Assignment 1

Heart.csv

```
# from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns

df = pd.read_csv('Heart.csv')
df.isnull().sum()
df.info()
df.count()
df.types
df==0
df[df==0].count()
(df==0).sum()
np.mean(df['Age'])    or  df['Age'].mean()
df.columns
data = df[['Age', 'Sex', 'ChestPain', 'RestBP', 'Chol']]
train, test = train_test_split(data, test_size=0.25, random_state=1)

actual = np.concatenate((np.ones(45), np.zeros(450), np.ones(5)))
actual
predicted = np.concatenate((np.ones(100), np.zeros(400)))
predicted

from sklearn.metrics import ConfusionMatrixDisplay
ConfusionMatrixDisplay.from_predictions(actual, predicted)

from sklearn.metrics import classification_report
                            accuracy_score

print(classification_report(actual, predicted))
accuracy_score(actual, predicted)
```

```
num
pan
seaborn
sklearn  linearReg
sklearn.model_selection  train_test_split
df=pd.read

test, train = train_test_split
    (data, test_size=0.25, rs=1

actual = np.concatenate((np.ones(45),
                         np.zeros(400), np.
                                        5))
actual

predicted = np.concatenate((np.ones(100),
                            np.zeros(400)
                           ))
predicted

import ConfusionMatrixDisplay
classification_report
& accuracy_score
print(c_r(actual, pred))
```

Assignment 2 Temperatures

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('temperature.csv')
df.head()

x = df['YEAR']
y = df['ANNUAL']

plt.title('Temp Plot of IND')
plt.xlabel('Year')
plt.ylabel('Annual Avg Temp')
plt.scatter(x,y)

x.shape
x = x.values
x = x.reshape(117,1)
x.shape

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()

regressor.fit(x,y)
regressor.coef_
regressor.intercept_

regressor.predict([[2024]])
predicted = regressor.predict(x)
predicted

abs(y - predicted)
np.mean(abs(y-predicted))

from sklearn.metrics import mean_absolute_error
mean_absolute_error(y, predicted)

from sklearn.metrics import mean_squared_error
"  "  "  "  "  r2_score

mean_squared_error(y,predicted)
r2_score(y, predicted)
```

```
pandas
matplot
seaborn

df=pd.read_csv('  ')

x = df['YEAR']
y = df['ANNUAL']

Plot

plt.title
plt.xlabel
ylabel
plt.scatter

Reshape

x = x.values
x = x.reshape(117,1)

from sklearn.linear_model import LR
regressor = LinearRegression()
regressor.fit(x,y)
regressor.coef_
.intercept_

regressor.predict([[2024]])
predicted = regressor.predict(x)
predicted

from sklearn import mean_absolute_error
mean_absolute_error(y, predicted)
same for
mean_squared_error
& r2_score
r2_score(y, predicted)

Plot - next pg.

alternative to find
accuracy score
> regressor.score(x,y)

(r2 means Rsquare)

sns.regplot(
```

sns.regplot(x='YEAR', y='ANNUAL', data=df) ✓ - for plotting

OR

plt.title('')
plt.xlabel('Year')
ylabel('Annual')
plt.scatter(x, y, label='actual', color='r')
plt.plot(x, predicted, label='predicted', color='g')
plt.legend()

Assignment 3 GRE Admission

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv('  ')
df = df.drop('Serial No', axis = 1)
df.shape
from sklearn.preprocessing import Binarizer
bi = Binarizer(threshold = 0.75)
df["Chance of Admit"] = bi.fit_transform(df[['Chance of Admit']])
df.head()
df.columns
x = df[['GRE Score', 'TOEFL Score', 'Rating', 'SOP', 'LOR', 'CGPA']]
y = df['Chance of Admit']
y = y.astype('int')
sns.countplot(x=y)
y.value_counts()
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 1)

from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LogisticRegression

model_dt = DecisionTreeRegressor(random_state = 1)
    rf = R   F  R    (''    ''   = '')
    lr = LogisticRegression(random_state=1, solver = 'lbfgs', max_iterator = 1000)

model_rf.fit(x_train, y_train)
  '' lr '' ( '' , '' )

y_pred_df = model_dt.predict(x_test)
''  ''  = ''  rf. ''  ''
''  ''  = ''  lr. ''  ''
```

```
y_pred_rf = y_pred_rf.reshape(1,-1)
y_pred_rf = bi_fit_transform(y_pred_rf)
y_pred_rf = y_pred_rf.reshape(125)

from sklearn.metrics import ConfusionMatrixDisplay, accuracy_score, classification_report

# Decision Tree

ConfusionMatrixDisplay.from_predictions(y_test, y_pred_dt)
                                                  lr
                                                  rf
plt.title('Decision Tree')
plt.show()  L  R
            R  F
print(f"Accuracy is {accuracy_score(y_test, y_pred_dt)}")
                                                  lr
                                                  rf
print(classification_report(y_test, y_pred_dt))
                            y_pred_lr
                            y_pred_rf
```

Clustering

```
import pandas as pd
import sklearn
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from google.colab import files
uploaded = files.upload()
import io
df = pd.read_csv(io.BytesIO(uploaded['Mall_Customers.csv']))
df

df.columns

x=df.iloc[:,[3,4]].values
x

from sklearn.cluster import kMeans
wcss_list = []

for i in range(1,11):
    kmeans = kMeans(n_clusters=i, init = 'k-means++')
    kmeans.fit(x)
    wcss_list.append(kmeans.inertia_)

plt.plot(range(1,11), wcss_list)
plt.title('The elbow method graph')
plt.xlabel('Number of clusters')
plt.ylabel('wcss_lists')
plt.show()

wcss_list
```