Exercise-7

PRABUDDHIRAJ YADAV 24BCE10988

Code -

Exercise-7 24BCE10988.py

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import accuracy_score, mean_squared_error, r2_score

# Load data
df = pd.read_csv('US COVID dataset.csv')

# Convert 'date' to dummy variables
dummies = pd.get_dummies(df, columns=['date'], dtype='int64')
data1 = dummies.dropna()

# 1. Analysis of all parameters
X = data1[['death', 'deathIncrease', 'inIcuCumulative', 'inIcuCurrently', 'hospitalizedIncrease',
           'hospitalizedCurrently', 'hospitalizedCumulative', 'negative', 'negativeIncrease',
           'onVentilatorCumulative', 'onVentilatorCurrently', 'positive', 'positiveIncrease',
           'totalTestResultsIncrease']]
Y = data1['totalTestResults']

# Create figure for plotting all graphs
plt.figure(figsize=(10, 6))
for col in X.columns:
    sns.scatterplot(x=Y, y=X[col], label=col)
plt.title("All Parameters vs Total Test Results")
plt.xlabel("Total Test Results")
plt.ylabel("Features (All X Parameters)")
plt.legend()
plt.show()

# Perform train-test split without fixing the random state
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)  # Random split each time

# Train and evaluate the regression model
regr = LinearRegression()
regr.fit(X_train, Y_train)
print("Linear Regression score of all parameters: ", regr.score(X_test, Y_test))

# Train and evaluate the KNN Regression model
knn_reg = KNeighborsRegressor(n_neighbors=5)
knn_reg.fit(X_train, Y_train)
knn_pred = knn_reg.predict(X_test)
knn_r2 = r2_score(Y_test, knn_pred)
```

Exercise-7 24BCE10988.py

```
knn_pred = knn_reg.predict(X_test)
knn_r2 = r2_score(Y_test, knn_pred)
print("KNN Regression R² score (all parameters): ", knn_r2)

# Logistic Regression (Binary Classification)
threshold = Y.median()
Y_binary = (Y > threshold).astype(int)

# Check if both classes (0 and 1) are present in the data before fitting Logistic Regression
if Y_binary.nunique() > 1:  # Only proceed if there are both 0 and 1 in the data
    X_train_binary, X_test_binary, Y_train_binary, Y_test_binary = train_test_split(X, Y_binary, test_size=0.2)  # Random split each time
    log_reg = LogisticRegression(max_iter=10000)
    log_reg.fit(X_train_binary, Y_train_binary)
    log_reg_pred = log_reg.predict(X_test_binary)
    log_reg_accuracy = accuracy_score(Y_test_binary, log_reg_pred)
    print("Logistic Regression Accuracy (all parameters): ", log_reg_accuracy)
else:
    print("Logistic Regression: Only one class in the target variable, skipping model fitting.")

# 2. Analysis of Increasing Parameters
X1 = data1[['deathIncrease', 'hospitalizedIncrease', 'negativeIncrease', 'positiveIncrease','totalTestResultsIncrease']]
Y1 = data1['totalTestResults']

# Plot for increasing parameters
plt.figure(figsize=(10, 6))
for col in X1.columns:
    sns.scatterplot(x=Y1, y=X1[col], label=col)
plt.title("Increasing Parameters vs Total Test Results")
plt.xlabel("Total Test Results")
plt.ylabel("Increasing Parameters (X1)")
plt.legend()
plt.show()

# Perform train-test split without fixing the random state
X1_train, X1_test, Y1_train, Y1_test = train_test_split(X1, Y1, test_size=0.2)  # Random split each time
regr.fit(X1_train, Y1_train)
print("Linear Regression score of increasing parameters: ", regr.score(X1_test, Y1_test))

# Train and evaluate the KNN Regression model
knn_reg.fit(X1_train, Y1_train)
knn_pred = knn_reg.predict(X1_test)
knn_r2 = r2_score(Y1_test, knn_pred)
print("KNN Regression R² score (increasing parameters): ", knn_r2)

```

Exercise-7 24BCE10988.py

```
knn_r2 = r2_score(Y1_test, knn_pred)
print("KNN Regression R² score (increasing parameters): ", knn_r2)

# Logistic Regression (Binary Classification)
Y1_binary = (Y1 > threshold).astype(int)

# Check if both classes (0 and 1) are present in the data before fitting Logistic Regression
if Y1_binary.nunique() > 1:  # Only proceed if there are both 0 and 1 in the data
    X1_train_binary, X1_test_binary, Y1_train_binary, Y1_test_binary = train_test_split(X1, Y1_binary, test_size=0.2)  # Random split each time
    log_reg.fit(X1_train_binary, Y1_train_binary)
    log_reg_pred = log_reg.predict(X1_test_binary)
    log_reg_accuracy = accuracy_score(Y1_test_binary, log_reg_pred)
    print("Logistic Regression Accuracy (increasing parameters): ", log_reg_accuracy)
else:
    print("Logistic Regression: Only one class in the target variable, skipping model fitting.")

# 3. Analysis of Positive and Negative cases
X2 = data1[['positive', 'negative']]
Y2 = data1['totalTestResults']

# Plot for positive and negative cases
plt.figure(figsize=(10, 6))
for col in X2.columns:
    sns.scatterplot(x=Y2, y=X2[col], label=col)
plt.title("Positive and Negative Cases vs Total Test Results")
plt.xlabel("Total Test Results")
plt.ylabel("Positive / Negative Cases")
plt.legend()
plt.show()

# Perform train-test split without fixing the random state
X2_train, X2_test, Y2_train, Y2_test = train_test_split(X2, Y2, test_size=0.2)  # Random split each time
regr.fit(X2_train, Y2_train)
print("Linear Regression score of Positive and Negative cases: ", regr.score(X2_test, Y2_test))

# Train and evaluate the KNN Regression model
knn_reg.fit(X2_train, Y2_train)
knn_pred = knn_reg.predict(X2_test)
knn_r2 = r2_score(Y2_test, knn_pred)
print("KNN Regression R² score (positive/negative cases): ", knn_r2)

# Logistic Regression (Binary Classification)
Y2_binary = (Y2 > threshold).astype(int)
```

Exercise-7 24BCE10988.py

```
# Logistic Regression (Binary Classification)
Y2_binary = (Y2 > threshold).astype(int)

# Check if both classes (0 and 1) are present in the data before fitting Logistic Regression
if Y2_binary.nunique() > 1:  # Only proceed if there are both 0 and 1 in the data
    X2_train_binary, X2_test_binary, Y2_train_binary, Y2_test_binary = train_test_split(X2, Y2_binary, test_size=0.2)  # Random split each time
    log_reg.fit(X2_train_binary, Y2_train_binary)
    log_reg_pred = log_reg.predict(X2_test_binary)
    log_reg_accuracy = accuracy_score(Y2_test_binary, log_reg_pred)
    print("Logistic Regression Accuracy (positive/negative cases): ", log_reg_accuracy)
else:
    print("Logistic Regression: Only one class in the target variable, skipping model fitting.")

# 4. Analysis of Hospitalized people
X3 = data1[['inIcuCumulative', 'onVentilatorCumulative']]
Y3 = data1['hospitalizedCumulative']

# Plot for hospitalized people
plt.figure(figsize=(10, 6))
for col in X3.columns:
    sns.scatterplot(x=Y3, y=X3[col], label=col)
plt.title("Hospitalized People (ICU and Ventilator) vs Hospitalized Cumulative")
plt.xlabel("Hospitalized Cumulative")
plt.ylabel("ICU / Ventilator Cumulative")
plt.legend()
plt.show()

# Perform train-test split without fixing the random state
X3_train, X3_test, Y3_train, Y3_test = train_test_split(X3, Y3, test_size=0.2)  # Random split each time
regr.fit(X3_train, Y3_train)
print("Linear Regression score of Hospitalized data: ", regr.score(X3_test, Y3_test))

# Train and evaluate the KNN Regression model
knn_reg.fit(X3_train, Y3_train)
knn_pred = knn_reg.predict(X3_test)
knn_r2 = r2_score(Y3_test, knn_pred)
print("KNN Regression R² score (hospitalized data): ", knn_r2)

# Logistic Regression (Binary Classification)
Y3_binary = (Y3 > threshold).astype(int)

# Check if both classes (0 and 1) are present in the data before fitting Logistic Regression
if Y3_binary.nunique() > 1:  # Only proceed if there are both 0 and 1 in the data
    X3_train_binary, X3_test_binary, Y3_train_binary, Y3_test_binary = train_test_split(X3, Y3_binary, test_size=0.2)  # Random split each time
    log_reg.fit(X3_train_binary, Y3_train_binary)
    log_reg_pred = log_reg.predict(X3_test_binary)
```

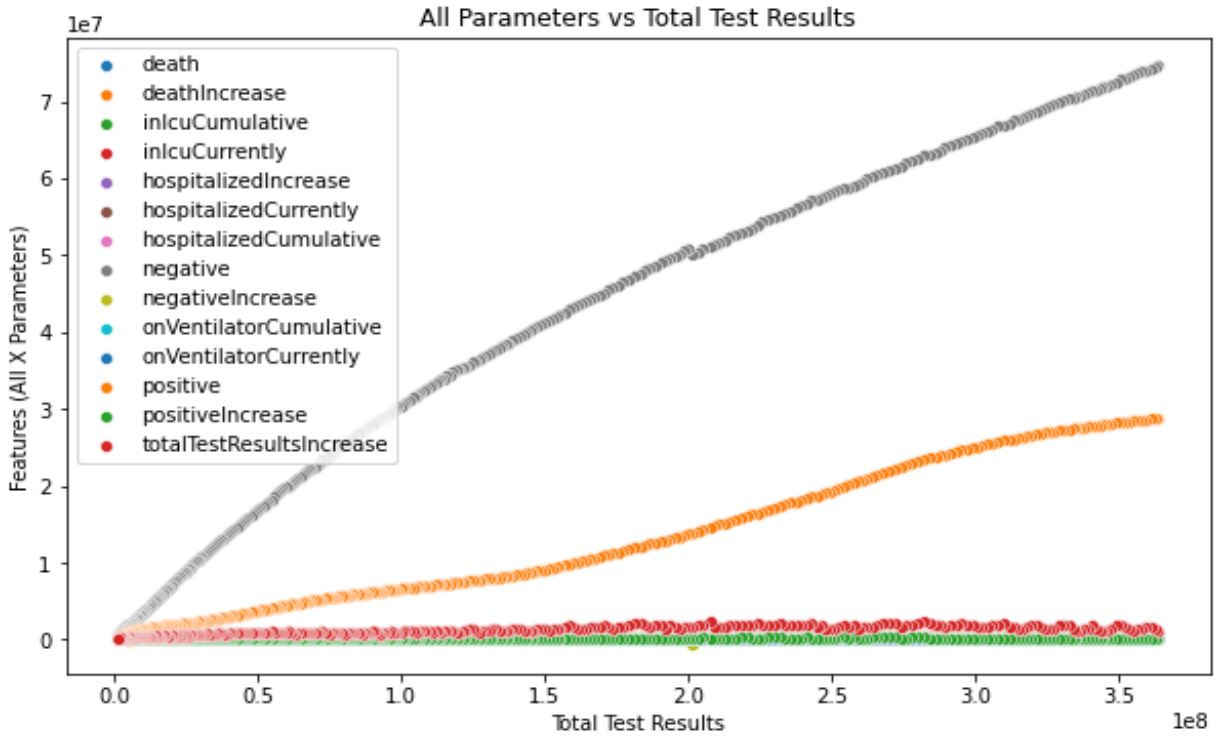
Exercise-7 24BCE10988.py

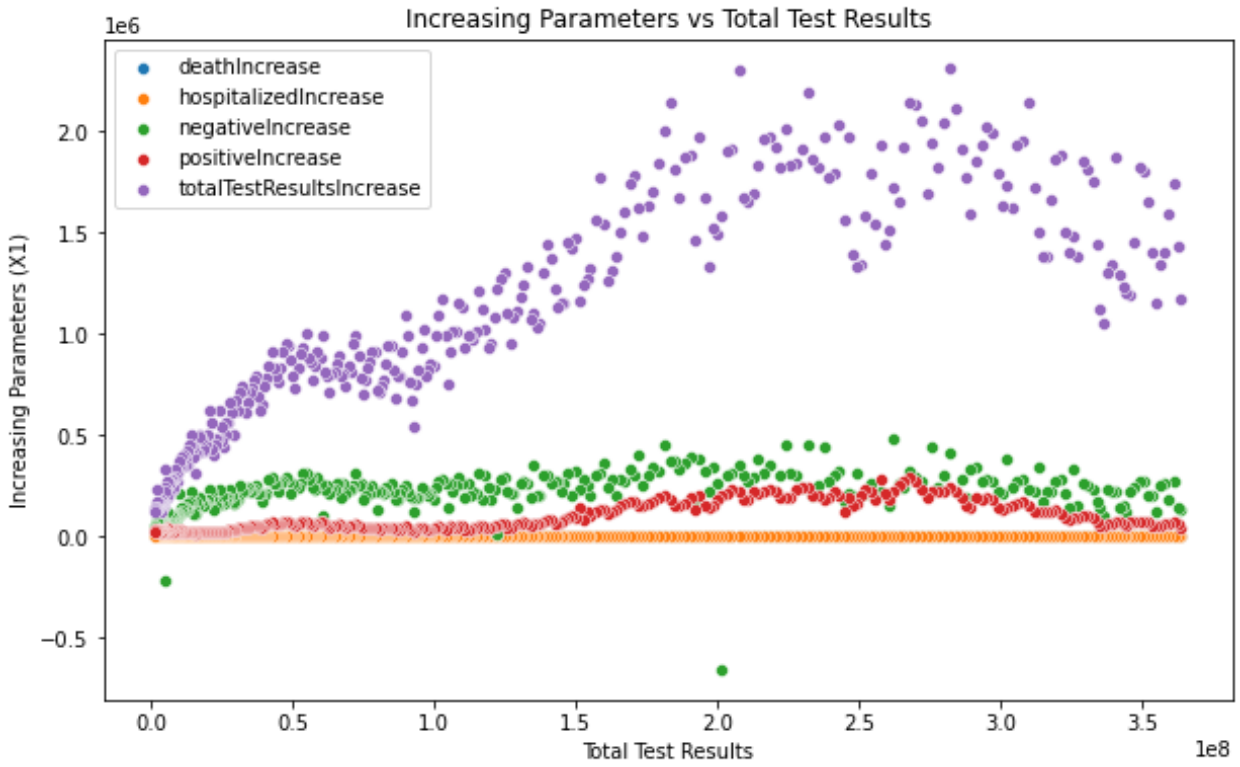
```
    X3_train_binary, X3_test_binary, Y3_train_binary, Y3_test_binary = train_test_split(X3, Y3_binary, test_size=0.2)  # Random split each time
    log_reg.fit(X3_train_binary, Y3_train_binary)
    log_reg_pred = log_reg.predict(X3_test_binary)
    log_reg_accuracy = accuracy_score(Y3_test_binary, log_reg_pred)
    print("Logistic Regression Accuracy (hospitalized data): ", log_reg_accuracy)
else:
    print("Logistic Regression: Only one class in the target variable, skipping model fitting.")

# 5. Analysis of Deaths
X4 = data1[['death']]
Y4 = data1['totalTestResults']

# Plot for deaths
plt.figure(figsize=(10, 6))
sns.scatterplot(x=Y4, y=X4['death'], label='Death vs Total Test Results')
plt.title("Deaths vs Total Test Results")
plt.xlabel("Total Test Results")
plt.ylabel("Deaths")
plt.legend()
plt.show()

# Perform train-test split without fixing the random state
X4_train, X4_test, Y4_train, Y4_test = train_test_split(X4, Y4, test_size=0.2)  # Random split each time
regr.fit(X4_train, Y4_train)
print("Linear Regression score of total deaths: ", regr.score(X4_test, Y4_test))

# Train and evaluate the KNN Regression model
knn_reg.fit(X4_train, Y4_train)
knn_pred = knn_reg.predict(X4_test)
knn_r2 = r2_score(Y4_test, knn_pred)
print("KNN Regression R² score (total deaths): ", knn_r2)

# Logistic Regression (Binary Classification)
Y4_binary = (Y4 > threshold).astype(int)

# Check if both classes (0 and 1) are present in the data before fitting Logistic Regression
if Y4_binary.nunique() > 1:  # Only proceed if there are both 0 and 1 in the data
    X4_train_binary, X4_test_binary, Y4_train_binary, Y4_test_binary = train_test_split(X4, Y4_binary, test_size=0.2)  # Random split each time
    log_reg.fit(X4_train_binary, Y4_train_binary)
    log_reg_pred = log_reg.predict(X4_test_binary)
    log_reg_accuracy = accuracy_score(Y4_test_binary, log_reg_pred)
    print("Logistic Regression Accuracy (total deaths): ", log_reg_accuracy)
else:
    print("Logistic Regression: Only one class in the target variable, skipping model fitting.")
```

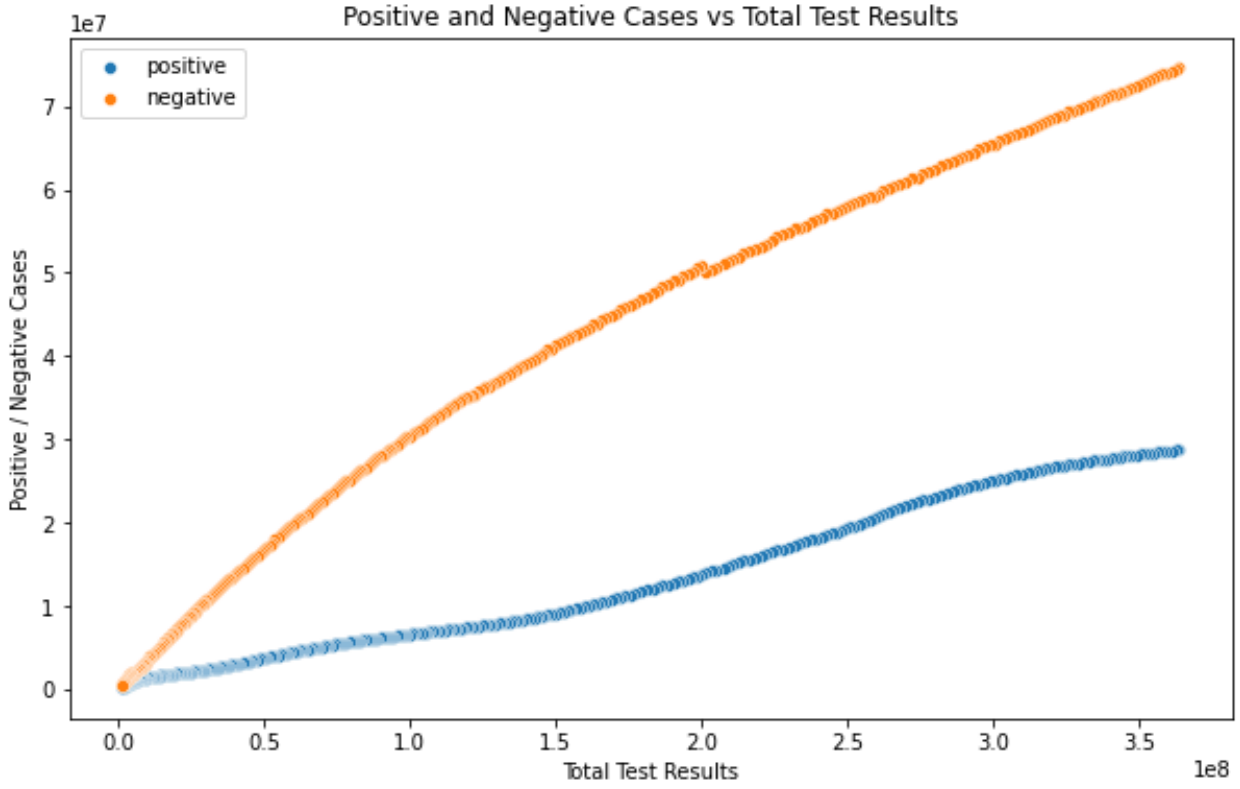
Output –

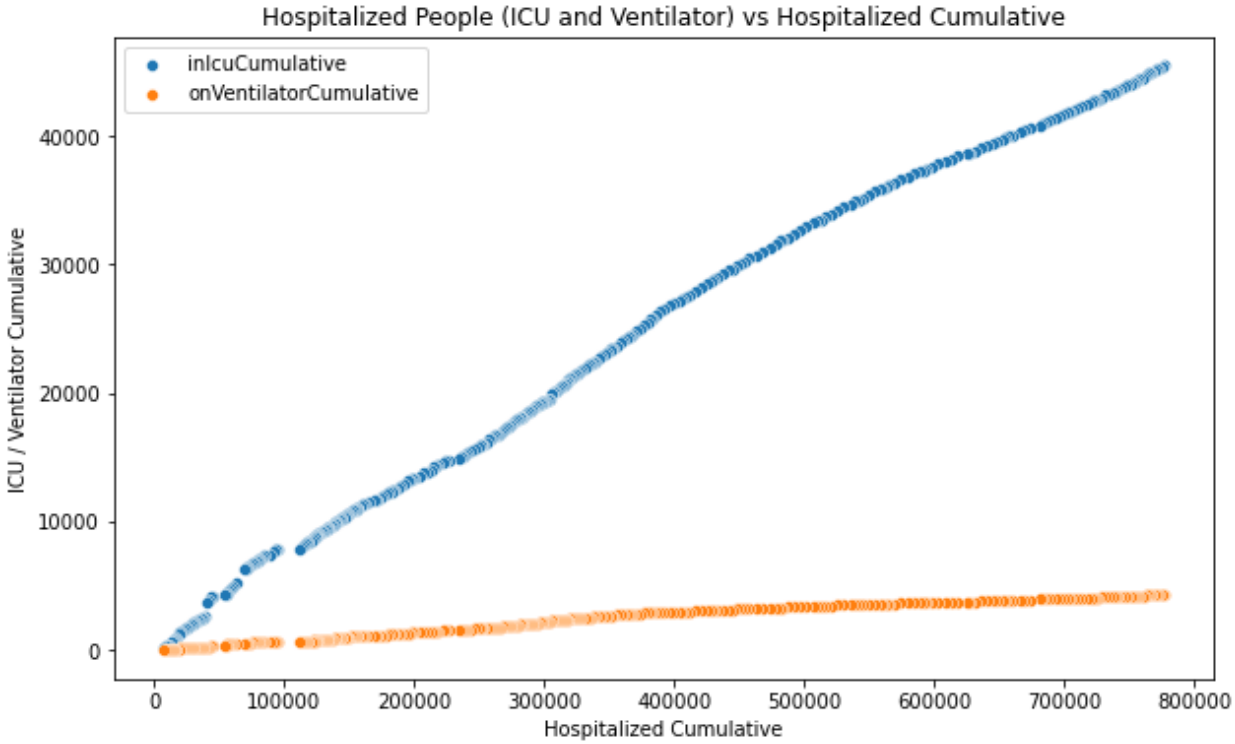
```
Console 1/A

In [13]: runfile('C:/Users/assus/.spyder-py3/Exercise-7 24BCE10988.py',
wdir='C:/Users/assus/.spyder-py3')
Linear Regression score of all parameters:  0.999797242116339
KNN Regression R² score (all parameters):  0.999945340918256
Logistic Regression Accuracy (all parameters):  0.9855072463768116
Linear Regression score of increasing parameters:  0.7520167944236643
KNN Regression R² score (increasing parameters):  0.8130271488935201
Logistic Regression Accuracy (increasing parameters):  0.9565217391304348
Linear Regression score of Positive and Negative cases:
0.9983779517168581
KNN Regression R² score (positive/negative cases):  0.999932733528919
Logistic Regression Accuracy (positive/negative cases):
0.4057971014492754
Linear Regression score of Hospitalized data:  0.9969272693269274
KNN Regression R² score (hospitalized data):  0.9998519296233882
Logistic Regression: Only one class in the target variable, skipping model
fitting.
Linear Regression score of total deaths:  0.9740109161706936
KNN Regression R² score (total deaths):  0.9997759214560676
Logistic Regression Accuracy (total deaths):  0.9855072463768116
```

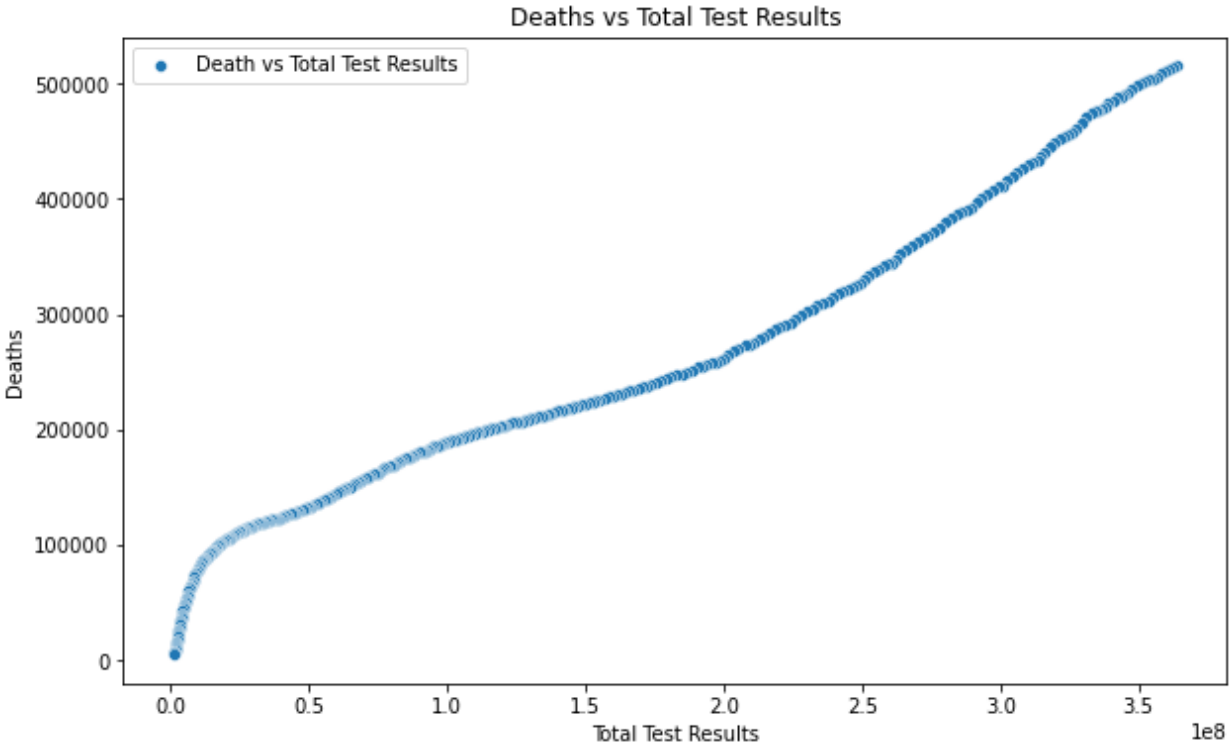
Graphs -




Increasing Parameters vs Total Test Results
1e6
deathIncrease
hospitalizedIncrease
negativeIncrease
positiveIncrease
totalTestResultsIncrease
Increasing Parameters (X1)
2.0
1.5
1.0
0.5
0.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Total Test Results
1e8


Positive and Negative Cases vs Total Test Results
1e7
positive
negative
Positive / Negative Cases
7
6
5
4
3
2
1
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Total Test Results
1e8


Hospitalized People (ICU and Ventilator) vs Hospitalized Cumulative
inIcuCumulative
onVentilatorCumulative
ICU / Ventilator Cumulative
Hospitalized Cumulative
0
10000
20000
30000
40000
0
100000
200000
300000
400000
500000
600000
700000
800000


Deaths vs Total Test Results
Death vs Total Test Results
Deaths
Total Test Results
1e8
0
100000
200000
300000
400000
500000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

Github Link-

https://github.com/Prabuddhiraj/US-COVID-dataset-Analysis.git