CSE343: Machine Learning

Assignment-1

Divyajeet Singh (2021529)

September 3, 2023

1 Section A (Theoretical)

(a) (2 marks)

If two variables exhibit a strong correlation with a third variable, does this necessarily imply that they will also display a high degree of correlation with each other? Provide a reasoned explanation, supported by an illustrative example.

Solution

No. When two variables exhibit a strong correlation with a third variable, they may or may not display a high degree of correlation with each other. This is because the correlation between two variables is a measure of the linear relationship between them. It does not imply that the two variables are dependent on each other.

As an example, consider two independent variables X_1 and X_2. Consider a variable W, which is a linear combination of X_1 and X_2, let's say

$$W = X_1 + X_2 \tag{1}$$

Then, X_1 and W are strongly correlated, and X_2 and W are also strongly correlated. However, X_1 and X_2 are not correlated at all.

(b) (2 marks)

What is the defining criteria(s) for a mathematical function to be categorized as a logistic function? Briefly explain which of the following functions are valid logistic functions: $\sinh(x)$, $\cosh(x)$, $\tanh(x)$, and $\text{signum}(x)$.

Solution

Mathematically, a function $f : \mathbb{R} \to \mathbb{R}$ is a logistic function if it is bounded, continuous, and can be written in the form

$$f(x) = \frac{A}{1 + e^{-B(x-x_0)}} + C \tag{2}$$

where the parameters are

- $A \in \mathbb{R}$: the curve's maximum asymptote when $C = 0$
- $B \in \mathbb{R}^+$: the logistic growth rate or steepness of the curve
- $x_0 \in \mathbb{R}$: the x-value of the sigmoid's midpoint
- $C \in \mathbb{R}$: a scalar to shift the curve vertically

This means that the function must be limited within its two horizontal asymptotes. Given the above constraints, the hyperbolic functions $\sinh(x)$ and $\cosh(x)$ are not valid logistic functions, simply because they are unbounded over $\mathbb{R}$.

$$\lim_{x\to\pm\infty}\sinh(x) = \lim_{x\to\pm\infty}\frac{e^x - e^{-x}}{2} \to \pm\infty$$
$$\lim_{x\to\pm\infty}\cosh(x) = \lim_{x\to\pm\infty}\frac{e^x + e^{-x}}{2} \to +\infty$$

The function $\tanh(x)$ can be written in the form of (2) with the assignment $A = 2$, $B = 2$, $x_0 = 0$, and $C = -1$. Moreover, $\tanh(x)$ is continuous and bounded over $\mathbb{R}$. Hence, it is a valid logistic function. A simple derivation is given below.

$$\begin{aligned}\tanh(x) &= \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \\ &= \frac{1-e^{-2x}}{1+e^{-2x}} = \frac{2}{1+e^{-2x}} - \frac{1+e^{-2x}}{1+e^{-2x}} \\ &= \frac{2}{1+e^{-2x}} - 1\end{aligned}$$

The function $\text{signum}(x)$ is not a valid logistic function, because it is not continuous over $\mathbb{R}$.

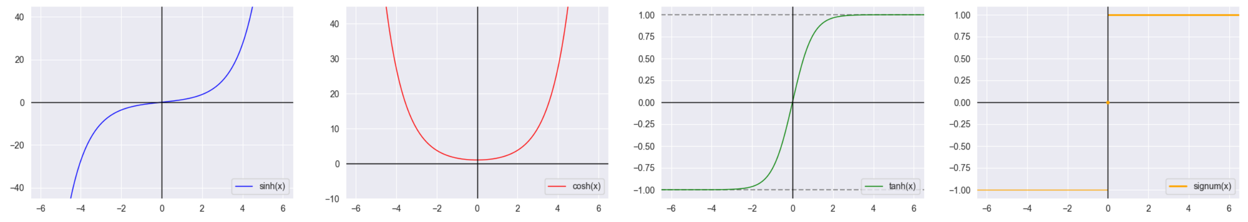

Figure 1: Plots of $\sinh(x)$, $\cosh(x)$, $\tanh(x)$, and $\text{signum}(x)$

(c) (2 marks)

Which validation technique is beneficial for very sparse datasets and why? Briefly explain how it is different from the traditional k-fold cross-validation technique.

Solution

Leave-one-out Cross-Validation (LooCV) is beneficial for sparse datasets. On a dataset with N samples, LooCV trains a model on $N-1$ samples and tests it on the remaining sample. This is repeated N times, selecting a different sample for testing each time. This way, N different models are trained and tested, and the average error is reported.

For sparse datasets with small number of samples, this is beneficial because it ensures that the models are trained on as large of a subset of the data as possible, which should ideally decrease both bias and variance.

Comparitively, in the k-fold Cross Validation technique, the dataset is divided into k folds of size $\frac{N}{k}$ each. k models are trained on k different subsets of size $\frac{N}{k}\cdot(k-1)$ and tested on the remaining fold, where $k \leq N$ and $k \in \mathbb{N}$. With $k = N$, this technique is equivalent to LooCV.

(d) (2 marks)

Find the coefficients of the least square regression line for a set of n data points (x_i, y_i) in slope-intercept form.

Solution

The Least Square Regression technique of linear regression assumes that the label for each data point is a linear combination of the features. In other words, the label y_i $(0 \le i \le n)$ for a data point (x_i, y_i) is given by

$$y_i = b_0 + b_1 x_i + \epsilon_i \tag{3}$$

where ϵ_i is the error, assumed to be randomly drawn from an unknown normal distribution $\epsilon \sim N[0, \sigma^2]$. Hence, the problem is to find parameters b_0 and b_1 that give the line that best fits the data.

Given n data points, this is an optimization problem to minimize the mean residual sum of squares.

$$\begin{align}
\min_{\mathbf{b}} J(\mathbf{b} := (b_0, b_1)) &= \min_{\mathbf{b}} \frac{1}{n} \sum_{i=1}^{n} \epsilon_i^2 \tag{4}\\
&= \min_{\mathbf{b}} \frac{1}{n} \sum_{i=1}^{n} (y_i - b_0 - b_1 x_i)^2 \tag{5}
\end{align}$$

This can be solved by finding the gradient of $J(\mathbf{b})$ and setting it to 0.

$$\begin{align}
\frac{\partial J(\mathbf{b})}{\partial b_0} = 0 &\implies \frac{2}{n} \sum_{i=1}^{n} (y_i - b_0 - b_1 x_i) \cdot (-1) = 0 \tag{6}\\
&\implies \frac{1}{n} \sum_{i=1}^{n} (b_0 + b_1 x_i - y_i) = 0 \tag{7}\\
&\implies b_0 + b_1 \cdot \frac{1}{n} \sum_{i=1}^{n} x_i - \frac{1}{n} \sum_{i=1}^{n} y_i = 0 \tag{8}\\
&\implies b_0 = \frac{1}{n} \sum_{i=1}^{n} y_i - b_1 \left(\frac{1}{n} \sum_{i=1}^{n} x_i \right) \tag{9}\\
&\implies b_0 = \bar{y} - b_1 \bar{x} \tag{10}
\end{align}$$

$$\begin{align}
\frac{\partial J(\mathbf{b})}{\partial b_1} = 0 &\implies \frac{2}{n} \sum_{i=1}^{n} (y_i - b_0 - b_1 x_i) \cdot (-x_i) = 0 \tag{11}\\
&\implies b_1 \sum_{i=1}^{n} x_i^2 + b_0 \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} x_i y_i = 0 \tag{12}\\
&\implies b_1 \sum_{i=1}^{n} x_i^2 + (\bar{y} - b_1 \bar{x}) \cdot n\bar{x} = \sum_{i=1}^{n} x_i y_i \tag{13}\\
&\implies b_1 \left(\sum_{i=1}^{n} x_i^2 - n\bar{x}^2 \right) = \sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y} \tag{14}\\
&\implies b_1 = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2} \tag{15}
\end{align}$$

So, the line $Y = b_0 + b_1 X$ is the line of best fit for the given data points, where b_0 and b_1 are given by the above equations.

(e) (1 mark)

What parameters are to be estimated in the simple linear regression model $Y = \alpha + \beta x + \epsilon$, where $\epsilon \sim N[0, \sigma^2]$?

Solution

In a simple linear model of regression, the parameters to be estimated are α, β, and σ. The parameter β is the slope of the regression line, α is its intercept, and σ is the standard deviation of the error term ϵ.

The parameters α and β are estimated using the least squares method. The parameter σ^2 is estimated as the variance of the residuals/errors, i.e. the difference between the observed and estimated values of Y.

$$\sigma^2 = \frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \implies \sigma = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n-2}} \tag{16}$$

where we divide by $n-2$ instead of n because we have estimated two parameters, which means two degrees of freedom are lost.

The error term ϵ is assumed to be a fixed random number drawn from a normal distribution with mean 0 and variance σ^2. Hence, there is no need to estimate the error.

(f) (1 mark)

In a study of the relationship between the mean daily temperature for the month and monthly charges on the electric bill, the following data was gathered:

$$X = \text{mean daily temperature} = [20, 30, 50, 60, 80, 90]$$
$$Y = \text{monthly electric bill charges} = [125, 110, 95, 90, 110, 130]$$

Which of the following seems the most likely model?

(a) $Y = \alpha + \beta x + \epsilon \quad \beta < 0$
(b) $Y = \alpha + \beta x + \epsilon \quad \beta > 0$
(c) $Y = \alpha + \beta_1 x + \beta_2 x^2 + \epsilon \quad \beta_2 < 0$
(d) $Y = \alpha + \beta_1 x + \beta_2 x^2 + \epsilon \quad \beta_2 > 0$

Solution

Given X and Y, it is easy to notice that as the mean daily temperature increases, the monthly electric bill charges first decrease and then increase again. Hence, it is likely that the relationship between X and Y is quadratic.

Let the model be $Y = \alpha + \beta_1 x + \beta_2 x^2 + \epsilon$. Then, we can observe the following

$$Y = \alpha + \beta_1 x + \beta_2 x^2 + \epsilon \tag{17}$$
$$\implies \frac{dY}{dx} = \beta_1 + 2\beta_2 x \tag{18}$$
$$\implies \frac{d^2Y}{dx^2} = 2\beta_2 > 0 \implies \beta_2 > 0 \tag{19}$$

The drop and rise in Y values with increase in x values means that the first derivative of Y with respect to x increases from negative to positive, indicating that the second derivative is positive. This relationship is highlighted in (19).

The model cannot be linear, since the second derivative of a linear model will be 0 for all values of $x \in \mathbb{R}$. Hence, the most likely model is (d).

2 Section B (Scratch Implementation)

This problem is about implementing the Logistic Regression algorithm with Stochastic Gradient Descent from scratch. The **Diabetes Healthcare Dataset** was used for this problem. The implementation can be found in `utils.py`.

Convergence of the Model

We start with a base model with $\alpha = 0.005$ and 70 epochs. This model gives a fairly consistent performance and convergess to a loss of less than 0.5 within less than 70 epochs. The loss and accuracy plots for this model are shown in Figure 2.

This model also displays low bias and low variance, with an average accuracy of about 75%+.

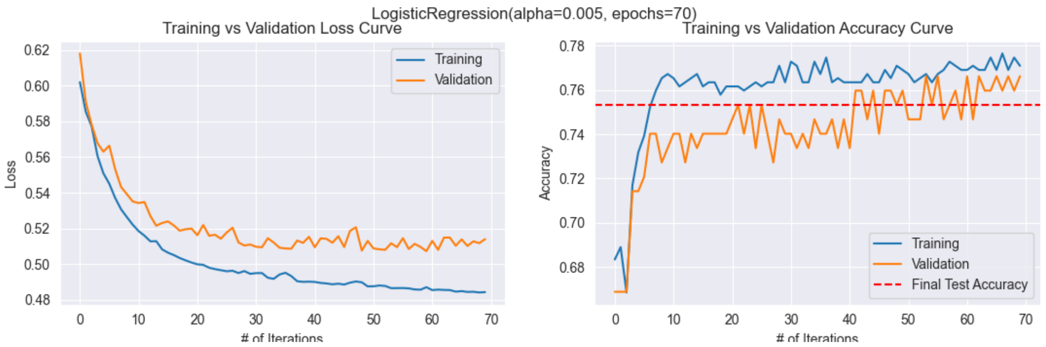

Figure 2: Loss and Accuracy plots for the base model

Hyperparameter Testing - Models with different Learning Rates

Different models were trained with different values of the learning rate. The results of the models are compared in the following figures.

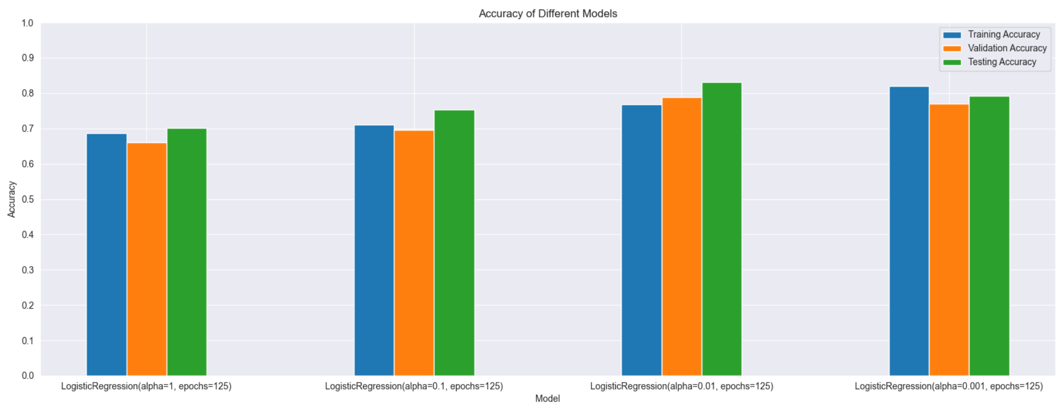

Figure 3: Training, Testing, and Validation Accuracy for different learning rates

Figure 3 suggests that learning rates 1.0 and 0.1 are too high, and so the model does not converge to a good estimate of the weights. With sufficient number of epochs the models with $\alpha = 0.01$ and $\alpha = 0.001$ perform well. However, the model with $\alpha = 0.001$ converges slower than the model with $\alpha = 0.01$ and hence takes more epochs to train.

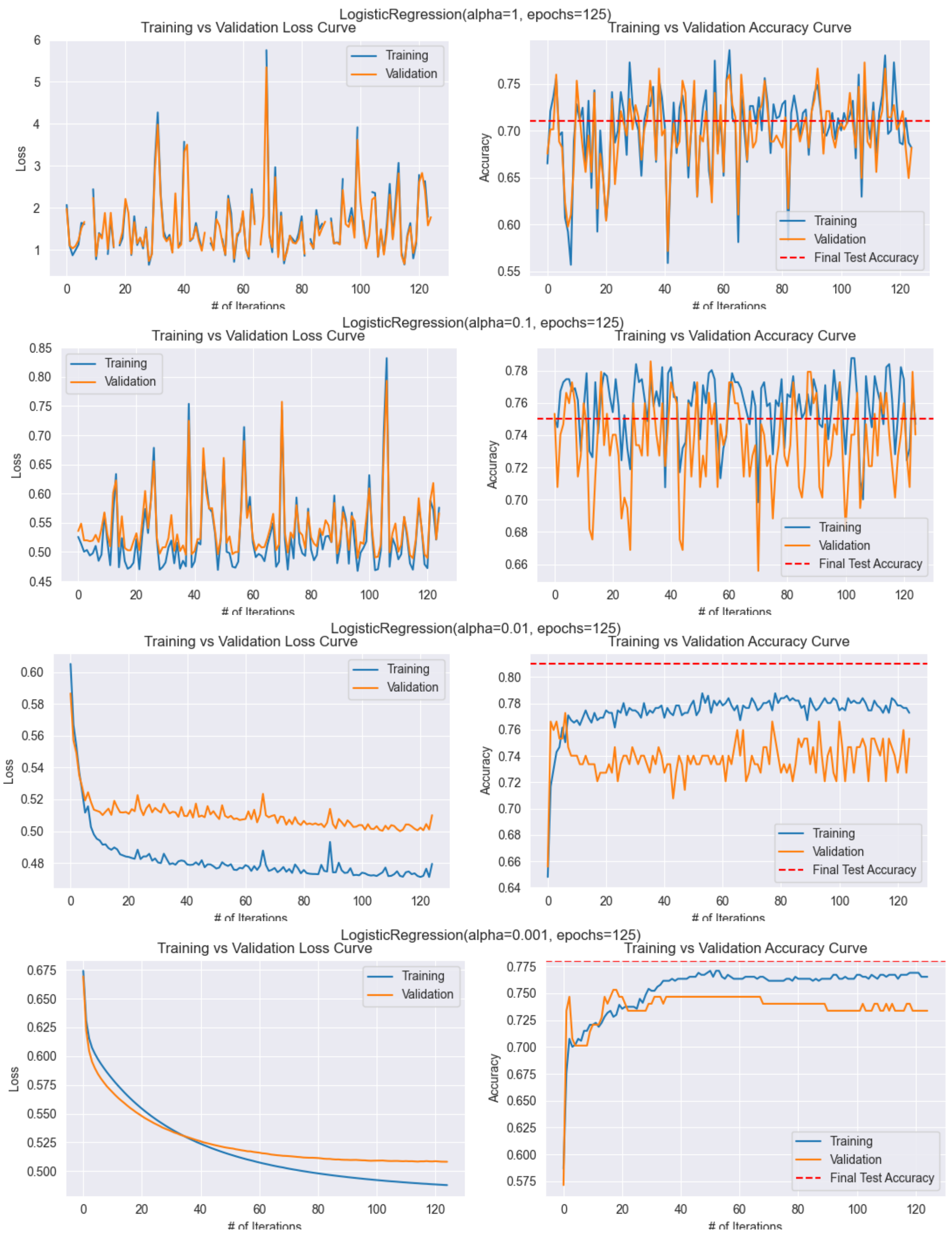

Figure 4: Loss and Accuracy plots for models with different learning rates

Figure 4 clearly demonstrates the difference in convergence between the models. The model with $\alpha = 0.001$ converges more smoothly, and with more than 50 epochs, it sometimes performs better than the other models.

We also look at the confusion matrices of the four models to see how they perform on the test set. The confusion matrix is structured as follows.

$$CM = \begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix} \tag{20}$$

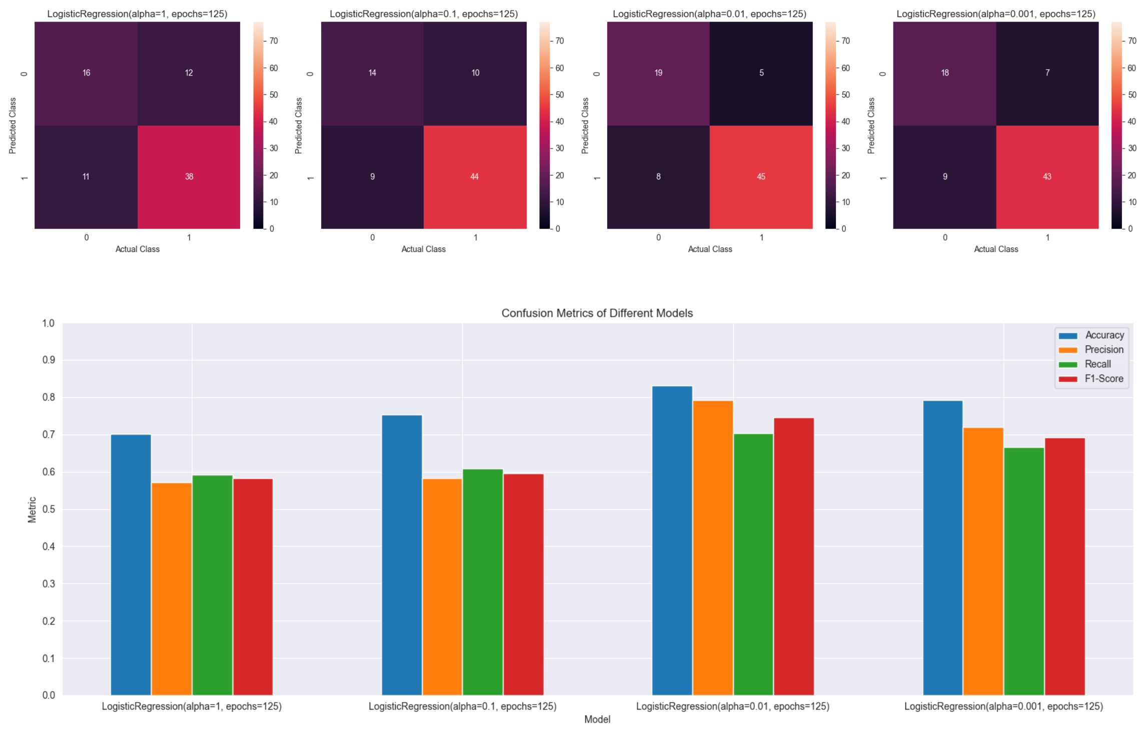

Figure 5: Confusion matrices and metrics for models with different learning rates

Figure 5 shows a comparison between the confusion matrices and metrics - Accuracy, Precision, Recall, and F1-Score of the four models. The model with $\alpha = 0.01$ performs the best, with the highest true results and least false results, and best metrics.

Adding Regularization to the Models

We now add regularization to the models to see if it improves the performance. The regularization terms L_1 and L_2 are added to the cross-entropy loss $J(\theta)$ to get the regularized loss functions $J_{L_1}(\theta)$ and $J_{L_2}(\theta)$.

$$L_1 = \lambda \sum_{j=1}^{d} |\theta_j| \tag{21}$$

$$L_2 = \lambda \sum_{j=1}^{d} \theta_j^2 \tag{22}$$

$$J_{L_k}(\theta) = J(\theta) + L_k \tag{23}$$

We search over a range of values of λ to find the best value for both regularization terms. The training and validation loss and accuracy per iteration is given in Figure 6.

Regularization tends to make the performance of the models more consistent. With any sufficient learning rate, adding regularization to the loss and gradient helped decrease the variance

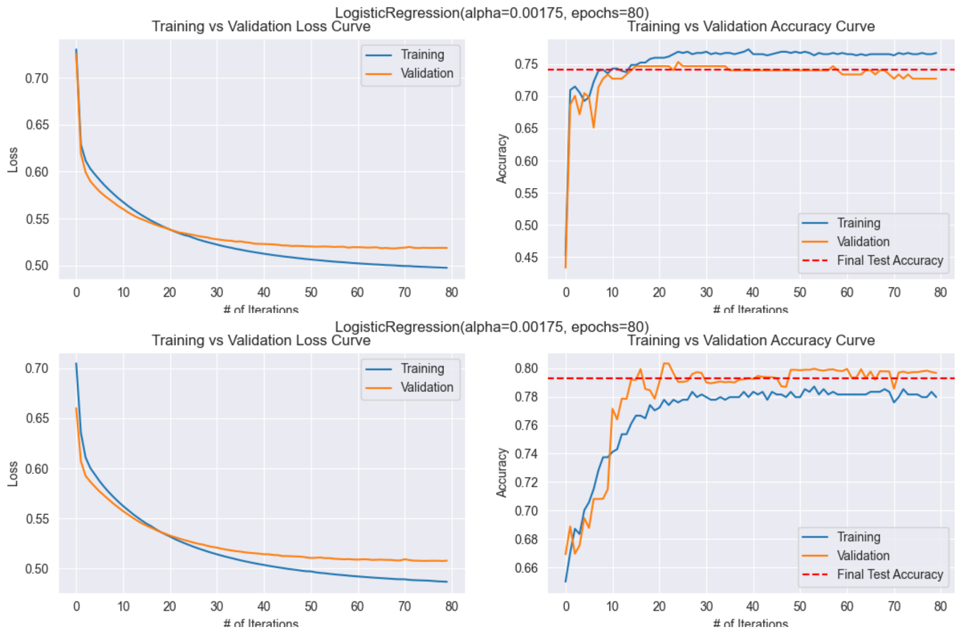

Figure 6: Training and Validation Loss and Accuracy for L_1 and L_2 regularization

in the performance on training and validation sets.

For L_1 regularization, the best value of λ was in $[0.001, 0.0001]$. For L_2 regularization, the best value of λ was around 10^{-4}.

Changing the logistic function to $\tanh(x)$

We now change the logistic function from the sigmoid function to the hyperbolic tangent function $\tanh(x)$. The range of $\tanh(x)$ is $[-1, 1]$, so it was normalized to $[0, 1]$ by adding 1 and dividing by 2. Our new hypothesis function becomes

$$h_\theta(x) = \frac{1}{2} \cdot (1 + \tanh(\theta^T x)) \tag{24}$$

We still choose to use the Bianry Cross-Entropy loss function, since it is a good measure of the performance of the model. However, the gradient was changed according to the new hypothesis function.

The training and validation loss and accuracy per iteration is given in Figure 7. On average,

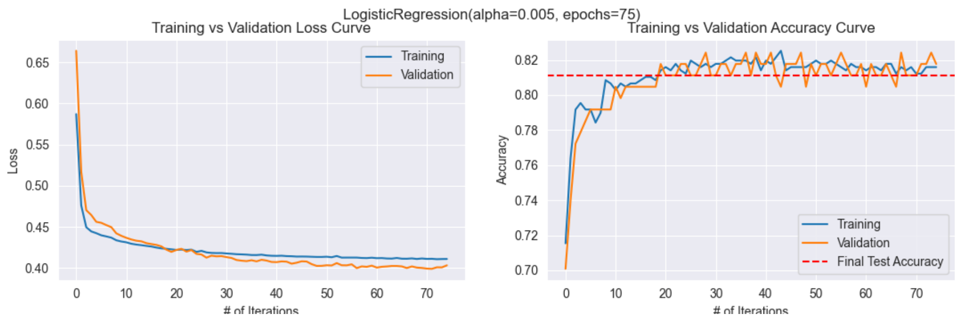

Figure 7: Training and Validation Loss and Accuracy for $\tanh(x)$

the model using $\tanh(x)$ performed better than the model using the sigmoid function. Using $\tanh(x)$ also makes the model converge in a lot less epochs, and the model converges to a better loss value. This could be attributed to the difference in gradients of the two functions.

Implementing Mini-Batch Gradient Descent

Finally, the model was trained using Mini-Batch Gradient Descent, with different values for the batch-size parameter. It was observed that as batch-size increases, the model converges faster. With batch-size, the model performance also increases, however, it stabilizes when batch-size is close to half the size of the training set.

The average performance is also better than model using SGD. This is because in each epoch, the model gets exposed to more training data, and hence, learns better.

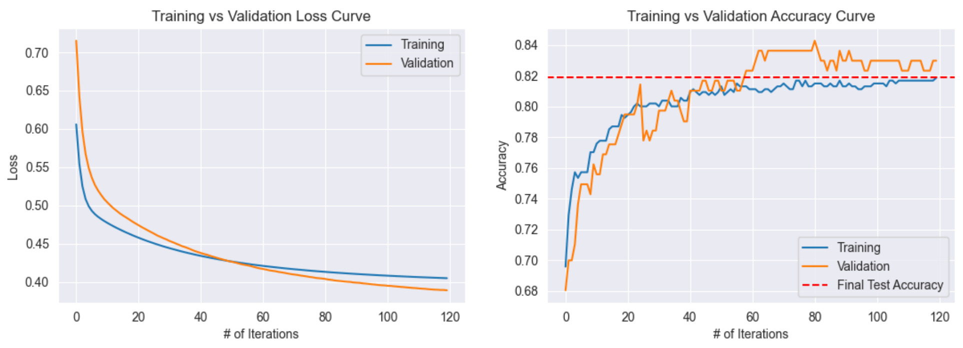

Figure 8: Training and Validation Loss and Accuracy for Mini-Batch Gradient Descent

Figure 8 shows the training and validation loss and accuracy per iteration for an instance of Mini-BGD, at batch-size = 275. It is hence, easy to note that Mini-BGD models converge a little faster than SGD models.

3 Section C (Algorithm Implementation using Packages)

This problem is about implementing the Linear Regression algorithm using the `scikit-learn` package. The **CO_2 Emissions Dataset** was used for this problem. The data analysis and model training can be fiund in the main notebook.

Data Analysis

The data was analyzed to find relevant features, and the features that are correlated to the target variable, `CO2 Emissions (g/km)`. The distributions of the features were also plotted to see if they are normally distributed.

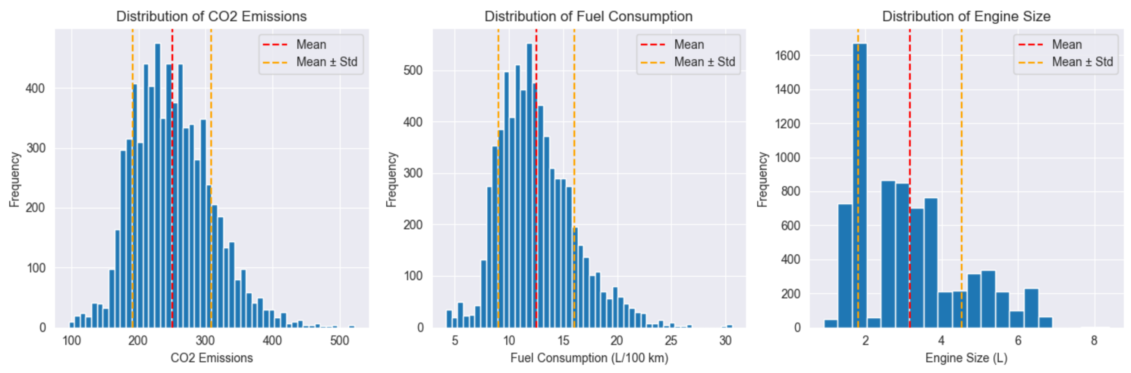

Figure 9: Distribution of some variables

Figure 9 shows the distribution of the target variable, and features like `Engine Size(L)` and `Fuel Consumption Comb (L/100 km)`. The distribution of each of them is almost gaussian, with a slight skew towards the left.

Figure 10 shows the pairplot of the features against each other. It is easy to note that most features show a strong positive correlation with each other. The feature `Fuel Consumption Comb (mpg)` shows a strong negative correlation with the other features. Moreover, the target variable also shows a storng correlation with all features. The correlation matrix of the data is given in Figure 11.

Figure 12 shows the boxplot of the feature `Vehicle Class`. It is easy to note that the median `CO2 Emissions` for `Vehicle Class VAN - PASSENGER` is the highest, and the median `CO2 Emissions` for `Vehicle Class STATION WAGON - SMALL` is the lowest. Moreover, the `Vehicle Class` of `COMPACT` and `MID-SIZE` have the largest variance (and hence number of outliers).

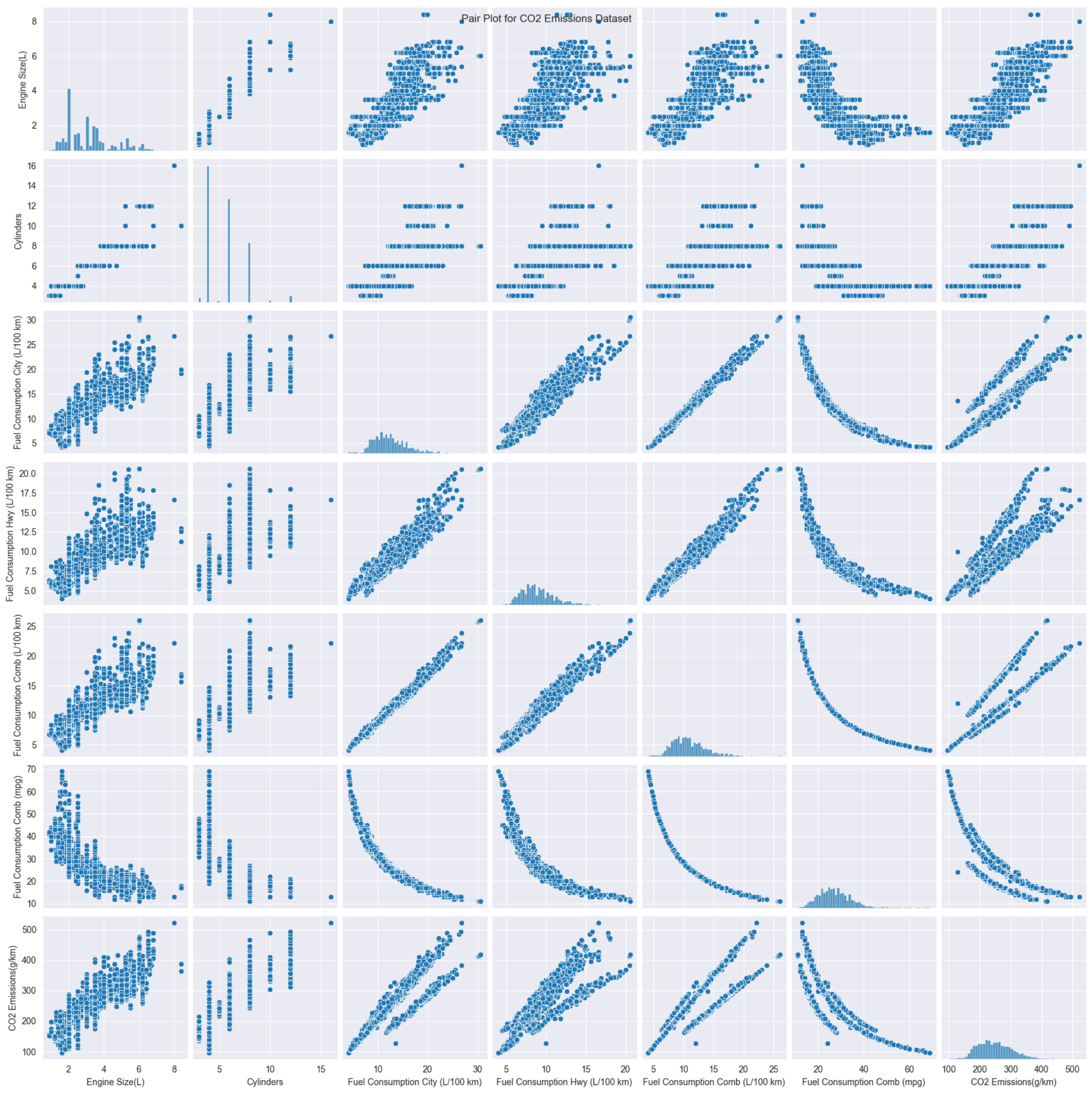

Figure 10: Pairplot of the features

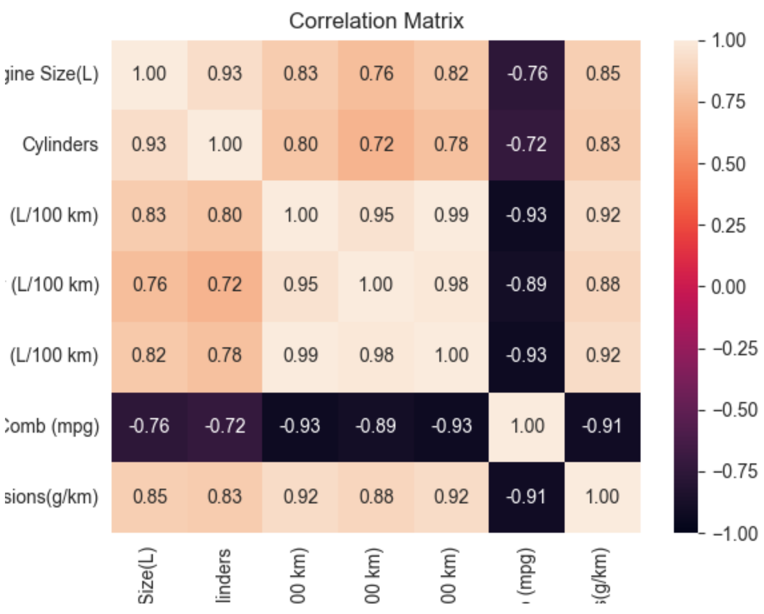

Figure 11: Correlation matrix of the features

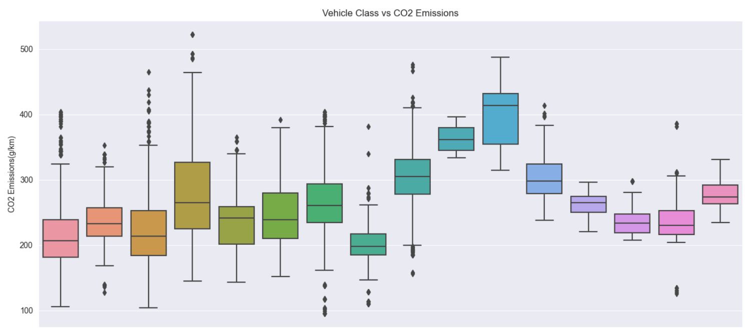

Figure 12: Boxplot of `Vehicle Class` against `CO2 Emissions`

Figure 13 shows the distribution of number of samples of each class of `Vehicle Class`. Clearly, the number of samples in each class is not balanced. This could lead to a bias in the model. However, since this is not the target variable, it is not a major concern.

Key Insights

With the above analysis, the important insights drawn from the data are summarized here.

- Most features show strong positive correlation with each other, apart from the feature `Fuel Consumption Comb (mpg)`, which shows strong negative correlation with the all other features.
- `VAN - PASSENGER` has the highest average `CO2 Emissions (g/km)`, while the `Vehicle Class` of `STATION WAGON - SMALL` has the lowest, with a few outliers.
- The `Vehicle Class` of `COMPACT` has the highest number of samples, and `PICKUP TRUCK - SMALL` has the lowest number of samples in the dataset.
- The distribution of the target variable and `Fuel Consumption Comb (mpg)` are gaussian, with a slight skew to the left.

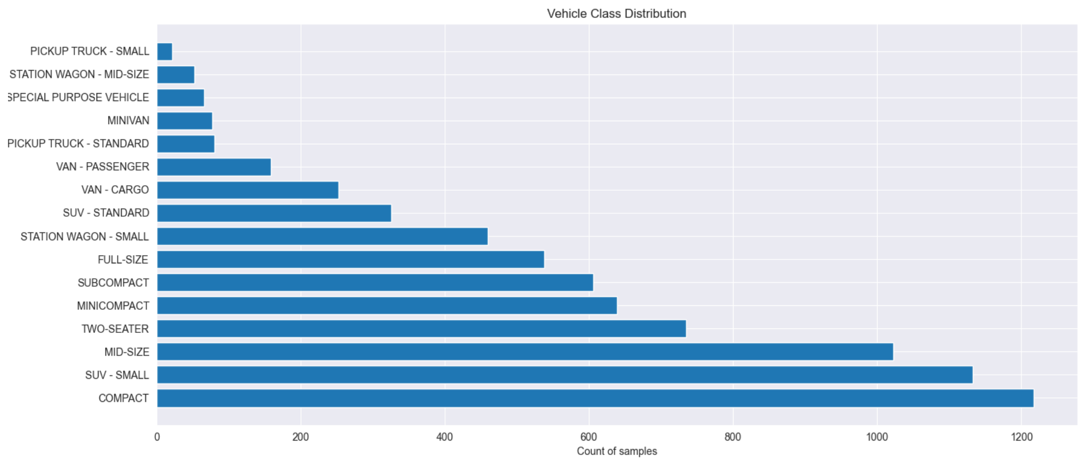

Figure 13: Distribution of number of samples of `Vehicle Class`

- The `Fuel Type` distribution suggests that the `Z` and `D` types dominate the dataset, while the `X` and `E` types are the least common.
- The pairplot suggests that the number of `Cylinders` in the engine shows high variation in correlating with the other features.
- The pairplot also displays that the `Engine Size` and `Fuel Consumption (L/100)` features are the most correlated with the target, `CO2 Emissions (g/km)`. The features with negative correlation are also evident.
- Overall, all features are strongly correlated with the target variable, which is a good indicator that the model will perform well.

Dataset Visualization using 2-dimensional t-SNE Algorithm

We use the t-Stochastic Neighbor Embedding Algorithm to reduce the dataset to 2 dimensions. This algorithm preserves the unnderlying structure of the dataset. So, ideally, the clusters in the data in higher dimensions should be preserved in the 2-dimensional visualization.

The algorithm is applied on two separate copies of the dataset. One encodes categorical features using label encoding, and the other encodes categorical features using one-hot encoding. The numerical features are standardized before applying the algorithm.

Figure 14 shows the 2-dimensional visualization of the dataset. It is clear from the visualization that the clusters are preserved. They are also well-separated. The data points are also not very sparsely or very densely distributed. In both plots, the data points are fairly separated, with very minute differences. This means that the model should perform well with both encodings.

Model Training

From this subsection, we use two variants of the dataset. The first has its categorical values encoded using Label Encoding, whereas we use One-Hot Encoding for the second variant. The numerical features are standardized to have zero mean and unit variance. The Label Encoder encodes the categorical features as integers, while the One-Hot Encoder encodes the categorical features as one-hot vectors.

The tabulated results of all the different models trained are given in the following tables. The

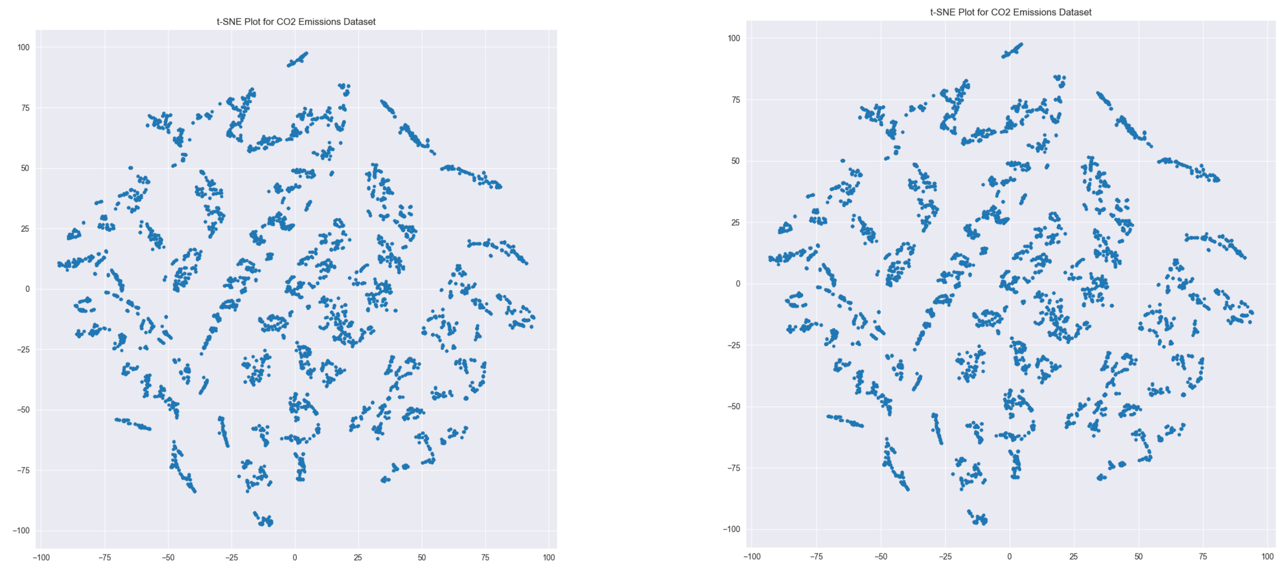

Figure 14: 2-dimensional t-SNE visualization of the dataset (Label and One-Hot Encoded)

models were trained using different encodings, regressors, and hyperparameters. The MSE, RMSE, R^2-Score, Adjusted R^2-Score, and MAE are reported for each model.

Linear Regression with Label Encoding

The results of the Linear Regression models trained with Label Encoded data are given in Table 1. A few models were also trained with PCA for different values of dimensions. This was done to check how the number of dimensions affects the performance of the model.

`n` (PCA)	Dataset	MSE	RMSE	R^2-Score	Adj. R^2-Score	MAE
-	Train	283.0803	16.8249	0.9178	0.9177	10.9406
-	Test	306.8297	17.5165	0.9078	0.9071	11.3096
4	Train	389.2899	19.7304	0.8858	0.8857	13.5305
4	Test	441.0239	21.0006	0.8732	0.8728	14.7514
6	Train	380.2578	19.5002	0.8885	0.8884	13.3269
6	Test	436.2294	20.8861	0.8746	0.8740	14.5930
8	Train	345.7971	18.5956	0.8986	0.8984	12.6924
8	Test	394.8307	19.8703	0.8865	0.8858	13.9460
10	Train	336.9742	18.3569	0.9012	0.9010	12.6153
10	Test	383.4623	19.5822	0.8897	0.8890	13.8268

Table 1: Results of different `LinearRegression` models trained using Label Encoding

Linear Regression with One-Hot Encoding

The results of the Linear Regression models trained with One-Hot Encoded data are given in Table 2. This is a repeat of the experiment done above, but for a different encoding of data. Surprisingly, almost all models trained with One-Hot Encoding perform significantly better than the models trained with Label Encoding.

The model trained on the original dataset (One-Hot Encoded) performs the best, with an R^2-Score of 1.0 and MSE of 3.01×10^{-27} on the testing data. This indicates that the model works almost perfectly. The models trained with PCA also perform well, but the performance is inversely proportional to the number of principal components selected.

`n` (PCA)	Dataset	MSE	RMSE	R^2-Score	Adj. R^2-Score	MAE
-	Train	2.95×10^{-27}	5.43×10^{-14}	1.0	1.0	3.90×10^{-14}
-	Test	3.01×10^{-27}	5.49×10^{-14}	1.0	1.0	3.92×10^{-14}
4	Train	387.6435	19.6887	0.8863	0.8862	13.5284
4	Test	439.1840	20.9567	0.8737	0.8734	14.7233
6	Train	364.7109	19.0974	0.8930	0.8929	13.0232
6	Test	423.8275	20.5871	0.8781	0.8776	14.3517
8	Train	340.8897	18.4632	0.9000	0.8999	12.7788
8	Test	394.8174	19.8700	0.8865	0.8858	14.0380
10	Train	325.2329	18.0342	0.9046	0.9045	12.2897
10	Test	374.1246	19.3423	0.8924	0.8917	13.6176

Table 2: Results of different `LinearRegression` models trained using One-Hot Encoding

Stochastic Gradient Descent Regression with different Encodings

We also implemented linear regression using the `SGDRegressor`, which uses Stochastic Gradient Descent to train the model. The results of the models trained with different encodings are given in Table 3.

Clearly, the one-hot encoded model outperforms the label encoded model. But both models perform well, and the performance is comparable with the usual `LinearRegression` model.

Encoding	Dataset	MSE	RMSE	R^2-Score	Adj. R^2-Score	MAE
Label	Train	288.1083	16.9737	0.9164	0.9162	10.8516
Label	Test	314.9450	17.7466	0.9054	0.9046	11.2728
One-Hot	Train	0.0025	0.0501	0.9999	0.9999	0.0349
One-Hot	Test	0.0023	0.0486	0.9999	0.9999	0.0345

Table 3: Results of different `SGDRegressor` models trained with different encodings

Regression with Regularization

We subject the Linear Regression Models to L_1 (Lasso) and L_2 (Ridge) regularization. Ideally, regularization should reduce the variance in the model, and hence, avoid overfitting.

Table 4 shows the results of the models trained with regularization.

Regularization	Dataset	MSE	RMSE	R^2-Score	Adj. R^2-Score	MAE
L_1 (Lasso)	Train	1.00×10^{-4}	0.0100	1.0	1.0	0.0079
L_1 (Lasso)	Test	9.81×10^{-5}	0.0099	1.0	1.0	0.0078
L_2 (Ridge)	Train	1.15×10^{-7}	0.0003	1.0	1.0	0.0002
L_2 (Ridge)	Test	1.09×10^{-7}	0.0003	1.0	1.0	0.0002

Table 4: Results of different regularization models on Linear Regression

All the above models were trained with the regularization parameter $\lambda = 0.01$. Clearly, L_2 regularization works better than L_1 regularization. However, the R^2-Scores of both the models were equal to 1.0. The MSE and MAE are also very low, but not as low as the regular one-hot encoded linear regression models, which is ideal.

References

1. Wikipedia: Logistic Function
2. Estimating variance in Linear Regression in $\mathbb{R}$
3. Confusion Matrix, Accuracy, Precision, Recall, F1-Score - GeeksForGeeks
4. Regularization - GeeksForGeeks Guide
5. Box Plot - Byjus
6. *t*-SNE Manifold - `scikit-learn` official documentation