Step 1: Data

1. 34 columns each having 3047 data samples are included in the given dataset.

2. It will predict the target death rates.

3. Minimum and Maximum values in each respective row.

avgAnnCount	6.0
avgDeathsPerYear	3
TARGET_deathRate	59.7
incidenceRate	201.3
medIncome	22640
popEst2015	827
povertyPercent	3.2
studyPerCap	0.0
binnedInc	(34218.1, 37413.8]
MedianAge	22.3
MedianAgeMale	22.4
MedianAgeFemale	22.3
Geography	Abbeville County, South Carolina
AvgHouseholdSize	0.0221
PercentMarried	23.1
PctNoHS18_24	0.0
PctHS18_24	0.0
PctSomeCol18_24	7.1
PctBachDeg18_24	0.0
PctHS25_Over	7.5
PctBachDeg25_Over	2.5
PctEmployed16_Over	17.6
PctUnemployed16_Over	0.4
PctPrivateCoverage	22.3
PctPrivateCoverageAlone	15.7
PctEmpPrivCoverage	13.5
PctPublicCoverage	11.2
PctPublicCoverageAlone	2.6
PctWhite	10.199155
PctBlack	0.0
PctAsian	0.0
PctOtherRace	0.0
PctMarriedHouseholds	22.99249
BirthRate	0.0

Minimum Values

avgAnnCount	38150.0
avgDeathsPerYear	14010
TARGET_deathRate	362.8
incidenceRate	1206.9
medIncome	125635
popEst2015	10170292
povertyPercent	47.4
studyPerCap	9762.308998
binnedInc	[22640, 34218.1]
MedianAge	624.0
MedianAgeMale	64.7
MedianAgeFemale	65.7
Geography	Zavala County, Texas
AvgHouseholdSize	3.97
PercentMarried	72.5
PctNoHS18_24	64.1
PctHS18_24	72.5
PctSomeCol18_24	79.0
PctBachDeg18_24	51.8
PctHS25_Over	54.8
PctBachDeg25_Over	42.2
PctEmployed16_Over	80.1
PctUnemployed16_Over	29.4
PctPrivateCoverage	92.3
PctPrivateCoverageAlone	78.9
PctEmpPrivCoverage	70.7
PctPublicCoverage	65.1
PctPublicCoverageAlone	46.6
PctWhite	100.0
PctBlack	85.947799
PctAsian	42.619425
PctOtherRace	41.930251
PctMarriedHouseholds	78.075397
BirthRate	21.326165

Maximum Values

The minimum value in the entire dataset = 0.0

The maximum value in the entire dataset = 10170292.0

4. There are 34 columns, hence 34 features in each data sample.

5. Columns with missing values along with the total cells missing:

PctSomeCol18_24 → 2285, PctEmployed16_Over → 152, PctPrivateCoverageAlone → 609

6. The label of this dataset is "TARGET_deathRate."

7. 70% for training, 15% for validation, and 15% for testing.

8. I began by reading a CSV file called 'cancer_reg.csv' and stored its contents in a DataFrame named **df**.

Next, I needed to address missing values in the dataset. To do this, I calculated the sum of missing values for each row in **df** using **df.isnull().sum(axis=1)**. Then, I looped through the columns, a total of 34, to check for any missing values in each column. Whenever I found a column with missing values, I filled those gaps by replacing them with the mean value of that column using **df.iloc[:, i].fillna(df.iloc[:, i].mean(), inplace=True)**.

Following this, I performed label encoding on two specific columns, 'Geography' and 'binnedInc', using Scikit-Learn's **LabelEncoder**. This transformation converted the categorical data in those columns into numerical format.

To define the features (X) and target (y) for my machine learning model, I excluded the 'TARGET_deathRate' column from the DataFrame to create X. I set y to be the 'TARGET_deathRate' column.

Then, I applied a logarithmic transformation to both X and y using **np.log1p()**. This transformation helped me make the data more suitable for modeling, especially if the data had skewed distributions or large ranges.

For evaluating the model's performance, I split the data into three sets: a training set (70% of the data), a validation set (15%), and a testing set (15%). This was accomplished with the **train_test_split** function, ensuring that I had separate datasets for training and testing.

Standardization was the next step in my data preparation. I used StandardScaler to scale the feature data. First, I fitted the scaler to the training data, and then I applied the same scaler to transform the validation and testing data. Standardization is important for ensuring that all features have zero mean and unit variance, which can aid certain machine learning algorithms in performing better.

Lastly, I performed clipping on the predictions and validation data to prevent them from going below a specified minimum value. This was done using **np.clip()**, which constrained the values within a predefined range.

Step 2: Model

Model	MSE
Linear regression	0.0046
ANN-oneL-16	0.0049
ANN-twoL-32-8	0.0037
ANN-threeL-32-16-8	0.0033
ANN-fourL-32-16-8-4	0.0029

Step 3: Objective

Mean Squared Error (MSE) is the loss function I used to train my models.

Step 4: Optimization

I did some research about Adam optimizer on my own and hence have used Adam optimizer for my model to train which also helped me to increase my accuracy.

Step 5: Model Selection

Model	LR: 0.1 (100 epochs, 64 batch size)	LR: 0.01 (1000 epochs, 64 batch size)	LR: 0.001 (1000 epochs, 64 batch size)	LR: 0.0001 (1000 epochs, 64 batch size)
ANN-oneL-16	0.76	0.84	0.79	0.50
ANN-twoL-32-8	0.79	0.84	0.72	0.51
ANN-threeL-32-16-8	0.81	0.88	0.72	0.54
ANN-fourL-32-16-8-4	0.82	0.88	0.73	0.69

The best performing model was the ANN-fourL-32-16-8-4 model at 1000 epochs, 64 batch size and 0.01 learning rate which gave me a R-squared score of 0.88.

Step 6: Model Performance

For the R-squared score table (Step 5):

1. Higher learning rates (e.g., LR: 0.1) generally lead to better R-squared scores, but there's a trade-off with stability and convergence.
2. Increasing the number of epochs (e.g., 1000 epochs) tends to improve model performance, as reflected in the higher R-squared scores.
3. The choice of model architecture (e.g., the number of hidden layers and units) also affects performance, with more complex models (e.g., ANN-fourL-32-16-8-4) generally outperforming simpler ones (e.g., ANN-oneL-16).
4. Very low learning rates (e.g., LR: 0.0001) can lead to suboptimal model performance, as seen in the lower R-squared scores.

For the MSE table (Step 2):

1. The ANN models (Artificial Neural Networks) are performing better than the simple Linear Regression model in terms of MSE. This suggests that the neural networks are capturing more complex relationships within the data.
2. As you increase the depth of the neural network (going from one hidden layer to four hidden layers), the MSE decreases, indicating that deeper architectures can model the data more effectively.
3. Among the neural network models, "ANN-fourL-32-16-8-4" has the lowest MSE, suggesting that it is the best-performing model among the ones listed.
4. The specific choice of model and its architecture can have a significant impact on the quality of predictions, and it's important to select the model that best fits your dataset and problem.