Binary Classification: Predicting Next-Day Rainfall Using Logistic Regression

➢ Objective

The purpose of this project is to develop a binary classification model that predicts whether it will rain the next day using historical weather observations. Logistic regression is employed to estimate the probability of next-day rainfall. To illustrate both conceptual understanding and practical implementation, the project includes two versions of the model: one built from scratch without any machine learning libraries, and another using the Scikit-learn library.

➢ Dataset Description

- Source: Australia weather data (weatherAUS.csv) from Kaggle
- Number of rows/records: ~145,000
- Number of columns/attributes: 22
- Target Variable: RainTomorrow
- Key Features:
  - Humidity3pm
  - Sunshine
  - Pressure3pm
  - Cloud3pm
  - Pressure9am
  - WindGustSpeed

➢ Python Environment and Libraries

The implementation was carried out in Google Colab. The project utilizes several Python libraries, including:

- Scikit-learn
- NumPy
- Pandas
- Matplotlib
- Seaborn

These libraries were used for data preprocessing, exploratory analysis, visualization, and model development.

➢ Data Processing

- Data Extraction: The original CSV file (weatherAUS.csv) was loaded into a Pandas DataFrame to extract and manage feature values. NumPy was then utilized for matrix-based operations during further processing and model preparation.
- Data Partitioning
    - Part-01: Dataset split into training and validation sets using a 60:40 ratio.
    - Part-02: Same dataset used again, this time split into training and validation sets at a 70:30 ratio to evaluate the effect of different splits on model performance.

- Scaling: Feature values were normalized using min-max scaling to bring all numerical attributes to the range 0 to 1.

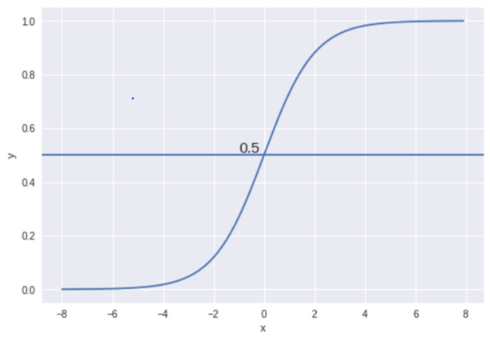
$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

- Imputation: Missing values were handled using a multivariate feature imputation approach, ensuring that relationships between features were preserved.
- Feature Selection: Important features were identified using a RandomForestClassifier, enabling the model to focus on the most relevant variables.
- Data Skew Correction: Skewness in numeric features was corrected using square root and logarithmic transformations, resulting in more symmetric distributions and improved model performance.

➢ Logistic Regression Implementation

- Hypothetical Function: Logistic Regression is used for binary classification tasks. In this implementation, we applied the sigmoid activation function to a linear combination of features to model the probability of the positive class. The hypothetical function $h(x)$is defined as:

$$z = W_1X_1 + W_2X_2 + W_3X_3 + W_4X_4 + W_5X_5 + W_6X_6 + W_7X_7 + W_8X_8 + b$$
$$h(x) = \text{sigmoid}(z) = \frac{1}{1+e^{-z}}$$



sigmoid function

Here, all weights were initialized to zero:

```
# Initializing weight array

wt = np.array([[0, 0, 0, 0, 0, 0, 0, 0, 0]])
```

The sigmoid function outputs values in the range [0,1], with $h(0) = 0.5$. We use a threshold of 0.5 to determine the predicted class:

- $Y = 1$ (rain) if $h(z) \geq 0.5$
- $Y = 0$ (not rain) if $h(z) < 0.5$

This simple approach allows the model to classify rainfall based on the computed probability.

- Cost Function and Gradient Descent: A cost function is created for our model, and the goal is to minimize the cost. The cost function for a single training example can be calculated as follows:

$$cost = \begin{cases} -log(h(x), & \text{if } y = 1 \\ -log(1 - h(x)), & \text{if } y = 0 \end{cases}$$

The cost for all training instances represented by J(θ) can be calculated by averaging the costs of all training samples.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^i log(h(x^i)) + (1 - y^i)log(1 - h(x^i))]$$

where *m* is the number of training samples.

Here, y represents the original output variable, and h represents the predicted output variable. Our objective is to keep costs as low as feasible. Now I must adjust the theta values such that the forecast is as near to the original output variable as possible. To reduce the cost function, I have used gradient descent. The gradient with respect to any parameter may be calculated as follows:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h(x^i) - y^i)\, x_j^i$$

Gradient Descent

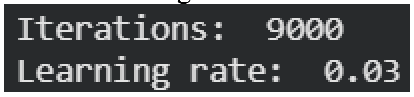
➢ Training the model

I have used 9000 iterations to train the model and minimize the cost function. In every iteration, a learning rate of 0.03 is used as a coefficient for the gradient descent to adjust the weight values. The number of iteration and learning rate may be adjusted to obtain a better value of cost function.

The adjustment of weight values in every iteration can be written as

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Theta is the weight values, and alpha is the learning rate.



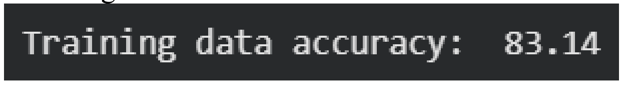
```
Iterations:  9000
Learning rate:  0.03
```

Gradient descent with above formula is used to converge at a local minimum for cost function against weight variables.

After n iteration for a given learning rate, the weight values are generated. With this, the training of the model is complete for the training dataset. Now, we will calculate the accuracy of the model. Accuracy is calculated as the percentage correct prediction of the model against actual values('Outcome').

Accuracy = Sum [| prediction – actual |] / total
Total = total number of instances (or rows)

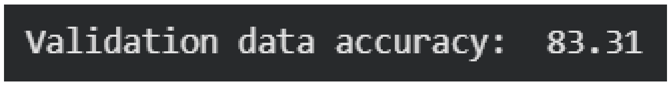
The accuracy achieved after training of the model is around 83.14 %.



```
Training data accuracy:  83.14
```

➢ Validation of the model

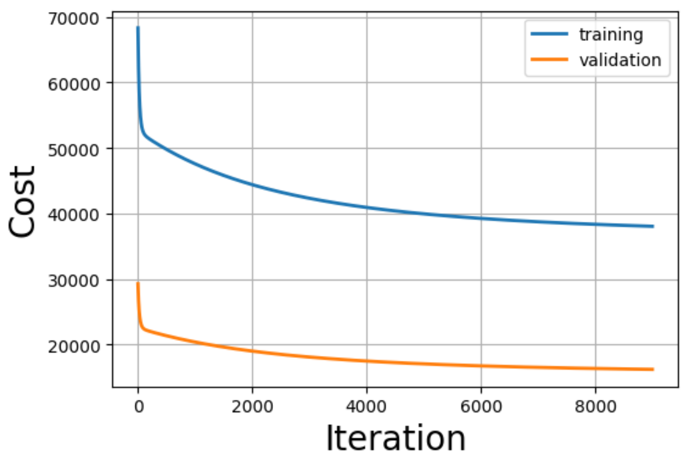
After completing the training of the model, I have validated the model with a validation dataset. The accuracy obtained for the validation dataset shows that our model is making a good prediction. The accuracy for the validation dataset achieved is around 83.31 %.



```
Validation data accuracy:  83.31
```

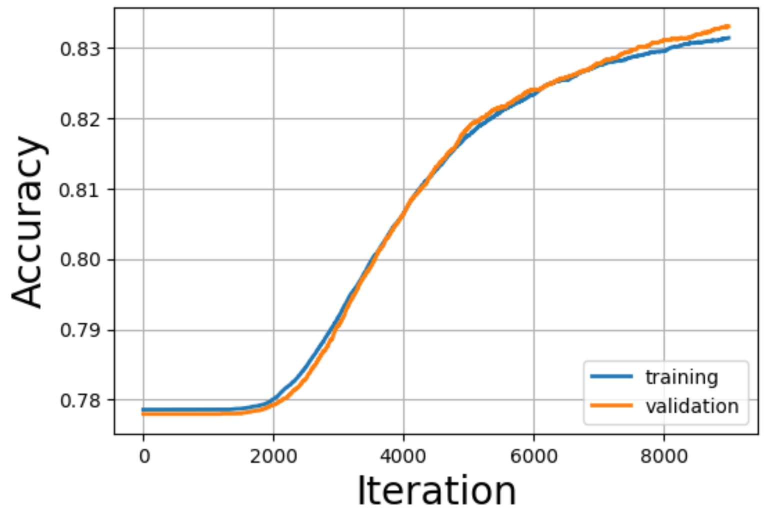
➢ The Plot of cost Function

Cost function comparison for training data and validation data set is below



- **The Plot of Accuracy**

  Accuracy *comparison* for training data and validation data set is below



- **Measuring the strength of model:**

  This machine learning model is making a good prediction of around 83%. Thus, this logistic regression model can be used for the rainfall prediction.