Basic Project on Regression, Classification and ANN

1.Regression

Step 1: Import library

```
import pandas as pd
```

```
import numpy as np
```

```
import seaborn as sns
```

Step 2: Import Data

```
df=pd.read_csv('https://github.com/YBI-Foundation/Dataset/raw/main/Fruits.csv')
```

```
df.head()
```

	Fruit Category	Fruit Name	Fruit Weight	Fruit Width	Fruit Length	Fruit Colour Score
0	1	Apple	192	8.4	7.3	0.55
1	1	Apple	180	8.0	6.8	0.59

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59 entries, 0 to 58
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Fruit Category      59 non-null     int64  
 1   Fruit Name          59 non-null     object 
 2   Fruit Weight        59 non-null     int64  
 3   Fruit Width         59 non-null     float64
 4   Fruit Length        59 non-null     float64
 5   Fruit Colour Score  59 non-null     float64
dtypes: float64(3), int64(2), object(1)
memory usage: 2.9+ KB
```

```
df.describe()
```

```
Fruit Category  Fruit Weight  Fruit Width  Fruit Length  Fruit Colour Score
```

```
df.shape
```

```
(59, 6)
```

```
df.columns
```

```
Index(['Fruit Category', 'Fruit Name', 'Fruit Weight', 'Fruit Width',
       'Fruit Length', 'Fruit Colour Score'],
      dtype='object')
```

Step 3. Define y X

```
y = df['Fruit Colour Score']
```

```
y.shape
```

```
(59,)
```

```
y
```

```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: diabetes, Length: 768, dtype: int64
```

```
#X=df.drop(['Fruit Colour Score','Fruit Name'],axis=1)
#or
X=df[['Fruit Category', 'Fruit Weight', 'Fruit Width','Fruit Length']]
```

```
X.shape
```

```
(59, 4)
```

```
X
```

	Fruit Category	Fruit Weight	Fruit Width	Fruit Length
0	1	192	8.4	7.3
1	1	180	8.0	6.8
2	1	176	7.4	7.2
3	1	178	7.1	7.8
4	1	172	7.4	7.0
5	1	166	6.9	7.3
6	1	172	7.1	7.6
7	1	154	7.0	7.1
8	1	164	7.3	7.7
9	1	152	7.6	7.3
10	1	156	7.7	7.1
11	1	156	7.6	7.5
12	1	168	7.5	7.6
13	1	162	7.5	7.1
14	1	162	7.4	7.2
15	1	160	7.5	7.5
16	1	156	7.4	7.4
17	1	140	7.3	7.1
18	1	170	7.6	7.9
19	2	86	6.2	4.7
20	2	84	6.0	4.6
21	2	80	5.8	4.3

22	2	80	5.9	4.3
23	2	76	5.8	4.0
24	2	342	9.0	9.4
25	2	356	9.2	9.2
26	2	362	9.6	9.2
27	2	204	7.5	9.2
28	2	140	6.7	7.1
29	2	160	7.0	7.4
30	2	158	7.1	7.5
31	2	210	7.8	8.0
32	2	164	7.2	7.0
33	2	190	7.5	8.1
34	2	142	7.6	7.8
35	2	150	7.1	7.9
36	2	160	7.1	7.6
37	2	154	7.3	7.3
38	2	158	7.2	7.8
39	2	144	6.8	7.4
40	2	154	7.1	7.5
41	2	180	7.6	8.2
42	2	154	7.2	7.2
43	3	97	7.2	10.3
44	3	70	7.3	10.5

Step 4: Splitting Data

```
47 3 98 7.3 9.7
```

```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(X,y,train_size=0.7,random_state=2529)
```

```
50 3 65 6.0 8.2
```

```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

```
((41, 4), (18, 4), (41,), (18,))
```

Step 5: Creating Model

```
50 3 58 5.9 8.1
```

```
#from sklearn.linear_model import LinearRegression
#model=LinearRegression()

from sklearn.neighbors import KNeighborsRegressor
model=KNeighborsRegressor()

#from sklearn.tree import DecisionTreeRegressor
#model=DecisionTreeRegressor()
```

Step 6: Training Model

```
model.fit(X_train,y_train)
```

```
KNeighborsRegressor()
```

Step 7: Predicting Model

```
y_pred=model.predict(X_test)
```

```
y_pred
```

```
array([0.712, 0.712, 0.808, 0.794, 0.794, 0.796, 0.78 , 0.762, 0.84 ,
       0.812, 0.766, 0.812, 0.796, 0.732, 0.794, 0.844, 0.816, 0.712])
```

Step 8: Accuracy

```
from sklearn.metrics import mean_absolute_percentage_error
```

```
mean_absolute_percentage_error(y_test,y_pred)
```

```
0.10590529996817882
```

2. Classification

Steps 1 and 2 are common

```
df.columns
```

```
Index(['Fruit Category', 'Fruit Name', 'Fruit Weight', 'Fruit Width',
       'Fruit Length', 'Fruit Colour Score'],
      dtype='object')
```

Step 3. Define y X

```
y = df['Fruit Category']
```

```
y.shape
```

```
(59,)
```

```
df
```

	Fruit Category	Fruit Name	Fruit Weight	Fruit Width	Fruit Length	Fruit Colour Score
0	1	Apple	192	8.4	7.3	0.55
1	1	Apple	180	8.0	6.8	0.59
2	1	Apple	176	7.4	7.2	0.60
3	1	Apple	178	7.1	7.8	0.92
4	1	Apple	172	7.4	7.0	0.89
5	1	Apple	166	6.9	7.3	0.93
6	1	Apple	172	7.1	7.6	0.92
7	1	Apple	154	7.0	7.1	0.88
8	1	Apple	164	7.3	7.7	0.70
9	1	Apple	152	7.6	7.3	0.69
10	1	Apple	156	7.7	7.1	0.69
11	1	Apple	156	7.6	7.5	0.67
12	1	Apple	168	7.5	7.6	0.73
13	1	Apple	162	7.5	7.1	0.83
14	1	Apple	162	7.4	7.2	0.85
15	1	Apple	160	7.5	7.5	0.86
16	1	Apple	156	7.4	7.4	0.84
17	1	Apple	140	7.3	7.1	0.87
18	1	Apple	170	7.6	7.9	0.88
19	2	Orange	86	6.2	4.7	0.80
20	2	Orange	84	6.0	4.6	0.79
21	2	Orange	80	5.8	4.3	0.77

22	2	Orange	80	5.9	4.3	0.81
23	2	Orange	76	5.8	4.0	0.81
24	2	Orange	342	9.0	9.4	0.75
25	2	Orange	356	9.2	9.2	0.75
26	2	Orange	362	9.6	9.2	0.74
27	2	Orange	204	7.5	9.2	0.77
28	2	Orange	140	6.7	7.1	0.72
29	2	Orange	160	7.0	7.4	0.81
30	2	Orange	158	7.1	7.5	0.79
31	2	Orange	210	7.8	8.0	0.82
32	2	Orange	164	7.2	7.0	0.80
33	2	Orange	190	7.5	8.1	0.74
34	2	Orange	142	7.6	7.8	0.75
35	2	Orange	150	7.1	7.9	0.75
36	2	Orange	160	7.1	7.6	0.76
37	2	Orange	154	7.3	7.3	0.79
38	2	Orange	158	7.2	7.8	0.77
39	2	Orange	144	6.8	7.4	0.75
40	2	Orange	154	7.1	7.5	0.78
41	2	Orange	180	7.6	8.2	0.79
42	2	Orange	154	7.2	7.2	0.82
43	3	Lemon	97	7.2	10.3	0.70
44	3	Lemon	70	7.3	10.5	0.72

45	3	Lemon	93	7.2	9.2	0.72
46	3	Lemon	80	7.3	10.2	0.71
47	3	Lemon	98	7.3	9.7	0.72

```
X=df[['Fruit Weight', 'Fruit Width','Fruit Length', 'Fruit Colour Score']]
```

```
X.shape
```

```
(59, 4)
```

```
X
```

	Fruit Weight	Fruit Width	Fruit Length	Fruit Colour Score
0	192	8.4	7.3	0.55
1	180	8.0	6.8	0.59
2	176	7.4	7.2	0.60
3	178	7.1	7.8	0.92
4	172	7.4	7.0	0.89
5	166	6.9	7.3	0.93
6	172	7.1	7.6	0.92
7	154	7.0	7.1	0.88
8	164	7.3	7.7	0.70
9	152	7.6	7.3	0.69
10	156	7.7	7.1	0.69
11	156	7.6	7.5	0.67
12	168	7.5	7.6	0.73
13	162	7.5	7.1	0.83
14	162	7.4	7.2	0.85
15	160	7.5	7.5	0.86
16	156	7.4	7.4	0.84
17	140	7.3	7.1	0.87
18	170	7.6	7.9	0.88
19	86	6.2	4.7	0.80
20	84	6.0	4.6	0.79
21	80	5.8	4.3	0.77

22	80	5.9	4.3	0.81
23	76	5.8	4.0	0.81
24	342	9.0	9.4	0.75
25	356	9.2	9.2	0.75
26	362	9.6	9.2	0.74
27	204	7.5	9.2	0.77
28	140	6.7	7.1	0.72
29	160	7.0	7.4	0.81
30	158	7.1	7.5	0.79
31	210	7.8	8.0	0.82
32	164	7.2	7.0	0.80
33	190	7.5	8.1	0.74
34	142	7.6	7.8	0.75
35	150	7.1	7.9	0.75
36	160	7.1	7.6	0.76
37	154	7.3	7.3	0.79
38	158	7.2	7.8	0.77
39	144	6.8	7.4	0.75
40	154	7.1	7.5	0.78
41	180	7.6	8.2	0.79
42	154	7.2	7.2	0.82
43	97	7.2	10.3	0.70
44	70	7.3	10.5	0.72

45	93	7.2	9.2	0.72

Step 4: Splitting Data

48	87	7.3	10.1	0.72

```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(X,y,train_size=0.7,random_state=2529)
```

51	58	6.0	7.5	0.72

```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

```
((41, 4), (18, 4), (41,), (18,))
```

54	58	6.1	8.5	0.71

Step 5: Creating Model

57	70	6.5	8.5	0.72

```
#from sklearn.linear_model import LogisticRegression
#model=LogisticRegression()

from sklearn.neighbors import KNeighborsClassifier
model=KNeighborsClassifier()

#from sklearn.tree import DecisionTreeClassifier
#model=DecisionTreeClassifier()
```

Step 6: Training Model

```
model.fit(X_train,y_train)
```

```
KNeighborsClassifier()
```

Step 7: Predicting Model

```
y_pred=model.predict(X_test)
```

```
y_pred
```

```
array([3, 3, 2, 1, 1, 1, 2, 2, 1, 2, 2, 2, 2, 3, 1, 1, 1, 3])
```

Step 8: Accuracy

```
from sklearn.metrics import confusion_matrix, classification_report
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[4, 2, 0],
       [3, 3, 0],
       [0, 2, 4]])
```

```
print(classification_report(y_test,y_pred))
```

```
                  precision    recall  f1-score   support

               1       0.57      0.67      0.62         6
               2       0.43      0.50      0.46         6
               3       1.00      0.67      0.80         6

        accuracy                           0.61        18
       macro avg       0.67      0.61      0.63        18
```

```
weighted avg       0.67      0.61      0.63        18
```

3. Artificial Nueral Network

```
import pandas as pd
```

```
data = pd.read_csv('https://github.com/YBI-Foundation/Dataset/raw/main/Diabetes.csv')
```

```
data.head()
```

	pregnancies	glucose	diastolic	triceps	insulin	bmi	dpf	age	diabetes
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
data.columns
```

```
Index(['pregnancies', 'glucose', 'diastolic', 'triceps', 'insulin', 'bmi',
       'dpf', 'age', 'diabetes'],
      dtype='object')
```

```
y=data['diabetes']
```

```
X=data[['pregnancies', 'glucose', 'diastolic', 'triceps', 'insulin', 'bmi',
```