NAME – Aruna Vattikota

SPECIFICATION - COMPUTER SCIENCE AND ENGINEERING

COLLEGE - KASIREDDY NARAYAN REDDY COLLEGE OF ENGINEERING AND RESEARCH

YEAR OF STUDY - 4 TH YEAR

MAJOR PROJECT 1

Choose any dataset of ur choice and apply suitable REGRESSION/CLASSIFIER

#Dataset - '/content/Asia_cup_1984_to_2018.csv'

#1.Take a dataset and create dataframe

```
import pandas as pd
df = pd.read_csv("/content/Asia_cup_1984_to_2018.csv")
 df
```

	Match id	Team_1	Team_2	Format	Ground	Year	Toss	Selection
0	1	Pakistan	Sri Lanka	ODI	Sharjah	1984	Sri Lanka	Bowling
1	2	India	Sri Lanka	ODI	Sharjah	1984	India	Bowling
2	3	India	Pakistan	ODI	Sharjah	1984	India	Batting
3	4	Sri Lanka	Pakistan	ODI	Colombo(PSS)	1986	Sri Lanka	Bowling
4	5	Bangladesh	Pakistan	ODI	Moratuwa	1986	Pakistan	Bowling
...	...	...	...	...	...	...	...	...
109	110	India	Pakistan	ODI	Dubai(DSC)	2018	Pakistan	Batting
110	111	Afghanistan	Bangladesh	ODI	Abu Dhabi	2018	Bangladesh	Batting
111	112	Afghanistan	India	ODI	Dubai(DSC)	2018	Afghanistan	Batting
112	113	Bangladesh	Pakistan	ODI	Abu Dhabi	2018	Bangladesh	Batting
113	114	Bangladesh	India	ODI	Dubai(DSC)	2018	India	Bowling

114 rows × 28 columns

#To display the information present in the table

```
 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 114 entries, 0 to 113
Data columns (total 28 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Match id              114 non-null    int64
 1   Team_1                114 non-null    object
 2   Team_2                114 non-null    object
 3   Format                114 non-null    object
 4   Ground                114 non-null    object
 5   Year                  114 non-null    int64
 6   Toss                  114 non-null    object
 7   Selection             114 non-null    object
 8   fi_score              114 non-null    float64
 9   fi_wickets            114 non-null    float64
 10  fi_4s                 114 non-null    float64
```

https://colab.research.google.com/drive/1dBBOL_TAkkbb0stNCtlLuAIgX307yeM6#scrollTo=Lu7MLi1k2avG&printMode=true


```
 11  fi_6s                 114 non-null    float64
 12  fi_extra              114 non-null    float64
 13  fi_run_rate           114 non-null    float64
 14  fi_avg_str_rate       114 non-null    float64
 15  fi_max_score          114 non-null    float64
 16  fi_max_indv_wickets   114 non-null    float64
 17  Player Of The Match   113 non-null    object
 18  Result                114 non-null    object
 19  si_score              114 non-null    float64
 20  si_wickets            114 non-null    float64
 21  si_4s                 114 non-null    float64
 22  si_6s                 114 non-null    float64
 23  si_extras             114 non-null    float64
 24  si_run_rate           114 non-null    float64
 25  si_avg_str_rate       114 non-null    float64
 26  si_max_indv_wickets   114 non-null    float64
 27  si_max_indv_wickets.1 114 non-null    float64
dtypes: float64(18), int64(2), object(8)
memory usage: 25.1+ KB
```

df.shape #114 rows and 28 columns

(114, 28)

df.size #total no of elements

3192

#To check the number to null values present

df.isnull()

	Match id	Team_1	Team_2	Format	Ground	Year	Toss	Selection	fi_score	fi_wick
0	False	False	False	False	False	False	False	False	False	F
1	False	False	False	False	False	False	False	False	False	F
2	False	False	False	False	False	False	False	False	False	F
3	False	False	False	False	False	False	False	False	False	F
4	False	False	False	False	False	False	False	False	False	F
...	...	...	...	...	...	...	...	...	...	
109	False	False	False	False	False	False	False	False	False	F
110	False	False	False	False	False	False	False	False	False	F
111	False	False	False	False	False	False	False	False	False	F
112	False	False	False	False	False	False	False	False	False	F
113	False	False	False	False	False	False	False	False	False	F

114 rows × 28 columns

```
# To display 1st 5 rows indexes
df.head()
```

	Match id	Team_1	Team_2	Format	Ground	Year	Toss	Selection	fi_sco
0	1	Pakistan	Sri Lanka	ODI	Sharjah	1984	Sri Lanka	Bowling	18
1	2	India	Sri Lanka	ODI	Sharjah	1984	India	Bowling	9
2	3	India	Pakistan	ODI	Sharjah	1984	India	Batting	18
3	4	Sri Lanka	Pakistan	ODI	Colombo(PSS)	1986	Sri Lanka	Bowling	11
4	5	Bangladesh	Pakistan	ODI	Moratuwa	1986	Pakistan	Bowling	9

5 rows × 28 columns

```
#To display last 5 row indexes
df.tail()
```

	Match id	Team_1	Team_2	Format	Ground	Year	Toss	Selection	f
109	110	India	Pakistan	ODI	Dubai(DSC)	2018	Pakistan	Batting	
110	111	Afghanistan	Bangladesh	ODI	Abu Dhabi	2018	Bangladesh	Batting	
111	112	Afghanistan	India	ODI	Dubai(DSC)	2018	Afghanistan	Batting	
112	113	Bangladesh	Pakistan	ODI	Abu Dhabi	2018	Bangladesh	Batting	
113	114	Bangladesh	India	ODI	Dubai(DSC)	2018	India	Bowling	

5 rows × 28 columns

#2.preprocessing - Filtering of Data(to remove Format columns)

df_numeric = df_numeric.drop(['Match id'],axis =1)#axis = 1 -column,axis = 0 -row

df_numeric

	Year	fi_score	fi_wickets	fi_4s	fi_6s	fi_extra	fi_run_rate	fi_avg_str_rat
0	1984	187.0	9.0	9.0	3.0	21.0	4.06	52.0
1	1984	97.0	0.0	9.0	0.0	14.0	4.47	60.4
2	1984	188.0	4.0	13.0	3.0	17.0	4.08	60.2
3	1986	116.0	10.0	10.0	0.0	14.0	3.42	37.8
4	1986	94.0	10.0	0.0	0.0	9.0	2.64	24.6
...	...	...	...	...	...	...	...	..
109	2018	238.0	1.0	24.0	6.0	1.0	6.02	91.3
110	2018	246.0	7.0	20.0	3.0	8.0	4.92	76.1
111	2018	252.0	8.0	17.0	11.0	7.0	5.04	54.1
112	2018	239.0	10.0	17.0	1.0	9.0	4.89	73.2
113	2018	222.0	10.0	17.0	4.0	7.0	4.57	49.9

114 rows × 19 columns

#We want to consider only the numeric data

#so we will create a new dataframe with only numeric data

df_numeric = df.select_dtypes(include = ['float64','int64'])

df_numeric

	Match id	Year	fi_score	fi_wickets	fi_4s	fi_6s	fi_extra	fi_run_rate	fi_avg_
0	1	1984	187.0	9.0	9.0	3.0	21.0	4.06	
1	2	1984	97.0	0.0	9.0	0.0	14.0	4.47	
2	3	1984	188.0	4.0	13.0	3.0	17.0	4.08	
3	4	1986	116.0	10.0	10.0	0.0	14.0	3.42	
4	5	1986	94.0	10.0	0.0	0.0	9.0	2.64	
...	...	...	...	...	...	...	...	...	
109	110	2018	238.0	1.0	24.0	6.0	1.0	6.02	
110	111	2018	246.0	7.0	20.0	3.0	8.0	4.92	
111	112	2018	252.0	8.0	17.0	11.0	7.0	5.04	
112	113	2018	239.0	10.0	17.0	1.0	9.0	4.89	
113	114	2018	222.0	10.0	17.0	4.0	7.0	4.57	

114 rows × 20 columns

#To display the table information which contains only numeric data

df_numeric.info()

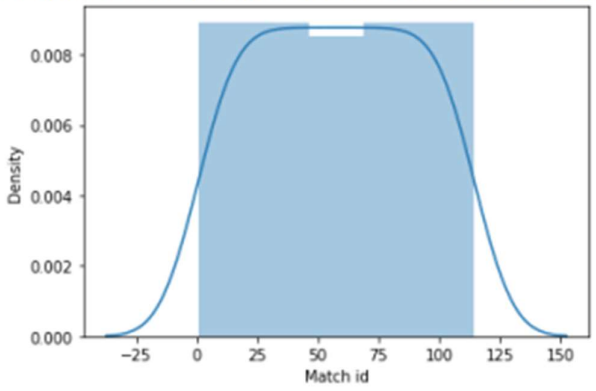
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 114 entries, 0 to 113
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Match id              114 non-null    int64
 1   Year                  114 non-null    int64
 2   fi_score              114 non-null    float64
 3   fi_wickets            114 non-null    float64
 4   fi_4s                 114 non-null    float64
 5   fi_6s                 114 non-null    float64
 6   fi_extra              114 non-null    float64
 7   fi_run_rate           114 non-null    float64
 8   fi_avg_str_rate       114 non-null    float64
 9   fi_max_score          114 non-null    float64
 10  fi_max_indv_wickets   114 non-null    float64
 11  si_score              114 non-null    float64
 12  si_wickets            114 non-null    float64
 13  si_4s                 114 non-null    float64
 14  si_6s                 114 non-null    float64
 15  si_extras             114 non-null    float64
 16  si_run_rate           114 non-null    float64
 17  si_avg_str_rate       114 non-null    float64
 18  si_max_indv_wickets   114 non-null    float64
 19  si_max_indv_wickets.1 114 non-null    float64
dtypes: float64(18), int64(2)
memory usage: 17.9 KB
```

#VISUALIZATION

import seaborn as sns

sns.distplot(df['Match id'])# # distribution plot

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fe1c9426590>
```



#4.divide the data into i/p and o/p

#output - Smog_level

#input - All the columns except the Smog_level column

x = df_numeric.iloc[:,0:6].values

x

```
array([[1.000e+00, 1.984e+03, 1.870e+02, 9.000e+00, 9.000e+00, 3.000e+00],
       [2.000e+00, 1.984e+03, 9.700e+01, 0.000e+00, 9.000e+00, 0.000e+00],
       [3.000e+00, 1.984e+03, 1.880e+02, 4.000e+00, 1.300e+01, 3.000e+00],
       [4.000e+00, 1.986e+03, 1.160e+02, 1.000e+01, 1.000e+01, 0.000e+00],
       [5.000e+00, 1.986e+03, 9.400e+01, 1.000e+01, 0.000e+00, 0.000e+00],
       [6.000e+00, 1.986e+03, 1.320e+02, 3.000e+00, 0.000e+00, 0.000e+00],
       [7.000e+00, 1.986e+03, 1.950e+02, 5.000e+00, 1.500e+01, 1.000e+00],
       [8.000e+00, 1.988e+03, 1.940e+02, 7.000e+00, 5.000e+00, 0.000e+00],
       [9.000e+00, 1.988e+03, 9.900e+01, 8.000e+00, 5.000e+00, 0.000e+00],
       [1.000e+01, 1.988e+03, 2.540e+02, 1.000e+01, 1.400e+01, 4.000e+00],
       [1.100e+01, 1.988e+03, 1.110e+02, 6.000e+00, 0.000e+00, 0.000e+00],
       [1.200e+01, 1.988e+03, 1.430e+02, 6.000e+00, 8.000e+00, 0.000e+00],
       [1.300e+01, 1.988e+03, 1.180e+02, 8.000e+00, 0.000e+00, 0.000e+00],
       [1.400e+01, 1.988e+03, 1.800e+02, 4.000e+00, 1.000e+01, 3.000e+00],
       [1.500e+01, 1.990e+03, 1.710e+02, 1.000e+00, 1.200e+01, 3.000e+00],
       [1.600e+01, 1.990e+03, 1.780e+02, 1.000e+01, 9.000e+00, 0.000e+00],
       [1.700e+01, 1.990e+03, 1.780e+02, 9.000e+00, 0.000e+00, 0.000e+00],
       [1.800e+01, 1.991e+03, 2.050e+02, 3.000e+00, 7.000e+00, 1.000e+00],
       [1.900e+01, 1.995e+03, 1.630e+02, 1.000e+01, 1.200e+01, 1.000e+00],
       [2.000e+01, 1.995e+03, 1.260e+02, 1.000e+01, 8.000e+00, 0.000e+00],
       [2.100e+01, 1.995e+03, 1.690e+02, 1.000e+01, 1.100e+01, 2.000e+00],
       [2.200e+01, 1.995e+03, 1.510e+02, 8.000e+00, 7.000e+00, 0.000e+00],
       [2.300e+01, 1.995e+03, 2.060e+02, 2.000e+00, 2.400e+01, 1.000e+00],
       [2.400e+01, 1.995e+03, 1.780e+02, 9.000e+00, 5.000e+00, 3.000e+00],
       [2.500e+01, 1.995e+03, 2.330e+02, 2.000e+00, 1.500e+01, 2.000e+00],
       [2.600e+01, 1.997e+03, 2.390e+02, 1.000e+01, 9.000e+00, 1.000e+00],
       [2.700e+01, 1.997e+03, 2.100e+02, 1.000e+01, 1.500e+01, 1.000e+00],
       [2.800e+01, 1.997e+03, 2.310e+02, 4.000e+00, 2.300e+01, 0.000e+00],
       [2.900e+01, 1.997e+03, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00],
       [3.000e+01, 1.997e+03, 2.960e+02, 4.000e+00, 3.100e+01, 4.000e+00],
       [3.100e+01, 1.997e+03, 1.300e+02, 8.000e+00, 1.000e+01, 0.000e+00],
       [3.200e+01, 1.997e+03, 2.400e+02, 2.000e+00, 1.900e+01, 6.000e+00],
       [3.300e+01, 2.000e+03, 1.750e+02, 6.000e+00, 8.000e+00, 1.000e+00],
       [3.400e+01, 2.000e+03, 2.490e+02, 6.000e+00, 1.700e+01, 4.000e+00],
       [3.500e+01, 2.000e+03, 2.050e+02, 1.000e+01, 1.800e+01, 2.000e+00],
       [3.600e+01, 2.000e+03, 8.700e+01, 1.000e+01, 7.000e+00, 0.000e+00],
       [3.700e+01, 2.000e+03, 2.510e+02, 1.000e+01, 1.900e+01, 4.000e+00],
       [3.800e+01, 2.000e+03, 1.930e+02, 3.000e+00, 1.400e+01, 3.000e+00],
       [3.900e+01, 2.000e+03, 2.770e+02, 4.000e+00, 1.600e+01, 6.000e+00],
       [4.000e+01, 2.004e+03, 2.210e+02, 9.000e+00, 1.500e+01, 0.000e+00],
       [4.100e+01, 2.004e+03, 2.600e+02, 6.000e+00, 1.700e+01, 1.000e+00]
```

y = df_numeric.iloc[:,6]

y

0 21.0

1 14.0

2 17.0

3 14.0

4 9.0

……..

109 1.0

110 8.0

111 7.0

112 9.0

113 7.0

Name: fi_extra, Length: 114, dtype: float64

#5.TRAIN AND TEST VARIABLES

#sklearn.model_seletion - package , train_test_split - library

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state = 0)
#whatever data splitting/data allocation happens to the x_train,x_test,y_train,y_test vari
#By default the training variables get 75% and testing variables get 25%

print(x.shape) #114 rows and 28 columns
print(x_train.shape) #114 rows and columns (75%)
print(y_test.shape) # 114 rows and 28 cos(25%)
```

(114, 6)

(85, 6)

(29,)

```
print(y.shape) #114 rows and 28 cos
print(y_train.shape) #114 rows and 28 cols (75%)
print(y_test.shape) # 114 rows and 28 cols (25%)
```

(114,)

(85,)

(29,)

#6.SCALING or NORMALISATION -DONE ONLY FOR INPUTS

```
from sklearn.preprocessing import MinMaxScaler scaler = MinMaxScaler
scaler = MinMaxScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.fit_transform(x_test)
#7.RUN a CLASSIFIER/REGRESSOR/CLUSTER
from sklearn.linear_model import LinearRegression
```

```
model = LinearRegression()

#8.MODEL FITTING

model.fit(x_train,y_train)

LinearRegression()

#9.PREDICT THE OUTPUT

y_pred = model.predict(x_test)# By taking the input testing data, we predict the output

y_pred # PREDICTED VALUES

array([19.20388763, 11.74969049, 7.97064147, 16.22057976, 14.96798661, 13.86047521,
4.04739576, 11.35167488, 5.95182177, 11.66279533, 15.61103646, 9.74944617, 13.46438146,
10.69219968, 2.40998563, 2.08202763, 17.18781815, 12.18514381, 15.81112326, 11.79051201,
18.31871304, 4.35290049, 17.39604029, 12.82253352, 15.5967685 , 11.23292956, 11.81577119,
4.11380785, 5.55326804])

y_test # ACTUAL VALUES

33 17.0

10 15.0

90 10.0

7 7.0

24 9.0

73 14.0

113 7.0

22 17.0

94 4.0

2 17.0

48 38.0

89 6.0

51 32.0

71 18.0

105 4.0

93 14.0

59 3.0

66 21.0

16 20.0
```

13 12.0

68 24.0

106 7.0

26 18.0

50 37.0

82 11.0

8 13.0

30 19.0

102 4.0

91 5.0

Name: fi_extra, dtype: float64

print(x_train[10]) # these are scaled/normalised values

[0.67857143 0.82352941 1. 0.4 0.97142857 0.36363636]

#INDIVIDUAL PREDICTION

model.predict([x_train[10]])

array([12.96713636]

MAJOR PROJECT-2

► Create any of the Image Processing Projects using Numpy and OpenCV.(Projects done in the class are not accepted) (One can use the haarcasacde models if necessary)

CODE :

```
import cv2

import numpy as np

from tkinter.filedialog import *

 photo = askopenfilename()

img = cv2.imread(photo)

grey = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

grey = cv2.medianBlur(grey, 5)

edges = cv2.adaptiveThreshold(grey, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 9,9)

#cartoonize

color = cv2.bilateralFilter(img, 9, 250, 250)

cartoon = cv2.bitwise_and(color, color, mask = edges)

cv2.imshow("Image", img)

cv2.imshow("Cartoon", cartoon)

#save

cv2.imwrite("cartoon.jpg", cartoon)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

OUTPUT :

Github Link : https://github.com/Aruna-vattikota/Rinex-Ml.git