Assignment 1.1 Advanced Reading: Machine Learning Tools, Process and Your Learnings

Name: Ery Jay P. Pisalbon Section: CPE31S3 Instructor: Dr.Alonica Villanueva

1) Summarize the following:

1.1) What is overfitting and underfitting? Provide examples.

Overfitting and Underfitting are **both causes for poor performance of machine learning algorithms but may vary on two different instances**:

1. Overfitting occurs when a machine learning model **works well in training** (100% accuracy) but **works poorly when we feed new data to the model**.

   Example: A student **memorizes the formula for every physics problem but doesn't understand its concepts behind it**

2. While Underfitting occurs when a machine learning model **works both poorly on training phase and test phase.**

   Example: A student **doesn't understand the module or learning materials on a specific course that results in grading on exams or quiz**

1.2) Define cross fold validation. How is it useful?

Cross-fold validation is a resampling technique used to **evaluate the performance** of a machine learning model on a **limited data sample.** Cross-fold validation is a useful tool to **prevent overfitting to your model.**

2) Demonstrate data splitting for training and testing data. Using any preprocessing library in Python on the iris dataset provided in the module.

In [1]:
```
!pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in e:\anacondanavigator\lib\site-package
s (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in e:\anacondanavigator\lib\site-packag
es (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.3.2 in e:\anacondanavigator\lib\site-package
s (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in e:\anacondanavigator\lib\site-packag
es (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in e:\anacondanavigator\lib\site
-packages (from scikit-learn) (2.2.0)
```

In [2]:
```
from sklearn.model_selection import train_test_split
import pandas as pd

iris = pd.read_csv("Dataset\Iris.csv") # getting the dataset
iris.head() # loading the first five records in the dataset
```

Out[2]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [3]:
```
iris.drop('Id',axis=1,inplace=True) # removing unnecesarry columns to predict Y val
```

In [6]:
```
iris.head() # checking
```

Out[6]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [7]:
```
#separating the features and targets
X = iris.drop('Species', axis=1) #features
Y = iris['Species'] #target
```

In [8]:
```
#checking
X
```

Out[8]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

In [9]:
```
Y
```

Out[9]:
```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
             ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: Species, Length: 150, dtype: object
```

In [10]:
```
# partitioning the features and targets for training and testing
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.3, random_st
#test size - percentage of the dataset that will go to testing
#random_state - saves the partition to memory if ever you want to use the same part
```

In [11]:
```
#checking
X_train
```

Out[11]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
13	4.3	3.0	1.1	0.1
102	7.1	3.0	5.9	2.1
67	5.8	2.7	4.1	1.0
34	4.9	3.1	1.5	0.1
98	5.1	2.5	3.0	1.1
...	...	...	...	...
63	6.1	2.9	4.7	1.4
70	5.9	3.2	4.8	1.8
81	5.5	2.4	3.7	1.0
11	4.8	3.4	1.6	0.2
95	5.7	3.0	4.2	1.2

105 rows × 4 columns

In [17]:
```
X_test
```

Out[17]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
33	5.5	4.2	1.4	0.2
16	5.4	3.9	1.3	0.4
43	5.0	3.5	1.6	0.6
129	7.2	3.0	5.8	1.6
50	7.0	3.2	4.7	1.4
123	6.3	2.7	4.9	1.8
68	6.2	2.2	4.5	1.5
53	5.5	2.3	4.0	1.3
146	6.3	2.5	5.0	1.9
1	4.9	3.0	1.4	0.2
147	6.5	3.0	5.2	2.0
32	5.2	4.1	1.5	0.1
31	5.4	3.4	1.5	0.4
122	7.7	2.8	6.7	2.0
127	6.1	3.0	4.9	1.8
74	6.4	2.9	4.3	1.3
88	5.6	3.0	4.1	1.3
96	5.7	2.9	4.2	1.3
42	4.4	3.2	1.3	0.2
134	6.1	2.6	5.6	1.4
80	5.5	2.4	3.8	1.1
48	5.3	3.7	1.5	0.2
90	5.5	2.6	4.4	1.2
65	6.7	3.1	4.4	1.4
97	6.2	2.9	4.3	1.3
64	5.6	2.9	3.6	1.3
93	5.0	2.3	3.3	1.0
114	5.8	2.8	5.1	2.4
25	5.0	3.0	1.6	0.2
41	4.5	2.3	1.3	0.3

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
104	6.5	3.0	5.8	2.2
89	5.5	2.5	4.0	1.3
116	6.5	3.0	5.5	1.8
82	5.8	2.7	3.9	1.2
112	6.8	3.0	5.5	2.1
55	5.7	2.8	4.5	1.3
86	6.7	3.1	4.7	1.5
61	5.9	3.0	4.2	1.5
94	5.6	2.7	4.2	1.3
135	7.7	3.0	6.1	2.3
21	5.1	3.7	1.5	0.4
22	4.6	3.6	1.0	0.2
29	4.7	3.2	1.6	0.2
77	6.7	3.0	5.0	1.7
66	5.6	3.0	4.5	1.5

In [18]:
```
#total number of records in the dataset
len(iris)
```

Out[18]: 150

In [19]:
```
#number of training
len(X_train)
```

Out[19]: 105

In [20]:
```
#number of testing
len(X_test)
```

Out[20]: 45

In [21]:
```
len(iris) * 0.3
```

Out[21]: 45.0

3) Provide a sample confusion matrix and demonstrate the computation of TP, FP, FN, TN. How can we derive accuracy of the model based on the previously mentioned metrics?

n=80	Predicted YES	Predicted NO
Actual NO	TN=20	FP=20
Actual YES	FN= 5	TP=35

Accuracy = (TP+TN)/n Accuracy = (20+35)/80 Accuracy = 55/80 **Accuracy = 0.69**

We can derive the accuracy of the model **based on the correct predictions** (the metrics True Positive and True Negative) over the number of the sample in the dataset

Example: getting the correct answers you get in an exam and getting the percentage of how well did you answer the exam