**Programming Questions**

**Q.5) Logistic Regression.**

a) Logistic Regression model implemented from scratch. Final accuracy calculated is 0.667.

b)

Linear Model: fw(x1, x2) = w0 + w1 \* x1 + w2 \* x2

Logistic Regression Function: 1 / (1 + e ^ (-fw(x1, x2)))

f(x1, x2) = - 1 + 1.5 \* x1 + 0.5 \* x2

Logistic Model: P (y\_hat = 1|x1, x2) = sigmoid(f(x)) = 1 / 1 + e-(- 1 + 1.5 \* x1 + 0.5 \* x2)

Cross-entropy Error Function:

=> - [y \* log(y\_predict) + (1−y) \* log(1 − y\_predict)]

=> - [y \* log(sigmoid(f(x)) + (1-y) \* log (1 - sigmoid(f(x)))]

=> - [y \* log (sigmoid(- 1 + 1.5 \* x1 + 0.5 \* x2) + (1-y) \* log(1 - sigmoid(- 1 + 1.5 \* x1 + 0.5 \* x2)]

Output of cross-entropy error function consisting of an array: [1.05663407, 0.84177673, 0.67949171, 0.43167237, 0.44515933, 0.48033844]

1. Updated Logistic Model:

theta\_0 = -1.0189, theta\_1 = 1.5321, theta\_2 = 0.5118.

f (x1, x2) = - 1.0189 + 1.5321 \* x1 + 0.5118 \* x2

Cost: 3.3264

1. Ans:
2. Accuracy = 0.667
3. Precision = 1
4. Recall = 0.333

**Q.6 Kaggle – Taxi Fare Prediction.**

1. Pre-processing the dataset:

1. Load the data, and removed the columns having null, NAN values.
2. Checked for zeros in train and test data, dropped the zero rows.
3. Extracted the dates separately from pickup\_datetime column and dropped that column.
4. Calculated the distance between given latitudes and longitudes.
5. Finally calculated Harvesine distance between two points on earth.

2. Feature selection, Modelling:

a) Selected 7 important features and dropped which are not useful. b) Used scikit-learn’s ‘train\_test\_split’ to divide the training data into training set with validation set consisting of 20% of rows of original dataset.

3. Training the Model:

a) Trained the dataset using LinearRegression, RandomForestRegressor, XGBoostRegressor. b) The best result was achieved with XGBoostRegressor. c) In order to find a best result, fine-tuned the hyperparameters of classifiers.

4. Result of Different Models:

1. Linear Regression Model: Model 1 is implemented using Linear Regression function. It gave RMSE around 9.32, which was too large. Hence it was rejected.

Graphical user interface, application

Description automatically generated

1. Random Forest Regressor: Model 2 is implemented using Random Forest Regressor. It gave RMSE around 3.72, which was quite good.

Graphical user interface, application, Teams

Description automatically generated

1. XG Boost Regressor: Model 3 is implemented using XG Boost Regressor. It gave RMSE around 3.61, which was best among all.

Graphical user interface, application

Description automatically generated

1. XG Boost Regressor-2: Model 4 is implemented using XG Boost Regressor only, but by changing some hyperparameters. It gave RMSE around 3.615.

Graphical user interface, application, Teams

Description automatically generated

* So, the top two scores of Models of RMSE are: 3.606, 3.615. Both models are trained using XG Boost Regressor. For the first model, the hyperparameters are: learning rate = 0.3, maximum depth = 6 and estimators are 100. While for second model are: learning rate = 0.1, maximum depth = 8 and estimators are 200.
* Basically, XG Boost regressor is one of the best among all regression models. It has parallelized tree building and pruning using depth-first approach. It penalizes more complex models through both LASSO (L1) and Ridge (L2) regularization to avoid overfitting. This algorithm comes with built-in cross-validation method at each iteration which avoids specifying the exact number of boosting iterations required in a single run. After XG Boost, Random Forest Regressor is a better option.