Assessment-1

Name:M.Sohan Souri

Reg No:19MID0090

TASK 1: How to classify the iris species and draw the inference?

Dataset : Iris

The iris dataset contains 5 features (4 independent features and 1 dependent feature).

1. Feature : sepal length - Independent Feature
2. Feature : sepal width - Independent Feature
3. Feature : petal length - Independent Feature
4. Feature : petal width - Independent Feature
5. Feature : species - Dependent feature that provides the categorical value of species.

DataSet Link: https://www.kaggle.com/datasets/uciml/iris

In [146…

```
#Importing the required Libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import r2_score
```

In [2]:

```
#Reading and storing the data set in a variable
data=pd.read_csv("/Users/apple/Downloads/Iris.csv")
```

In [3]:

```
#Data Set Preview
data.head()
```

Out[3]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [4]:

```
#Basic info of the dataset
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   species            150 non-null    int64
dtypes: float64(4), int64(1)
memory usage: 6.0 KB
```

In [5]:

```
#Statistical Summary of the dataset
data.describe()
```

Out[5]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

In [6]:

```
data.columns
```

Out[6]:

```
Index(['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
       'petal width (cm)', 'species'],
      dtype='object')
```

In [7]:

```
#Data Wrangling
# delete a column
data = data.drop(columns = ['petal width (cm)'])
data.head()
```

Out[7]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	species
0	5.1	3.5	1.4	0
1	4.9	3.0	1.4	0
2	4.7	3.2	1.3	0
3	4.6	3.1	1.5	0
4	5.0	3.6	1.4	0

In [8]:

```
# scatterplot
colors = ['red', 'orange', 'blue']
species = ['Virginica','Versicolor','Setosa']
```

In [9]:

```
data['species'].value_counts()
data.head()
```

Out[9]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	species
0	5.1	3.5	1.4	0
1	4.9	3.0	1.4	0
2	4.7	3.2	1.3	0
3	4.6	3.1	1.5	0
4	5.0	3.6	1.4	0

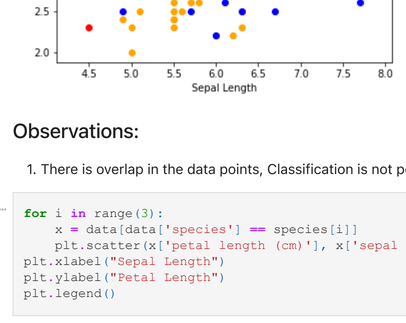
In [10]:

```
for i in range (0,data.shape[0]):
    if data["species"][i]==0:
        data["species"][i]="Virginica"
    elif data["species"][i]==1:
        data["species"][i]='Versicolor'
    else:
        data["species"][i]='Setosa'
```

In [11]:

```
for i in range(3):
    x = data[data['species'] == species[i]]
    plt.scatter(x['sepal length (cm)'], x['sepal width (cm)'], c = colors[i], label=species[i])
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.legend()
```

Out[11]: <matplotlib.legend.Legend at 0x7f9a42716190>



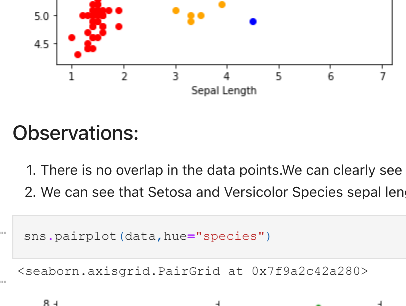
Observations:

1. There is overlap in the data points, Classification is not possible.

In [100…

```
for i in range(3):
    x = data[data['species'] == species[i]]
    plt.scatter(x['petal length (cm)'], x['sepal length (cm)'], c = colors[i], label=species[i])
plt.xlabel("Sepal Length")
plt.ylabel("Petal Length")
plt.legend()
```

Out[100… <matplotlib.legend.Legend at 0x7f9a2c2de610>



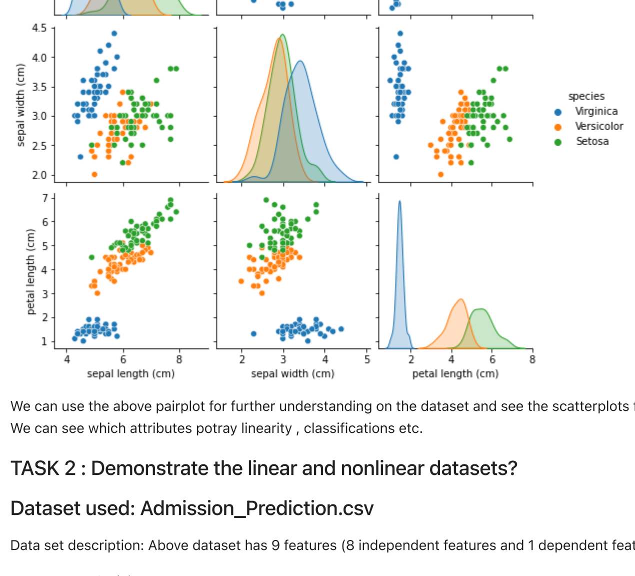
Observations:

1. There is no overlap in the data points.We can clearly see different specie groups in the plot.
2. We can see that Setosa and Versicolor Species sepal length increases with petal length.

In [102…

```
sns.pairplot(data,hue="species")
```

Out[102… <seaborn.axisgrid.PairGrid at 0x7f9a2c42a280>



We can use the above pairplot for further understanding on the dataset and see the scatterplots for all the attribute combinations. We can see which attributes potray linearity , classifications etc.

TASK 2 : Demonstrate the linear and nonlinear datasets?

Dataset used: Admission_Prediction.csv

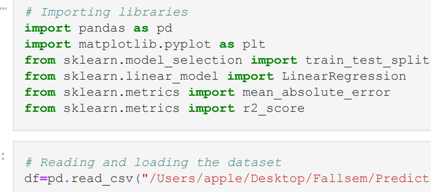
Data set description: Above dataset has 9 features (8 independent features and 1 dependent feature).

1. Feature: Serial No.
2. Feature: GRE Score – Numerical value
3. Feature: TOEFL Score – Numerical value
4. Feature: University Rating – Categorical value (1-5) Feature
5. Feature: SOP – Categorical value (1-5)
6. Feature: LOR – Categorical value (1-5)
7. Feature: CGPA – Numerical value
8. Feature: Research – Categorical value(1,0)
9. Feature: Chance of Admit – Dependent feature which conveys the probability/ chance of getting admitted into college based on above independent feature values.

Dataset Link:https://www.kaggle.com/datasets/adityadeshpande23/admissionpredictioncsv

Linear Regression:

A linear technique to modelling the connection between a scalar answer and one or more explanatory factors is known as linear regression. Simple linear regression is used when there is only one explanatory variable; multiple linear regression is used when there are more than one.



In [147…

```
# Importing libraries
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split # For splitting the dataset into training and testing data
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
```

In [15]:

```
# Reading and loading the dataset
df=pd.read_csv("/Users/apple/Desktop/Fallsem/Predictive Analytics/Admission_Predict.csv")
```

In [103…

```
#Statistical Summary of the dataset
df.describe()
```

Out[103…

	GRE_Score	TOEFL_Score	University_Rating	SOP	LOR_	CGPA	Research	Chance_of_Admit_
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000
mean	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.547500	0.724350
std	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.498362	0.142609
min	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.000000	0.340000
25%	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.000000	0.640000
50%	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.000000	0.730000
75%	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.000000	0.830000
max	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.000000	0.970000

In [104…

```
#Basic data information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   GRE_Score          400 non-null    int64
 1   TOEFL_Score        400 non-null    int64
 2   University_Rating  400 non-null    int64
 3   SOP                400 non-null    float64
 4   LOR_               400 non-null    float64
 5   CGPA               400 non-null    float64
 6   Research           400 non-null    int64
 7   Chance_of_Admit_   400 non-null    float64
dtypes: float64(4), int64(4)
memory usage: 25.1 KB
```

In [18]:

```
df.head() # To view first 5 rows of the dataset/
```

Out[18]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

Data Wrangling

In [19]:

```
df.rename(columns={"Chance of Admit":"Chance_of_Admit"})
```

Out[19]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65
...	...	...	...	...	...	...	...	...	...
395	396	324	110	3	3.5	3.5	9.04	1	0.82
396	397	325	107	3	3.0	3.5	9.11	1	0.84
397	398	330	116	4	5.0	4.5	9.45	1	0.91
398	399	312	103	3	3.5	4.0	8.78	0	0.67
399	400	333	117	4	5.0	4.0	9.66	1	0.95

400 rows × 9 columns

In [20]:

```
df.columns = [c.replace(' ','_') for c in df.columns]
df.columns
```

Out[20]:

```
Index(['Serial_No.', 'GRE_Score', 'TOEFL_Score', 'University_Rating', 'SOP',
       'LOR_', 'CGPA', 'Research', 'Chance_of_Admit_'],
      dtype='object')
```

In [21]:

```
df.Chance_of_Admit_
```

Out[21]:

```
0      0.92
1      0.76
2      0.72
3      0.80
4      0.65
       ...
395    0.82
396    0.84
397    0.91
398    0.67
399    0.95
Name: Chance_of_Admit_, Length: 400, dtype: float64
```

In [23]:

```
serial_number=df["Serial_No."]
df.drop("Serial_No.",axis=1,inplace=True)
```

In [24]:

```
# Missing values count
df.isnull().sum()
```

Out[24]:

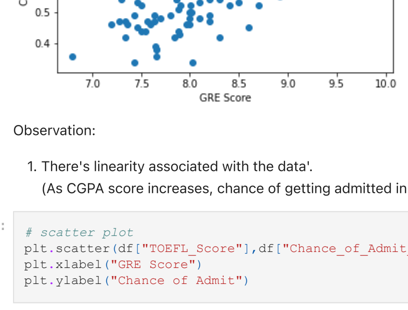
```
GRE_Score            0
TOEFL_Score          0
University_Rating    0
SOP                  0
LOR_                 0
CGPA                 0
Research             0
Chance_of_Admit_     0
dtype: int64
```

Visualization Plots

In [94]:

```
# scatter plot

plt.scatter(df["CGPA"],df["Chance_of_Admit_"])
plt.xlabel("GRE Score")
plt.ylabel("Chance of Admit")
```

Out[94]: Text(0, 0.5, 'Chance of Admit')



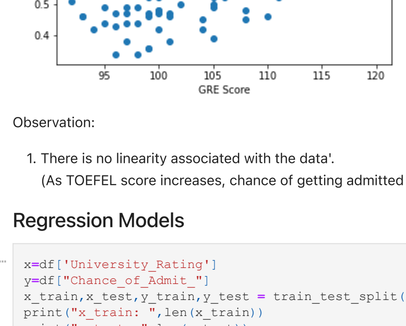
Observation:

1. There's linearity associated with the data'.
   (As CGPA score increases, chance of getting admitted in college increases)

In [97]:

```
# scatter plot
plt.scatter(df["TOEFL_Score"],df["Chance_of_Admit_"])
plt.xlabel("GRE Score")
plt.ylabel("Chance of Admit")
```

Out[97]: Text(0, 0.5, 'Chance of Admit')



Observation:

1. There is no linearity associated with the data'.
   (As TOEFEL score increases, chance of getting admitted in college is not directly proportional)

Regression Models

In [157…

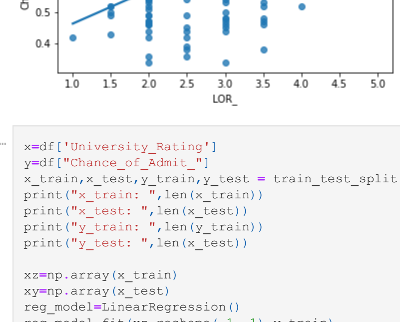
```
x=df['University_Rating']
y=df["Chance_of_Admit_"]
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
print("x_train: ",len(x_train))
print("x_test: ",len(x_test))
print("y_train: ",len(y_train))
print("y_test: ",len(x_test))

xz=np.array(x_train)
xy=np.array(x_test)
reg_model=LinearRegression()
reg_model.fit(xz.reshape(-1, 1),y_train)
y_pred=reg_model.predict(xy.reshape(-1, 1))

print("Regression co-efficients: ",reg_model.coef_)
print("Regression intercept: ",reg_model.intercept_)
print("Mean absolute error: ",round(mean_absolute_error(y_test,y_pred),2))
print("Regression score: ",reg_model.score(xy.reshape(-1, 1),y_test))
```

```
x_train:  280
x_test:  120
y_train:  280
y_test:  120
Regression co-efficients:  [0.08873586]
Regression intercept:  0.4513958482462419
Mean absolute error:  0.08
Regression score:  0.5064041542783597
```

In [114…

```
sns.regplot(x = "LOR_",
            y = "Chance_of_Admit_",
            ci = None,
            data = df)
```

Out[114… <AxesSubplot:xlabel='LOR_', ylabel='Chance_of_Admit_'>



In [158…

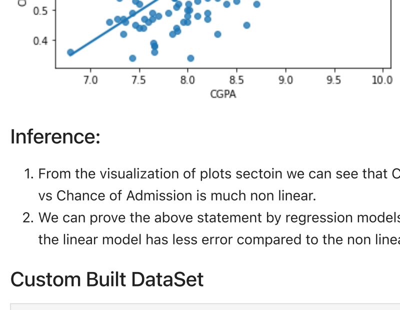
```
x=df['University_Rating']
y=df["Chance_of_Admit_"]
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
print("x_train: ",len(x_train))
print("x_test: ",len(x_test))
print("y_train: ",len(y_train))
print("y_test: ",len(x_test))

xz=np.array(x_train)
xy=np.array(x_test)
reg_model=LinearRegression()
reg_model.fit(xz.reshape(-1, 1),y_train)
y_pred=reg_model.predict(xy.reshape(-1, 1))

print("Regression co-efficients: ",reg_model.coef_)
print("Regression intercept: ",reg_model.intercept_)
print("Mean absolute error ",round(mean_absolute_error(y_test,y_pred),2))
print("Regression score: ",reg_model.score(xy.reshape(-1, 1),y_test))
```

```
x_train:  280
x_test:  120
y_train:  280
y_test:  120
Regression co-efficients:  [0.08774343]
Regression intercept:  0.45980984601026603
Mean absolute error  0.08
Regression score:  0.43056696484817814
```

In [108…

```
sns.regplot(x = "CGPA",
            y = "Chance_of_Admit_",
            ci = None,
            data = df)
```

Out[108… <AxesSubplot:xlabel='CGPA', ylabel='Chance_of_Admit_'>



Inference:

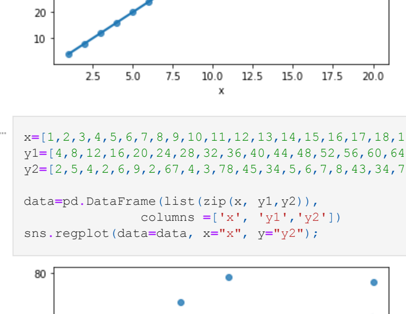
1. From the visualization of plots sectoin we can see that CGPAvs Chance of admission scatter plot can be linear where as TOEFEL vs Chance of Admission is much non linear.
2. We can prove the above statement by regression models. We built two models of the same above cases and we could see that the linear model has less error compared to the non linear model. CGPA case has less error when compared to TOEFEL case.

Custom Built DataSet

In [ ]:

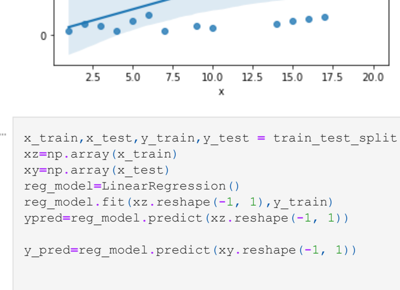
In [182…

```
x=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
y1=[4,8,12,16,20,24,28,32,36,40,44,48,52,56,60,64,68,72,76,80]
y2=[2,5,4,2,6,9,2,67,4,3,78,45,34,5,6,7,8,43,34,76]

data=pd.DataFrame(list(zip(x, y1,y2)),
                  columns =['x', 'y1','y2'])
sns.regplot(data=data, x="x", y="y1");
```



In [183…

```
x=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
y1=[4,8,12,16,20,24,28,32,36,40,44,48,52,56,60,64,68,72,76,80]
y2=[2,5,4,2,6,9,2,67,4,3,78,45,34,5,6,7,8,43,34,76]

data=pd.DataFrame(list(zip(x, y1,y2)),
                  columns =['x', 'y1','y2'])
sns.regplot(data=data, x="x", y="y2");
```



In [168…

```
x_train,x_test,y_train,y_test = train_test_split(x,y1,test_size=0.3)
xz=np.array(x_train)
xy=np.array(x_test)
reg_model=LinearRegression()
reg_model.fit(xz.reshape(-1, 1),y_train)
ypred=reg_model.predict(xz.reshape(-1, 1))

y_pred=reg_model.predict(xy.reshape(-1, 1))



print("Mean absolute error for linear dataset:",round(mean_absolute_error(y_test,y_pred),2))
print("Rsquare Score:",r2_score(y_train,ypred))

x_train,x_test,y_train,y_test = train_test_split(x,y2,test_size=0.3)
xz=np.array(x_train)
xy=np.array(x_test)
reg_model=LinearRegression()
reg_model.fit(xz.reshape(-1, 1),y_train)
ypred=reg_model.predict(xz.reshape(-1, 1))

y_pred=reg_model.predict(xy.reshape(-1, 1))


print("Mean absolute error for linear dataset:",round(mean_absolute_error(y_test,y_pred),2))
print("Rsquare Score:",r2_score(y_train,ypred))
```

```
Mean absolute error for linear dataset: 0.0
Rsquare Score: 1.0
Mean absolute error for linear dataset: 16.31
Rsquare Score: 0.23251768430740527
```

Inference:

1. From the first plot we can see that the regression line is touching all the points, and there is no error, hence the plot is linear
2. From the second plot we can see that regression line in not touching all the points , hence there is lot of error. So we can say the data are non linear.

In [ ]: