1. Introduction

There were two datasets for Part A which were "FIT1043-Essay-Features.csv", and "FIT1043-Essay-Features-Submission.csv". These datasets were provided and it is utilised throughout the whole PartA. There were a total of 4 main Task given and few sub task provided inside the 4 main task.

Part A

A1.1

a supervised machine learning is a type of machine learning where it makes predictions and decisions through algorithm given on a labeled dataset. labeled data a learning algorithm annd a model are the essentials components of supervised learning.

labeled data is the foundation of supervised machine learning where is has input and output pairs. The input is the data that are going to be predicted while the output is the corresponding target. Input is also known as features and the output is also known as the outcome.

the train dataset is to allow the algoritm uses the input and output pairs from it and adjust the model's parameters. while the model learns by finding patterns and relationships to make accurate predictions the test dataset is not used during the model training phase. However it is utitlise when measuring the generalise results of the new data. it is also used to calculate the model's perfomance

A1.2

In [1]:
```python
import pandas as pd
features_data = pd.read_csv('FIT1043-Essay-Features.csv')
features = features_data.iloc[:, :-1]
labels = features_data.iloc[:, -1]
features
```

Out[1]:

	essayid	chars	words	commas	apostrophes	punctuations	avg_word_length	sentences	questions	avg_word_sentence	POS
0	1457	2153	426	14	6	0	5.053991	16	0	26.625000	423.99527
1	503	1480	292	9	7	0	5.068493	11	0	26.545455	290.99310
2	253	3964	849	19	26	1	4.669022	49	2	17.326531	843.99054
3	107	988	210	8	7	0	4.704762	12	0	17.500000	207.65378
4	1450	3139	600	13	8	0	5.231667	24	1	25.000000	594.65215
...	...	...	...	...	...	...	...	...	...	...	..
1327	1151	2404	467	16	10	0	5.147752	22	0	21.227273	462.98706
1328	1015	1182	241	0	14	0	4.904564	16	0	15.062500	238.65546
1329	1345	1814	363	5	11	0	4.997245	13	3	27.923077	362.32964
1330	344	1427	287	5	8	0	4.972125	13	1	22.076923	284.65727
1331	1077	2806	542	24	6	0	5.177122	22	3	24.636364	538.98888

1332 rows × 18 columns

In [2]:
```
labels
```

Out[2]:
```
0       4
1       4
2       4
3       3
4       4
       ..
1327    4
1328    3
1329    3
1330    3
1331    4
Name: score, Length: 1332, dtype: int64
```

A1.3

In [3]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=0)
```

A2.1

Binary classifications involves with distinguishing between tow classes, like spam or not a spam. While multi-class classification deals with more than two classes such as categorising objects in images or classifying text into various topics. Multi-class has mutiple catergories to predict while binaray only has two classes.

A2.2

normalising/scaled data means that adjusting the numerical values of features in a dataset to a consistent scale. The purpose of this it to allow the machine learning algorithm is effective and not affect by other variations in feature scale.

In [4]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

A2.3

a.

SVM also known as supervised machine learning model uses classification alogrithms for two_group classification problems. the concepts an characteristic of SVM are such as binary classification, kernel tricks, margin and many more

b.

SVM kernel is a function that allows low dimensional input space transformed into a higher dimensional space. for example, converting non seperable problem into a separable problem

c.

In [5]:
```
from sklearn.svm import SVC
svm = SVC(kernel='linear')
svm = svm.fit(x_train, y_train)
```

A2.4

In [6]:
```python
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(x_train, y_train)
classifier
```

Out[6]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

A3.1

In [7]:
```python
svm.predict(x_test)
```

Out[7]:
```
array([4, 3, 3, 2, 3, 4, 5, 4, 2, 4, 4, 2, 4, 3, 3, 4, 3, 4, 3, 4, 3, 3,
       3, 3, 3, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 3, 3, 4, 3, 3, 4, 2, 3, 3,
       3, 3, 3, 3, 2, 3, 4, 3, 4, 4, 4, 4, 4, 3, 3, 5, 3, 4, 3, 3, 5, 4,
       4, 4, 4, 4, 3, 3, 4, 5, 4, 3, 3, 4, 4, 3, 4, 4, 4, 3, 3, 4, 4, 2,
       4, 4, 3, 4, 4, 3, 1, 3, 3, 4, 3, 2, 4, 4, 3, 4, 4, 2, 4, 4, 3, 4,
       2, 2, 2, 3, 3, 4, 3, 4, 3, 3, 2, 3, 3, 4, 4, 3, 3, 4, 4, 3, 4, 4,
       4, 3, 4, 3, 3, 4, 4, 3, 3, 4, 4, 3, 3, 3, 3, 3, 4, 4, 4, 3, 4, 4,
       4, 3, 3, 3, 3, 3, 3, 4, 3, 3, 3, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4,
       4, 4, 4, 2, 3, 4, 3, 3, 3, 4, 3, 3, 3, 3, 3, 4, 4, 3, 2, 2, 4, 4,
       3, 4, 3, 4, 4, 3, 3, 4, 3, 3, 3, 3, 4, 4, 3, 3, 3, 3, 4, 4, 3, 4,
       4, 4, 4, 4, 4, 4, 2, 4, 3, 3, 3, 3, 4, 3, 4, 4, 4, 4, 5, 4, 4, 4,
       3, 3, 3, 4, 4, 3, 4, 4, 3, 4, 3, 4, 4, 4, 3, 2, 2, 3, 3, 3, 4, 3,
       3, 3, 4], dtype=int64)
```

In [8]:
```
classifier.predict(x_test)
```

Out[8]:
```
array([4, 3, 3, 3, 3, 4, 4, 4, 3, 4, 4, 2, 3, 3, 3, 4, 3, 1, 4, 4, 3, 2,
       3, 4, 3, 4, 4, 3, 3, 3, 4, 4, 3, 5, 4, 2, 4, 3, 3, 3, 3, 2, 4, 3,
       3, 2, 3, 3, 2, 3, 4, 3, 5, 4, 3, 4, 3, 3, 3, 4, 3, 4, 4, 4, 4, 3,
       3, 4, 3, 4, 4, 3, 4, 5, 4, 3, 4, 4, 3, 4, 4, 4, 4, 4, 3, 4, 4, 3,
       4, 3, 3, 4, 4, 3, 2, 3, 4, 4, 4, 2, 4, 4, 4, 4, 4, 1, 4, 4, 3, 4,
       3, 3, 3, 3, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 3, 1, 4, 4, 5,
       4, 4, 4, 3, 3, 4, 4, 2, 3, 4, 4, 3, 3, 3, 4, 3, 4, 4, 4, 3, 4, 5,
       4, 4, 2, 3, 3, 3, 4, 4, 3, 3, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 4, 2,
       3, 4, 4, 2, 3, 4, 4, 3, 3, 4, 3, 3, 3, 3, 3, 4, 3, 3, 2, 3, 4, 4,
       3, 4, 4, 4, 4, 4, 3, 4, 3, 4, 2, 3, 4, 4, 2, 3, 3, 3, 3, 4, 3, 3,
       4, 3, 4, 4, 4, 3, 3, 3, 3, 4, 4, 2, 5, 3, 4, 4, 4, 4, 5, 3, 4, 4,
       3, 4, 4, 3, 4, 4, 5, 4, 3, 4, 3, 4, 4, 4, 3, 2, 2, 3, 2, 4, 4, 3,
       3, 3, 4], dtype=int64)
```

A3.2

In [9]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, svm.predict(x_test))
cm
```

Out[9]:
```
array([[ 0,  2,  0,  0,  0,  0],
       [ 1, 10,  9,  0,  0,  0],
       [ 0,  6, 81, 28,  0,  0],
       [ 0,  0, 30, 85,  4,  0],
       [ 0,  0,  1,  8,  1,  0],
       [ 0,  0,  0,  1,  0,  0]], dtype=int64)
```

In [10]:
```
cm2 = confusion_matrix(y_test, classifier.predict(x_test))
cm2
```

Out[10]:
```
array([[ 0,  2,  0,  0,  0,  0],
       [ 0,  7, 11,  2,  0,  0],
       [ 2,  7, 66, 39,  1,  0],
       [ 1,  3, 32, 77,  6,  0],
       [ 0,  0,  2,  7,  1,  0],
       [ 0,  0,  1,  0,  0,  0]], dtype=int64)
```

A3.3

In [11]:
```
from sklearn.metrics import accuracy_score
accuracy1 = accuracy_score(y_test, svm.predict(x_test))
accuracy2 = accuracy_score(y_test, classifier.predict(x_test))
accuracy1
```

Out[11]:
```
0.6629213483146067
```

In [12]:
```
accuracy2
```

Out[12]:
```
0.5655430711610487
```

accuracy1 is more accurate than accuracy2 therefore the SVM is more accurate

A4.1

In [13]:
```
features_submission_data = pd.read_csv('FIT1043-Essay-Features-Submission.csv')
score = svm.predict(features_submission_data)
score
```

Out[13]:
```
array([4, 3, 3, 4, 4, 4, 3, 3, 3, 2, 3, 4, 4, 3, 4, 3, 4, 4, 3, 3, 3, 3,
       4, 4, 4, 4, 4, 4, 4, 3, 2, 4, 3, 3, 4, 3, 4, 4, 3, 3, 3, 4, 3, 3,
       2, 3, 3, 4, 4, 3, 2, 4, 4, 4, 3, 4, 3, 4, 4, 4, 3, 3, 3, 4, 3, 4,
       4, 4, 3, 4, 4, 4, 5, 3, 3, 3, 3, 4, 4, 4, 4, 3, 4, 3, 4, 1, 4, 4,
       1, 3, 3, 3, 3, 3, 4, 3, 4, 4, 3, 3, 4, 5, 1, 4, 3, 4, 3, 3, 5, 4,
       4, 3, 4, 4, 4, 3, 2, 4, 2, 3, 4, 4, 4, 3, 3, 4, 3, 5, 3, 3, 4, 4,
       2, 3, 4, 3, 3, 4, 2, 4, 4, 4, 4, 3, 3, 4, 4, 3, 4, 4, 4, 3, 4, 3,
       5, 3, 3, 4, 4, 4, 4, 3, 2, 2, 4, 5, 3, 3, 2, 4, 3, 4, 3, 3, 4, 3,
       4, 3, 3, 4, 4, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 5, 4, 5, 3, 5, 4, 3,
       4], dtype=int64)
```

A4.2

In [14]:
```
essayid = features_submission_data['essayid'].values
```

In [15]:
```
d1 = {
    "essayid": essayid,
    "score" : score
     }
df = pd.DataFrame(d1)
new_csv = "new_csv"
df.to_csv(new_csv, index=False)
```

PART B

The datasets used in Part B were taken from [https://www.kaggle.com/datasets/ilayaraja07/data-cleaning-feature-imputation/?select=Students_Performance_mv.csv (https://www.kaggle.com/datasets/ilayaraja07/data-cleaning-feature-imputation/?select=Students_Performance_mv.csv)](https://www.kaggle.com/datasets/ilayaraja07/data-cleaning-feature-imputation/?select=Students_Performance_mv.csv).

In [16]:
```
partb_dataset = pd.read_csv("Students_Performance_mv.csv")
missing_data = partb_dataset.isnull().sum()
missing_data
```

Out[16]:
```
gender                          0
race/ethnicity                 11
parental level of education    21
lunch                          12
test preparation course         4
math score                      0
reading score                   0
writing score                   0
dtype: int64
```

In [17]:
```
partb_dataset.dropna(inplace=True)
missing_data = partb_dataset.isnull().sum()
missing_data
```

Out[17]:
```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
dtype: int64
```

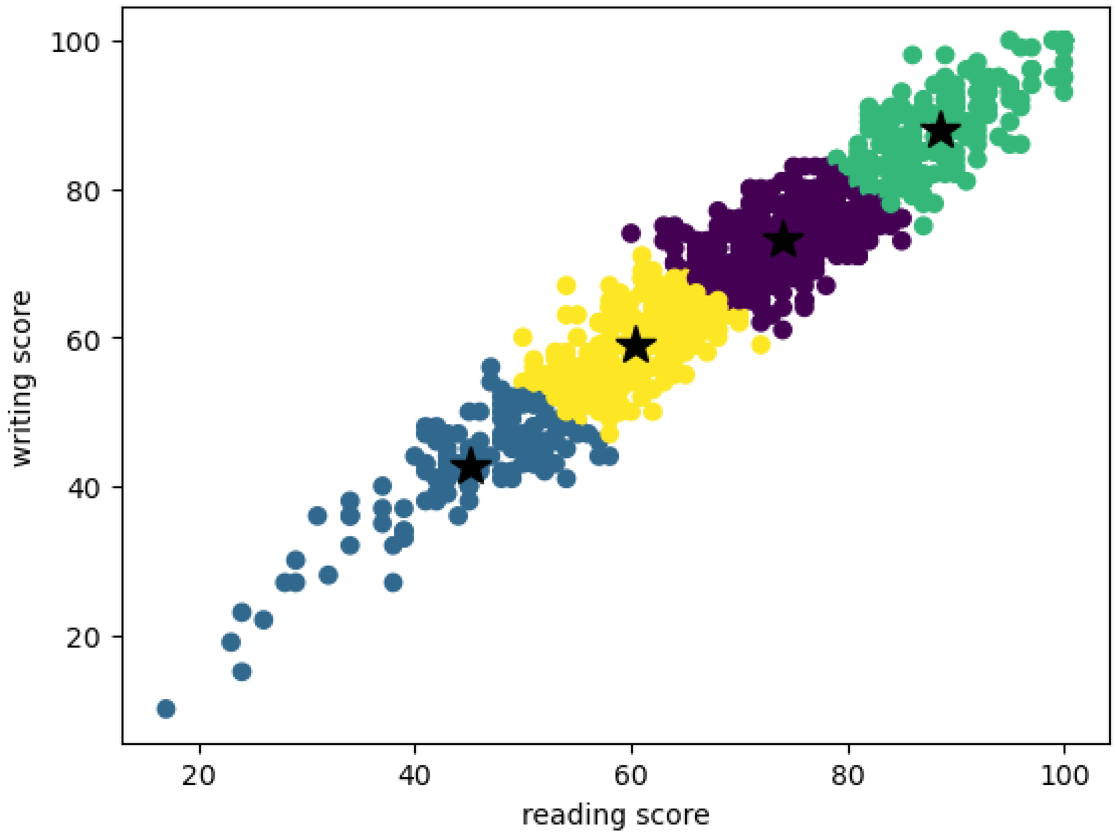
In [18]:

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=4).fit(partb_dataset[['reading score','writing score']])
plt.scatter(x=partb_dataset['reading score'],
            y=partb_dataset['writing score'],
            c=kmeans.labels_)
plt.plot(kmeans.cluster_centers_[:,0],
         kmeans.cluster_centers_[:,1],
         'k*',
         markersize=15)
plt.xlabel('reading score')
plt.ylabel('writing score')
plt.show
```

```
C:\Users\tanje\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value o
f `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\tanje\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to ha
ve a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by s
etting the environment variable OMP_NUM_THREADS=4.
  warnings.warn(
```

Out[18]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```




```
the above diagram is dataset used with its result of the k-means clustering. There were a total of 4 subgroups
of it while the blue colored part of the data range from 10 to 60 for the reading score and 5 to 58 for
writing score. For the yellow part of the data, the reading score range from 50 to 75, and 48 to 73 for
writing score. Following, the purple part of the data reading score range from 60 to 87 and 60 to 82 for the
writing score. Lastly the green part of the data reading score range from 80 to 100, while the writing score
range from 75 to 100. All of these data values are an estimation based on the graph.
```