THỰC HÀNH: CÁC GIẢI THUẬT PHÂN LOẠI CƠ BẢN

A. MỤC TIÊU CHƯƠNG

Mục tiêu của phần thực hành các giải thuật phân loại dữ liệu (Data Classification Algorithms) được thiết kế để giúp sinh viên phát triển kỹ năng áp dụng các giải thuật phân loại vào việc giải quyết các bài toán thực tế trong khoa học dữ liệu. Cụ thể, phần thực hành nhằm đạt được các mục tiêu sau: Hiểu và triển khai các thuật toán phân loại: Cây quyết định và rừng cây (Decision Tree & Random Forest), Support Vector Machine (SVM) và Bayes ngây thơ (Naive Bayes) thông qua ngôn ngữ lập trình như Python và sử dụng các thư viện Scikit-learn, SciPy, …

Hướng dẫn thực hiện việc triển khai xây dựng một mô hình học máy cổ điển với các bước:
- Tiền xử lý dữ liệu: Hướng dẫn cách chuẩn bị dữ liệu, bao gồm xử lý giá trị thiếu, chuẩn hóa dữ liệu, và mã hóa các biến phân loại để phù hợp với yêu cầu của các thuật toán phân loại.
- Đánh giá và tối ưu hóa mô hình: sinh viên sẽ thực hành đánh giá hiệu suất mô hình thông qua các chỉ số như accuracy, precision, recall, F1-score và sử dụng các kỹ thuật như cross-validation, grid search để tối ưu hóa tham số.

- Phân tích và diễn giải kết quả: Phát triển khả năng diễn giải kết quả phân loại, nhận diện các vấn đề như quá khớp (overfitting) và đề xuất cải tiến.

B. KẾT CẤU THỰC HÀNH

Thực hành bao gồm 3 phần là
- Giải thuật cây quyết định và rừng cây
- Giải thuật Support Vector Machine (SVM)
- Giải thuật Bayes ngây thơ

C. NỘI DUNG THỰC HÀNH

2.1. GIẢI THUẬT 1: CÂY QUYẾT ĐỊNH VÀ RỪNG CÂY

2.1.1. Ôn tập lý thuyết

+ Quy trình khai phá dữ liệu CRISP – DM (Cross Industry Standard Process for Data Mining) là gì ? Quy trình khai phá dữ liệu SEMMA (Sample, Explore, Modify, Model, Access) là gì?

+ Cây quyết định hoạt động như thế nào? Hãy giải thích các thành phần chính (nút gốc, nút lá, nhánh) và cách cây đưa ra dự đoán.

+ Các tiêu chí phân tách (splitting criteria) như Gini Index, Entropy, hay Information Gain được sử dụng trong cây quyết định là gì? Chúng khác nhau ra sao?

+ Rừng cây (Random Forest) là gì? Nó khác gì so với một cây quyết định đơn lẻ? Tại sao Random Forest

thường có hiệu suất tốt hơn cây quyết định trong các bài toán phân loại?

+ Những ưu điểm và hạn chế của cây quyết định và Random Forest là gì? Trong trường hợp nào thì cây quyết định có thể hoạt động kém hiệu quả?

+ Viết đoạn code mẫu bằng Python (sử dụng Scikit-learn) để xây dựng một mô hình cây quyết định không? Hãy mô tả các bước thực hiện

+ Làm thế nào để triển khai một mô hình Random Forest trong Python? Bạn thường thiết lập các tham số nào (ví dụ: n_estimators, max_depth)?

+ Làm thế nào để đánh giá tầm quan trọng của các đặc trưng (feature importance) trong Random Forest bằng Python?

+ Điều chỉnh siêu tham số (hyperparameter tuning) cho cây quyết định hoặc Random Forest chưa? Hãy mô tả cách bạn sử dụng GridSearchCV hoặc RandomizedSearchCV

2.1.2. Bài làm mẫu

Bài toán 1: Xây dựng cây quyết định và rừng cây với dữ liệu lấy từ https://www.kaggle.com/datasets/deceneu/default-of-credit-card-clients

Nhiệm vụ 1: Xây dựng cây quyết định bằng thư viện Scikit-Learn

1. Tải một số package mà chúng tôi sử dụng và package graphviz, để vẽ cây quyết định

```
import numpy as np #numerical computation
import pandas as pd #data wrangling
import matplotlib.pyplot as plt #plotting package
#Next line helps with rendering plots
%matplotlib inline
import matplotlib as mpl #add'l plotting functionality
mpl.rcParams['figure.dpi'] = 400 #high res figures
import graphviz #to visualize decision trees
```

2. Nạp dữ liệu vào bộ nhớ, phân tích và loại bỏ những features không liên quan đến bài toán cần giải quyết

```
df = pd.read_csv('data.csv') #Load the cleaned data
features_response = df.columns.tolist() #Get a list of column names
#Make a list of columns to remove that aren't features or the response
variable
items_to_remove = ['ID', 'SEX', 'PAY_2', 'PAY_3',\
                   'PAY_4', 'PAY_5', 'PAY_6',\
                   'EDUCATION_CAT', 'graduate school',\
                   'high school', 'none',\
                   'others', 'university']
features_response = [item for item in features_response if item not
in items_to_remove]
features_response
```

3. Chuẩn bị dữ liệu cho tập train và tập test

```
from sklearn.model_selection import train_test_split
```

```
from sklearn import tree
#Split the data into training and testing sets using the same random seed
X_train, X_test, y_train, y_test = \
    train_test_split(df[features_response[:-1]].values,
    df['default payment next month'].values,
    test_size=0.2, random_state=24)
```

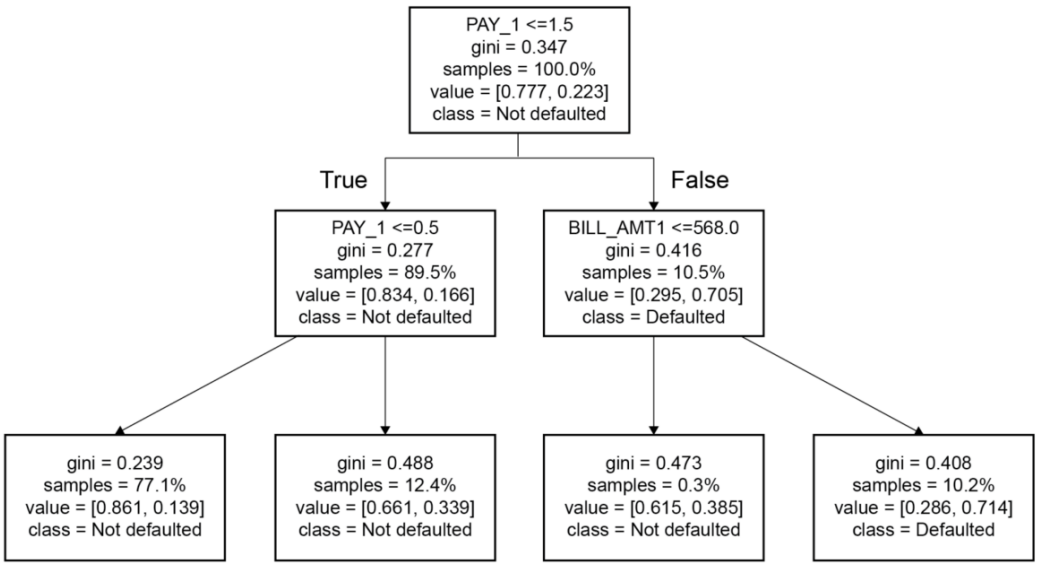
4. Xây dựng cây quyết định từ lớp DecisionTreeClassifier có trong thư viện Scikit-Learn

```
# the tree will grow to a depth of at most 2
dt = tree.DecisionTreeClassifier(max_depth=2)
dt.fit(X_train, y_train)
```

5. Hiển thị cây quyết định với package **graphviz**

```
dot_data = tree.export_graphviz(dt,
                out_file=None,
                filled=True,
                rounded=True,
                feature_names=\
                  features_response[:-1],
                  proportion=True,
                  class_names=['Not defaulted', 'Defaulted'])
graph = graphviz.Source(dot_data)
graph
```

Kết quả ta được cây quyết định như hình bên dưới



Hình 2.1 - Cây quyết định

Nhiệm vụ 2: Tìm tham số tối ưu cho cây quyết định bằng GridSearchCV và vẽ biểu đồ đánh giá mô hình với các tham số khác nhau

1. Thực hiện 1, 2 và 3 như ở nhiệm vụ 1 để tải thư viện, nạp dữ liệu và chuẩn bị dữ liệu
2. Tạo cây quyết định và xác định các giá trị tham số có thể chọn để tìm tham số tối ưu

```
from sklearn.model_selection import GridSearchCV
params = {'max_depth':[1, 2, 4, 6, 8, 10, 12]} #parameters
dt = tree.DecisionTreeClassifier() #tree modal
cv = GridSearchCV(dt, param_grid=params, scoring='roc_auc',
                  n_jobs=None, refit=True, cv=4, verbose=1,
                  error_score=np.nan,
                  return_train_score=True) # cv is the best model.
cv.fit(X_train, y_train)
```

Sử dụng ChatGPT để tìm hiểu thêm các khái niệm bên dưới

scoring = 'roc_auc': the ROC AUC metric

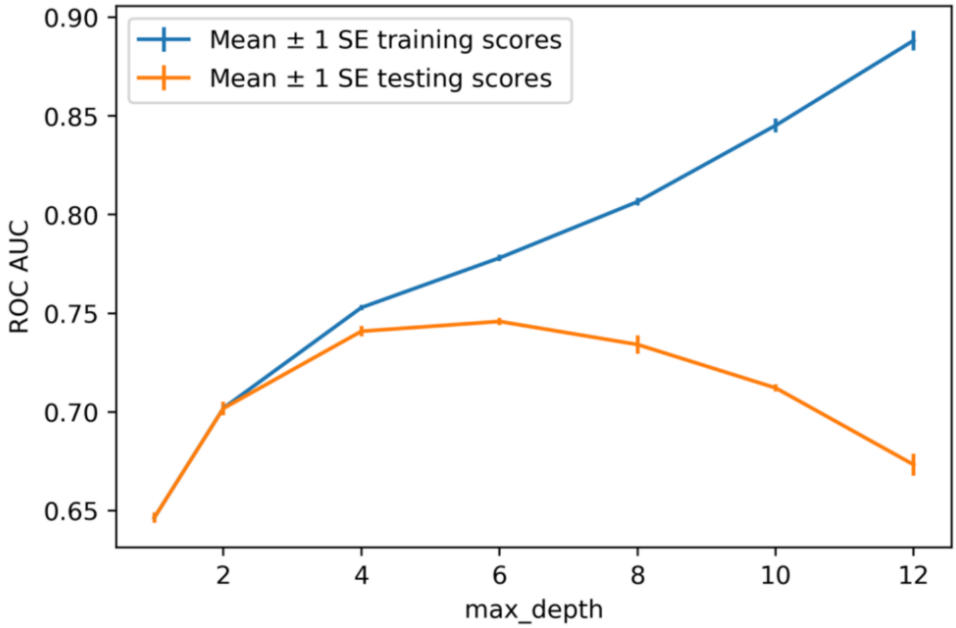
cv = 4: 4-fold cross-validation

return_train_score = true: để đánh giá bias và variance

3. Vẽ biểu đồ đánh giá mô hình với các tham số chiều sâu của cây khác nhau

```
cv_results_df = pd.DataFrame(cv.cv_results_)
#View the names of the remaining columns in the results DataFrame
cv_results_df.columns
ax = plt.axes()
ax.errorbar(cv_results_df['param_max_depth'],
            cv_results_df['mean_train_score'],
            yerr=cv_results_df['std_train_score']/np.sqrt(4),
            label='Mean $\pm$ 1 SE training scores')
ax.errorbar(cv_results_df['param_max_depth'],
            cv_results_df['mean_test_score'],
            yerr=cv_results_df['std_test_score']/np.sqrt(4),
            label='Mean $\pm$ 1 SE testing scores')
ax.legend()
plt.xlabel('max_depth')
plt.ylabel('ROC AUC')
```

Kết quả ta được biểu đồ so sánh kết quả đánh giá mô hình với các tham số khác nhau



Hình 2.2 - Biểu đồ đánh giá hiệu quả thực hiện cây quyết định với các chiều sâu khác nhau

Nhiệm vụ 3: Xây dựng rừng cây (random forest)

1. Thực hiện 1, 2 và 3 như ở nhiệm vụ 1 để tải thư viện, nạp dữ liệu và chuẩn bị dữ liệu

2. Tạo rừng cây với lớp RandomForestClassifier trong Scikit-Learn

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier\
  (n_estimators=10, criterion='gini', max_depth=3,
   min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0,
   max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0,
   bootstrap=True, oob_score=False, n_jobs=None,
   random_state=4, verbose=0, warm_start=False, class_weight=None)
```

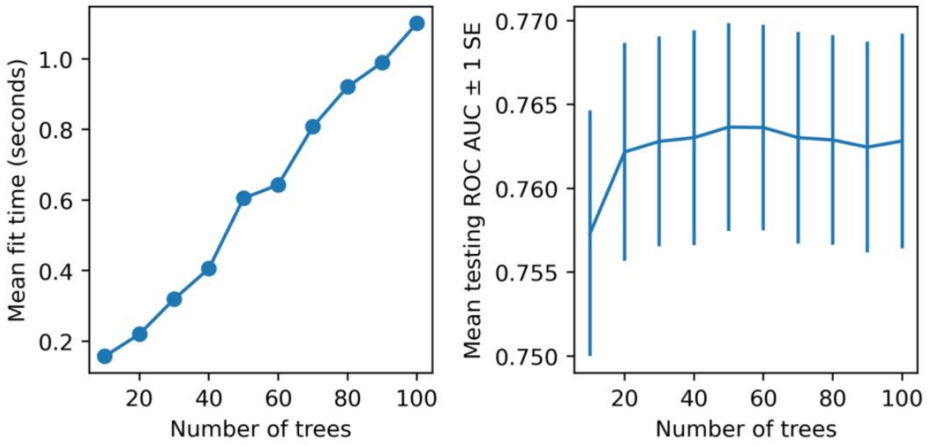
3. Tìm tham số tối ưu cho mô hình rừng cây và thực hiện train với tham số tối ưu đó

```
#a parameter grid for this exercise in order to search the numbers of
trees, ranging from 10 to 100 by 10s
rf_params_ex = {'n_estimators':list(range(10,110,10))}
cv_rf_ex = GridSearchCV(rf, param_grid=rf_params_ex,
                        scoring='roc_auc', n_jobs=None,
                        refit=True, cv=4, verbose=1,
                        error_score='np.nan',
                        return_train_score=True)
cv_rf_ex.fit(X_train, y_train)
```

4. Vẽ biểu đồ đánh giá mô hình rừng cây với các tham số số cây có trong rừng khác nhau

```
cv_rf_ex_results_df = pd.DataFrame(cv_rf_ex.cv_results_)
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(6, 3))
axs[0].plot(cv_rf_ex_results_df['param_n_estimators'],
            cv_rf_ex_results_df['mean_fit_time'],
            '-o')
axs[0].set_xlabel('Number of trees')
axs[0].set_ylabel('Mean fit time (seconds)')
axs[1].errorbar(cv_rf_ex_results_df['param_n_estimators'],
                cv_rf_ex_results_df['mean_test_score'],
                yerr=cv_rf_ex_results_df['std_test_score']/np.sqrt(4))
axs[1].set_xlabel('Number of trees')
axs[1].set_ylabel('Mean testing ROC AUC $\pm$ 1 SE ')
plt.tight_layout()
```

Kết quả thực hiện:

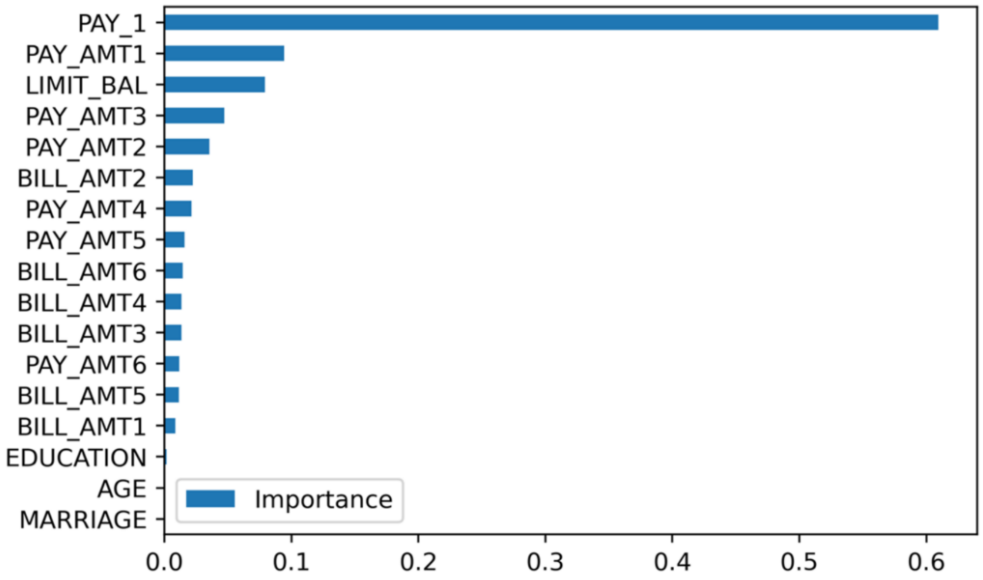


Hình 2.3 - Biểu đồ thể hiện mối quan hệ giữa số cây với Mean Fit Time và Mean Testing ROC AUC

5. Xem tham số tốt nhất của rừng cây, xem mức độ quan trọng của từng feature trong mô hình với tham số tốt nhất

```
# {'n_estimators': 50}
cv_rf_ex.best_params_
# the feature names and importances
feat_imp_df = pd.DataFrame({
    'Importance':cv_rf_ex.best_estimator_.feature_importances_},
    index=features_response[:-1])
feat_imp_df.sort_values('Importance', ascending=True).plot.barh()
```

Kết quả thực hiện



Hình 2.4 - Biểu đồ hiển thị mức độ quan trọng của từng feature

2.1.3. Bài tập thực hành 1

Xây dựng cây quyết định và rừng cây trên dữ liệu Titanic lấy từ
https://www.kaggle.com/code/dmilla/introduction-to-decision-trees-titanic-dataset

2.1.4. Bài tập thực hành 2

Xây dựng cây quyết định và rừng cây trên dữ liệu bệnh tiểu đường. Dữ liệu lấy từ
https://www.kaggle.com/code/tumpanjawat/diabetes-eda-random-forest-hp

2.2. GIẢI THUẬT 2: SUPPORT VECTOR MACHINE (SVM)

2.2.1. Ôn tập lý thuyết

+ Giải thuật Support Vector Machine hoạt động như thế nào? Hãy giải thích khái niệm về ranh giới phân tách (hyperplane) và lề (margin)

+ Các vector hỗ trợ (support vectors) có vai trò gì trong SVM? Tại sao chúng quan trọng?

+ Sự khác biệt giữa SVM với lề cứng (hard margin) và lề mềm (soft margin) là gì? Khi nào nên sử dụng lề mềm?

+ Hàm nhân (kernel) trong SVM là gì? Hãy giải thích các loại kernel phổ biến (linear, polynomial, RBF) và khi nào nên sử dụng chúng

+ Tham số C trong SVM có ý nghĩa gì? Nó ảnh hưởng như thế nào đến hiệu suất của mô hình?

+ Viết đoạn code mẫu bằng Python (sử dụng Scikit-learn) để xây dựng một mô hình SVM cho bài toán phân loại không? Hãy mô tả các bước thực hiện

+ Hàm nào trong Scikit-learn để chuẩn hóa dữ liệu (scaling) trước khi áp dụng SVM? Tại sao bước này quan trọng?

2.2.2. Bài làm mẫu

Bài toán 1: Thực hiện các nhiệm vụ trong bài toán 1 để xây dựng mô hình với giải thuật SVM cho dữ liệu Iris-data lấy từ https://www.kaggle.com/code/xvivancos/tutorial-knn-in-the-iris-data-set

Nhiệm vụ 1: Xây dựng mô hình SVM để phân loại các loài hoa cẩm chướng

1. Tải dữ liệu về, nạp dữ liệu, xem thông tin các feature có trong tập dữ liệu và chuẩn bị dữ liệu cho xây dựng mô hình

```
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
# Download&Load dữ liệu iris từ datasets của scikit-learn
iris = datasets.load_iris()
# Hiển thị mô ta dữ liệu, chỉ có trong các bộ dữ liệu chuẩn và mở để học tập và nghiên cứu
print(iris.DESCR)
# Từ tập dữ liệu ban đầu, tách lấy ma trận biểu diễn các đặc trưng và nhãn.
data = iris.data
target = iris.target
# TODO: Chia dữ liệu và nhãn thành 2 tập dữ liệu huấn luyện và dữ liệu kiểm tra theo tỉ lệ 80:20
X_train, X_test, y_train, y_test = train_test_split(data, target, test_size
                                        = 0.2, random_state=101)
```

2. Tạo mô hình SVM với dữ liệu đã chuẩn bị

```
from sklearn import svm
# khởi tạo mô hình phân lớp
clf = svm.SVC()
# Sử dụng phương thức 'fit' để huấn luyện mô hình với dữ liệu huấn luyện và nhãn huấn luyện
# fit (X,Y) với X là tập các đối tượng, Y là tập nhãn tương ứng của đối tượng.
clf.fit(X_train, y_train)
```

3. Đánh giá độ chính xác của mô hình

```
# Tính độ chính xác trên tập huấn luyện và tập kiểm tra
train_acc = clf.score(X_train,y_train)
val_acc = clf.score(X_test,y_test)
print('Training accuracy: {}'.format(train_acc))
print('Validation accuracy: {}'.format(val_acc))
```

4. Tìm tham số kernel tối ưu cho mô hình SVM

```
# best_svm, best_val_acc và best_kernel lần lượt là các biến lưu mô hình
```

```
tốt nhất,
# độ chính xác cao nhất trên tập kiểm tra và kernel tốt nhất
kernels = ['linear', 'poly', 'rbf', 'sigmoid']
best_svm = None
best_val_acc = -1
best_kernel = None
# Huấn luyện các mô hình dựa trên dữ liệu huấn luyện và tham số kernel
# Tính toán độ chính xác trên tập huấn luyện và tập kiểm tra để tìm được mô hình tốt nhất
for i in range(4):
    clf = svm.SVC(kernel=kernels[i], probability=True)
    clf.fit(X_train, y_train)
    tmp_val_acc = clf.score(X_test, y_test)
    if (tmp_val_acc > best_val_acc):
        best_val_acc = tmp_val_acc
        best_svm = clf
        best_kernel = kernels[i]
# Hiển thị mô hình tốt nhất cùng với độ chính xác
print("Best validation accuracy : {} with kernel: {}".format(best_val_acc, best_kernel))
# Mô hình tốt nhất của bạn nên có độ chính xác xấp xỉ 86,67%
```

Ghi chú: Sinh viên có thể sử dụng GridSearchCV để tìm tham số tối ưu cho mô hình SVM trên.

Bài toán 2: Xây dựng mô hình dựa vào giải thuật SVM trên dữ liệu hình ảnh Handwritten-Digit-MNIST-SVM. Dữ liệu lấy từ https://www.kaggle.com/code/nishan192/mnist-digit-recognition-using-svm

Nhiệm vụ 1: Tìm hiểu về cách biểu diễn và hiển thị các ảnh từ tập dữ liệu là hình ảnh

1. Import thư viện và tải dữ liệu là tập các hình ảnh viết tay từ số 0 đến số 9

```
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt

from sklearn.datasets import load_digits
digits = load_digits(n_class=10)
```

2. Khảo sát thông tin có trong digits

```
#thông tin toàn bộ dữ liệu đã tải về
digits
#xem thông tin của một hình dưới dạng ma trận 8 x 8
digits['data'][0].reshape(8,8)
```

```
#xem thông tin của một hình dưới dạng mảng
digits['data'][0]
#xem thông tin 9 nhãn đầu tiên
digits['target'][0:9]
```

3. Vẽ ra hình dựa vào dữ liệu dạng ma trận 8 x 8

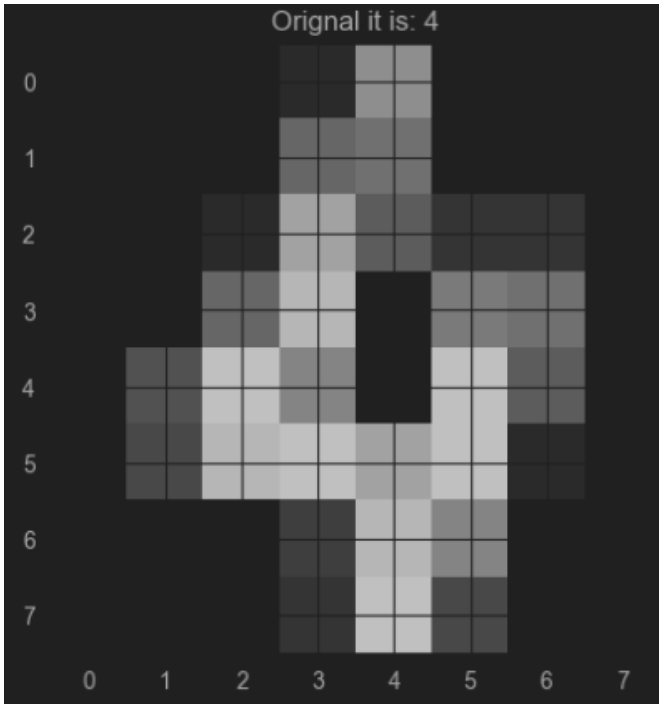
```
# Each Digit is represented in digits.images as a matrix of 8x8 = 64 pixels. Each of the 64 values represent
# a greyscale. The Greyscale are then plotted in the right scale by the imshow method.
fig, ax = plt.subplots(8,8, figsize=(10,10))
for i, axi in enumerate(ax.flat):
    axi.imshow(digits.images[i], cmap='binary')
    axi.set(xticks=[], yticks=[])
```

Kết quả thực hiện

4. Vẽ một ảnh từ ma trận 8 x 8

```
# Hàm vẽ 1 ảnh có kích thước 8 x 8 (ảnh lấy từ ma images)
def view_digit(index):
    plt.imshow(digits.images[index] , cmap = plt.cm.gray_r)
    plt.title('Orignal it is: '+ str(digits.target[index]))
    plt.show()
# vẽ ảnh ở vị trí thứ 4
view_digit(4)
```

Kết quả thực hiện



Nhiệm vụ 2: Xây dựng mô hình SVM để nhận diện chữ viết tay từ 0 – 9

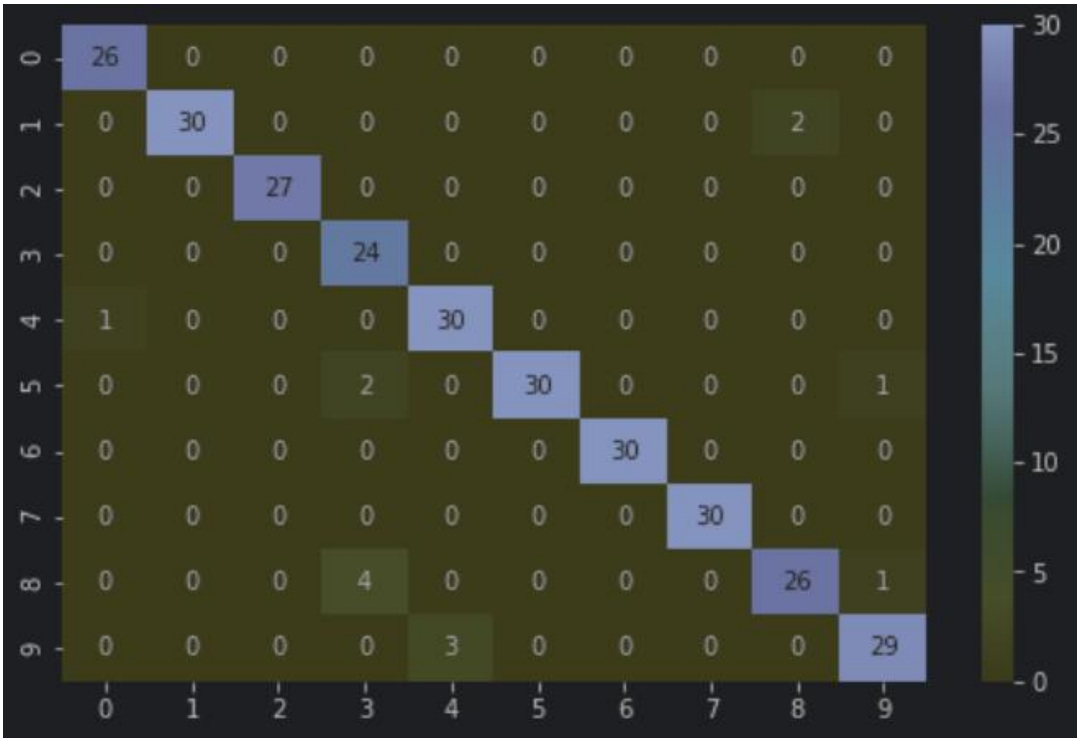
1. Chuẩn bị dữ liệu và xây dựng mô hình SVM

```
# Thực hiện import các thư viện cần thiết để xây dựng mô hình SVM
# Thực hiện bước 1 của nhiệm vụ 1
from sklearn import svm
main_data = digits['data']
targets = digits['target']
svc = svm.SVC(gamma=0.001 , C = 100)
# GAMMA is a parameter for non linear hyperplanes.
# The higher the gamma value it tries to exactly fit the training data set
# C is the penalty parameter of the error term.
#  It controls the trade off between smooth decision boundary and classifying the training points correctly.
svc.fit(main_data[:1500] , targets[:1500])
predictions = svc.predict(main_data[1501:])
# list(zip(predictions , targets[1501:]))
```

2. Đánh giá hiệu quả của mô hình với Confusion Matrix

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
cm = confusion_matrix(predictions, targets[1501:])
conf_matrix = pd.DataFrame(data = cm)
plt.figure(figsize = (8,5))
sns.heatmap(conf_matrix, annot=True,fmt='d',cmap="YlGnBu");
```

Kết quả thực hiện



Hình 2.5 - Confusion matrix trình bày dưới dạng head map

3. In kết quả dạng văn bản

```
from sklearn.metrics import classification_report
print(classification_report(predictions, targets[1501:]))
```

Kết quả thực hiện

	precision	recall	f1-score	support
0	0.96	1.00	0.98	26
1	1.00	0.94	0.97	32
2	1.00	1.00	1.00	27
3	0.80	1.00	0.89	24
4	0.91	0.97	0.94	31
5	1.00	0.91	0.95	33
6	1.00	1.00	1.00	30
7	1.00	1.00	1.00	30
8	0.93	0.84	0.88	31
9	0.94	0.91	0.92	32

Hình 2.6 - Thông tin đánh giá về hiệu quả của mô hình trên từng lớp

2.2.3. Bài tập thực hành 1

Xây dựng mô hình từ giải thuật SVM trên dữ liệu bệnh tiểu đường. Dữ liệu lấy từ https://www.kaggle.com/code/tumpanjawat/diabetes-eda-random-forest-hp

2.2.4. Bài tập thực hành 2

Xây dựng mô hình từ giải thuật SVM trên dữ liệu các con thú trong rừng. Dữ liệu lấy từ https://www.kaggle.com/code/kareemellithy/animal-condition-predict-svm-knn

2.3. GIẢI THUẬT 3: BAYES NGÂY THƠ (NAÏVE BAYES)

2.3.1. Ôn tập lý thuyết

+ Giải thuật Naive Bayes hoạt động như thế nào? Hãy giải thích định lý Bayes và giả định "ngây thơ" trong thuật toán này?

+ Các loại mô hình Naive Bayes (Gaussian, Multinomial, Bernoulli) khác nhau ra sao? Khi nào nên sử dụng từng loại?

+ Tại sao Naive Bayes được gọi là "ngây thơ"? Giả định về tính độc lập của các đặc trưng ảnh hưởng như thế nào đến hiệu suất của mô hình?

+ Ưu điểm và hạn chế của Naive Bayes so với các thuật toán phân loại khác như SVM hoặc Random Forest là gì?

+ Viết đoạn code mẫu bằng Python (sử dụng Scikit-learn) để xây dựng một mô hình Naive Bayes (ví dụ: Gaussian Naive Bayes) không? Hãy mô tả các bước thực hiện

+ Làm thế nào để xử lý dữ liệu phân loại (categorical data) trước khi áp dụng Multinomial Naive Bayes trong Python?

+ Naive Bayes thường được sử dụng trong phân loại văn bản (text classification). Bạn có thể giải thích cách triển khai Naive Bayes cho bài toán này không?

2.3.2. Bài làm mẫu

Bài toán 1: Xây dựng mô hình dữ liệu bằng giải thuật Bayes ngây thơ trên tập dữ liệu lấy tại https://www.kaggle.com/code/zabihullah18/email-spam-detection

Nhiệm vụ 1: Phân loại sử dụng Naïve Bays

1. Import thư viện và nạp dữ liệu vào notebook

```
#Importing the Necessary libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix,
                                        classification_report
data = pd.read_csv('spam.csv', encoding='latin-1')
#display the first 5 rows
data.head()
```

2. Xử lý dữ liệu trước khi xây dựng mô hình từ dữ liệu

```
# Drop the columns with NaN values
data = data.drop(columns=['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],
```

```
axis=1)
# Rename columns for clarity:
data.columns = ['label', 'text']
# Separate features (X) and target labels (y)
X =  data.drop('label', axis=1)
y = data['label']


# Split the data into training and testing sets (80% training, 20%
testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

3. Xây dựng vector hóa nội dung HAM | SPAM của tập train và tập test

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()


# Fit and transform the training data (X_train)
X_train_vectorized = vectorizer.fit_transform(X_train['text'])
# Transform the test data (X_test)
X_test_vectorized = vectorizer.transform(X_test['text'])
```

4. Xây dựng mô hình Naïve Bayes

```
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB()
classifier.fit(X_train_vectorized, y_train)
```

5. Đánh giá hiệu quả của mô hình

```
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report


# Make predictions on the test data
y_pred = classifier.predict(X_test_vectorized)
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)


print(f"Accuracy: {accuracy:.2f}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
```

```
print(classification_rep)
```

Kết quả thực hiện

```
Accuracy: 0.98
Confusion Matrix:
[[963   2]
 [ 16 134]]
Classification Report:
              precision    recall  f1-score   support

         ham       0.98      1.00      0.99       965
        spam       0.99      0.89      0.94       150

    accuracy                           0.98      1115
   macro avg       0.98      0.95      0.96      1115
weighted avg       0.98      0.98      0.98      1115
```

Hình 2.7 - Báo cáo đánh giá về hiệu quả của mô hình

2.3.3. Bài tập thực hành 1

Xây dựng mô hình Naïve ngây thơ trên tập dữ liệu hành vi của khách hàng lấy tại https://www.kaggle.com/code/arezalo/customer-behaviour-prediction-naive-bayes

2.3.4. Bài tập thực hành 2

Xây dựng mô hình Naïve ngây thơ trên tập dữ liệu mushroom. Dữ liệu lấy tại https://www.kaggle.com/datasets/uciml/mushroom-classification/data

D. TÓM TẮT THỰC HÀNH

Với 3 phần là phân loại với giải thuật cây quyết định và rừng cây, giải thuật support vector machine (SVM) và giải thuật Bayes ngây thơ được trình bày trong chương đã phần nào giúp sinh viên có thể tiến hình triển phân loại dữ liệu trên một số tập dữ liệu cơ bản.

Với việc thực hành giải quyết các bài tập trên lớp và các bài tập ở nhà giúp sinh viên hiểu rõ hơn các khái niệm lý thuyết đã học và áp dụng tốt vào việc giải quyết các **bài toán phân loại** trong thực tế.