Rana Fahad Aman

Artificial Intelligence Lab_8

30 November 2023

In-Lab Task 1

In [1]:

```python
# Importing libraries needed
# Note that Keras is generally used for deep learning as well
from keras.models import Sequential
from keras.layers import Dense, Dropout
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np
from sklearn import linear_model
from sklearn import preprocessing
from sklearn import tree
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
import pandas as pd
import csv
import seaborn as sns
import matplotlib.pyplot as plt
print("Libraries Imported Successfully!")
```

```
Libraries Imported Successfully!
```

In [2]:

```python
#========================#
# Read Data and fix seed
#========================#
# Fix random seed for reproducibility
np.random.seed(7)
df = pd.read_csv("Alumni Giving Regression (Edited).csv", delimiter=",")
df = df.dropna()
df.head()
```

Out[2]:

	A	B	C	D	E	F
0	24	0.42	0.16	0.59	0.81	0.08
1	19	0.49	0.04	0.37	0.69	0.11
2	18	0.24	0.17	0.66	0.87	0.31
3	8	0.74	0.00	0.81	0.88	0.11
4	8	0.95	0.00	0.86	0.92	0.28

In [3]:

```python
df.info()
```

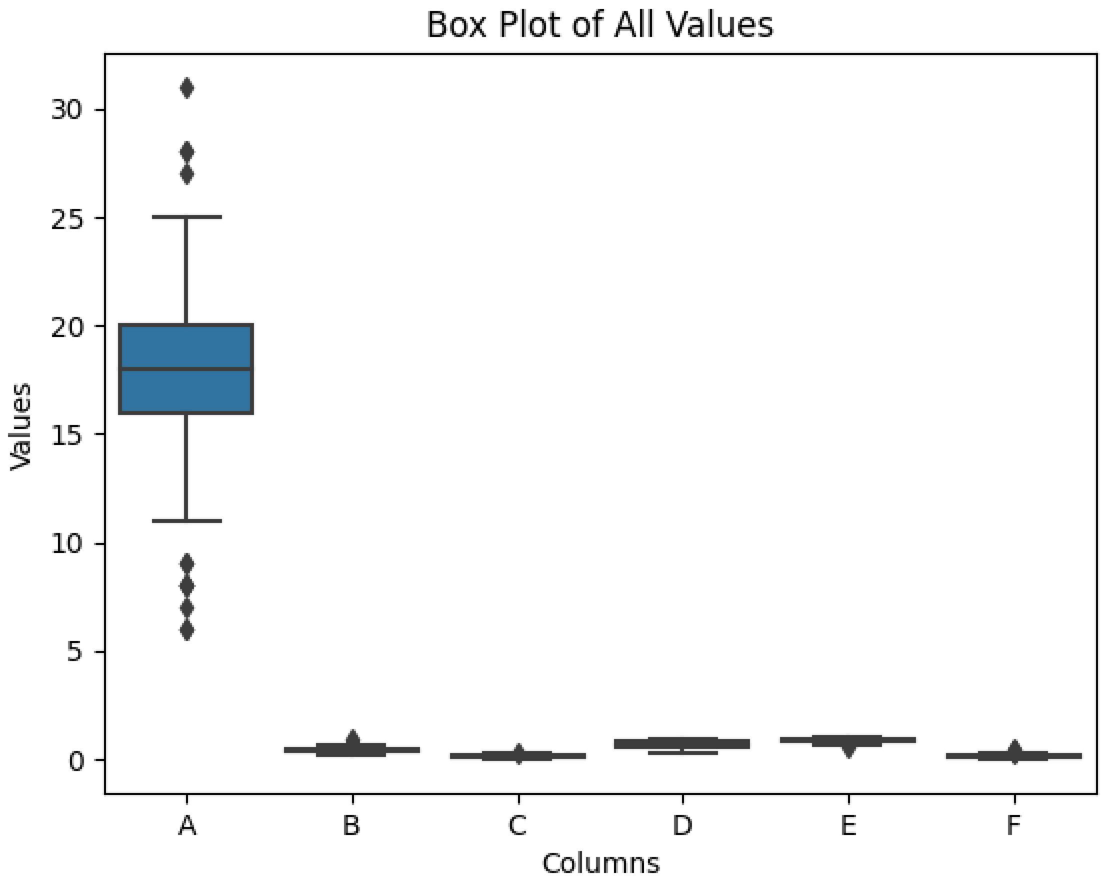
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 123 entries, 0 to 122
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   A       123 non-null    int64  
 1   B       123 non-null    float64
 2   C       123 non-null    float64
 3   D       123 non-null    float64
 4   E       123 non-null    float64
 5   F       123 non-null    float64
dtypes: float64(5), int64(1)
memory usage: 5.9 KB
```

In [4]:
```
df.describe()
```

Out[4]:

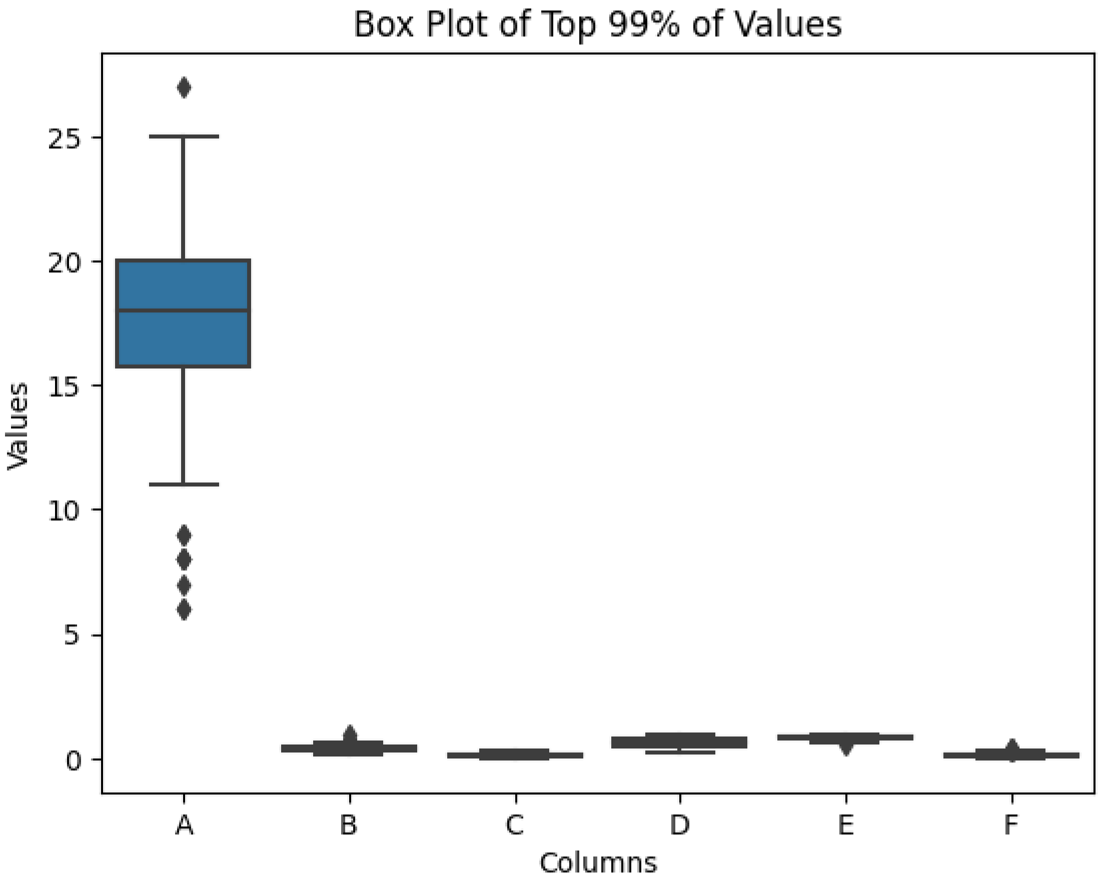
	A	B	C	D	E	F
count	123.000000	123.000000	123.000000	123.000000	123.000000	123.000000
mean	17.772358	0.403659	0.136260	0.645203	0.841138	0.141789
std	4.517385	0.133897	0.060101	0.169794	0.083942	0.080674
min	6.000000	0.140000	0.000000	0.260000	0.580000	0.020000
25%	16.000000	0.320000	0.095000	0.505000	0.780000	0.080000
50%	18.000000	0.380000	0.130000	0.640000	0.840000	0.130000
75%	20.000000	0.460000	0.180000	0.785000	0.910000	0.170000
max	31.000000	0.950000	0.310000	0.960000	0.980000	0.410000

In [5]:
```
sns.boxplot(data=df).set(xlabel='Columns',ylabel='Values', title='Box Plot of All Values
plt.show()
```



In-Lab Task 2

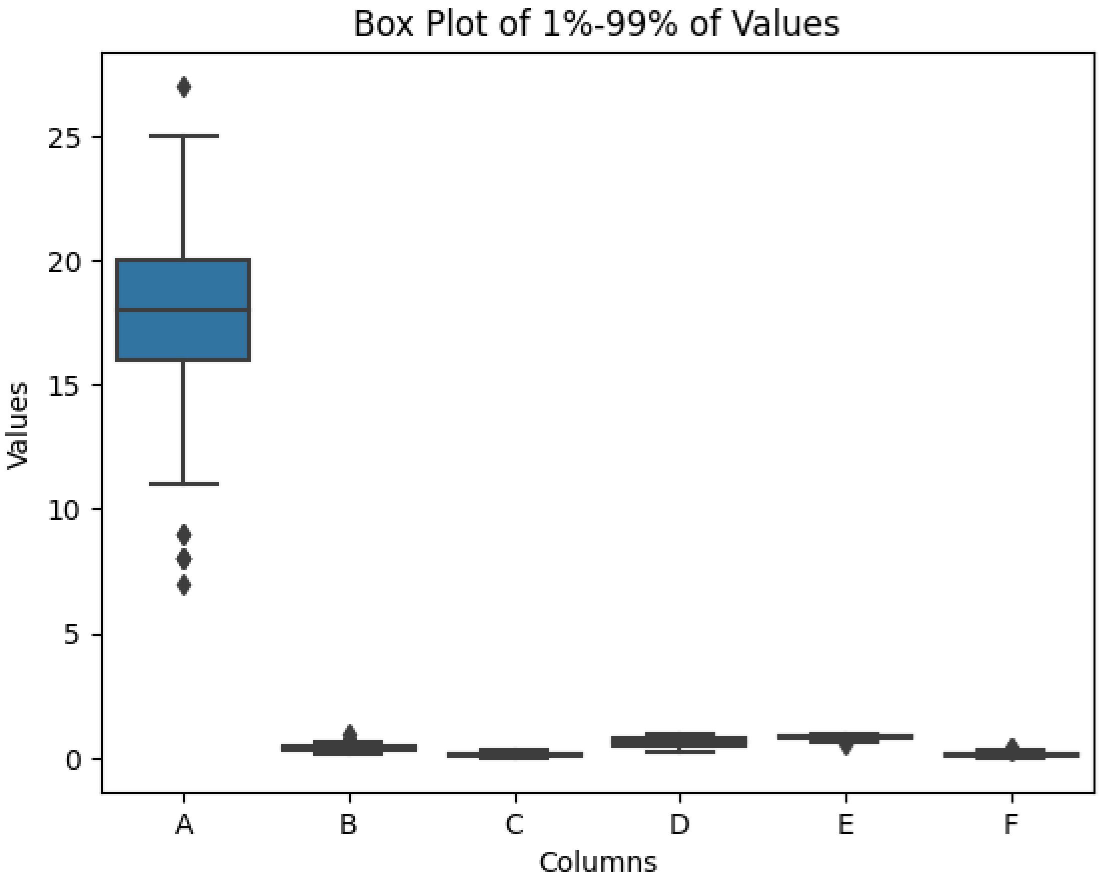
In [6]:
```
quantile99 = df.iloc[:,0].quantile (0.99)
df1 = df[df.iloc[:,0] < quantile99]
sns.boxplot(data=df1).set(xlabel='Columns',ylabel='Values', title='Box Plot of Top 99% o
plt.show()
```



Box Plot of Top 99% of Values

In [7]:

```
quantile1 = df.iloc[:,0].quantile (0.01)
quantile99 = df.iloc[:,0].quantile (0.99)
df2 = df[(df.iloc[:,0] > quantile1) & (df.iloc[:,0] <quantile99)]
sns.boxplot(data=df2).set(xlabel='Columns',ylabel='Values', title='Box Plot of 1%-99% of
plt.show()
```



Box Plot of 1%-99% of Values

In [8]:

```
model3 = RandomForestRegressor()
X_train = df[['A','B','C','D','F']]
y_train = df[['E']]
y_train = np.array(y_train).ravel() # Makes the target variable "y_train" into a 1D arra
model3.fit(X_train, y_train)
RF = model3
importances = RF.feature_importances_
std = np.std([tree.feature_importances_ for tree in RF.estimators_], axis = 0)
indices = np.argsort(importances)[::-1]
# Print the feature ranking
print("========================")
print("    Feature Ranking")
print("========================")
for f in range(X_train.shape[1]):
    print("{Feature#%s}=>(%f)" %(indices[f], importances[indices[f]]*100))
print("========================")
```

```
========================
    Feature Ranking
========================
{Feature#3}=>(89.246726)
{Feature#2}=>(3.788848)
{Feature#1}=>(3.062046)
{Feature#4}=>(2.152351)
{Feature#0}=>(1.750029)
========================
```

In-Lab Task3

In [9]:

```
indices_top3= indices[:3]
print("The top 3 features indexes:",indices_top3)
Y_position=5
TOP_N_FEATURE = 3
X = df.iloc[:, indices_top3]
Y = df.iloc[:,Y_position]
# create model
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.20, random_state =

#Model 1 linear regression
model1 = linear_model.LinearRegression()
model1.fit(X_train, y_train)
y_pred_train1 = model1.predict(X_train)
print("            ======================")
print("                  Regression")
print("================================================")
RMSE_train1 = mean_squared_error(y_train, y_pred_train1)
print("Regression TrainSet: RMSE {}".format(RMSE_train1))
print("================================================")
y_pred1 = model1.predict(X_test)
RMSE_test1 = mean_squared_error(y_test,y_pred1)
print("Regression Testset: RMSE {}".format(RMSE_test1))
print("================================================")
```

```
The top 3 features indexes: [3 2 1]
            ======================
                  Regression
================================================
Regression TrainSet: RMSE 0.0027963867542767 71
================================================
Regression Testset: RMSE 0.004386394878107668
================================================
```

In [ ]: