Name: Rishabh Acharya
Roll Number: B22CS090

Report for PA2 (Lab 3, 4)

General step in all questions:
'random_state' parameter has been used to ensure reproducibility of the results obtained.

Question 1

- Given the titanic dataset, the task is to train a decision tree made from scratch, and test for accuracy and other metrics.
- PassengerID, Ticket, Cabin columns have been dropped as they seemed to be irrelevant.
- Embarked column had 2 missing values. The 2 rows containing the missing values have been dropped as dropping 2 rows won't affect the dataset significantly.
- The missing ages have been filled in using the following steps:
    - ➔ Firstly, I have checked the mean of the overall dataset. It came out around 29.7.
    - ➔ The problem with filling in missing ages with this value is that, for names containing 'Master,' it obviously specifies that the concerned person has age less than 18. For something that obvious, filling in the age with age > 18 will be illogical.
    - ➔ Regardless of Pclass, a boolean mask is created which filters out the rows containing names containing 'Master,' and the mean age is found out.
    - ➔ Missing ages corresponding to those names are filled in with the aforementioned mean age, and appropriately inserted back into the dataframe.
    - ➔ For the rest missing values, a function is made which creates boolean masks based on Pclass and gender.
    - ➔ 6 types of masks are created (3 types of Pclass and 2 types of gender) and 6 filters are made accordingly.
    - ➔ For each filtered dataframe, mean age is found out and missing ages are filled in in that specific dataframe.
- Name column is dropped because it doesn't contribute to determining survivability.
- Onehot Encoding is performed for Embarked and Pclass columns as they are not ranked categorical data and it is need to convert categorical data to numerical data so that the decision tree can work on it.
- Gender is mapped as 'male': 1 and 'female': 0 for the above mentioned reason.
- Before applying all the preprocessing mentioned above, the dataset is split into training, validation and test datasets because data leakage is unwanted (Explanation: A model never sees the test data while or before training. If calculation of mean age is done on

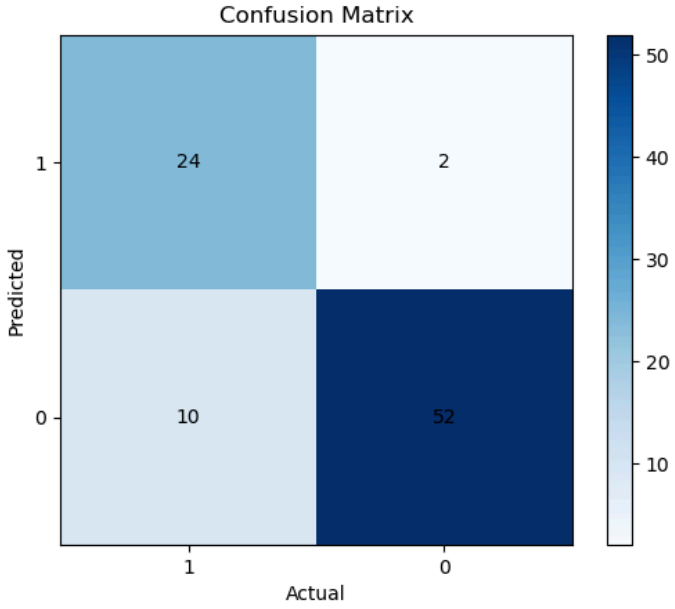
the whole dataset, it will obviously include the test dataset as well, resulting in data leakage. Hence, the same preprocessing steps are followed but on 3 datasets independently).

- Node class is created to act as a node for the decision tree. Contains all the required variables such as left and right child nodes (if any), filtered dataset, depth, label, etc.
- DecisionTree_Classifier is the class that contains all the necessary functions concerning the decision tree.
- fit() method is called to create the decision tree.
- categorical_continuous() determines which features are categorical and which are continuous.
- For each node, its entropy is stored in order to calculate the information gain.
- From fit(), the recursive function build_tree() is entered.
- This function prints the validation accuracy everytime 2 nodes are added.
- split() function determines the best split possible, based on the highest information gain obtained.
- If a feature is categorical, every possible split is accounted for and the information gains are compared. Finally, split with the highest info gain, along the left and right child nodes, are stored.
- If a feature is continuous, conTocat() returns the list of average value between every 2 continuous data points of that feature column, and for each average value obtained, we find a split and the corresponding info gain.
- After taking into consideration both continuous and categorical features, split with the highest info gain is obtained, along with the left and right child nodes obviously.
- Limitations: If highest info gain isn't > 0 or if maximum allowed depth of tree is reached, splitting is terminated.
- The same steps are followed every time new nodes are created or termination in one node occurs.
- Max depth = 8 is used as the model was overfitting to the training dataset otherwise. 8 seemed optimal to both the training and test data.
- find_predictions() returns the predicted class label (survived or not survived) after splitting the given dataframe, based on the decision tree created.
- Metrics such as precision, recall, F1 score are calculated by simply finding the true positives, false positives, true negatives and false negatives.
- infer() asks for all the categories that were originally present in the dataset, performs the same preprocessing steps and gives the corresponding survivability.
- Confusion Matrix is created by obtaining true positives, false positives, true negatives and false negatives and plotting it using matplotlib.

Observation:

Classification of features into ordinal, nominal and continuous:

Ordinal	Nominal	Continuous
Pclass	Survived	Age
	Names	Fare
	Sex	
	Ticket	
	Embarked	
	Cabin	



Precision: 0.9230769230769231
Recall: 0.7058823529411765
F1 Score: 0.8000000000000002
Training Set Accuracy: 88.2636655948553 %
Test Set Accuracy: 86.36363636363636 %

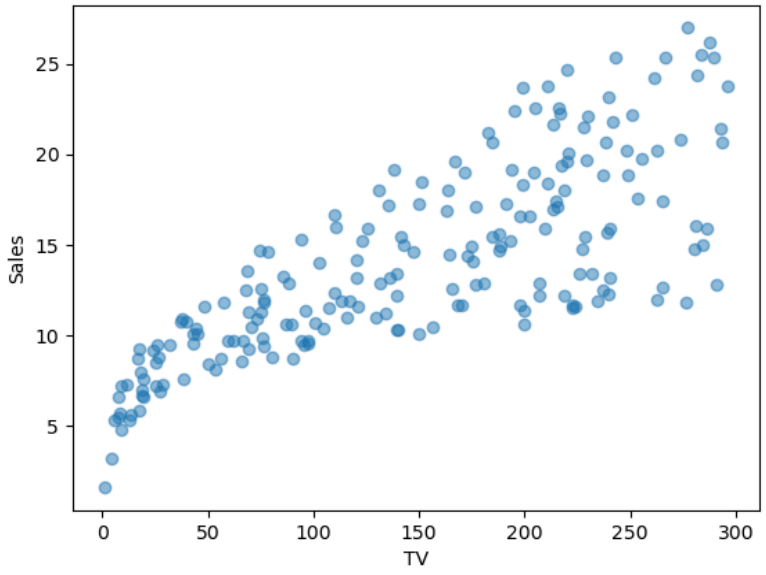
Conclusion:

Trained model does a good job of predictions and doesn't overfit. Precision, recall and F1 scores are good.

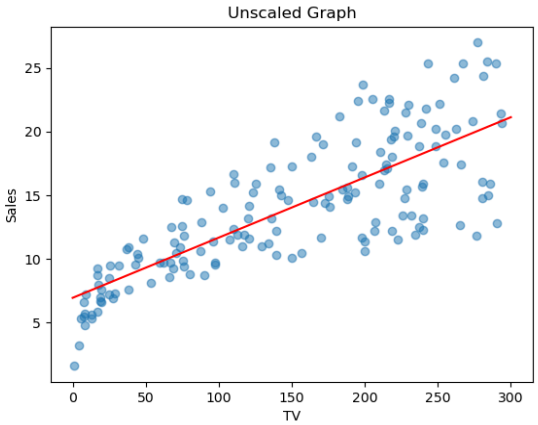
Question 2

- Question asks to make a linear regression model and fit a line to the given data.
- A general class for Linear Regression will be used, which can work for multivariable linear regression as well (Q3).
- Z normalization is done for both x and y dataset, based on the test data. This is done for faster convergence and to prevent the MSE from overshooting (this was happening if I hadn't normalized the data).
- Weight corresponds to the slope and bias corresponds to the intercept over here.
- Gradient descent is used to update the slope and intercept.
- From the formula of MSE, partial differential is done with respect to weights and bias, and the resulting formula is used to calculate the gradients. These gradients are supplemented with a learning rate to determine the magnitude of the step taken to reach minima.
- Learning rate is kept low in order to prevent a huge step and a potential overshoot.
- Vectorization is used to calculate the predictions and gradients. It means that operations are applied to the matrix at once, instead of looping over the samples. This enhances the speed of training.

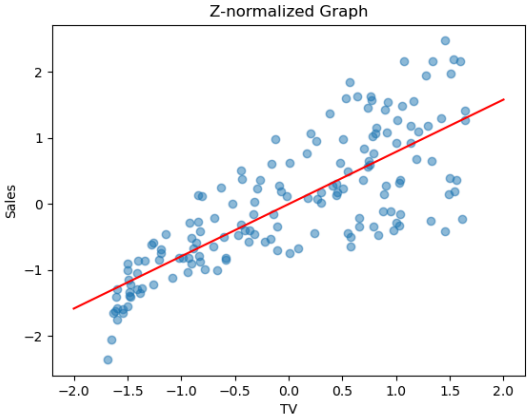
Observation:



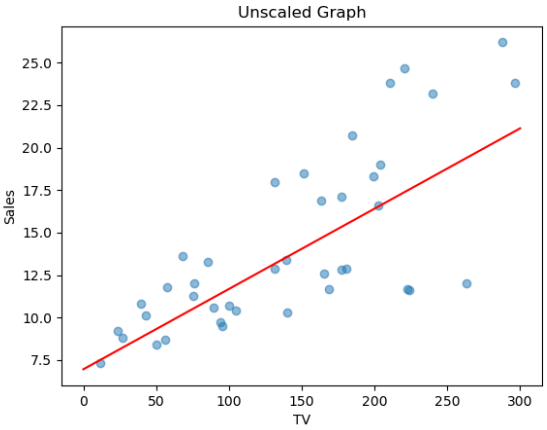
Scatter Plot (Whole Dataset)



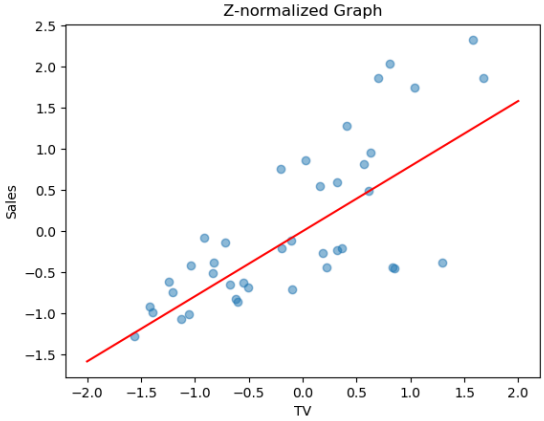
Best Fit Line on Unscaled Graph (Training set)



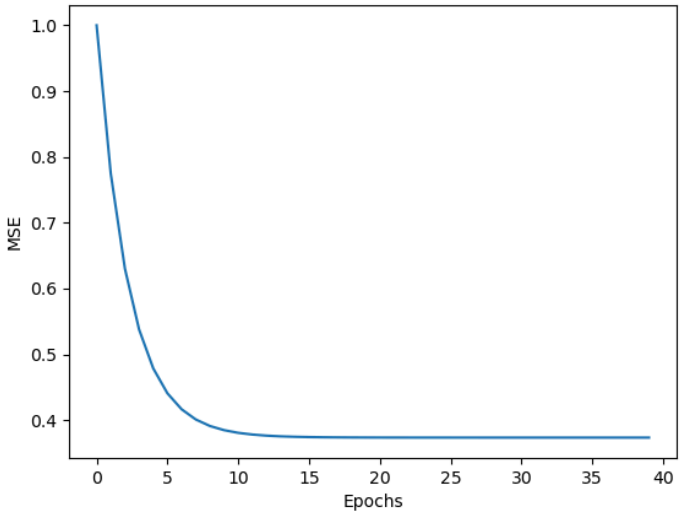
Best Fit Line on Normalized Scales (Training set)



Best Fit Line on Unscaled Graph (Test set)



Best Fit Line on Scaled Graph (Test set)



MSE vs Epochs Graph

Plotting MSE vs Epochs graph shows that the model has converged and reached minimum MSE with learning rate = 0.1 and after 40 epochs.

Training MSE after 40 epochs: 0.3734998555005562
Test MSE: 0.4125772567595091
Test Mean Absolute Error (MAE): 0.5101535646116397

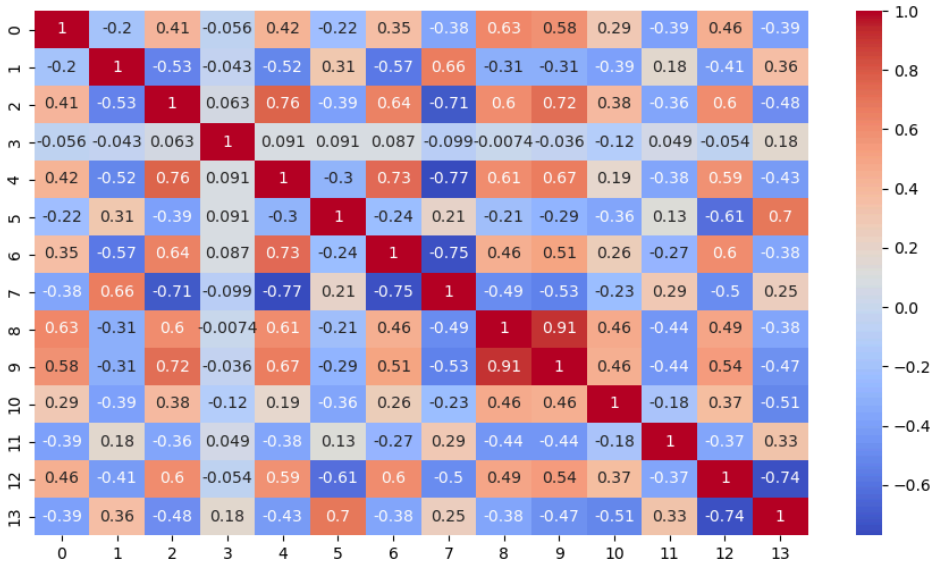
Conclusion:

Model has converged and reached minima.

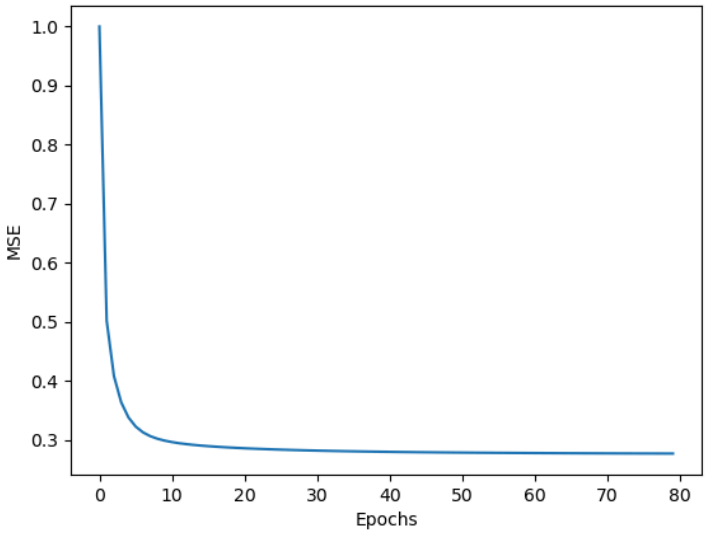
Question 3

- Steps taken to approach the problem are quite similar to that of question 3.
- Dataset is processed to align the columns corresponding to a row.
- Instead of just one feature column, we have multiple feature columns now.
- Instance of the same Linear Regression class is used to attain minima.

Observation:



Correlation heatmap showing the correlation coefficient between every 2 variables in original dataset



MSE vs Epochs graph

Plotting MSE vs Epochs graph shows that the model has converged and reached minimum MSE with learning rate = 0.1 and after 80 epochs.

Training MSE after 80 epochs: 0.27716192505253934
Test Set MSE: 0.20421951686682666
Test Set MAE: 0.32926640344199753

Conclusion:

Model has converged and reached minima.