TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

INTRODUCTION TO MACHINE LEARNING

MID-TERM REPORT INTRODUCTION TO MACHINE LEARNING

STUDENT 1: 522H0120 – Nguyễn Đình Việt Hoàng
STUDENT 2: 521H0473 – Nguyễn Minh Quân
LECTURER: TS. Lê Anh Cường

THÀNH PHỐ HỒ CHÍ MINH

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

NHẬP MÔN HỌC MÁY

BÀI BÁO CÁO GIỮA KỲ MÔN NHẬP MÔN HỌC MÁY

SINH VIÊN 1: 522H0120 – Nguyễn Đình Việt Hoàng
SINH VIÊN 2: 521H0473 – Nguyễn Minh Quân
GIẢNG VIÊN HƯỚNG DẪN: TS. Lê Anh Cường

THÀNH PHỐ HỒ CHÍ MINH

LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn thầy Lê Anh Cường vì đã giảng dạy tụi em nhiệt tình môn Nhập môn học máy. Chúng em muốn bày tỏ sự cảm kích sâu sắc với sự tận tâm và kiến thức chuyên môn mà thầy đã chia sẻ với chúng em. Qua những buổi học của thầy chúng em đã có cơ hội hiểu biết thêm về các khía cạnh cơ bản của Machine Learning qua việc giải thích chi tiết và áp dụng thực tế. Thầy đã giúp chúng em nắm vững kiến thức và áp dụng chúng vào thực tế. Cuối cùng, chúng em xin cảm ơn đến thầy Lê Anh Cường vì sự tận tâm và sự hỗ trợ quý báu trong suốt quá trình học môn Nhập môn học máy. Những kiến thức và kỹ năng mà chúng em đã học được sẽ luôn có giá trị và ảnh hưởng đến sự phát triển của chúng em trong tương lai. Chúng em xin chân thành cảm ơn thầy Lê Anh Cường và chúc thầy sức khỏe, thành công và hạnh phúc.

TP. Hồ Chí Minh, ngày 30 tháng 3 năm 2024
Tác giả
(Ký tên và ghi rõ họ tên)
Nguyễn Minh Quân
Nguyễn Đình Việt Hoàng

PHIẾU ĐÁNH GIÁ CỦA GIẢNG VIÊN HƯỚNG DẪN

Tên giảng viên hướng dẫn: .....

Ý kiến nhận xét: .....

.....

.....

.....

Điểm tổng theo phiếu đánh giá rubrik: .....

TP. Hồ Chí Minh, ngày 29 tháng 3 năm 2024

Giảng viên hướng dẫn

(Ký tên và ghi rõ họ tên)

BÁO CÁO ĐƯỢC HOÀN THÀNH

TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Nhóm chúng em xin cam đoan đây là bài báo cáo của riêng chúng em và được sự hướng dẫn của thầy Lê Anh Cường. Các nội dung nghiên cứu, kết quả trong bài báo cáo này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong bài báo cáo còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào nhóm chúng em xin hoàn toàn chịu trách nhiệm về nội dung bài báo cáo giữa kỳ học kỳ 2/2023-2024 của mình. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do chúng em gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 30 tháng 3 năm 2024

Tác giả

(Ký tên và ghi rõ họ tên)

Nguyễn Minh Quân

Nguyễn Đình Việt Hoàng

MỤC LỤC

QUESTION 1:

(1) Present the steps in solving a classification problem or regression problem using machine learning methods.

Outline the process of tackling a classification problem using machine learning techniques.

Step 1 - Data Gathering:

Begin by collecting relevant data pertaining to the issue at hand. This data can be sourced from various outlets such as databases, CSV files, or .data files.

Step 2 - Data Preprocessing:

Data often requires refinement before being inputted into a machine learning model. Preprocessing tasks may involve eliminating noisy data, handling missing values, normalizing data, encoding categorical variables, and dividing the data into training and testing sets.

Step 3 - Model Selection:

Depending on the problem's nature (classification or regression) and the data's specific requirements, choose an appropriate machine learning model. Options include decision trees, linear regression, support vector machines (SVM), among others.

Step 4 - Model Training:

Utilize the training dataset to teach the model how to associate input features with labels (for classification) or target values (for regression). This typically involves optimizing the model's parameters to minimize a loss function.

Step 5 - Model Evaluation:

After training the model, assess its performance on a separate testing dataset that it hasn't encountered before. This evaluation aids in gauging the model's ability to generalize to new data. Common metrics for classification problems include accuracy, balanced accuracy (F1-score), and confusion matrix, while regression problems often employ metrics like mean absolute error (MAE), mean squared error (MSE), and coefficient of determination (R-squared).

(2) Perform the steps in question 1 on a classification or regression problem using different machine learning methods. Evaluate and compare the results of applying these methods;
(3) Find solutions to improve the accuracy of the models mentioned in question

(2).

Classification

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report
from sklearn.preprocessing import MinMaxScaler
from warnings import filterwarnings
from sklearn.metrics import accuracy_score,confusion_matrix
filterwarnings(action='ignore')
```

Load data

```
wine = pd.read_csv("winequality-red.csv")
wine.sample(15)
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
1081	7.9	0.300	0.68	8.3	0.050	37.5	289.0	0.99316	3.01	0.51	12.3	7
84	6.3	0.300	0.48	1.8	0.069	18.0	61.0	0.99590	3.44	0.78	10.3	6
927	8.4	0.670	0.19	2.2	0.093	11.0	75.0	0.99736	3.20	0.59	9.2	4
713	8.0	0.430	0.36	2.3	0.075	10.0	48.0	0.99760	3.34	0.46	9.4	5
262	8.0	0.520	0.03	1.7	0.070	10.0	35.0	0.99575	3.34	0.57	10.0	5
182	7.2	0.730	0.02	2.5	0.076	16.0	42.0	0.99720	3.44	0.52	9.3	5
303	7.4	0.670	0.12	1.6	0.186	5.0	21.0	0.99600	3.39	0.54	9.5	5
895	7.1	0.590	0.01	2.3	0.080	27.0	43.0	0.99550	3.42	0.58	10.7	6
933	7.4	0.610	0.01	2.0	0.074	13.0	38.0	0.99748	3.48	0.65	9.8	5

```
wine.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

```
wine.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983	5.636023
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668	0.807569
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	5.000000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000	6.000000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000	6.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.000000

```
wine.isnull().sum()
```

+ Code + Text

Connect | Colab AI

```
[ ] 1 wine.isnull().sum()
```

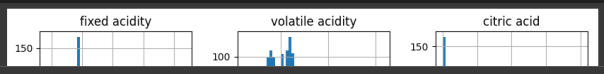
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

```
1 wine.groupby('quality').mean()
```

quality	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
3	8.360000	0.884500	0.171000	2.635000	0.122500	11.000000	24.900000	0.997464	3.398000	0.570000	9.955000
4	7.779245	0.693962	0.174151	2.694340	0.090679	12.264151	36.245283	0.996542	3.381509	0.596415	10.265094
5	8.167254	0.577041	0.243686	2.528855	0.092736	16.983847	56.513950	0.997104	3.304949	0.620969	9.899706
6	8.347179	0.497484	0.273824	2.477194	0.084956	15.711599	40.869906	0.996615	3.318072	0.675329	10.629519
7	8.872362	0.403920	0.375176	2.720603	0.076588	14.045226	35.020101	0.996104	3.290754	0.741256	11.465913
8	8.566667	0.423333	0.391111	2.577778	0.068444	13.277778	33.444444	0.995212	3.267222	0.767778	12.094444

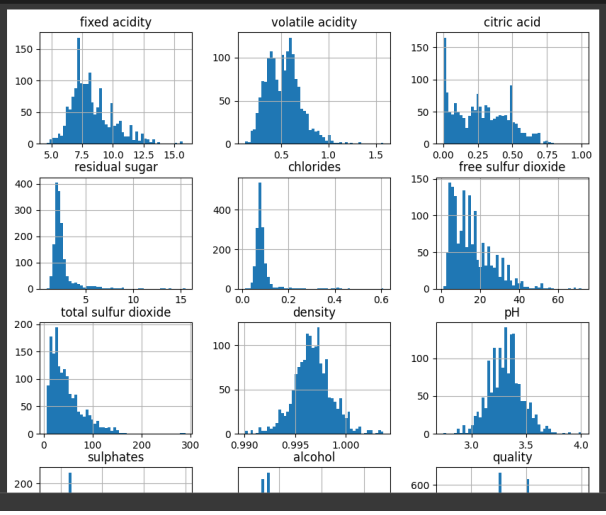
Data visualization

```
[ ] 1 wine.hist(figsize=(10,10),bins=50)
    2 plt.show()
```



Data visualization

```
1 wine.hist(figsize=(10,10),bins=50)
2 plt.show()
```



Data normalization

```
[ ]  1  wine['quality'].unique()
```

```
array([5, 6, 7, 4, 8, 3])
```

```
[ ]  1  wine['goodquality'] = [1 if x >=7 else 0 for x in wine['quality']]
     2  wine.sample(5)
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	goodquality
1156	8.5	0.180	0.51	1.75	0.071	45.0	88.0	0.99524	3.33	0.76	11.8	7	1
140	8.4	0.745	0.11	1.90	0.090	16.0	63.0	0.99650	3.19	0.82	9.6	5	0
704	9.1	0.765	0.04	1.60	0.078	4.0	14.0	0.99800	3.29	0.54	9.7	4	0
1316	5.4	0.740	0.00	1.20	0.041	16.0	46.0	0.99258	4.01	0.59	12.5	6	0
1575	7.5	0.520	0.40	2.20	0.060	12.0	20.0	0.99474	3.26	0.64	11.8	6	0

```
     1  df = wine.values
     2  X = wine.values[:,:-1]
     3  Y = wine.values[:,-1]
```

```
[ ]  1  scaler = MinMaxScaler()
     2  scaler.fit(X)
     3  X_scaled = scaler.fit_transform(X)
     4  print(df)
```

```
[[ 7.4   0.7   0.    ...  9.4   5.    0.  ]
 [ 7.8   0.88  0.    ...  9.8   5.    0.  ]
 [ 7.8   0.76  0.04  ...  9.8   5.    0.  ]
 ...
 [ 6.3   0.51  0.13  ... 11.    6.    0.  ]
 [ 5.9   0.645 0.12  ... 10.2   5.    0.  ]
 [ 6.    0.31  0.47  ... 11.    6.    0.  ]]
```

Data preparing for Machine learning modeling


```
 [ 5.9   0.645 0.12  ... 10.2   5.    0.  ]
 [ 6.    0.31  0.47  ... 11.    6.    0.  ]]
```

Data preparing for Machine learning modeling


```
[ ]  1  X_train, X_test, Y_train, Y_test = train_test_split(X,Y,random_state=7, test_size=0.2)
```

```
[ ]  1  model_res=pd.DataFrame(columns=['Model','Score'])
```

Training

+ Code + Text

```
     1  def deploy_logistic_classifier(model, X_test):
     2      y_pred = model.predict(X_test)
     3      model_res.loc[len(model_res)] = ['LogisticRegression', accuracy_score(Y_test,y_pred)]
     4      print("Predict (Logistic Classifier):", y_pred)
     5      print(classification_report(Y_test, y_pred))
     6
     7  def deploy_svm_classifier(model, X_test):
     8      y_pred = model.predict(X_test)
     9      model_res.loc[len(model_res)] = ['SVM Classifier', accuracy_score(Y_test,y_pred)]
    10      print("Predict (SVM Classifier):", y_pred)
    11      print(classification_report(Y_test, y_pred))
    12
    13  def deploy_neural_network_classifier(model, X_test):
    14      y_pred = model.predict(X_test)
    15      model_res.loc[len(model_res)] = ['Neural Network Classifier', accuracy_score(Y_test,y_pred)]
    16      print("Predict (Neural Network Classifier):", y_pred)
    17      print(classification_report(Y_test, y_pred))
```

Prediction and Evaluation

```
[ ]  1  model_logistic = LogisticRegression()
     2  model_logistic.fit(X_train, Y_train)
     3  deploy_logistic_classifier(model_logistic, X_test)
     4  model_res
```

Prediction and Evaluation

```python
model_logistic = LogisticRegression()
model_logistic.fit(X_train, Y_train)
deploy_logistic_classifier(model_logistic, X_test)
model_res
```

```
Predict (Logistic Classifier): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 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. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1.
 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 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. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.
 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 1. 1. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0.
 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 1. 0. 0. 0.
 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1.
 0. 1. 0. 1. 1. 0. 0. 0.]
              precision    recall  f1-score   support

         0.0       1.00      0.99      1.00       279
         1.0       0.95      1.00      0.98        41

    accuracy                           0.99       320
   macro avg       0.98      1.00      0.99       320
weighted avg       0.99      0.99      0.99       320
```

	Model	Score
0	LogisticRegression	0.99375

```python
model_svm = SVC()
model_svm.fit(X_train, Y_train)
deploy_svm_classifier(model_svm, X_test)
model_res
```

```
Predict (SVM Classifier): [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. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
```

```python
model_svm = SVC()
model_svm.fit(X_train, Y_train)
deploy_svm_classifier(model_svm, X_test)
model_res
```

```
Predict (SVM Classifier): [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. 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. 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. 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. 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. 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.
 0. 0. 0. 0. 0. 0. 0. 0.]
              precision    recall  f1-score   support

         0.0       0.87      1.00      0.93       279
         1.0       0.00      0.00      0.00        41

    accuracy                           0.87       320
   macro avg       0.44      0.50      0.47       320
weighted avg       0.76      0.87      0.81       320
```

	Model	Score
0	LogisticRegression	0.993750
1	SVM Classifier	0.871875

```python
model_neural_network = MLPClassifier()
model_neural_network.fit(X_train, Y_train)
deploy_neural_network_classifier(model_neural_network, X_test)
model_res
```

```
Predict (Neural Network Classifier): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 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. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1.
 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.
```

```
model_neural_network = MLPClassifier()
model_neural_network.fit(X_train, Y_train)
deploy_neural_network_classifier(model_neural_network, X_test)
model_res
```

```
Predict (Neural Network Classifier): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1.
 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 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. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 1. 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. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 1. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0.
 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 1. 0. 0. 0.
 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1.
 0. 1. 0. 1. 1. 0. 0. 0.]
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       279
         1.0       1.00      1.00      1.00        41

    accuracy                           1.00       320
   macro avg       1.00      1.00      1.00       320
weighted avg       1.00      1.00      1.00       320
```

	Model	Score
0	LogisticRegression	0.993750
1	SVM Classifier	0.871875
2	Neural Network Classifier	1.000000

```
model_res = model_res.sort_values(by='Score',ascending=False)
model_res
```

	Model	Score
2	Neural Network Classifier	1.000000
0	LogisticRegression	0.993750

```
model_res = model_res.sort_values(by='Score',ascending=False)
model_res
```

	Model	Score
2	Neural Network Classifier	1.000000
0	LogisticRegression	0.993750
1	SVM Classifier	0.871875

QUESTION 2:

(1) Let's study yourself and present what is the overfitting problem of machine learning models? Describe overfitting phenomena.

Explain the overfitting issue in machine learning models and describe the phenomenon.

a. Overfitting in Machine Learning Models:

Overfitting arises when a machine learning model captures intricate details and noise from the training data to such an extent that it performs poorly on unseen data. Essentially, the model becomes overly complex, identifying spurious patterns that do not extend beyond the training dataset. This leads to a high variance in predictions, resulting in superior performance on the training set compared to the test set or any new data. (Overfitting xảy ra khi một mô hình học máy nắm bắt các chi tiết tinh vi và nhiễu từ dữ liệu huấn luyện đến mức nó hoạt động kém trên dữ liệu chưa thấy. Cơ bản, mô hình trở nên quá phức tạp, nhận biết các mô hình giả mạo không mở rộng ra khỏi tập dữ liệu huấn luyện. Điều này dẫn đến

sự thay đổi lớn trong các dự đoán, dẫn đến hiệu suất vượt trội trên tập huấn luyện so với tập kiểm tra hoặc bất kỳ dữ liệu mới nào.)

b. Causes of Overfitting:

• Excessive features: An abundance of features relative to observations can cause the model to tightly fit to the training data's specific details. (• Quá nhiều đặc trưng: Sự dư thừa của các đặc trưng so với các quan sát có thể khiến mô hình khớp chặt chẽ với các chi tiết cụ thể của dữ liệu huấn luyện.)

• Model complexity: A highly intricate model with numerous parameters is more prone to overfitting. (• Độ phức tạp của mô hình: Một mô hình phức tạp với nhiều tham số có xu hướng overfitting hơn.)

• Inadequate training data: A limited dataset amplifies the risk of overfitting as the model may not encounter enough data variance to accurately learn underlying patterns. (• Dữ liệu huấn luyện không đủ: Một tập dữ liệu hạn chế làm tăng nguy cơ overfitting vì mô hình có thể không gặp đủ sự thay đổi dữ liệu để học một cách chính xác các mô hình cơ bản.)

(2) Please present solutions to avoid overfitting, from general solutions to solutions for each specific machine learning method (including learned machine learning methods and you can be expanded to other learning methods).

Propose methods to mitigate overfitting, encompassing general strategies and techniques tailored to specific machine learning methods. (Đề xuất các phương pháp giảm thiểu overfitting, bao gồm các chiến lược chung và các kỹ thuật được điều chỉnh cho các phương pháp học máy cụ thể.)

a. General Strategies: (a. Các chiến lược chung:)

• Simplify the model: Opt for a simpler model with fewer parameters, such as linear models, if they suit the problem adequately. (• Đơn giản hóa mô hình: Chọn một mô hình đơn giản hơn với ít tham số, như các mô hình tuyến tính, nếu chúng phù hợp với vấn đề.)

• Cross-validation: Employ cross-validation methodologies like k-fold cross-validation to ensure stable model performance across different data subsets. (• Cross-validation: Sử dụng các phương pháp cross-validation như k-fold cross-validation để đảm bảo hiệu suất mô hình ổn định trên các tập con dữ liệu khác nhau.)

• Data augmentation: Expand the size and diversity of the training set by introducing slightly modified copies of existing data or newly generated synthetic data to alleviate overfitting. (Tăng cường dữ liệu: Mở rộng kích thước và đa dạng của tập huấn luyện bằng cách giới

thiệu các bản sao được chỉnh sửa nhẹ của dữ liệu hiện có hoặc dữ liệu tổng hợp mới được tạo ra để giảm thiểu overfitting.)

• Feature selection/reduction: Decrease input feature count using methods like Principal Component Analysis (PCA) or manually selecting pertinent features to reduce complexity. (• Lựa chọn/giảm đặc trưng: Giảm số lượng đặc trưng đầu vào bằng cách sử dụng các phương pháp như Principal Component Analysis (PCA) hoặc lựa chọn thủ công các đặc trưng liên quan để giảm độ phức tạp.)

b. Specific Techniques for Machine Learning: (b. Các kỹ thuật cụ thể cho học máy:)

• Decision Trees: Implement pruning techniques, specify a maximum tree depth, or set minimum samples per leaf to regulate growth. (• Cây quyết định: Thực hiện các kỹ thuật cắt tỉa, xác định độ sâu tối đa của cây, hoặc đặt số mẫu tối thiểu cho mỗi lá để điều chỉnh sự phát triển.)

• Support Vector Machines (SVMs): Opt for an appropriate kernel and regularization parameter (C) to prevent overfitting. Utilizing simpler kernels, like linear, can be beneficial for certain datasets. (• Máy vector hỗ trợ (SVMs): Chọn một kernel phù hợp và tham số điều chuẩn © để ngăn chặn overfitting. Sử dụng các kernel đơn giản hơn, như tuyến tính, có thể có lợi cho một số tập dữ liệu.)

• K-nearest neighbor: Determine the number of neighbors (k) via cross-validation to ensure the model does not overemphasize noise in the data prematurely. (• K-nearest neighbor: Xác định số lượng hàng xóm (k) thông qua cross-validation để đảm bảo rằng mô hình không nhấn mạnh quá sớm nhiễu trong dữ liệu.)

(3) Present the problems mentioned in questions 1 and 2 through one or more real data sets (for classification and/and regression problems). Note that you must choose data sets and machine learning method where overfitting occurs.

Overfitting Classification

Overfitting classification

```
1  import warnings
2  warnings.filterwarnings("ignore")
3  import numpy as np
4  import matplotlib.pyplot as plt
5  import pandas as pd
6
7  df = pd.read_csv('fruit.csv')
8  df.sample(10)
```

	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
30	3	orange	selected_seconds	158	7.1	7.5	0.79
32	3	orange	selected_seconds	164	7.2	7.0	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79
58	4	lemon	unknown	118	6.1	8.1	0.70
56	4	lemon	unknown	116	5.9	8.1	0.73
36	3	orange	turkey_navel	160	7.1	7.6	0.76
43	4	lemon	spanish_belsan	194	7.2	10.3	0.70
48	4	lemon	spanish_belsan	174	7.3	10.1	0.72
26	3	orange	spanish_jumbo	362	9.6	9.2	0.74
27	3	orange	selected_seconds	204	7.5	9.2	0.77

```
[12] 1  X = df.iloc[:,3:6].values
     2  Y = df.iloc[:,0].values
     3  print(X)
     4  Y
```

```
[[192.    8.4   7.3]
 [180.    8.    6.8]
 [176.    7.4   7.2]
 [ 86.    6.2   4.7]
```

0s completed at 10:46 PM

+ Code + Text

RAM Disk Colab AI

```
[13] 1  from sklearn.model_selection import train_test_split
     2  X_train,X_test, y_train, y_test = train_test_split(X,Y,test_size = 0.2,random_state = 0)
```

```
[22] 1  from sklearn import tree
     2
     3  clf_2 = tree.DecisionTreeClassifier(min_samples_split = 2)
     4
     5  clf_2.fit(X_train, y_train)
     6
     7  pred_2 = clf_2.predict(X_test)
     8
     9  print(pred_2)
    10
```

```
[3 3 4 1 3 1 1 4 3 3 2 3]
```

```
[23] 1  pred_train = clf_2.predict(X_train)
     2  pred_train
```

```
array([3, 3, 3, 3, 4, 2, 1, 3, 4, 3, 3, 4, 1, 4, 3, 1, 2, 3, 1, 4, 1, 4,
       1, 1, 3, 1, 4, 4, 4, 3, 1, 1, 4, 3, 2, 1, 3, 1, 1, 1, 3, 4, 2, 1,
       4, 4, 4])
```

```
[36] 1  from sklearn.metrics import confusion_matrix, precision_score,recall_score , accuracy_score
     2  print("Accuracy score on test data",accuracy_score(y_test,pred_2))
     3  print("Accuracy score on training data",accuracy_score(y_train,pred_train))
```

```
Accuracy score on test data 0.75
Accuracy score on training data 1.0
```

Overfitting regression

```
[32] 1  import warnings
     2  warnings.filterwarnings("ignore")
     3  import matplotlib.pyplot as plt
     4  import numpy as np
     5  import matplotlib.pyplot as plt
```

0s completed at 10:46 PM

Overfitting Regression

Overfitting regression

```
[32] 1  import warnings
     2  warnings.filterwarnings("ignore")
     3  import matplotlib.pyplot as plt
     4  import numpy as np
     5  import matplotlib.pyplot as plt
     6  import pandas as pd
     7
     8  df = pd.read_csv('fruit.csv')
     9  df.sample(10)
```

	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
54	4	lemon	unknown	116	6.1	8.5	0.71
50	4	lemon	unknown	130	6.0	8.2	0.71
9	1	apple	braeburn	172	7.4	7.0	0.89
2	1	apple	granny_smith	176	7.4	7.2	0.60
51	4	lemon	unknown	116	6.0	7.5	0.72
6	2	mandarin	mandarin	80	5.9	4.3	0.81
12	1	apple	braeburn	154	7.0	7.1	0.88
34	3	orange	turkey_navel	142	7.6	7.8	0.75
1	1	apple	granny_smith	180	8.0	6.8	0.59
21	1	apple	cripps_pink	156	7.4	7.4	0.84

```
[27] 1  X = df.iloc[:, 3:6].values
     2  Y = df.iloc[:, 0].values
     3  print(X)
     4  Y
```

```
[28] 1  from sklearn.model_selection import train_test_split
     2  X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
```

0s completed at 10:46 PM

```
1  from sklearn.tree import DecisionTreeRegressor
2
3  reg_2 = DecisionTreeRegressor(min_samples_split=2)
4
5  reg_2.fit(X_train, y_train)
6
7  pred_2 = reg_2.predict(X_test)
8
9  print(pred_2)
```

```
[3. 3. 4. 1. 1. 1. 1. 4. 3. 1. 2. 3.]
```

```
[30] 1  pred_train = reg_2.predict(X_train)
     2  pred_train
```

```
array([3., 3., 3., 3., 4., 2., 1., 3., 4., 3., 3., 4., 1., 4., 3., 1., 2.,
       3., 1., 4., 1., 4., 1., 1., 3., 1., 4., 4., 4., 3., 1., 1., 4., 3.,
       2., 1., 3., 1., 1., 1., 3., 4., 2., 1., 4., 4., 4.])
```

```
1  from sklearn.metrics import mean_squared_error, r2_score
2
3  mse_2 = mean_squared_error(y_test, pred_2)
4  r2_2 = r2_score(y_test, pred_2)
5
6  print("Mean Squared Error (Test Set):", mse_2)
7  print("R-squared Score (Test Set):", r2_2)
8
9  mse_train = mean_squared_error(y_train, pred_train)
10 r2_train = r2_score(y_train, pred_train)
11
12 print("Mean Squared Error (Training Set):", mse_train)
13 print("R-squared Score (Training Set):", r2_train)
```

```
Mean Squared Error (Test Set): 1.0
R-squared Score (Test Set): 0.1955307262569832
Mean Squared Error (Training Set): 0.0
R-squared Score (Training Set): 1.0
```