# TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

# ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

---

## HỌC MÁY THỐNG KÊ

**HỌ VÀ TÊN: NGUYỄN MINH TIẾN**

**MSSV: 20522010**

**LỚP: DS102.M21**

**BÁO CÁO BÀI THỰC HÀNH 2**

**BÀI TOÁN PHÂN LỚP**

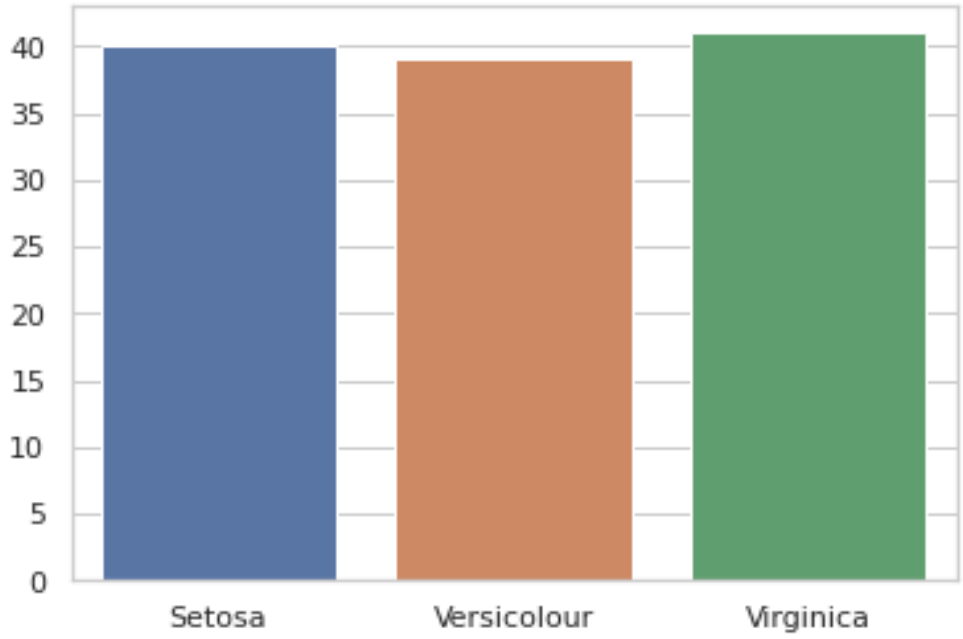
**Bài 1:** Hãy thống kê số lượng nhãn (label) trên tập training và tập test vừa chia. Vẽ biểu đồ phân bố nhãn (Gợi ý: sử dụng ***barplot*** trong thư viện ***seaborn***).

Tập train:

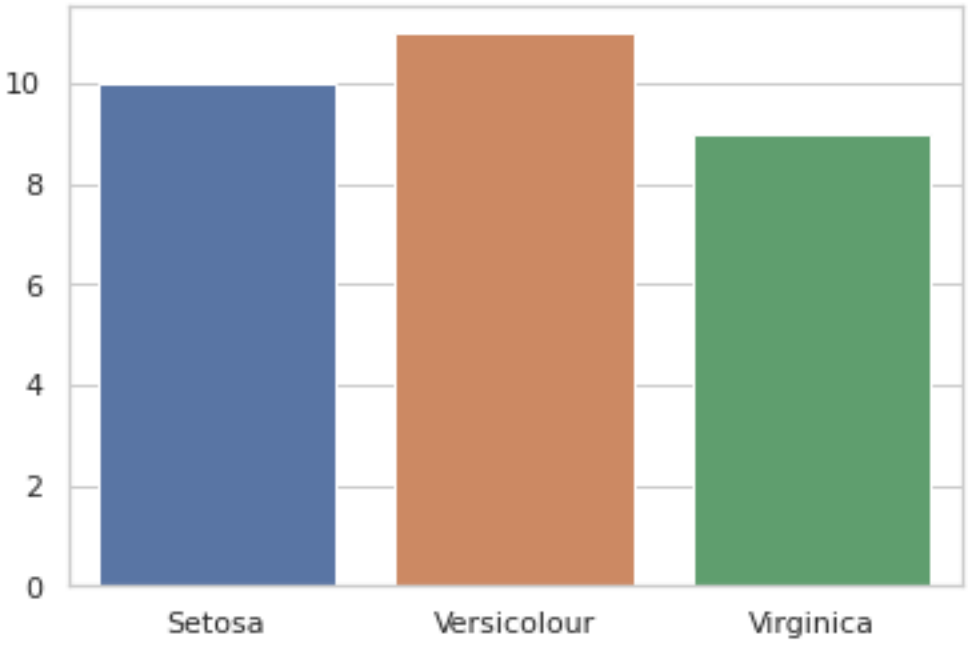
| **Nhãn** | **Số lượng** |
|---|---|
| 0 | 40 |
| 1 | 39 |
| 2 | 41 |

Tập test:

| **Nhãn** | **Số lượng** |
|---|---|
| 0 | 10 |
| 1 | 39 |
| 2 | 41 |




**Biểu đồ phân bố nhãn tập train**




**Biểu đồ phân bố nhãn tập test**

**Bài 2:** Thực hiện huấn luyện mô hình Logistic Regression trên bộ dữ liệu (tham khảo theo các bước đã hướng dẫn).

Entrée [51]:
```
#Cài Linear Regression để dự đoán
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
```

Out[51]:
```
LogisticRegression()
```

Entrée [52]:
```
y_pred = model.predict(X_test)
```

Entrée [53]:
```
print(y_test)
```

```
[1 2 0 2 0 1 2 0 1 1 1 0 2 0 2 2 1 2 2 2 1 0 1 1 2 0 0 2 2 2]
```

Entrée [54]:
```
print(y_pred)
```

```
[2 1 0 2 0 2 1 0 1 1 1 0 1 0 2 2 1 2 2 2 1 0 1 2 1 0 0 2 2 2]
```

**Bài 3**: Thực hiện huấn luyện mô hình K láng giềng gần nhất (KNN) trên bộ dữ liệu, sau đó so sánh độ chính xác (Accuracy) với mô hình LogisticRegression.

Entrée [56]:
```
#K=1
knn = neighbors.KNeighborsClassifier(n_neighbors = 1, p = 2)
knn.fit(X_train, y_train)
y_pred_1 = knn.predict(X_test)

print("Print results for 20 test data points:")
print("Predicted labels: ", y_pred_1[10:30])
print("Ground truth    : ", y_test[10:30])
```

```
Print results for 20 test data points:
Predicted labels:  [2 0 1 0 2 2 2 2 2 2 1 0 1 2 1 0 0 1 2 2]
Ground truth    :  [1 0 2 0 2 2 1 2 2 2 1 0 1 1 2 0 0 2 2 2]
```

Entrée [57]:
```
#Major_voting: K=10
knn = neighbors.KNeighborsClassifier(n_neighbors = 10, p = 2)
knn.fit(X_train, y_train)
y_pred_10 = knn.predict(X_test)

print("Print results for 20 test data points:")
print("Predicted labels: ", y_pred_10[10:30])
print("Ground truth    : ", y_test[10:30])
```

```
Print results for 20 test data points:
Predicted labels:  [1 0 1 0 2 2 1 2 2 2 1 0 1 2 1 0 0 1 2 2]
Ground truth    :  [1 0 2 0 2 2 1 2 2 2 1 0 1 1 2 0 0 2 2 2]
```

Entrée [58]:
```
#K=10, weighted
knn = neighbors.KNeighborsClassifier(n_neighbors = 10, p = 2, weights = 'distance')
knn.fit(X_train, y_train)
y_pred_10_weighted = knn.predict(X_test)

print("Print results for 20 test data points:")
print("Predicted labels: ", y_pred_10_weighted[10:30])
print("Ground truth    : ", y_test[10:30])
```

```
Print results for 20 test data points:
Predicted labels:  [2 0 1 0 2 2 1 2 2 2 1 0 1 2 1 0 0 1 2 2]
Ground truth    :  [1 0 2 0 2 2 1 2 2 2 1 0 1 1 2 0 0 2 2 2]
```

Entrée [59]:
```
#K=10, weighted uniform
knn = neighbors.KNeighborsClassifier(n_neighbors = 10, p = 2, weights = 'uniform')
knn.fit(X_train, y_train)
y_pred_10_weight_uniform = knn.predict(X_test)

print("Print results for 20 test data points:")
print("Predicted labels: ", y_pred_10_weight_uniform[10:30])
print("Ground truth    : ", y_test[10:30])
```

```
Print results for 20 test data points:
Predicted labels:  [1 0 1 0 2 2 1 2 2 2 1 0 1 2 1 0 0 1 2 2]
Ground truth    :  [1 0 2 0 2 2 1 2 2 2 1 0 1 1 2 0 0 2 2 2]
```

Độ chính xác của mô hình Logistic Regression: 76.67 %

Độ chính xác của mô hình 1NN: 76.33 %

Độ chính xác của mô hình 10NN: 83.33 %

Độ chính xác của mô hình 10NN weighted: 80.00 %

Độ chính xác của mô hình 10NN weighted uniform: 83.33 %

=> Từ kết quả trên ta rút ra kết luận rằng: cần đánh giá việc phân chia tập dữ liệu train và tập test theo đúng tỷ lệ để kết quả của mô hình được ổn định.

**Bài 4**: Đánh giá 2 mô hình vừa xây dựng trên 3 độ đo sau: precision_score, recall_score và f1_score sử dụng macro average.

Out[64]:

| | Logistic Regression | KNN_1 | KNN_10 | KNN_10_weighted |
|---|---|---|---|---|
| **precision** | 0.783333 | 0.779762 | 0.844444 | 0.811966 |
| **recall** | 0.786325 | 0.774929 | 0.849003 | 0.811966 |
| **F1-macro** | 0.78386 | 0.776325 | 0.845614 | 0.811966 |

Cả về precision, recall, F1-macro của 3 thuật toán trên không chênh lệch nhau bao nhiêu. Kết quả này vẫn thay đổi sau nhiều lần chạy thử.

**Bài 5***: Hãy sử dụng chiến lược tinh chỉnh tham số GridSearchCV để tìm ra bộ tham số tốt nhất cho mô hình Logistic Regression. So sánh kết quả với mô hình gốc.

Out[69]:

| | Logistic Regression | GridSearchCV |
|---|---|---|
| **precision** | 0.783333 | 0.783333 |
| **recall** | 0.786325 | 0.786325 |
| **F1-macro** | 0.78386 | 0.78386 |