In [35]:

```
#Importing all required Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pandas import Series

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import auc, accuracy_score, confusion_matrix, mean_squared_error
from sklearn.model_selection import cross_val_score, GridSearchCV, KFold, RandomizedSearchCV, train_test_split

from sklearn.tree import DecisionTreeClassifier
```

In [33]:

```
univdata = pd.read_csv("C:/Users/Admin/Desktop/Myprojectdata.csv", delimiter = ",",encoding='unicode_escape' )
univdata

trimColNames = [name.strip() for name in univdata.columns]
univdata.columns = trimColNames
```

In [ ]:

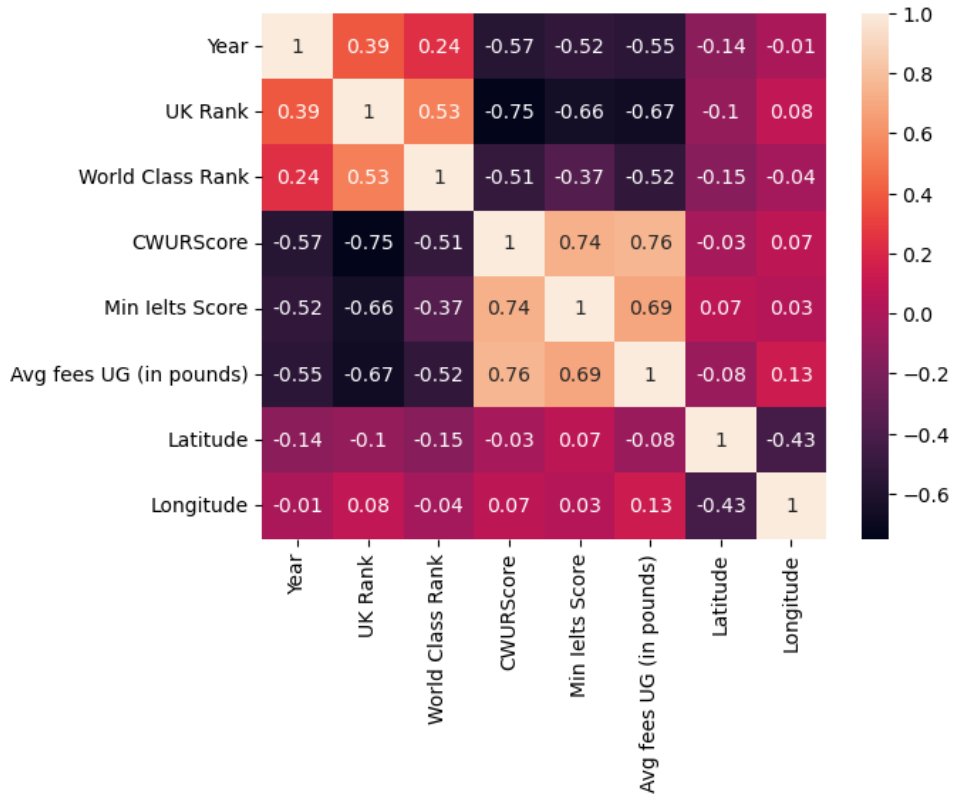
```
univdata['UnivName'] = univdata['UnivName'].astype(str)
print(univdata.dtypes)
univdata.shape
univdata.describe()

univdata = pd.get_dummies(univdata)
labels = np.array(univdata['UK Rank'])
univdata_list = list(univdata.columns)

univdata = np.array(univdata)
```

In [174]:

```
#Finding the Correlation Matrix between the predicted and the output Variable
correlation_matrix = univdata.iloc[:,:].corr().round(2)
sns.heatmap(data=correlation_matrix, annot=True)
# Year, UK Rank and World class rank have a strong relation and hence can help in finding best university
```

Out[174]:

<AxesSubplot:>

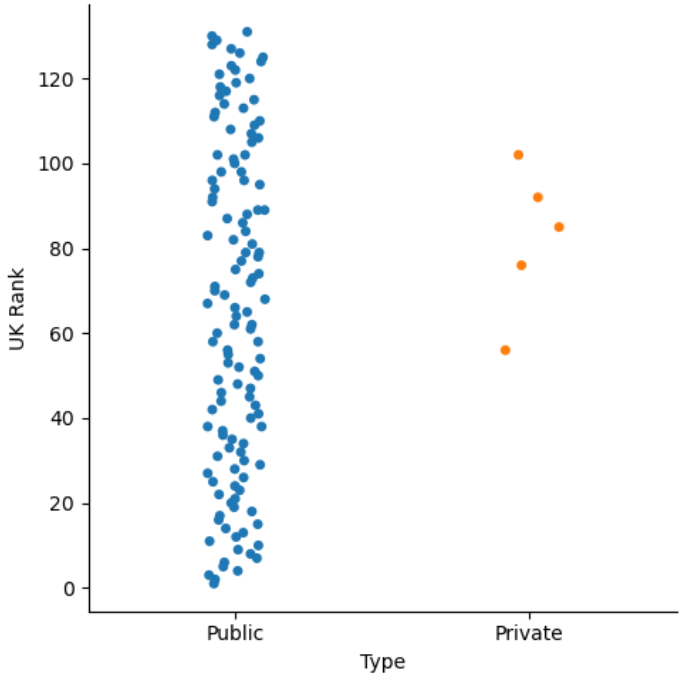


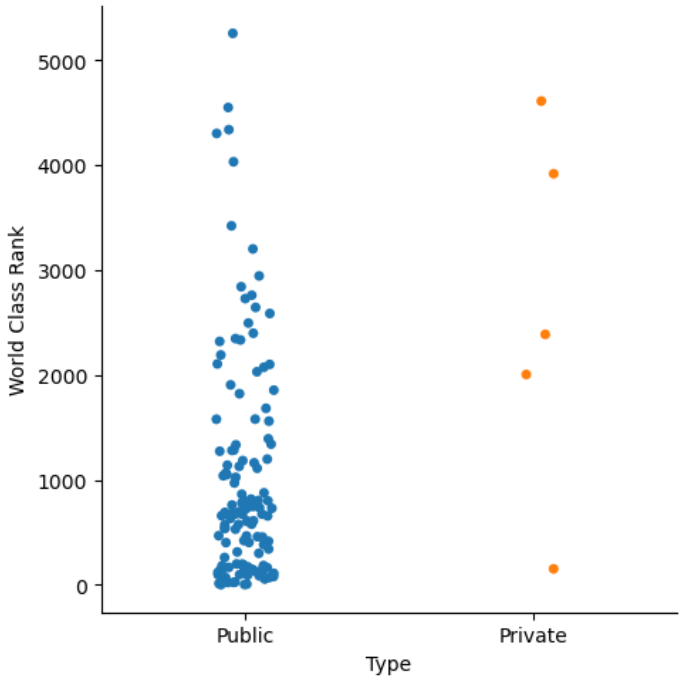
In [175]:

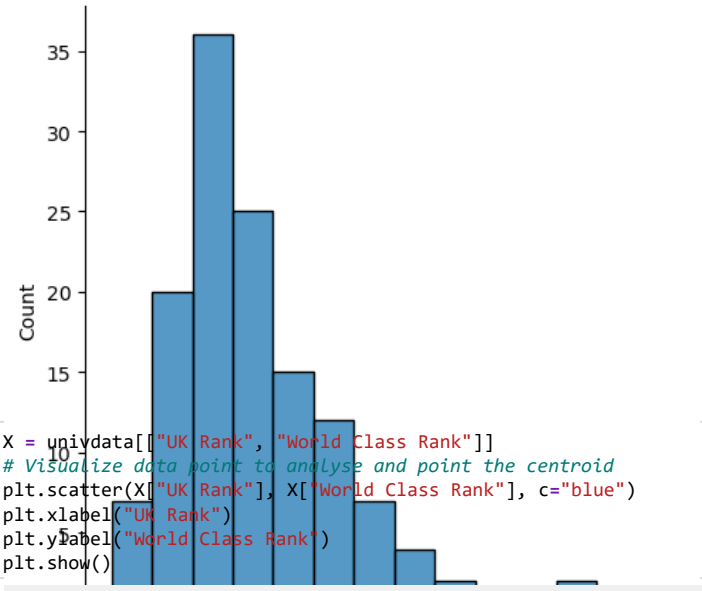
```
# Determining the Public and private universities as per UK Ranking
sns.catplot(data = univdata, x = "Type", y = "UK Rank")
# Determining the Public and private universities as per UK Ranking
sns.catplot(data = univdata, x = "Type", y = "World Class Rank")
#Result - There are less Private universities which has good world ranking

# To Categorize the colleges according to their fees
sns.displot(univdata['Avg fees UG (in pounds)'])
```

Out[175]:

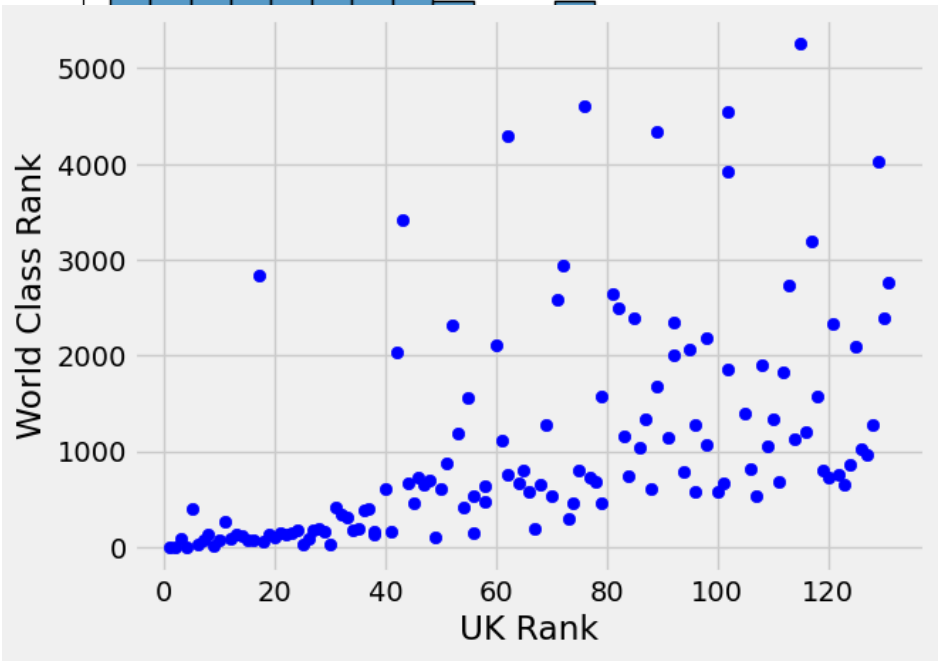
```
<seaborn.axisgrid.FacetGrid at 0x12b5f110700>
```







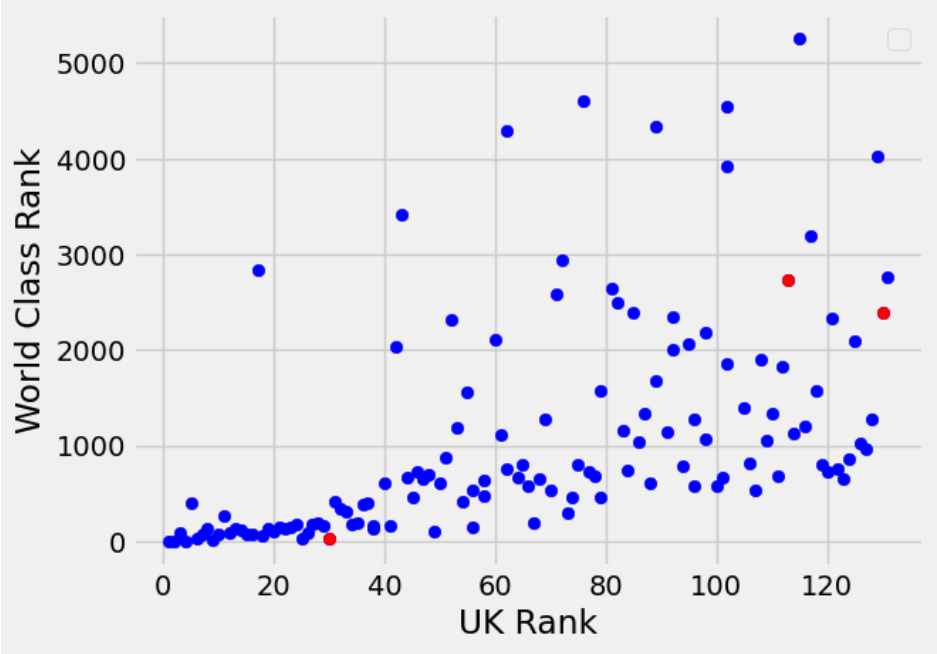

```
X = univdata[["UK Rank", "World Class Rank"]]
# Visualize data point to analyse and point the centroid
plt.scatter(X["UK Rank"], X["World Class Rank"], c="blue")
plt.xlabel("UK Rank")
plt.ylabel("World Class Rank")
plt.show()
```



In [39]:

```
K=3

# select random observation as a centriod
Centroids = (X.sample(n=K))
plt.scatter(X["UK Rank"], X["World Class Rank"], c="blue")
plt.scatter(Centroids["UK Rank"], Centroids["World Class Rank"], c="red")
plt.xlabel("UK Rank")
plt.ylabel("World Class Rank")
plt.legend()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored
when legend() is called with no argument.
```



In [38]:

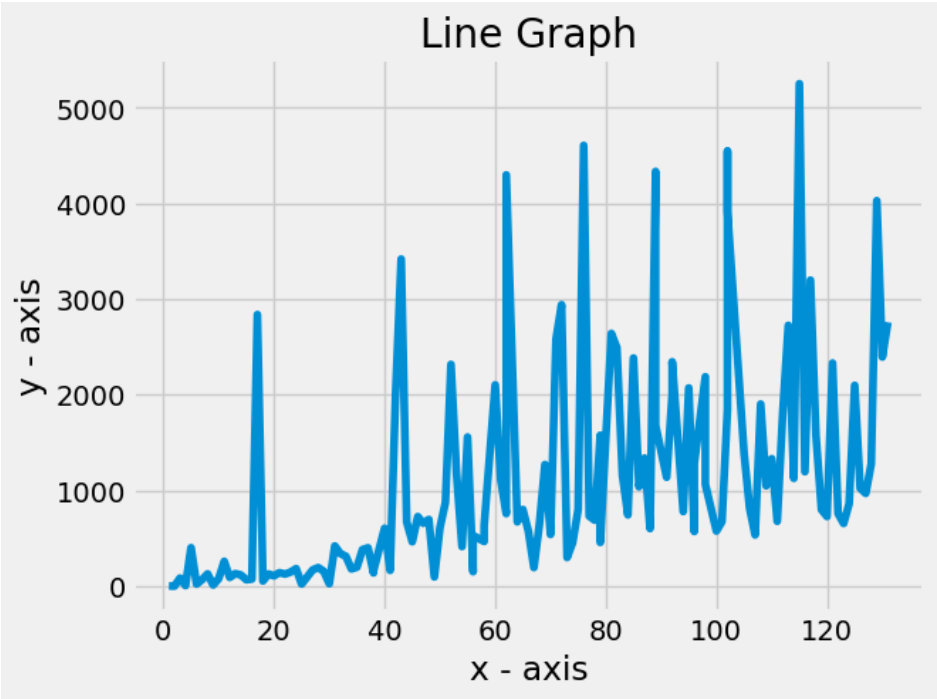
```
Centroids
```

Out[38]:

	UK Rank	World Class Rank
65	66	576
45	46	729
40	41	167

In [26]:

```
X = univdata["UK Rank"]
Y = univdata["World Class Rank"]
plt.plot(X, Y)
plt.xlabel('x - axis')
# naming the y axis
plt.ylabel('y - axis')
plt.title('Line Graph')
plt.show()
```



In [27]:

```
x = univdata.iloc[:,4:6] # 1t for rows and second for columns
x
```

Out[27]:

	UK Rank	World Class Rank
0	1	4
1	2	2
2	3	86
3	4	8
4	5	404
...	...	...
126	127	971
127	128	1281
128	129	4030
129	130	2397
130	131	2759

131 rows × 2 columns

In [28]:

```
identified_clusters = kMeans.fit_predict(x)
identified_clusters
```

Out[28]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 2, 0, 0, 0, 0, 2, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 2, 2, 0, 0, 0, 1, 0, 0, 2, 0, 2, 2, 0, 0, 2, 0, 0, 0,
       1, 2, 0, 2, 2, 0, 2, 0, 0, 2, 0, 0, 0, 2, 1, 1, 2, 0, 0, 2, 0, 0,
       0, 2, 2, 0, 1, 0, 2, 2, 0, 0, 2, 0, 0, 0, 2, 0, 0, 0, 1, 2, 2])
```

In [29]:

```
from sklearn.cluster import KMeans
kMeans = KMeans(3)
kMeans.fit(identified_clusters)
```

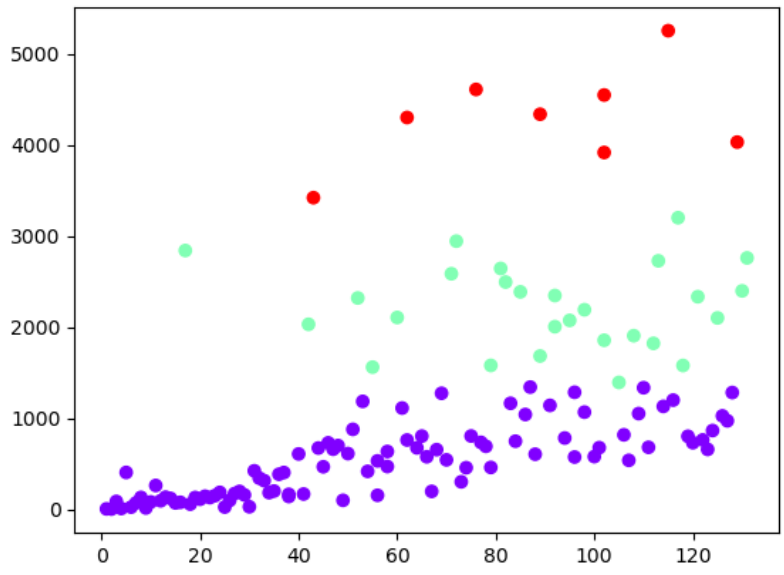
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_2224\471576108.py in <module>
      1 from sklearn.cluster import KMeans
      2 kMeans = KMeans(3)
----> 3 kMeans.fit(identified_clusters)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py in fit(self, X, y, sample_weight)
   1135             Fitted estimator.
   1136         """
-> 1137         X = self._validate_data(
   1138             X,
   1139             accept_sparse="csr",

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py in _validate_data(self, X, y, reset, validate_separatel
y, **check_params)
    564             raise ValueError("Validation should be done on X, y or both.")
    565         elif not no_val_X and no_val_y:
--> 566             X = check_array(X, **check_params)
    567             out = X
    568         elif no_val_X and not no_val_y:

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in check_array(array, accept_sparse, accept_
large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, e
stimator)
    767             # If input is 1D raise error
    768             if array.ndim == 1:
--> 769                 raise ValueError(
    770                     "Expected 2D array, got 1D array instead:\narray={}.\n"
    771                     "Reshape your data either using array.reshape(-1, 1) if "

ValueError: Expected 2D array, got 1D array instead:
array=[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 2. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 2. 1. 0. 0. 0. 0. 0.
 0. 0. 0. 2. 0. 0. 2. 0. 0. 0. 0. 2. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 2. 2.
 0. 0. 0. 1. 0. 0. 2. 0. 2. 2. 0. 0. 2. 0. 0. 0. 1. 2. 0. 2. 2. 0. 2. 0.
 0. 2. 0. 0. 0. 2. 1. 1. 2. 0. 0. 2. 0. 0. 0. 2. 2. 0. 1. 0. 2. 2. 0. 0.
 2. 0. 0. 0. 2. 0. 0. 0. 1. 2. 2.].
Reshape your data either using array.reshape(-1, 1) if your data has a single feature or array.reshape(1, -1) if it
contains a single sample.
```

In [244]:

```
data_with_clusters = data.copy()
data_with_clusters['Clusters'] = identified_clusters
plt.scatter(data_with_clusters['UK Rank'],data_with_clusters['World Class Rank'],c=data_with_clusters['Clusters'],cmap='rainbow')
```

Out[244]:

```
<matplotlib.collections.PathCollection at 0x12b5d7b2340>
```



In [207]:

```
#Training the Model

train_univdata, test_univdata,train_labels, test_labels = train_test_split(univdata, labels, test_size = 0.30, random_state = 189
print('Training Univdata Shape:', a_train.shape)
print('Testing Univdata Shape:', b_train.shape)
print('Training Labels Shape:', train_labels.shape)
print('Testing Labels Shape:', test_labels.shape)
```

```
Training Univdata Shape: (104, 7)
Testing Univdata Shape: (104,)
Training Labels Shape: (91,)
Testing Labels Shape: (40,)
```

In [ ]: