training R^2 Score 0.9698492462311558
Testing R^2 Score 0.9532163742690059
```

**Summary about the model:**

- High accuracy: ~0.95
- High precision: ~0.95, High recall: ~0.98
- High training R^2 score and High testing score, nearly the same
- => The good model

# Lựa chọn model phù hợp

**Trong danh sách các model có thể chọn => chọn ra một số model phù hợp nhất dựa trên ưu - khuyệt điểm và đặc điểm của dataset (ví dụ ở bài này là 4) => thực hiện việc kiểm tra trên các model được chọn => ra quyết định**

In [21]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
```

In [22]:
```
# Tính độ chính xác theo: Logistic, Naive Bayes, SVM, KNN
from sklearn.model_selection import cross_val_score
models = [
    RandomForestClassifier(n_estimators=200),
    SVC(kernel='linear'),
    GaussianNB(),
    LogisticRegression(solver='liblinear') # solver='lbfgs',
]
CV = 5 # số lần lặp
cv_df = pd.DataFrame(index=range(CV * len(models))) # lưu kết quả acc của 4 mode
entries = [] # lưu 2 thông tin là model_name và accuracies.mean()

for model in models: # duyệt từng model trong ds model
    model_name = model.__class__.__name__
    accuracies = cross_val_score(model, X, y, scoring='accuracy', cv=CV)
    print(accuracies)
    entries.append([model_name, accuracies.mean()])

cv_df = pd.DataFrame(entries, columns=['model_name', 'accuracy'])
```

```
[0.92173913 0.93913043 0.98230088 0.97345133 0.97345133]
[0.94782609 0.93043478 0.97345133 0.92035398 0.95575221]
[0.92173913 0.92173913 0.95575221 0.94690265 0.95575221]
[0.93913043 0.93913043 0.97345133 0.94690265 0.96460177]
```

In [ ]:
```
```

In [23]:
```
cv_df
```

Out[23]:

| | model_name | accuracy |
|---|---|---|
| 0 | RandomForestClassifier | 0.958015 |
| 1 | SVC | 0.945564 |
| 2 | GaussianNB | 0.940377 |
| 3 | LogisticRegression | 0.952643 |

In [24]:
```
plt.figure(figsize=(8,6))
plt.bar(cv_df['model_name'],cv_df['accuracy'])
plt.xlabel('model_name')
plt.ylabel('Mean of accuracies')
plt.xticks(rotation='vertical')
plt.title("Accuracies of Algorithms")
plt.show()
```