Assignment 3: Non-Linear Models and Validation Metrics

- **Full Name** = Hiu Sum Yuen
- **UCID** = 30162577

In this assignment, you will need to write code that uses non-linear models to perform classification and regression tasks. You will also be asked to describe the process by which you came up with the code. More details can be found below. Please cite any websites or AI tools that you used to help you with this assignment.

Question	Point
Part 1: Regression	**14.5**
Step 0: Import Libraries	
Step 1: Data Input	0.5
Step 2: Data Processing	0
Step 3: Implement Machine Learning Model	0.5
Step 4: Validate Model	0.5
Step 5: Visualize Results	3
Questions	6
Process Description	4
Part 2: Classification	**17.5**
Step 1: Data Input	2
Step 2: Data Processing	1.5
Step 3: Implement Machine Learning Model	
Step 4: Validate Mode	
Step 5: Visualize Results	4
Questions	6
Process Description	4
Part 3: Observations/Interpretation	**3**
Part 4: Reflection	**2**
Total	**37**
Bonus	
Part 5: Bonus Question	**3**

Import Libraries

```
!pip install seaborn
```

```
    Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.13.1)
    Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.25.2)
    Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.5.3)
    Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.10/dist-packages (from seaborn) (3.7.1)
    Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->se
    Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seabor
    Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->s
    Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->s
    Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->sea
    Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seabo
    Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->se
    Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4
    Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn) (2023.4)
    Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib!=3
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Part 1: Regression (14.5 marks)

For this section, we will be continuing with the concrete example from yellowbrick. You will need to compare these results to the results from the previous assignment. Please use the results from the solution if you were unable to complete Assignment 2

Step 1: Data Input (0.5 marks)

The data used for this task can be downloaded using the yellowbrick library: https://www.scikit-yb.org/en/latest/api/datasets/concrete.html

Use the yellowbrick function `load_concrete()` to load the concrete dataset into the feature matrix `X` and target vector `y`.

```
from yellowbrick.datasets import load_concrete
from sklearn.model_selection import train_test_split


# TO DO: Import concrete dataset from yellowbrick library
random_state_seed = 0
X, y = load_concrete()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = random_state_seed)
```

Step 2: Data Processing (0 marks)

Data processing was completed in the previous assignment. No need to repeat here.

This is just for your information and no action is required from you for this step.

Step 3: Implement Machine Learning Model (0.5 marks)

1. Import the Decision Tree, Random Forest and Gradient Boosting Machines regression models from sklearn
2. Instantiate the three models with `max_depth = 5`. Are there any other parameters that you will need to set?
3. Implement each machine learning model with `X` and `y`

Step 4: Validate Model (0.5 marks)

Calculate the average training and validation accuracy using mean squared error with cross-validation. To do this, you will need to set `scoring='neg_mean_squared_error'` in your `cross_validate` function and negate the results (multiply by -1)

Step 5: Visualize Results (3 marks)

1. Create a pandas DataFrame `results` with columns: Training accuracy and Validation accuracy, and index: DT, RF and GB
2. Add the accuracy results to the `results` DataFrame
3. Print `results`

```
# TO DO: ADD YOUR CODE HERE FOR STEPS 3-5
# Note: for any random state parameters, you can use random_state = 0
# HINT: USING A LOOP TO STORE THE DATA IN YOUR RESULTS DATAFRAME WILL BE MORE EFFICIENT


# Import the necessary libraries
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor


#instantiate a set of models
random_state_seed = 0
depth = 5
gamma = 0.15
models = []
DTR = DecisionTreeRegressor(random_state = random_state_seed, max_depth = depth)
RFR = RandomForestRegressor(random_state = random_state_seed, max_depth = depth, n_estimators = 50)
GBR = GradientBoostingRegressor(random_state = random_state_seed, max_depth = depth, n_estimators = 50, learning_rate= gamma)

for model in [DTR, RFR, GBR]:
    models.append(model)


# Import the necessary libraries
from sklearn.model_selection import cross_validate
```

```
# Train and cross validate the models
scoring_method = "neg_mean_squared_error"
accuracy_results = []
for model in models:
    model.fit(X_train, y_train)
    CV_result = cross_validate(model, X, y, scoring = scoring_method, return_train_score = True)
    mean_accuracy_train = np.mean(CV_result["train_score"]) * -1
    mean_accuracy_test = np.mean(CV_result["test_score"]) * -1
    accuracy_results.append([mean_accuracy_train, mean_accuracy_test])


# Show accuracy_results in a dataframe
results = pd.DataFrame(accuracy_results, columns = ["Training Accuracy", "Validation Accuracy"], index = ["DT", "RF", "GB"])
results
```

	Training Accuracy	Validation Accuracy
DT	47.918561	163.087775
RF	32.465128	158.028955
GB	4.904112	107.038631

Repeat the step above to print the R2 score instead of the mean-squared error. For this case, you can use `scoring='r2'`.

Due to the similarity of this to the main part of step 5, this part is 0.5 and the main part of step 5 is 2.5 of the total 3 points for this step.

```
# TO DO: ADD YOUR CODE HERE
# This would be similar to the main step, the main difference is the scoring.

# Train and cross validate the models
scoring_method = "r2"
accuracy_results = []
for model in models:
    model.fit(X_train, y_train)
    CV_result = cross_validate(model, X, y, scoring = scoring_method, return_train_score = True)
    mean_accuracy_train = np.mean(CV_result["train_score"])
    mean_accuracy_test = np.mean(CV_result["test_score"])
    accuracy_results.append([mean_accuracy_train, mean_accuracy_test])

# Show accuracy_results in a dataframe
results = pd.DataFrame(accuracy_results, columns = ["Training Accuracy", "Validation Accuracy"], index = ["DT", "RF", "GB"])
results
```

	Training Accuracy	Validation Accuracy
DT	0.822887	0.176210
RF	0.879663	0.166801
GB	0.982097	0.421480

Questions (6 marks)

1. How do these results compare to the results using a linear model in the previous assignment? Use values.
2. Out of the models you tested, which model would you select for this dataset and why?
3. If you wanted to increase the accuracy of the tree-based models, what would you do? Provide two suggestions.

YOUR ANSWERS HERE

1. Linear Regression provided results of MSE at 110.3, 95.6 for train and test respectively, and r2 at 0.609 and 0.636 for train and test respectively. Compared to the results above from models with higher depth, linear regression for this dataset is much more reliable; The above machine learning algorithms all show a drastic difference between train and test accuracy, where train accuracy is much higher than test accuracy, showing that these algorithms overfit the dataset.
2. I choose Gradient Boost; GB not only shows highest accuracy in both train and test respectively, it also overfits the least for this dataset out of the three algorithms.
3. Firstly, seeing as overfitting is a primary concern of poor validation accuracy when training accuracy is high, we should try models with lower depth. Secondly, training with more trees in RF and GB can provide more normalization, increasing the accuracy of validation.

Process Description (4 marks)

Please describe the process you used to create your code. Cite any websites or generative AI tools used. You can use the following questions as guidance:

1. Where did you source your code?
2. In what order did you complete the steps?
3. If you used generative AI, what prompts did you use? Did you need to modify the code at all? Why or why not?
4. Did you have any challenges? If yes, what were they? If not, what helped you to be successful?

DESCRIBE YOUR PROCESS HERE

1. sklearn website, stack overflow, and chatgpt for DT, RF, and GB importing and descriptions.
2. According to the questions step by step.
3. I asked chatgpt about quick descriptions on DT, RF, and GB, utilizing github copilot to code faster.
4. I originally oversaw negative accuracy when looking for r2 accuracy.

Part 2: Classification (17.5 marks)

You have been asked to develop code that can help the user classify different wine samples. Following the machine learning workflow described in class, write the relevant code in each of the steps below:

Step 1: Data Input (2 marks)

The data used for this task can be downloaded from UCI: https://archive.ics.uci.edu/dataset/109/wine

Use the pandas library to load the dataset. You must define the column headers if they are not included in the dataset

You will need to split the dataset into feature matrix `X` and target vector `y`. Which column represents the target vector?

Print the size and type of `X` and `y`

```
!pip install -U ucimlrepo
```

```
Collecting ucimlrepo
  Downloading ucimlrepo-0.0.3-py3-none-any.whl (7.0 kB)
Installing collected packages: ucimlrepo
Successfully installed ucimlrepo-0.0.3
```

```
# TO DO: Import wine dataset
from ucimlrepo import fetch_ucirepo
wine_dataset = fetch_ucirepo(id=109)
X, y = wine_dataset.data.features, wine_dataset.data.targets
```

```
X
```

	Alcohol	Malicacid	Ash	Alcalinity_of_ash	Magnesium	Total_phenols	Flavanc
0	14.23	1.71	2.43	15.6	127	2.80	
1	13.20	1.78	2.14	11.2	100	2.65	
2	13.16	2.36	2.67	18.6	101	2.80	
3	14.37	1.95	2.50	16.8	113	3.85	
4	13.24	2.59	2.87	21.0	118	2.80	
...	...	...	...	...	...	...	
173	13.71	5.65	2.45	20.5	95	1.68	
174	13.40	3.91	2.48	23.0	102	1.80	
175	13.27	4.28	2.26	20.0	120	1.59	
176	13.17	2.59	2.37	20.0	120	1.65	
177	14.13	4.10	2.74	24.5	96	2.05	

178 rows × 13 columns

y

	class
0	1
1	1
2	1
3	1
4	1
...	...
173	3
174	3
175	3
176	3
177	3

178 rows × 1 columns

```
# Print size and type of X and y
print(f"size of X: {X.size}")
print(f"type of X: {type(X)}")
print(f"size of y: {y.size}")
print(f"type of y: {type(y)}")
```

```
size of X: 2314
type of X: <class 'pandas.core.frame.DataFrame'>
size of y: 178
type of y: <class 'pandas.core.frame.DataFrame'>
```

Step 2: Data Processing (1.5 marks)

Print the first five rows of the dataset to inspect:

```
# TO DO: ADD YOUR CODE HERE
X.head()
```

	Alcohol	Malicacid	Ash	Alcalinity_of_ash	Magnesium	Total_phenols	Flavanoic
0	14.23	1.71	2.43	15.6	127	2.80	3.(
1	13.20	1.78	2.14	11.2	100	2.65	2.7
2	13.16	2.36	2.67	18.6	101	2.80	3.2
3	14.37	1.95	2.50	16.8	113	3.85	3.4
4	13.24	2.59	2.87	21.0	118	2.80	2.6

Check to see if there are any missing values in the dataset. If necessary, select an appropriate method to fill-in the missing values

```
# TO DO: ADD YOUR CODE HERE
# Check X
X.isnull().sum()
```

```
Alcohol                         0
Malicacid                       0
Ash                             0
Alcalinity_of_ash               0
Magnesium                       0
Total_phenols                   0
Flavanoids                      0
Nonflavanoid_phenols            0
Proanthocyanins                 0
Color_intensity                 0
Hue                             0
0D280_0D315_of_diluted_wines    0
Proline                         0
dtype: int64
```

```
# Check y
y.isnull().sum()
```

```
class    0
dtype: int64
```

How many samples do we have of each type of wine?

```
# TO DO: ADD YOUR CODE HERE
y.value_counts()
```

```
class
2        71
1        59
3        48
dtype: int64
```

Step 3: Implement Machine Learning Model

1. Import `SVC` and `DecisionTreeClassifier` from sklearn
2. Instantiate models as `SVC()` and `DecisionTreeClassifier(max_depth = 3)`
3. Implement the machine learning model with `X` and `y`

Step 4: Validate Model

Calculate the average training and validation accuracy using `cross_validate` for the two different models listed in Step 3. For this case, use `scoring='accuracy'`

Step 5: Visualize Results (4 marks)

There is no individual mark for Steps 3 and 4 and those grades are included within the four points.

Step 5.1: Compare Models (2 out of total 4 marks)

1. Create a pandas DataFrame `results` with columns: Training accuracy and Validation accuracy
2. Add the data size, training and validation accuracy for each dataset to the `results` DataFrame
3. Print `results`

```
# TO DO: ADD YOUR CODE HERE FOR STEPS 3-5
# Note: for any random state parameters, you can use random_state = 0
# HINT: USING A LOOP TO STORE THE DATA IN YOUR RESULTS DATAFRAME WILL BE MORE EFFICIENT


# Import the necessary libraries
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier


# Split the data into training and testing sets
random_state_seed = 0
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = random_state_seed)


# Instantiate a set of models
models = []
SVC_model = SVC()
DTC_model = DecisionTreeClassifier(max_depth = 3)
for model in [SVC_model, DTC_model]:
    models.append(model)


# Train and cross validate the models
scoring_method = "accuracy"
accuracy_results = []
for model in models:
    model.fit(X_train, y_train)
    CV_result = cross_validate(model, X, y, scoring = scoring_method, return_train_score = True)
    mean_accuracy_train = np.mean(CV_result["train_score"])
    mean_accuracy_test = np.mean(CV_result["test_score"])
    accuracy_results.append([len(X), mean_accuracy_train, mean_accuracy_test])

# Show accuracy_results in a dataframe
results = pd.DataFrame(accuracy_results, columns = ["Data size", "Training Accuracy", "Validation Accuracy"], index = ["SVC", "DT
results
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passe
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passe
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passe
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passe
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passe
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passe
  y = column_or_1d(y, warn=True)
```

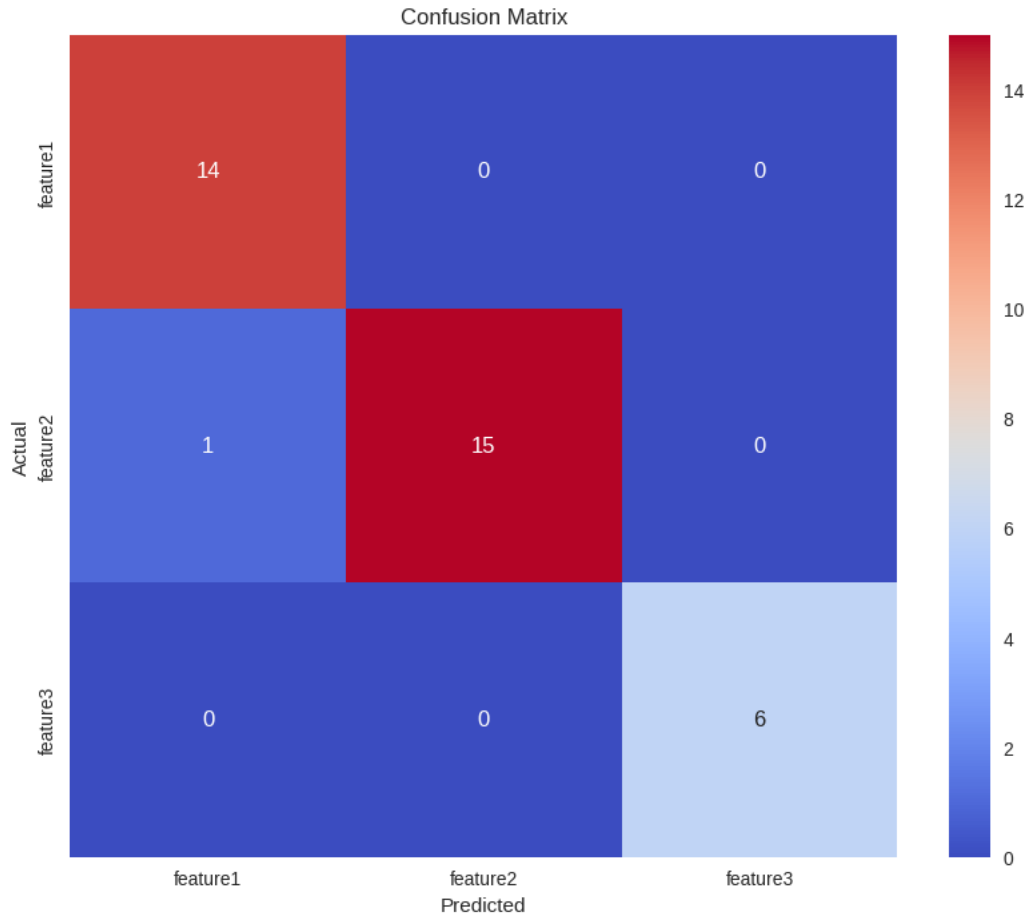
	Data size	Training Accuracy	Validation Accuracy
SVC	178	0.703743	0.663492
DTC	178	0.974756	0.882063

Step 5.2: Visualize Classification Errors (2 out of total 4 marks)

Which method gave the highest accuracy? Use this method to print the confusion matrix and classification report:

```
# TO DO: Implement best model
best_model = DTC_model


# TO DO: Print confusion matrix using a heatmap


# Import the necessary libraries
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Make prediction
y_pred = best_model.predict(X_test)


# Create confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)


# Plot confusion matrix
features = ["feature1", "feature2", "feature3"]
fig, ax = plt.subplots(figsize = (10, 8))
sns.heatmap(conf_matrix, annot = True, cmap = "coolwarm", xticklabels = features, yticklabels = features)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```




```
# TO DO: Print classification report


# Import the necessary libraries
from sklearn.metrics import classification_report


print(classification_report(y_test, y_pred))
```

```
                  precision    recall  f1-score   support

               1       0.93      1.00      0.97        14
               2       1.00      0.94      0.97        16
               3       1.00      1.00      1.00         6

        accuracy                           0.97        36
       macro avg       0.98      0.98      0.98        36
    weighted avg       0.97      0.97      0.97        36
```

Questions (6 marks)

1. How do the training and validation accuracy change depending on the method used? Explain with values.
2. What are two reasons why the support vector machines model did not work as well as the tree-based model?
3. How many samples were incorrectly classified in step 5.2?
4. In this case, is maximizing precision or recall more important? Why?

YOUR ANSWERS HERE

1. SVC performs poorly on the wine dataset at 0.70 and 0.66 for train and test respectively in comparison to DTC which had accuracies of 0.97 and 0.89 respectively. This shows that the higher degree of complexity by DTC performs better on the wine dataset. Also, both models show signs of slight overfitting since training accuracy is fairly higher than validation accuracy.
2. Firstly, SVC has a lower level of complexity compared to DTC, showing that this dataset cannot be separated into hyperplanes linearly with low inaccuracy. Secondly, we can increase the complexity of the SVC by providing more features.
3. There is 1 misclssification of false negative class 1 classification when it should be class 2.
4. Recall is already maximized at 14/14 in this case, so we should look to maximize recall which is at 14/15 since it can be improved on.

Process Description (4 marks)

Please describe the process you used to create your code. Cite any websites or generative AI tools used. You can use the following questions as guidance:

1. Where did you source your code?
2. In what order did you complete the steps?
3. If you used generative AI, what prompts did you use? Did you need to modify the code at all? Why or why not?
4. Did you have any challenges? If yes, what were they? If not, what helped you to be successful?

DESCRIBE YOUR PROCESS HERE

1. Lecture slides, sklearn website, stack overflow, and chatgpt for SVC, and decision tree classification importing and descriptions.
2. According to the questions step by step.
3. I asked chatgpt about quick descriptions on SVC, and decision tree classification, utilizing github copilot to code faster.
4. Recall and precision are topics that I had to google and wrap my head around as it is confusing to remember and understand even when I comprehend the concept previously.

Part 3: Observations/Interpretation (3 marks)

Describe any pattern you see in the results. Relate your findings to what we discussed during lectures. Include data to justify your findings.

ADD YOUR FINDINGS HERE

The pattern seen by this assignment is that even if decision tree models can be powerfully predictive algorithms, it is crucial to test around for the perfect depth to train your model on for your specific type of dataset; There is no "one size fits all" complexity as machine learning algorithms vary in performance based on the data you work with. Training a machine learning model with too high complexity can cause overfitting seen by Part 1 of this assignment, and vice versa seen by SVC in Part 2 of this assignment. The primary pattern we are expected to see here is identifying what degree of complexity works well with the data we are given.

Part 4: Reflection (2 marks)

Include a sentence or two about:

- what you liked or disliked,
- found interesting, confusing, challangeing, motivating while working on this assignment.

ADD YOUR THOUGHTS HERE

1. I found this assignment quite tidious in terms of the reflective questions of process description. I think that the challenges question is good to have, but "what order I completed the steps and how I source my code is going to be nearly the same everytime.
2. I am reinforcing a lot of the knowledge from lectures due to these assignments as it is quite practical to have us actually use these extentions. I enjoy the process of searching about information that is more specific whilst having the safety net of step by step guidence to what I should be doing next.
3. This assignment takes a long time to do.

Part 5: Bonus Question (3 marks)

Repeat Part 2 and compare the support vector machines model used to `LinearSVC(max_iter=5000)`. Does using `LinearSVC` improve the results? Why or why not?

Is `LinearSVC` a good fit for this dataset? Why or why not?

```
# TO DO: ADD YOUR CODE HERE
```