FIT1043 INTRODUCTION TO DATA SCIENCE

ASSIGNMENT 2

George Tan Juan Sheng
30884128

Introduction

This assignment is mainly on applying machine learning skills like classification and clustering based on the dataset given which is data on cars. The dataset contains different kinds of cars of different models with different specifications. We will be applying clustering and classificaton in order to construct a model that is able to get value from the data. We will be using clustering in order to group data as if they are unlabelled in order to find similarities between them and in the end find value based on the group of clusters. We will then apply classification to construct a predictive model to predict the output based on the input, which is our test data set. Classification methods like Decision Tree and Random Forest algorithm will also be compared to see which would be a better model to to predict the output. In the end, we will also be using the classification algorithm with the better result to predict the outputs from the kaggle test data.

Importing the necessary libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
```

After reading the file, we can see that the file provides the data of different kinds of car of different models such as Audi, BMW Porsche etc. The file provides data of the cars such as the brand of the car, in other words the manufacturer, the model, vehicle class, fuel capacity, fuel efficiency and many more.

Since our main objective is to apply classification and clustering, despite having quite a few informations in the file regarding the cars, only a few fields will be used in classification and clustering while the remaining fields will not be used.

Reading the File

We will start the assignment by reading the file 'FIT1043-vehicle-classifier.csv' as a DataFrame into df1 and df2. df1 will be used for clustering and df2 will be used for classification. This is because each classification and clustering uses different columns of the file and hence it is better to have two dataframes that have the same file to be used for each process so that when the fields in df1 changes, the fields in df2 will remain untouched.

The size of file 'FIT1043-vehicle-classifier.csv' is 156 x 14. In other words this file consists of 156 rows and 14 columns. df1 and df2 will have the same dimensions as they contain the same file.

In [2]:
```
df1 = pd.read_csv('FIT1043-vehicle-classifier.csv')
df2 = pd.read_csv('FIT1043-vehicle-classifier.csv')
df1.shape
```

Out[2]:
```
(156, 14)
```

Data Wrangling

Since both dataframe are of the same file, we will be using df1 just for showcasing. However the data wrangling process will be applied to both dataframes, df1 and df2.

This is the basic statistics of the file. Starting off, we may notice that the dataset has null values which may affect our analysis on this dataset. This is noticeable by seeing that the count for some columns are not 156, in other words this means that some columns have lesser values compared to others. The columns that have null values are Curb_weight and Fuel_efficiency.

In [3]:
```
df1.describe()
```

Out[3]:

	Engine_size (litres)	Horsepower	Wheelbase	Width	Length	Height	Curb_weight
count	156.000000	156.000000	156.000000	156.000000	156.000000	156.000000	155.000000
mean	3.060897	185.948718	107.499359	71.133974	187.301923	59.187179	3.378026
std	1.044653	56.700321	7.638996	3.415326	13.402491	7.922412	0.630502
min	1.000000	55.000000	92.600000	62.600000	149.400000	47.000000	1.895000
25%	2.300000	149.500000	103.000000	68.500000	177.575000	54.775000	2.971000
50%	3.000000	177.500000	107.000000	70.550000	187.900000	56.200000	3.342000
75%	3.575000	215.000000	112.200000	73.175000	196.125000	64.525000	3.799500
max	8.000000	450.000000	138.700000	79.900000	224.500000	104.500000	5.572000

Next, we will be removing the rows of columns which has null values. From the data, we can see that column Curb_weight and Fuel_efficiency has null values in its rows as the 'count' value is not 156, meaning that they have less data when compared to other columns, hence we will be removing the rows that contains of these columns that contains null values.

The reason why we chose to remove the rows that have null values instead of putting our own values in is because it is hard to predict the specifications of the vehicle as it is a very dependant factor, depending on the model of the car, the type of car and many more. Hence, it would be much more reasonable to just remove the rows that contain null values rather than putting in values on our predictions, in order to reduce errors.

In [4]:
```
df1 = df1.dropna(subset = ['Curb_weight','Fuel_efficiency'])
df2 = df2.dropna(subset = ['Curb_weight','Fuel_efficiency'])
```

Now, we can see that we have successfully removed the rows with null values and all the columns have the same amount of rows of data.

In [5]:
```
df1.describe()
```

Out[5]:

	Engine_size (litres)	Horsepower	Wheelbase	Width	Length	Height	Curb_weight
count	153.000000	153.000000	153.000000	153.000000	153.000000	153.000000	153.000000
mean	3.050327	185.071895	107.422876	71.069281	187.049020	59.250327	3.376797
std	1.046430	56.729054	7.690309	3.415912	13.402156	7.986857	0.634540
min	1.000000	55.000000	92.600000	62.600000	149.400000	47.000000	1.895000
25%	2.300000	148.000000	103.000000	68.500000	177.500000	54.700000	2.967000
50%	3.000000	175.000000	107.000000	70.400000	186.700000	56.200000	3.340000
75%	3.500000	215.000000	112.200000	73.100000	194.800000	64.900000	3.821000
max	8.000000	450.000000	138.700000	79.900000	224.500000	104.500000	5.572000

Next up, we can also notice that there is an outlier for field Height for Chevrolet Corvette. We can see that the car with the highest height is actually the Chevrolet Corvette, which has a height of 104.5 and this is drastically higher than the height of other big cars like SUVs or trucks. Moreover, the Corvette is a Coupe and is a Sports car, we can also search up Google to find out that it is practically impossible for a sports car to be that tall of a height, hence this large margin error of the outlier should be removed and we can do this by removing this row of data.

In [6]:
```
df1[df1['Height'] == 104.5]
```

Out[6]:

	Manufacturer	Model	Vehicle_class	Vehicle_alt_class	US_vehicle_type	Engine_size (litres)	Hor
24	Chevrolet	Corvette	Coupe	Sports	Passenger	5.7	

And here, we can see that we have successfully removed the row of car Chevrolet Corvette and the row is not present anymore.

Now we have df1 and df2 ready for use for analysis as the datas are wrangled, df1 will be used for clustering and df2 will be used for classification.

In [7]:
```
df1 = df1.drop(df1[df1['Model'] == 'Corvette'].index)
df2 = df2.drop(df2[df2['Model'] == 'Corvette'].index)
df1[df1['Model'] == 'Corvette']
```

Out[7]:

Manufacturer	Model	Vehicle_class	Vehicle_alt_class	US_vehicle_type	Engine_size (litres)	Horsep

This is the dimensions of the file after wrangling. It will be the same for df1 and df2

In [8]:
```
df1.shape
```

Out[8]:
```
(152, 14)
```

In [9]:
```
df2.shape
```

Out[9]:
```
(152, 14)
```

CLUSTERING

For clustering, we will be using df1 and we will remove the columns “Vehicle_class”, “Vehicle_alt_class” and “US_vehicle_type” as these columns will only be used during the classification part of the assignment.

In [10]:
```
df1 = df1.drop(columns = ['Vehicle_class','Vehicle_alt_class','US_vehicle_typ
```

Un-supervised Machine Learning

Moving on we will talk about un-supervised machine learning and implement clustering for our data. Un-supervised machine learning is actually referring to the input data being unlabelled and the algorithms learn to inherent structure from the input data. Clustering is one of the methods of un-supervised machine learning to discover the inherent groupings in the data. In other words,clustering groups the input that are similiar to each other into a group. For this assignment, we will be using the algorithm k-means to solve clustering problems.

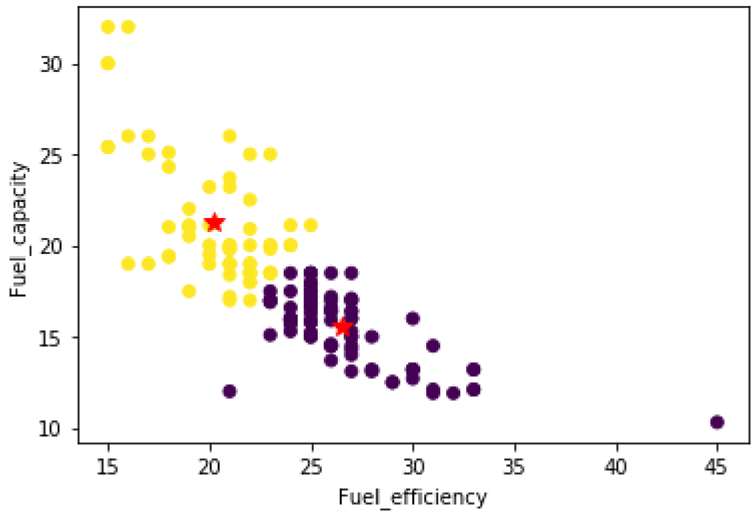
Clustering using 'Fuel_efficiency' and 'Fuel_capacity'

Before we start clustering, we would have to choose our inputs and decide on what dimension will our cluster be in. The inputs we have chosen today is Fuel_efficiency and Fuel_capacity and hence we will be starting off the clustering process with a 2-dimension visualization. We may use other inputs to cluster again if the results are not really ideal.

The reason why we would choose Fuel_efficiency and Fuel_capacity is because these variables are very reasonable for grouping cars. As we know, clustering's main purpose is to group data points that are similar into one group. Hence, cars that have similar Fuel_efficiency and Fuel_capacity have high chances of being grouped together correctly and meaningful data might be obtained from the clusters to know more about the cars that are grouped together in thatr cluster. And so, I would think that using Fuel_efficiency and Fuel_capacity may be a good fit for clustering purposes in order to group cars into groups of similar characteristics.

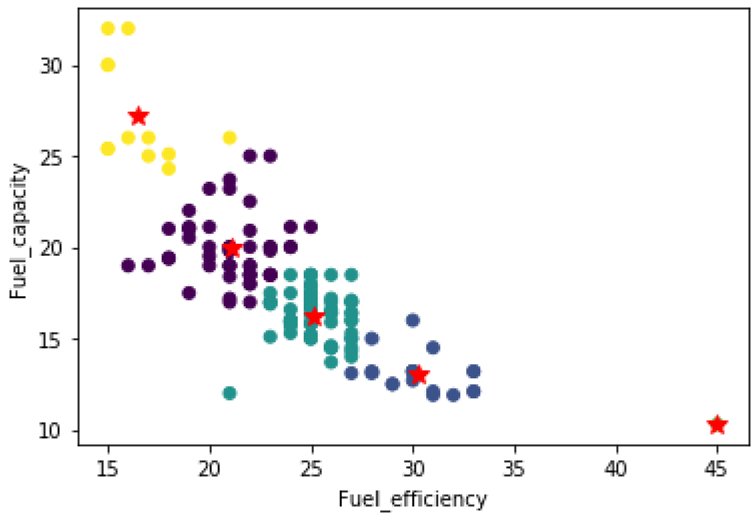
In [11]:
```
kmeans = KMeans(n_clusters = 2).fit(df1[['Fuel_efficiency','Fuel_capacity']])
plt.scatter(df1['Fuel_efficiency'],df1['Fuel_capacity'], c = kmeans.labels_)

plt.plot(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1], 'r*', mar
plt.xlabel('Fuel_efficiency')
plt.ylabel('Fuel_capacity')
```

Out[11]:
```
Text(0, 0.5, 'Fuel_capacity')
```



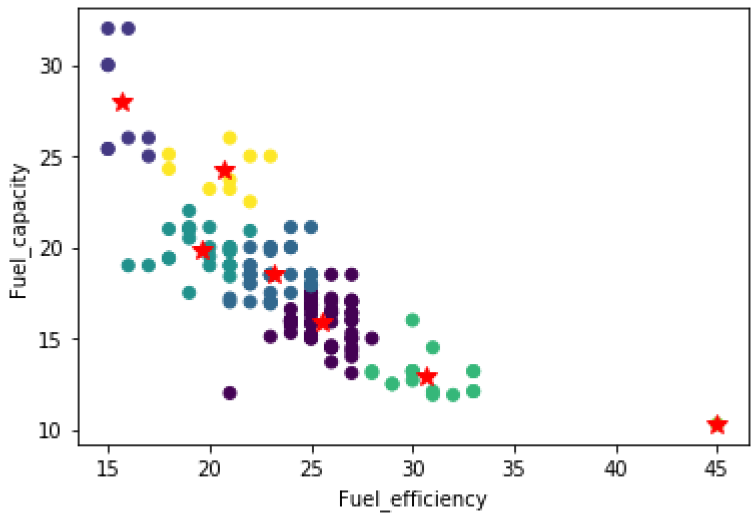
In [12]:
```
kmeans = KMeans(n_clusters = 5).fit(df1[['Fuel_efficiency','Fuel_capacity']])
plt.scatter(df1['Fuel_efficiency'],df1['Fuel_capacity'], c = kmeans.labels_)

plt.plot(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1], 'r*', mar
plt.xlabel('Fuel_efficiency')
plt.ylabel('Fuel_capacity')
```

Out[12]:
```
Text(0, 0.5, 'Fuel_capacity')
```



In [13]:
```
kmeans = KMeans(n_clusters = 7).fit(df1[['Fuel_efficiency','Fuel_capacity']])
plt.scatter(df1['Fuel_efficiency'],df1['Fuel_capacity'], c = kmeans.labels_)

plt.plot(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1], 'r*', mar
plt.xlabel('Fuel_efficiency')
plt.ylabel('Fuel_capacity')
```

Out[13]:
```
Text(0, 0.5, 'Fuel_capacity')
```



From all the results of the clustering above, we can see a clear pattern. We can see that the cars that are grouped together in one group are strongly related. For example, we can see that cars with higher fuel capacity and lower fuel efficiency are grouped together with other cars that have relatively high fuel capacity and low fuel efficiency, indicating that these cars that are grouped together may be of the same kind of car. One of the predictions may be saying that cars that have high fuel capacity and low fuel efficiency are cars that are big and heavy like trucks or SUVS.

And so, this is actually very true as if we were to look back the data we have gotten, in the first cluster which is at the top left part of the clustering, we can see that cars that have around less than fuel efficiency of 17 and fuel capacity of more than 27 are actually trucks or SUVs or MPVs. The clustering makes a lot of sense as the cars that are grouped together are actually very related and very similar as they are all larger cars which tend to have worse fuel mileage and larger fuel capacity. This can be meaningful as a salesman can actually recommend car choices to a customer according to the customer's preference towards the car's fuel usage and capacity

However, the clusters of the graph are not very distinct and hence it may be a little unclear to determine the different clusters. Hence, we will re-do this cluster in 3-Dimensional and try other inputs that may have more clearer clusters and compare if they have equally decent output.

In [14]:
```
df2.loc[(df2['Fuel_efficiency'] <= 17) & (df2['Fuel_capacity'] >= 27)]
```

Out[14]:

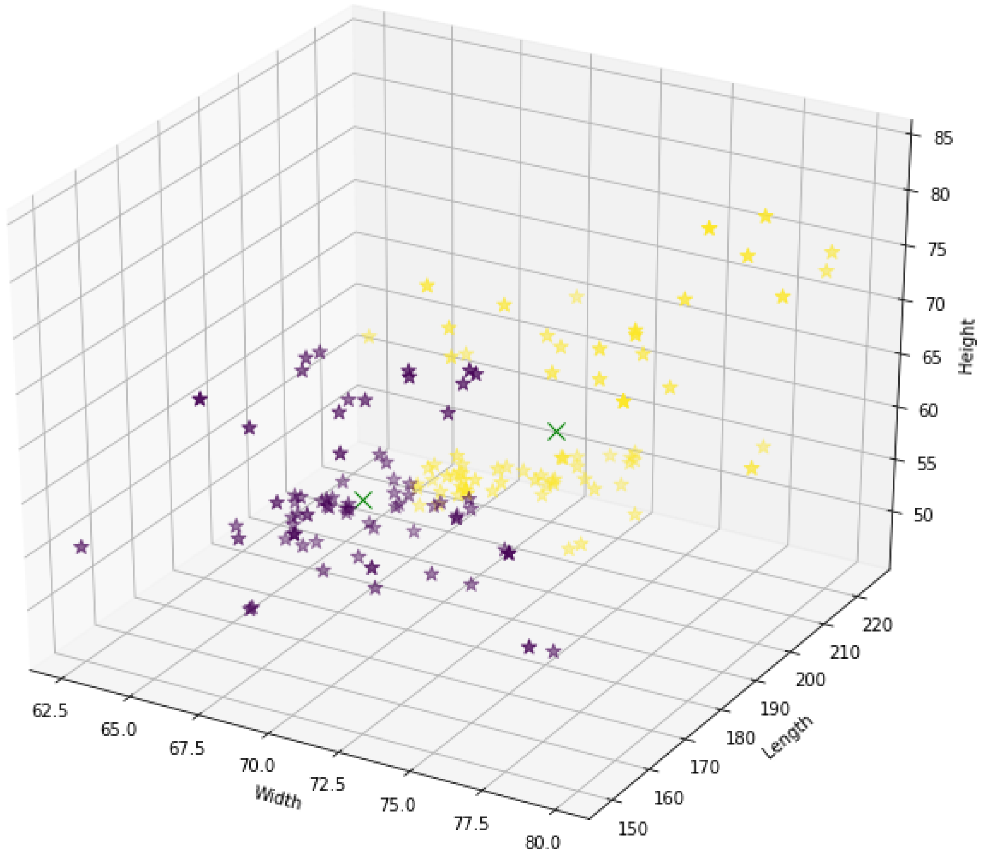
	Manufacturer	Model	Vehicle_class	Vehicle_alt_class	US_vehicle_type	Engine_size (litres)	Hc
18	Cadillac	Escalade	SUV	NaN	Car	5.7	
40	Dodge	Ram Wagon	Truck	NaN	Car	3.9	
41	Dodge	Ram Van	Truck	NaN	Car	3.9	
77	Lincoln	Navigator	MPV	SUV	Car	5.4	

df2 was used here to ilustrate the type of cars that are grouped together in the first cluster as df2 contains columns like Vehicle_class where df1 does not as the columns were removed earlier. We can see cars in the first cluster are cars of SUVS, trucks or MPVs.

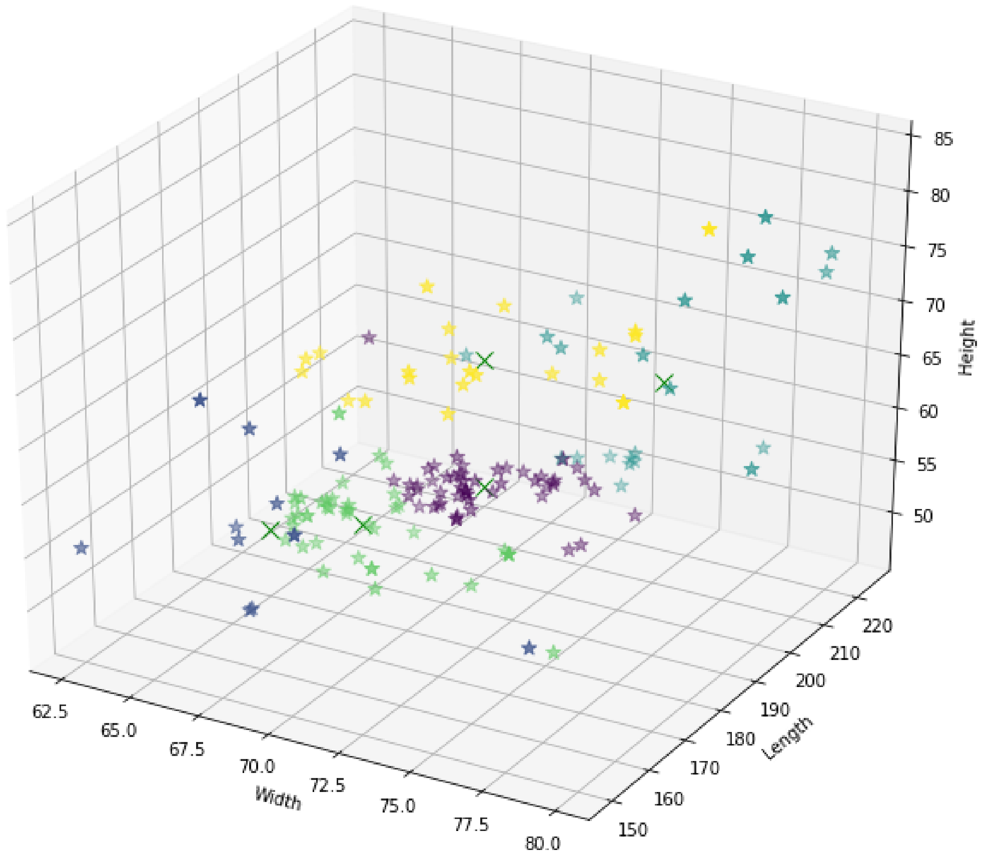
3-Dimensional Clustering using 'improved inputs' of 'Width', 'Length' and 'Height'

Although the previous clustering showed a pretty decent result with using inputs of Fuel_efficiency and Fuel_capacity, we would also re-do the clustering in 3_Dimensions and by using inputs of some physical characteristics of the car to see if the outcome would be any better as we already have a pretty decent clustering result. We will be using Height, Width and Length for this clustering.

In [15]:
```
kmeans = KMeans(n_clusters = 2).fit(df1[['Width','Length', 'Height']])


fig = plt.figure(figsize=(12,10))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(df1['Width'], df1['Length'], df1['Height'], c= kmeans.labels_, mar
plt.plot(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], kmeans.c
)


ax.set_xlabel('Width')
ax.set_ylabel('Length')
ax.set_zlabel('Height')

plt.show()
```

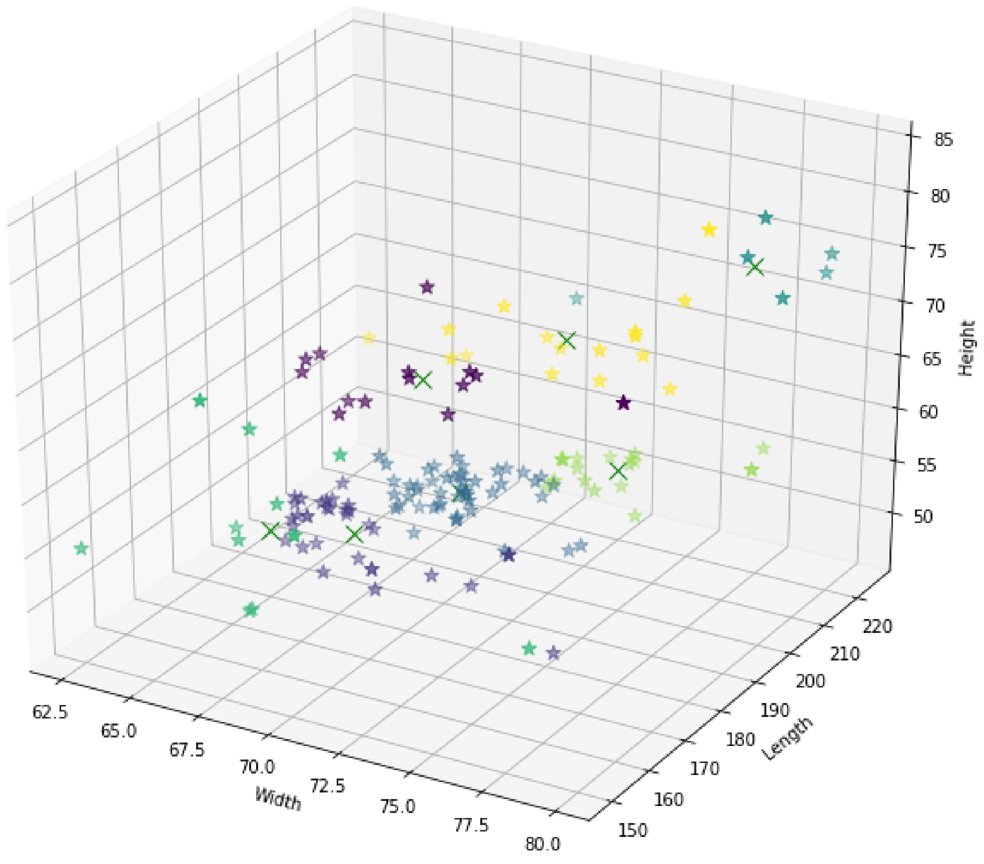


In [16]:
```
kmeans = KMeans(n_clusters = 5).fit(df1[['Width','Length', 'Height']])


fig = plt.figure(figsize=(12,10))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(df1['Width'], df1['Length'], df1['Height'], c= kmeans.labels_, mar
plt.plot(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], kmeans.c
)


ax.set_xlabel('Width')
ax.set_ylabel('Length')
ax.set_zlabel('Height')

plt.show()
```



In [17]:

```
kmeans = KMeans(n_clusters = 7).fit(df1[['Width','Length', 'Height']])


fig = plt.figure(figsize=(12,10))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(df1['Width'], df1['Length'], df1['Height'], c= kmeans.labels_, mar
plt.plot(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], kmeans.c
)


ax.set_xlabel('Width')
ax.set_ylabel('Length')
ax.set_zlabel('Height')

plt.show()
```



The clustering result in 3-Dimensions and by using 'Width', 'Length' and 'Height' have more distinct clusters, compared to the clustering result using 'Fuel_efficiency' and 'Fuel_capacity'. However, for this 3-Dimensional clustering graph, we can see the clustering is better as it is more leveled and obvious. However, although this clustering result has clearer clusters, this clustering does not show as much meaning as the clustering we had before, in other words it is harder to find meaningful similarities for the inputs of this graph.

In the clustering graph, we can see basically in each cluster, cars that have similar width, length and height will be put in the same cluster, but in the modern world nowdays many cars can have the same physical characteristics but are not the same type of cars. Cars in the same cluster are less likely to be similar or of the same type, as cars with similar width, length and height may most likely to be different cars as well.Hence, even though we have gotten to know that specific cars in a specific cluster have similar width, length and height, but what's next? We would have to apply some sort of method again in order to separate the cars in that cluster into groups of cars that are maybe of the same type or class. In another words, no meaningful value can be found between the cars that have similar width, length and height.

However, the previous clustering result using 'Fuel_efficiency' and 'Fuel_capacity' shows more meaning as we are able to say that cars that are in specific clusters may be cars of the same type as they have clear similarities, just like the cars may be trucks or SUVs that are big cars as all the cars in the cluster have high fuel capacity but lower fuel efficiency.

And so, we can conclude that the 3-dimensional clustering result using 'Width', 'Length' and 'Height' is actually not any better than the clustering result from 'Fuel_efficiency' and 'Fuel_capacity' as the clustering result from 'Fuel_efficiency' and 'Fuel_capacity' shows more meaning. After all, the main objective of clustering is to group data into clusters of similar characteristics and find value out of the cluster. This was accomplished by the clustering using 'Fuel_efficiency' and 'Fuel_capacity' as cars of the same clusters are cars that are actually similar in a way that is meaningful. But clustering using 'Width','Length' and 'Height' was not able to do so as cars in the same cluster may have similar width, length and height, but they have high chances of being different types of cars as well.

CLASSIFICATION

Binary and Multi-Class Classification

In classification, there is binary classification and multi-class classification. Binary classification is referring to classify the data into two groups, whether the data fullfils the qualitative property or not. For binary classification, it is basically to group the elements of a set into a yes or no group according to the property given.

Multi-class classification is referring to classify the data into more than two classes. This is different compared to binary classification as binary classification classifies the data into two groups of whether the data fullfils the property given or not, but multi-class classification can classify the data into one of three or even more classes.

Supervised Machine Learning

In supervised machine learning, we will be able to train the machine by giving it data. All training data is labelled and the algorithms learn to predict the output from the input data. The goal of supervised machine learning is that we train the machine so well by giving it alot of traning datasets, that the mapping function will be approximated so well and when we give a new input data to the machine, a correctly predicted output can be obtained.

Normally in supervised machine learning, datasets will be separated into training and testing datasets by using a ratio of 8:2 or even 9:1, in the sense that the training dataset should always be larger than the test dataset by a whole lot. Training datasets are labelled and used to train the machine to develop a predictive model based on the training dataset. After the model is formed, we use the testing datasets to test the model and compare the predicted value with the actual value in order to determine if the model is accurate.

Binary classification using the decision tree algorithm, where the labelled data is the “US_vehicle_type”

For this part of the classfication, we will be manipulating the values of the column US_vehicle_type. This column contains values of Car and Passenger, in other words each car is either a vehicle type of Car or Passenger. Hence, we will be changing the values 'Car' into '0' and 'Passenger' into '1' of each row in order to allow us to visualize the result later on.

In [18]:
```
df2.loc[(df2['US_vehicle_type'] == 'Car'),'US_vehicle_type']='0'
df2.loc[(df2['US_vehicle_type'] == 'Passenger'),'US_vehicle_type']='1'
```

We will also be only using two fields for our x data in order to easier visualize the result. We will be using field 'Width' and 'Height'.

In [19]:
```
X = df2.iloc[:,[8,10]].values                          #Index of Width and Hei
y = df2.iloc[:,4].values                               #US_vehicle_type is ind

X_train, X_test, y_train, y_test = train_test_split(
     X, y, test_size = 0.25, random_state = 0)

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

classifier = DecisionTreeClassifier(
    criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)

y_pred3c = classifier.predict(X_test)


cm3c = confusion_matrix(y_test, y_pred3c)               #Confusion Matrix for

accuracy3c = accuracy_score(y_test, y_pred3c)           #Accuracy on predictiv

cm3c
```

Out[19]:
```
array([[10,  0],
       [ 0, 28]], dtype=int64)
```

Multi-class classification using the decision tree algorithm, where the

labelled data is the “Vehicle_class”.

For multi-class classifications, we will be using 5 fields which are 'Wheelbase', 'Width', 'Length', 'Height' and 'Curb_weight'. We will be using these 5 fields out of the 9 fields and the reason for not using all 9 fields is to prevent constructing an overfitting decision tree, hence we will only be using these 5 fields which are based on the physical characteristics of the car.

In [20]:
```
X = df2.iloc[:,[7,8,9,10,11]].values     #Index of 'Wheelbase', 'Width', 'Len
y = df2.iloc[:,2].values                 #Vehicle_class is index 2 in the fil

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size = 0.25, random_state = 0)

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)


classifier = DecisionTreeClassifier(
    criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)

y_pred3d = classifier.predict(X_test)


cm3d = confusion_matrix(y_test, y_pred3d)                    #Confusion Matr

accuracy3d = accuracy_score(y_test, y_pred3d)                #Accuracy on pr
cm3d
```

Out[20]:
```
array([[ 1,  0,  0,  5,  0],
       [ 0,  3,  0,  0,  0],
       [ 0,  1,  3,  0,  0],
       [ 1,  0,  0, 21,  0],
       [ 0,  0,  2,  0,  1]], dtype=int64)
```

Multi-class classification with the “Vehicle_alt_class”

As there are many empty values for field Vehicle_alt_class, we will be filling up the empty values with the word 'Unknown'. This is so that the cars with no labels will be labelled with 'Unknown' as we do not know which additional class are the cars in. Hence, we will label those cars with no labelled additional class as Unknown. If the model predicts an output for the car which have characteristics that are undefined, the car will be labelled with 'Unknown'.

In [21]:
```
df2['Vehicle_alt_class'].fillna('Unknown', inplace = True)
df2.describe()
```

Out[21]:

	Engine_size (litres)	Horsepower	Wheelbase	Width	Length	Height	Curb_weight
count	152.000000	152.000000	152.000000	152.000000	152.000000	152.000000	152.000000
mean	3.032895	184.019737	107.442105	71.052632	187.097368	58.952632	3.377895
std	1.027359	55.398552	7.712040	3.420971	13.433067	7.110788	0.636492
min	1.000000	55.000000	92.600000	62.600000	149.400000	47.000000	1.895000
25%	2.300000	147.500000	103.000000	68.475000	177.475000	54.700000	2.964750
50%	3.000000	175.000000	107.000000	70.400000	187.250000	56.200000	3.341000
75%	3.500000	211.250000	112.200000	73.025000	195.125000	64.525000	3.821500
max	8.000000	450.000000	138.700000	79.900000	224.500000	83.600000	5.572000

In [22]:
```
X = df2.iloc[:,[7,8,9,10,11]].values #Index of 'Wheelbase', 'Width', 'Length'
y = df2.iloc[:,3].values            #Vehicle_alt_class is index 3 in the file


X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size = 0.25, random_state = 0)


sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)


classifier = DecisionTreeClassifier(
    criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)

y_pred3e = classifier.predict(X_test)


cm3e = confusion_matrix(y_test, y_pred3e)                       #Confusion

accuracy3e = accuracy_score(y_test, y_pred3e)                   #Accuracy c

cm3e
```

Out[22]:
```
array([[ 0,  0,  0],
       [ 0,  1,  1],
       [ 1,  4, 31]], dtype=int64)
```

Output predicted results from Question 3c, 3d and 3f in a two column CSV

In [23]:
```
predicted_results3c = pd.DataFrame({
    'ID' : list(i for i in range(1, len(y_pred3c)+1)),
    'Predicted Output' : y_pred3c
    })

predicted_results3d = pd.DataFrame({
    'ID' : list(i for i in range(1, len(y_pred3d)+1)),
    'Predicted Output' : y_pred3d
    })

predicted_results3e = pd.DataFrame({
    'ID' : list(i for i in range(1, len(y_pred3e)+1)),
    'Predicted Output' : y_pred3e
    })

predicted_results3c.to_csv(r'C:\Users\georg\Desktop\30884128_GeorgeTanJuanShe
predicted_results3d.to_csv(r'C:\Users\georg\Desktop\30884128_GeorgeTanJuanShe
predicted_results3e.to_csv(r'C:\Users\georg\Desktop\30884128_GeorgeTanJuanShe
```

Evaluation metrics and Output Evaluation

For our classification results, we have decided to use confusion matrix in order to evaluate our model. The reason why we would choose confusion matrix to evalute model is because this evaluation metrics allow us to calculate exactly how good our model is and our model's capability of predicting correct outputs.

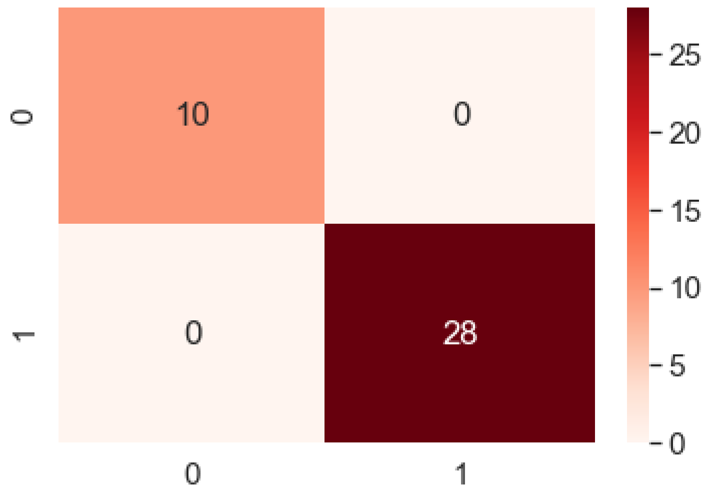
We are able to evaluate our predictive models based on its accuracy of predicitng the correct outputs. This accuracy can be calculated by using the confusion matrix. In a confusion matrix, there are values of true positive, true negative, false positive and false negative. Accuracy can be calculated by taking the sum of the values true positive and true negative, and then divided by the sum of values of true positive, true negative, false positive and false negative. Hence, predictive models with a higher accuracy basically indicates that this predictive model can predict outputs better than other predictive models.

And so, we evaluate our output of our classification predictive model based on the accuracy rate of the model.

Question 3(c) Binary Classification with 'US_vehicle_type' Evaluation

For Question 3(c), we can visualize the confusion matrix of binary classfication with US_vehicle_type. The visualization of the confusion matrix is as such below and the accuracy of this confusion matrix is 100%. Although it is practically impossible to have a predictive model that can predict outputs correctly all the time, our datasets for this assignment is rather small and the characteristics are rather obvious, hence we are able to get a 100% accuracy predictive model.

In [24]:
```
sns.set(font_scale=1.4)
sns.heatmap(cm3c, annot = True, cmap = 'Reds')
plt.show()
```



In [25]:
```
print(accuracy3c)
```

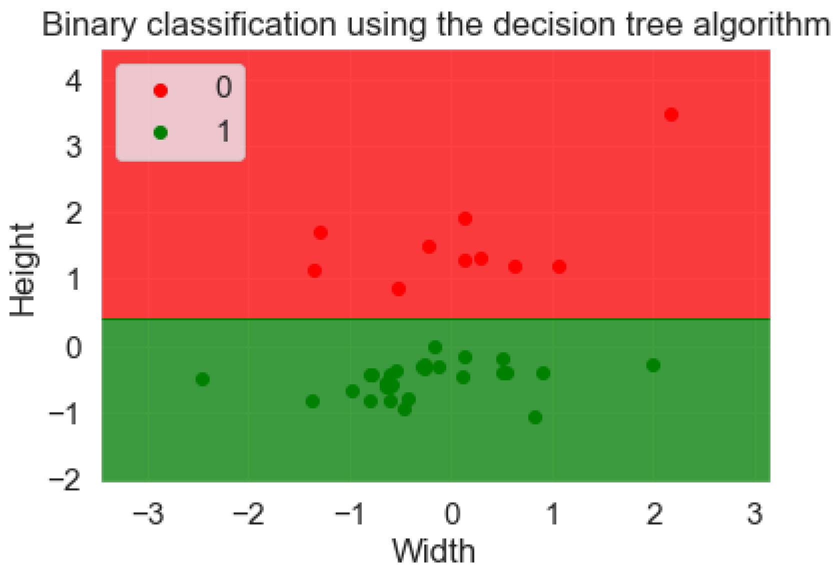
```
1.0
```

From the Listed Color Map as below, we are also able to visualize the results of our binary classification. We are able to see that the cars are separated into its category as whether the vehicle type is a Car or Passenger. 0 stands for Car while 1 stands for Passenger. The classification is rather clear the graph is split into red and green sections, showing that the cars in the green section have a vehicle type of Passenger while the cars in the red section have a vehicle type of Car.

In [26]:
```
X = df2.iloc[:,[8,10]].values
y = df2.iloc[:,4].values

X_train, X_test, y_train, y_test = train_test_split(
     X, y, test_size = 0.25, random_state = 0)

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

classifier = DecisionTreeClassifier(
    criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)



# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(
    np.arange(
        start = X_set[:, 0].min() - 1,
        stop = X_set[:, 0].max() + 1,
        step = 0.01
    ),
    np.arange(
        start = X_set[:, 1].min() - 1,
        stop = X_set[:, 1].max() + 1,
        step = 0.01
    )
)
plt.contourf(
    X1,
    X2,
    classifier.predict(
        np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
    alpha = 0.75,
    cmap = ListedColormap(('red', 'green'))
)
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(
        X_set[y_set == j, 0],
        X_set[y_set == j, 1],
        c = ListedColormap(('red', 'green'))(i),
        label = j
)
plt.title('Binary classification using the decision tree algorithm')
plt.xlabel('Width')
plt.ylabel('Height')
plt.legend()
plt.show()
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, which sho
uld be avoided as value-mapping will have precedence in case its length
```

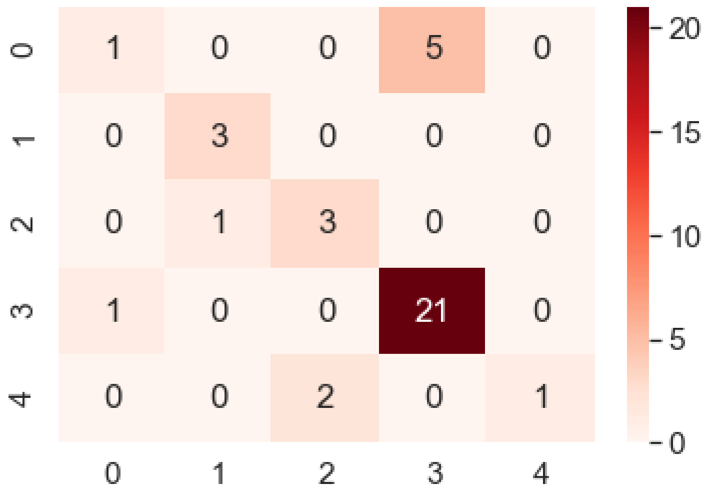
```
matches with 'x' & 'y'.  Please use a 2-D array with a single row if you
really want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which sho
uld be avoided as value-mapping will have precedence in case its length
matches with 'x' & 'y'.  Please use a 2-D array with a single row if you
really want to specify the same RGB or RGBA value for all points.
```



Question 3(d) Multi-Class Classification with 'Vehicle_class' Evaluation

For Question 3(d), we can visualize the confusion matrix of multi-class classfication with Vehicle_class. The confusion matrix is as such below and the accuracy of this confusion matrix is 76.32%. This is a relatively high accuracy for a predictive model but we do have a small dataset, hence we need to train and test this predictive model with larger datasets in order to correctly determine whether this predictive model is really that accurate or not. We will also discuss this further later as we will be comparing between the Decision Tree and Random Forest algorithm to see which algorithm gives a higher accuracy and better output.

In [27]:
```
sns.set(font_scale=1.4)
sns.heatmap(cm3d, annot = True, cmap = 'Reds')
plt.show()
```



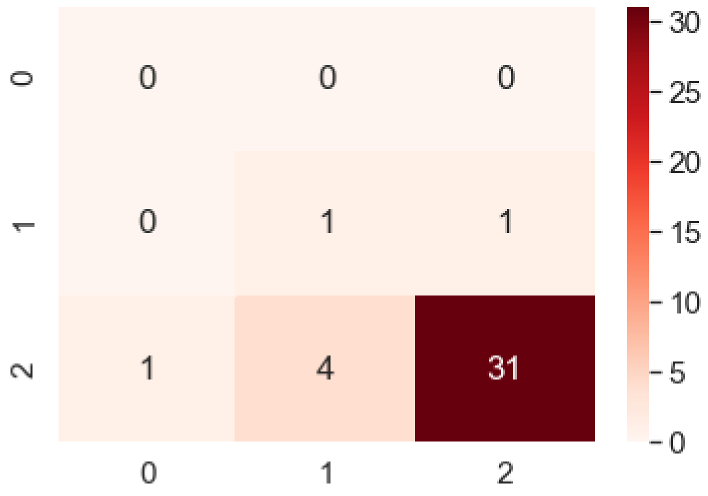
In [28]:
```
print(accuracy3d)
```

```
0.7631578947368421
```

Question 3(e) Multi-Class Classification with 'Vehicle_alt_class' Evaluation

For Question 3(e), we can visualize the confusion matrix of binary classfication with US_vehicle_type. The confusion matrix is as such below and the accuracy of this confusion matrix is 84.21%. This model is also relatively decent as it has a high accuracy. However, we would need to test this model again with larger datasets in order to better evaluate this model as the datasets we have now is very small, which would easily lead us to high accuracy. Only by training and testing this model with larger datasets, we can correctly determine if this model is really decent and accurate.

In [29]:
```
sns.set(font_scale=1.4)
sns.heatmap(cm3e, annot = True, cmap = 'Reds')
plt.show()
```



In [30]:
```
print(accuracy3e)
```

```
0.8421052631578947
```

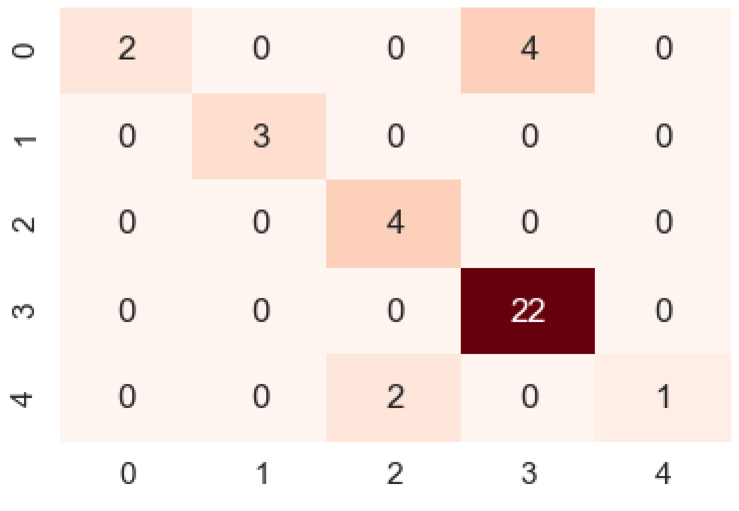
CONCLUSION

Comparison between Decision Tree and Random Forest Algorithm(Re-do from Part 3 (d))

Classification using Random Forest Algorithm

In [31]:

```
#Classification using Random Forest Algorithm
X = df2.iloc[:,[7,8,9,10,11]].values
y = df2.iloc[:,2].values

X_train, X_test, y_train, y_test = train_test_split(
     X, y, test_size = 0.25, random_state = 0)


sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)



classifier = RandomForestClassifier(
    n_estimators = 20,
    criterion = 'entropy',
    random_state = 0
)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix

cm = confusion_matrix(y_test, y_pred)
sns.set(font_scale=1.4)
sns.heatmap(cm, annot = True, cmap = 'Reds', cbar = False)
plt.show()
```



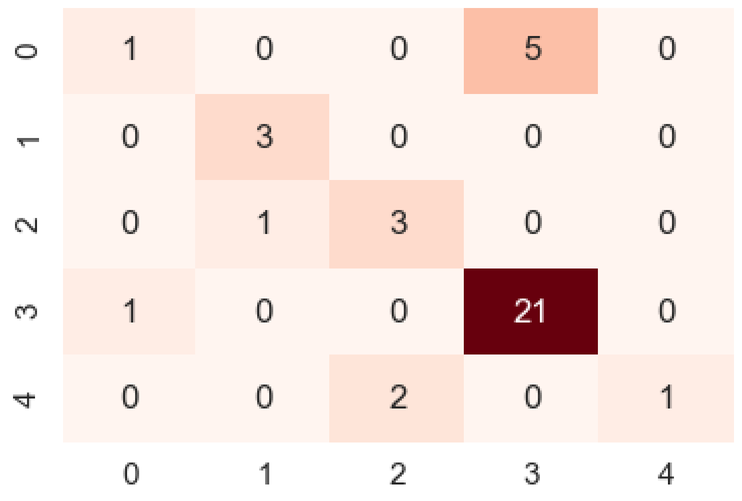
In [32]:
```
print(accuracy_score(y_test, y_pred))
```

```
0.8421052631578947
```

From the confusion matrix above, we are able to calculate the accuracy of the algorithm. The accuracy for this Random Forest algorithm is 84.21%

Classification using Decision Tree Algorithm

In [33]:
```
#Classification using Decision Tree Algorithm
X = df2.iloc[:,[7,8,9,10,11]].values
y = df2.iloc[:,2].values


X_train, X_test, y_train, y_test = train_test_split(
     X, y, test_size = 0.25, random_state = 0)


sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)


classifier = DecisionTreeClassifier(
    criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)

y_pred3d = classifier.predict(X_test)



cm = confusion_matrix(y_test, y_pred3d)
sns.set(font_scale=1.4)
sns.heatmap(cm, annot = True, cmap = 'Reds', cbar = False)
plt.show()
```



In [34]:
```
print(accuracy_score(y_test, y_pred3d))
```

```
0.7631578947368421
```

From the confusion matrix above that we have gotten from Part 3 (d), we are able to calculate the accuracy of the algorithm. The accuracy for the Decision Tree algorithm is 76.32%

Hence, we are able to conclude that the Random Forest algorithm has a better result than the Decision Tree algorithm as the Random Forest algorithm has a higher accuracy compared to Decision Tree algorithm

Output for Kaggle Submission

In [35]:
```
kaggle_file = pd.read_csv('FIT1043-kaggle-test-data.csv')
test_data = kaggle_file.iloc[:,[3,4,5,6,7]].values          #Index for colu

#Classification using Random Forest Algorithm
X = df2.iloc[:,[7,8,9,10,11]].values
y = df2.iloc[:,2].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
     X, y, test_size = 0.25, random_state = 0)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)


from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(
    n_estimators = 20,
    criterion = 'entropy',
    random_state = 0
)
classifier.fit(X_train, y_train)

test_data = sc.transform(test_data)
testresult_pred = classifier.predict(test_data)

predicted_kaggle = pd.DataFrame({
    'Id' : list(i for i in range(1, len(testresult_pred)+1)),
    'Predicted' : testresult_pred
})

predicted_kaggle.to_csv(r'C:\Users\georg\Desktop\30884128_GeorgeTanJuanSheng_
```

ASSIGNMENT CONCLUSION

Conclusion, we have successfully implemented the clustering and classification method onto the dataset that was given and have investigated our outputs. For clustering, it is important to find meaning out of the group of the clusters from the cluster results. In other words, if we can't find meaningful similarities between the data in the same cluster then the purpose of clustering would be defeated as the main purpose of clustering is to find similarities of the data of the same cluster. We must also choose the inputs wisely in order to construct a meaning clustering model. For this assignment, we have also successfully implemented a 3-dimemsional model for clustering, which gave clearer and more leveled clusters.

For classification, binary classification had gave 100% accuracy as our dataset was small. However if the dataset was bigger, the accuracy of the model would not be so high and it will definetely not be 100%. For multi-class classification, we had also concluded that the predictive model created using Random Forest algorithm had a higher accuracy compared to the model created using Decision Tree algorithm. This might or might not be true for all times as it really depends on our dataset. And so, it is important to implement both algorithms in order to find out which is able to create a better model so that we can get even more accurate results.

We have also succesfully predicted the outputs of the test data from kaggle and the accuracy we have gotten is 94.117%. All in all I would say it was a great experience from doing this assignment and I would also like to thank Dr Ian for all the guidance. That will be all and thank you for your time!

```
In [ ]:
```