Question 5:

(a). Please see the attached jupyter notebook in the folder “question 5”

(b). (i).

The probabilty of a record being 1 at the beginning is given by

$$P(\hat{y} = 1|x_1, x_2) = \frac{1}{1 + e^{-(-1+1.5x_1+0.5x_2)}}$$

The cross entropy error function is given by

$$E(w) = \sum_l y^l lnP(y^l = 1|x^l, w) + (1 - y^l)lnP(y^l = 0|x^l, w)$$

where w = [-1, 1.5, 0.5]

(ii). weights after one iteration = [-1.00316626 1.50535086 0.50196867]

The updated logistic model after one iteration is

$$P(\hat{y} = 1|x_1, x_2) = \frac{1}{1 + e^{-(-1.0032+1.5053x_1+0.502x_2)}}$$

(iii). After the model converges, Accuracy = 0.67, Recall = 1.0, Precision = 0.6

Question 6:

The top two scores achieved were 3.26377 and 3.38977. Both these scores were achieved by gradient boosting regressors. The first was achieved by a gradient boosting regressor with 500 trees and max depth 6. The second was achieved by a gradient boosting regressor with 200 trees and max depth 6.

For training the models, one million samples were randomly chosen from the training data using the command line utility shuf. These samples are present in sample.csv. The predictions are present in sample_submission.csv.

The data was lightly preprocessed. The column containing passenger count was dropped because new york city taxi laws prevent charging passengers based on passenger count. Nonsensical data was deleted. Date and time column was processed to generate useful features. The model giving the highest accuracy took 30 minutes to train.

The other models that I tried were k nearest neighbours and random forests. Hyperparameter optimisation was done on K nearest neighbours. It showed promising results with k = 2 for 1 million samples. Thus it was decided to use more data. But it was taking too long to train. Thus

it was not pursued any further. Random forests did not show much promise against gradient boosting either.

Gradient boosting probably worked better than k nearest neighbours because knn suffered from less data and curse of dimensionality. After feature engineering, there were 9 features in the data. It is not too high but it can possibly decrease the performance of knn.

The initial accuracy gained with gradient boosting was 4.21206. The initial accuracy gained by random forests was 5.39343. Seeing such a big difference in accuracy made me abandon random forests at the get go. However, fine tuning random forests may lead to better than initial results.

Gradient boosting probably performed better because random forests can only minimize variance whereas gradient boosting can minimize both bias and variance.