Lab Activity - 1

1. Implement a Linear Regression model in Machine Learning and fit the model to predict total vaccination by 15-Mar-2021 for India country.

Using the `country_vaccinatins` dataset.

```
import pandas as pd
from datetime import datetime
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_style("whitegrid")
```

```
covid_df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Datasets/co
print(covid_df.shape)
covid_df.head()
```

```
(4435, 15)
```

	country	iso_code	date	total_vaccinations	people_vaccinated	people_full
0	Albania	ALB	2021-01-10	0.0	0.0	
1	Albania	ALB	2021-01-11	NaN	NaN	
2	Albania	ALB	2021-01-12	128.0	128.0	
3	Albania	ALB	2021-01-13	188.0	188.0	
4	Albania	ALB	2021-01-14	266.0	266.0	

```
covid_df.isnull().sum()
```

```
country                                   0
iso_code                                304
```

```
date                                      0
total_vaccinations                     1519
people_vaccinated                      1952
people_fully_vaccinated                2773
daily_vaccinations_raw                 1968
daily_vaccinations                      154
total_vaccinations_per_hundred         1519
people_vaccinated_per_hundred          1952
people_fully_vaccinated_per_hundred    2773
daily_vaccinations_per_million          154
vaccines                                  0
source_name                               0
source_website                            0
dtype: int64
```

```
covid_df['date'].min(), covid_df['date'].max()
```

```
('2020-12-08', '2021-02-27')
```

```
covid_df['country'].value_counts()
```

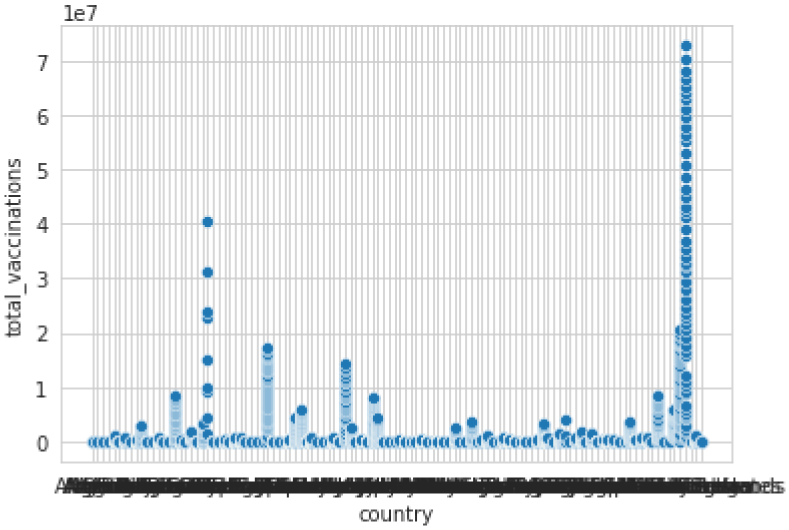
```
Lithuania         82
Scotland          76
United Kingdom    76
England           76
Wales             76
                  ..
Senegal            5
South Korea        3
Ukraine            3
Saint Helena       1
Greenland          1
Name: country, Length: 112, dtype: int64
```

```
covid_req_data = covid_df.loc[:, ['country', 'date', 'total_vaccinations']]
covid_req_data['date'] = pd.to_datetime(covid_req_data['date'])
covid_req_data.head(10)
```

	country	date	total_vaccinations
0	Albania	2021-01-10	0.0
1	Albania	2021-01-11	NaN

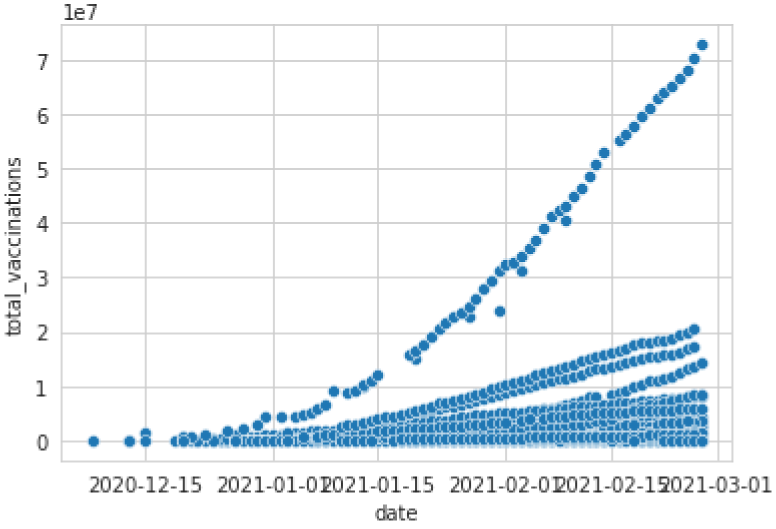
```
sns.scatterplot(x='country', y='total_vaccinations', data=covid_req_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5ae9ed5210>
```



```
sns.scatterplot(x='date', y='total_vaccinations', data=covid_req_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5ae9d03a90>
```



```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()


x1 = le.fit_transform(covid_req_data['country'])
x2 = le.fit_transform(covid_req_data['date'])
X = np.array([x1, x2]).T
covid_req_data['total_vaccinations'].fillna(0, inplace=True)
```

```
covid_req_data['total_vaccinations'].fillna(0, inplace=True)
y = covid_req_data['total_vaccinations'].values


X = covid_req_data.groupby(['date']).sum()
X['date_diff']=X['total_vaccinations']
count=0
for index, row in X.iterrows():
    row['date_diff']=count
    count+=1


x = X[['date_diff']]
y = X['total_vaccinations']


from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score


X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.33, 

rg = LinearRegression()
rg.fit(X_train, y_train)
y_pred = rg.predict(X_test)

print(f'Correlation Score: {rg.score(X_test, y_test)} \nMSE: {mean_squared_
```

```
    Correlation Score: 0.8332154888981802
    MSE: 864569548661889.8
```

2. Implement a Multiple Regression model in Machine Learning to fit the model. You can assume features / independent variables and dependent variable.

Use `winequality-red.csv` dataset.

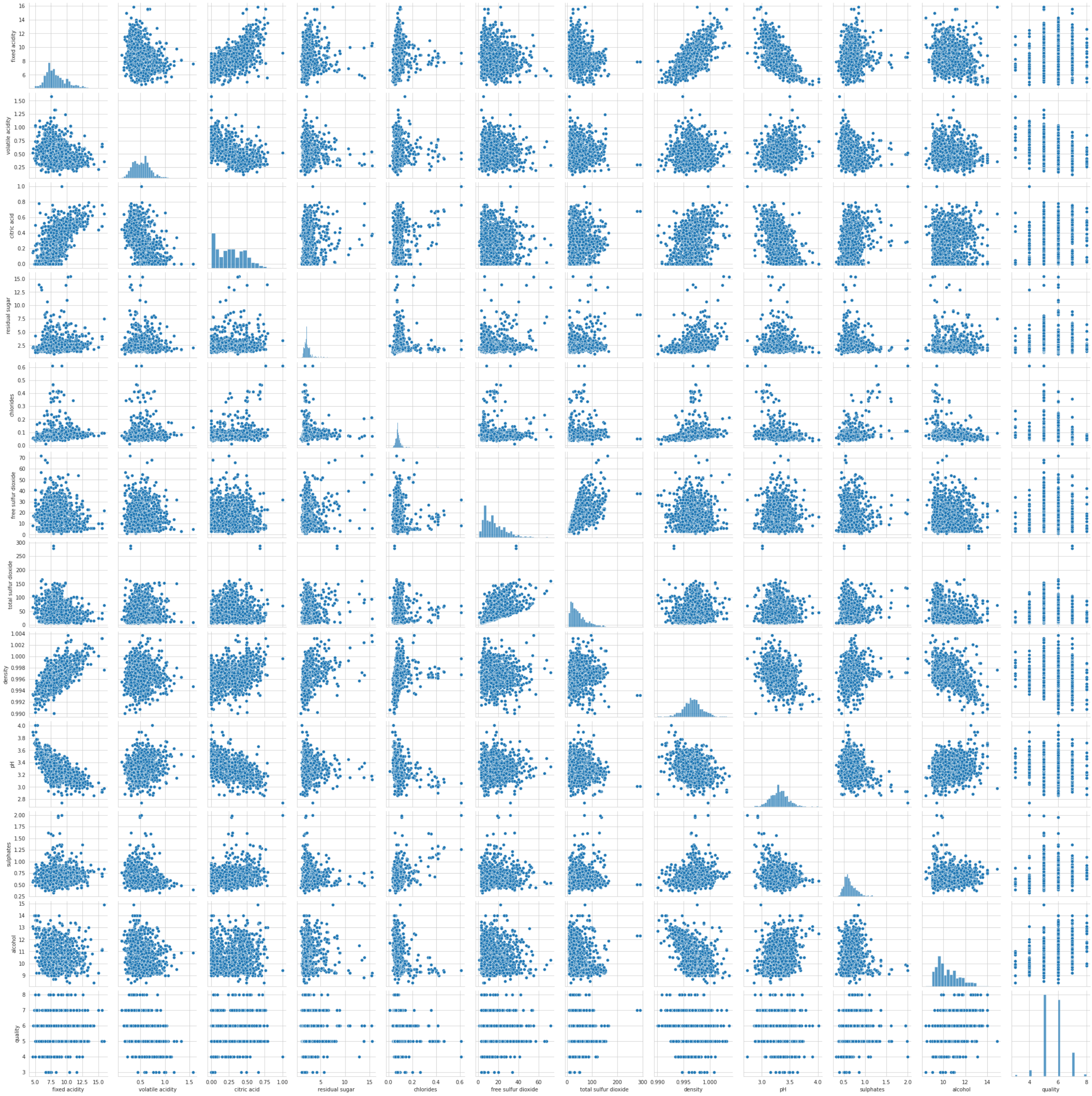
```
wine = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Datasets/winequa
print(wine.shape)
wine.head()
```

```
(1599, 12)
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.

```
sns.pairplot(wine)
```

```
<seaborn.axisgrid.PairGrid at 0x7f5ae8530f90>
```



```
X = wine.iloc[:, :-1].values
y = wine.loc[:, ['quality']].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
```

Multiple Linear Regression

```
from sklearn.linear_model import LinearRegression
```

```
rg = LinearRegression()

rg.fit(X_train, y_train)
y_pred = np.round(rg.predict(X_test))

print(f'Correlation Score: {r2_score(y_test, y_pred)} \nAccuracy Score: {a
```

```
    Correlation Score: 0.29698656219105857
    Accuracy Score: 0.6025
    MSE: 0.48
```

Polynomial Regression

```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 2)
x_poly = poly_reg.fit_transform(X_train)
linear_reg = LinearRegression()
linear_reg.fit(x_poly, y_train)
y_pred = np.round(linear_reg.predict(poly_reg.transform(X_test)))

print(f'Correlation Score: {r2_score(y_test, y_pred)} \nAccuracy Score: {a
```

```
    Correlation Score: 0.10658708945113693
    Accuracy Score: 0.6075
    MSE: 0.61
```

4. Implement a Decision Tree model in Machine Learning and fit the model towards the regression to predict housing price.

Use `melb_data.csv` dataset.

```
house = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Datasets/melb_
print(house.shape)
house.head()
```

```
(13580, 21)
```

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance
0	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5
1	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5
2	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5

```
house.isnull().sum()
```

```
Suburb              0
Address             0
Rooms               0
Type                0
Price               0
Method              0
SellerG             0
Date                0
Distance            0
Postcode            0
Bedroom2            0
Bathroom            0
Car                62
Landsize            0
BuildingArea     6450
YearBuilt        5375
CouncilArea      1369
Lattitude           0
Longtitude          0
Regionname          0
Propertycount       0
dtype: int64
```

```
values = {'BuildingArea': house['BuildingArea'].mean(), 'YearBuilt': house
house.fillna(value=values, inplace=True)


house.dropna(inplace=True)
house.drop(['Date'], axis=1, inplace=True)


objList = house.select_dtypes(include = "object").columns
print (objList)
```

```
Index(['Suburb', 'Address', 'Type', 'Method', 'SellerG', 'CouncilArea',
       'Regionname'],
      dtype='object')
```

```
#Label Encoding for object to numeric conversion
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

for feat in objList:
    house[feat] = le.fit_transform(house[feat].astype(str))

house.head(10)
```

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Distance	Postcode
0	0	2817	2	0	1480000.0	1	41	2.5	3067.0
1	0	8590	2	0	1035000.0	1	41	2.5	3067.0
2	0	843	3	0	1465000.0	3	41	2.5	3067.0
3	0	11974	3	0	850000.0	0	41	2.5	3067.0
4	0	1717	4	0	1600000.0	4	202	2.5	3067.0
5	0	4442	2	0	941000.0	1	198	2.5	3067.0
6	0	4384	3	0	1876000.0	1	202	2.5	3067.0
7	0	3398	2	0	1636000.0	1	202	2.5	3067.0
8	0	2251	1	2	300000.0	1	41	2.5	3067.0
9	0	3183	2	0	1097000.0	1	41	2.5	3067.0

```
X = house.drop(['Price'], axis=1)
y = house.loc[:, ['Price']]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
```

```
from sklearn.tree import DecisionTreeRegressor
dr = DecisionTreeRegressor()

dr.fit(X_train, y_train)
y_pred = np.round(dr.predict(X_test))

print(f'Correlation Score: {r2_score(y_test, y_pred)} \nMSE: {mean_squared
```

```
Correlation Score: 0.5482045701918146
MSE: 187114525545.24075
```

5. Implement a Logistic Regression model in Machine Learning and fit the model to predict heart rate of a person based on age and BMI.

Use `Framingham.csv` dataset.

```
data = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Datasets/framin
print(data.shape)
data.head()
```

```
(4240, 16)
   male  age  education  currentSmoker  cigsPerDay  BPMeds  prevalentStroke  pr
0     1   39        4.0              0         0.0     0.0                0
1     0   46        2.0              0         0.0     0.0                0
2     1   48        1.0              1        20.0     0.0                0
3     0   61        3.0              1        30.0     0.0                0
4     0   46        3.0              1        23.0     0.0                0
```

```
data["glucose"].fillna((data["glucose"].mode())[0], inplace=True)
data.dropna(inplace=True)
data.isnull().sum()
```

```
male               0
age                0
education          0
currentSmoker      0
cigsPerDay         0
BPMeds             0
prevalentStroke    0
prevalentHyp       0
diabetes           0
totChol            0
sysBP              0
diaBP              0
BMI                0
heartRate          0
glucose            0
TenYearCHD         0
dtype: int64
```

```
target1=data[data['TenYearCHD']==1]
target0=data[data['TenYearCHD']==0]

target1=resample(target1,replace=True,n_samples=len(target0),random_state=

target=pd.concat([target0,target1])

target['TenYearCHD'].value_counts()

data=target
```

```
data=target
np.shape(data)
```