PROJECT: FLIGHT PRICES PREDICTION

PROJECT STATEMENT:

In this project, we aim to build machine learning models to predict flight prices given various details about the flight such as departure time, arrival time, the day of the week etc.

MOTIVATION BEHIND THE PROJECT:

Everyone in our team hails from Northern India and have often faced problem while booking flight tickets. There have been several instances when we booked our ticket early to get at low cost but our estimate didn't turn out to be correct. The 'dynamic pricing' strategy followed by most of the airlines today leaves many customers stumped when it comes to predicting prices. When the problem statement for this ML Hackathon was left open, we decided to grab this opportunity to learn in depth about flight pricing techniques, the factors on which the price depends and applying the algorithms which we have learned as part of the course to tackle this problem.

DATASET:

We wanted to prepare our own dataset by scraping the online travel websites. However, for this, we would have had to start scraping the data at least two weeks ago. This is because most of the travel websites do not store prices of past flights which would have been required to make the prediction.

Hence, we searched online for an available dataset and were able to find one such dataset on Github. The link for the used dataset is-

https://github.com/humain-lab/airfare_prediction/tree/master/dataset

About the dataset:

The dataset which has been manually collected from 'airtickets.gr', consists of 1814 rows.

Records are for flights from Thessaloniki (SKG) - Greece --> Stuttgart (STR) - Germany

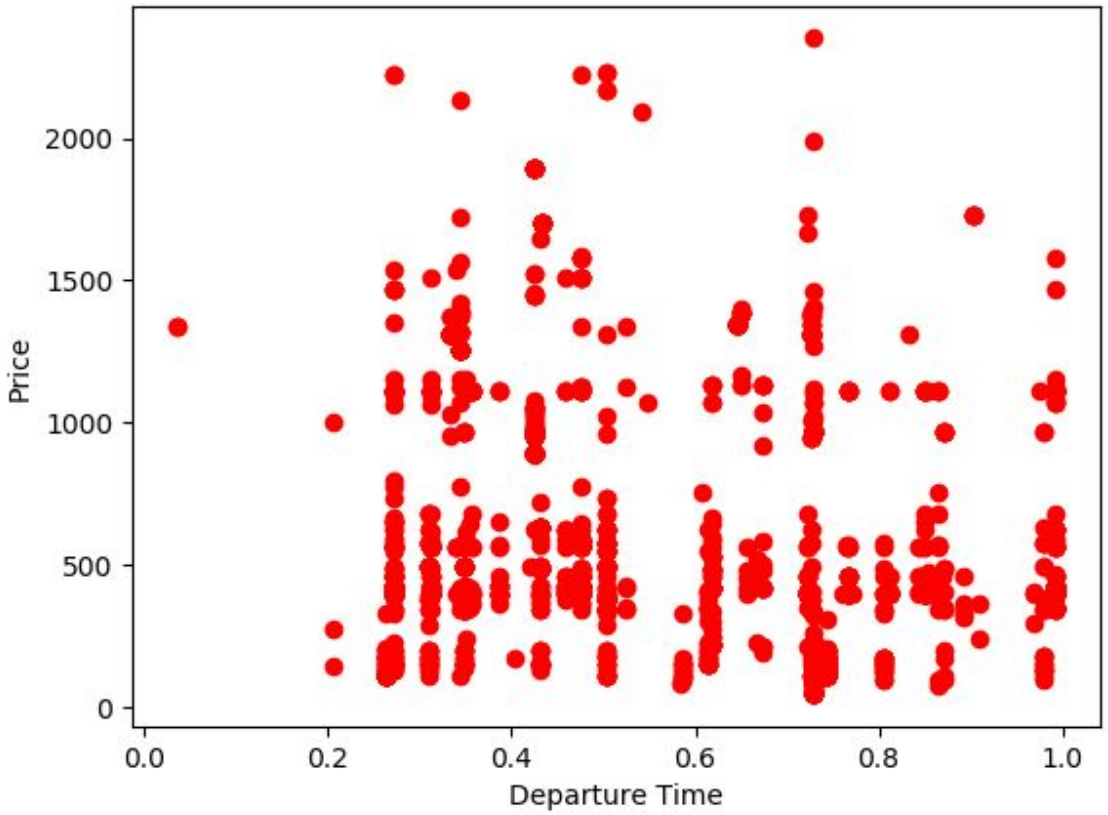
For every flight, the following features were considered:

F1: Feature 1 - departure time.
F2: Feature 2 - arrival time.
F3: Feature 3 - the number of free luggage (0, 1 or 2).
F4: Feature 4 - days left until departure.
F5: Feature 5 - the number of intermediate stops.
F6: Feature 6 - holiday day (yes or no).
F7: Feature 7 - overnight flight (yes or no).
F8: Feature 8 - the day of the week.

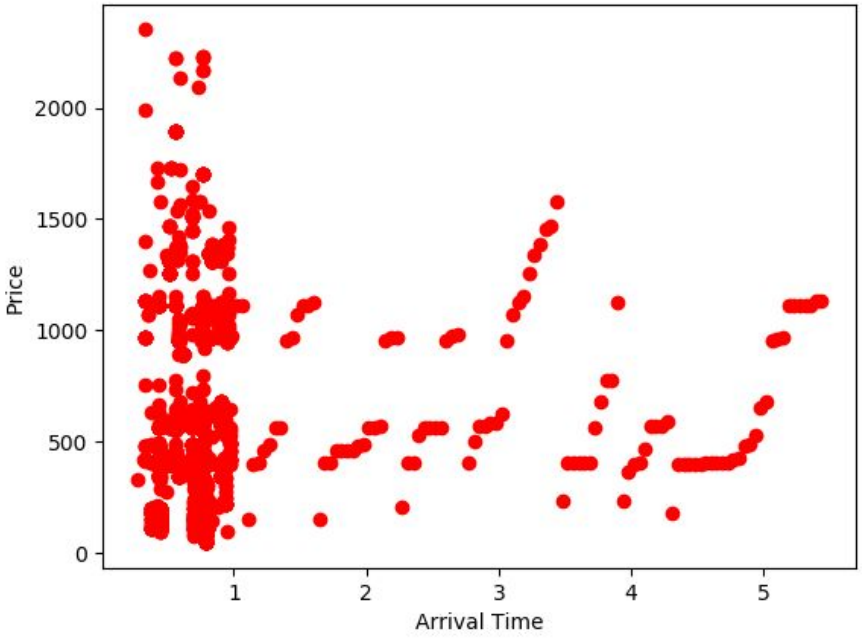
The 9th column of the dataset corresponds to the price of that particular flight.

ANALYSIS OF DATASET:
We plotted graphs of price vs all the 8 features and tried to observe patterns in the data. Here are the graphs:

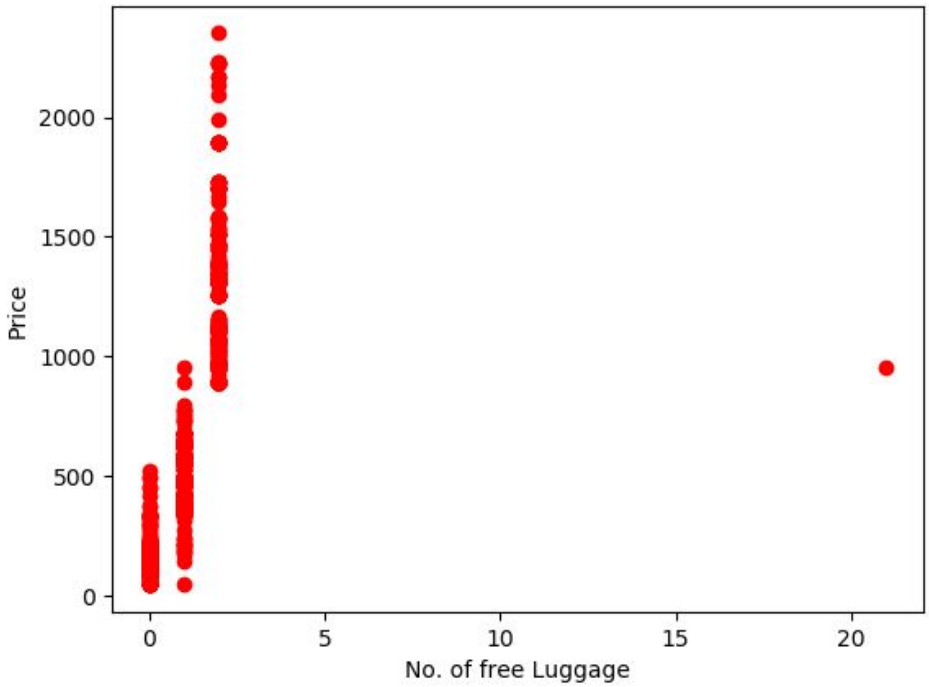
1. Price vs Departure Time: No particular pattern could be observed. The graph tells that the data we have is uniformly distributed wrt departure time between the range of 0.2 to 1.0.



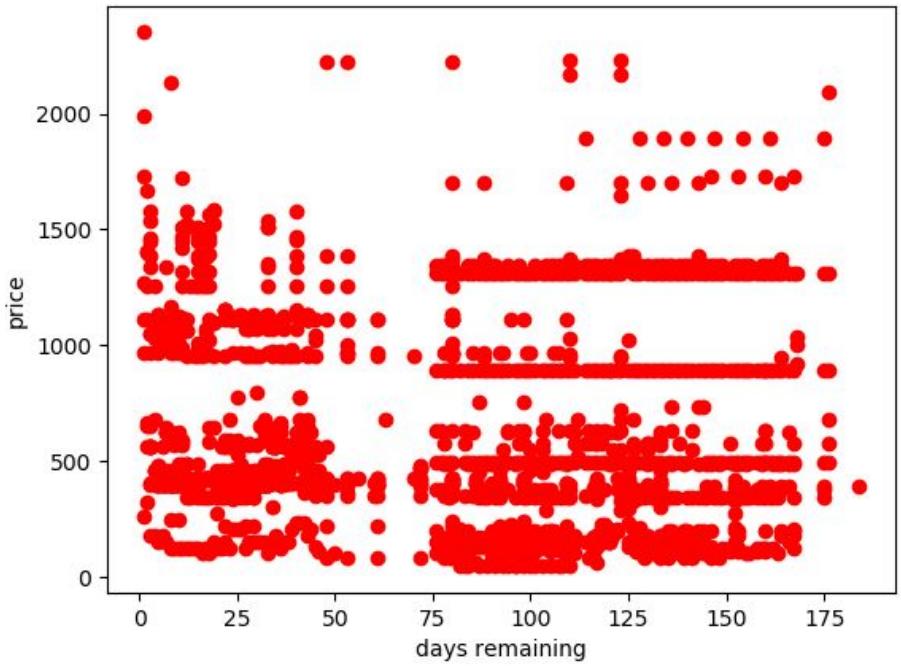
2. Price vs Arrival Time: Majority of the data points have their arrival time between 0.0 to 1.0 and in this range, there was no correlation between price and arrival time.



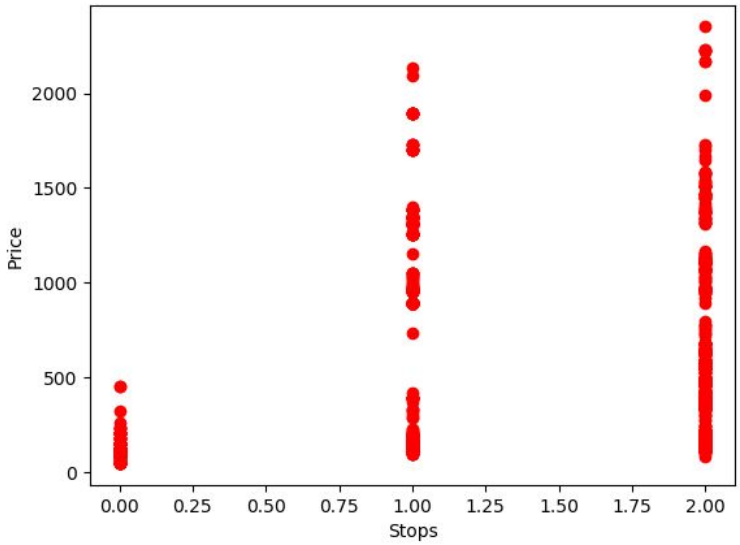
3. Price vs No. of free Luggage: Price is directly proportional to the number of free luggage. And one data point with the number of luggage as 20 is most probably an outlier.



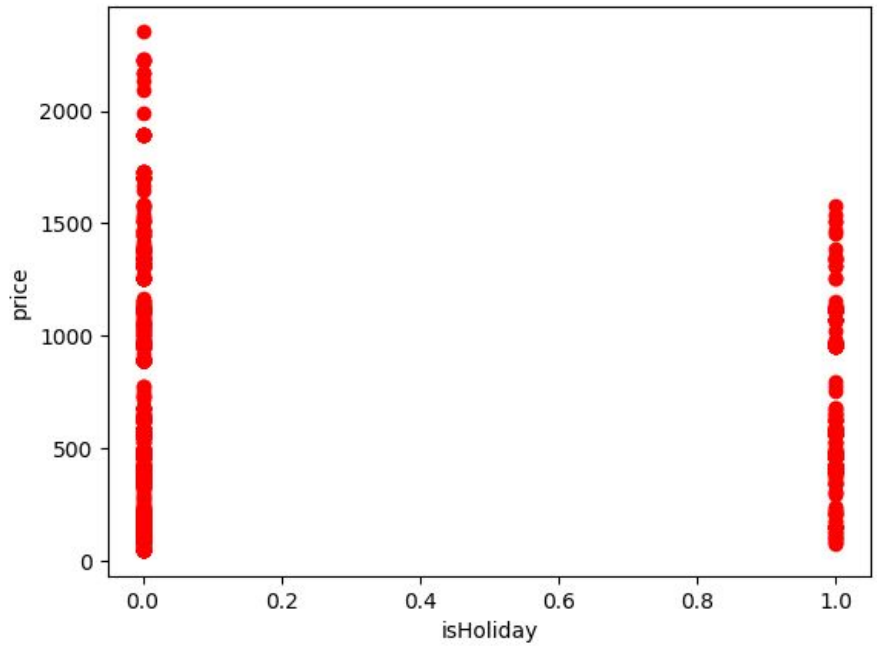
4. Price vs Days left for departure:



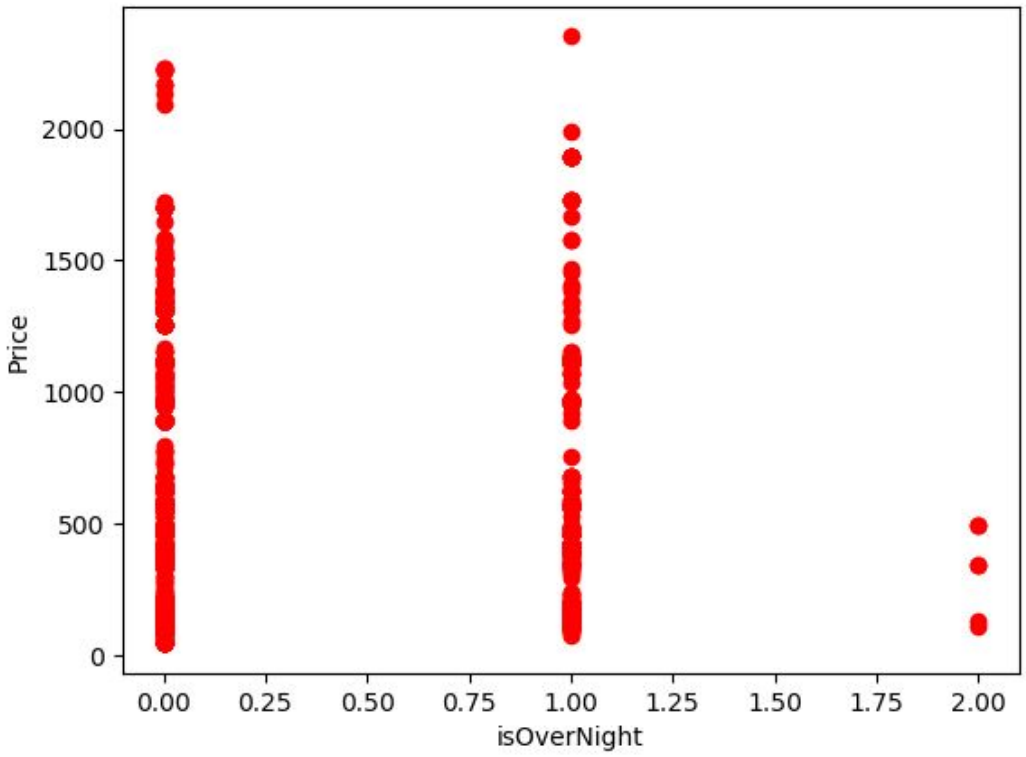
5. Price vs No. of Stops: The plot shows that we have flights with less number of stops are cheaper. Moreover, the plot shows that we have very less data for 0 stops flights.



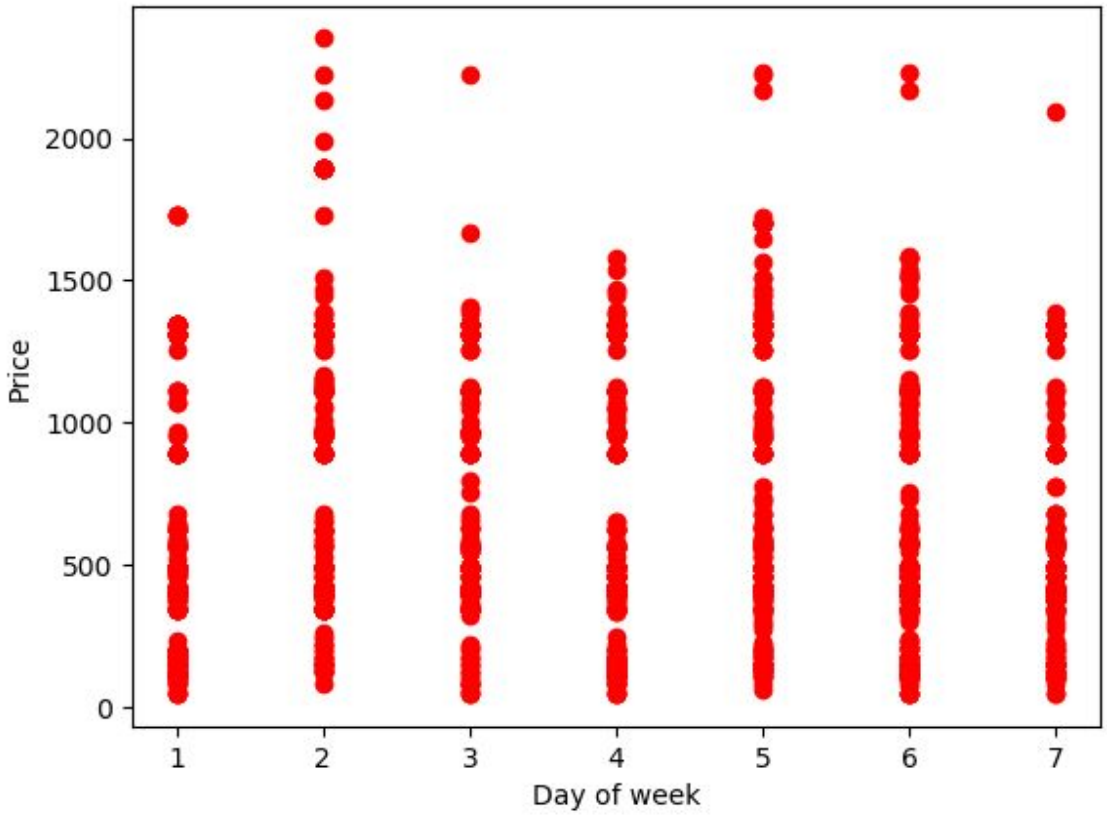
6. Price vs isHoliday:



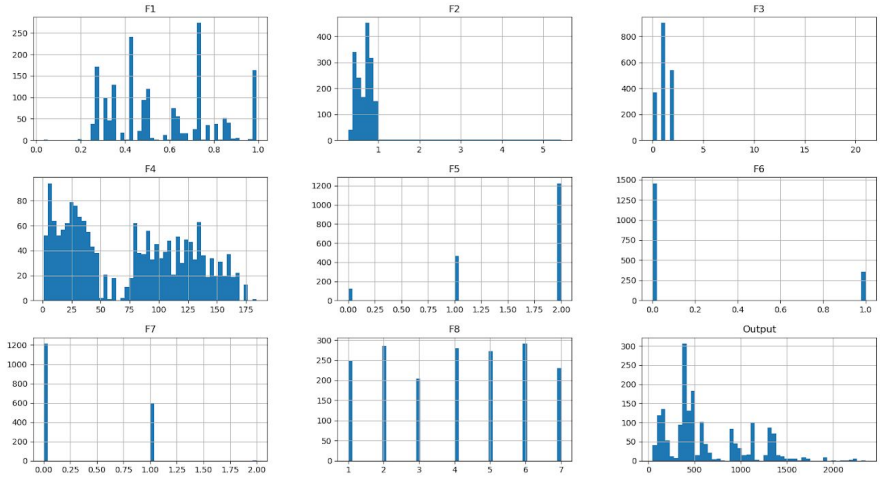
7. Price vs overnight



8. Price vs Day of the week: Price distribution for each day of the week is almost similar, which indicates that this feature is not much helpful in model building.



HISTOGRAMS:



APPROACH TAKEN:

1. DATA AUGMENTATION**:** We have very few data points with 0 number of intermediate stops. Moreover, price and number of intermediate stops are correlated as seen by the plot. Also, since “Day of Week” and price aren’t correlated, we can use this to augment our data. So, we added more rows having 0 number of intermediate rows using existing such rows and just changing the week of day column.
2. Since all the columns had real values (no text) Linear Regression was the first thing to come in mind. We fitted a Linear Regression model, firstly using all the 8 features, then by removing “day of week” feature. As expected, results were similar for both cases.
3. Results we got from Linear Regression had 0.71 scores, so we tried Random Forest Regressor next to increase our score.
4. We used scikit learn inbuilt libraries to build/train our models.

Machine Learning Models used in this project-

1. Linear Regression
   Simple linear regression is a statistical method that allows us to summarize and study relationships between two continuous (quantitative) variables:

- One variable denoted *x*, is regarded as the **predictor**, **explanatory**, or **independent** variable.
- The other variable denoted *y*, is regarded as the **response**, **outcome**, or **dependent** variable.

   Here, the features like the day of the week, holiday etc are the independent variables while the price of the flight is a dependent variable.

2. Random Forest Regression

   The **random forest** model is a type of additive model that makes predictions by combining decisions from a sequence of base models. More formally we can write this class of models as:

$$g(x)=f_0(x)+f_1(x)+f_2(x)+\ldots$$

   where the final model g is the sum of simple base models f_i. Here, each base classifier is a simple decision tree. This broad technique of using multiple models to obtain better predictive performance is called **model ensembling**. In random forests, all the base models are constructed independently using a **different subsample** of the data.

   The model requires various parameters to be entered. The parameters and their chosen values as follows :

a. **n_estimators** : integer, optional (default=10)

   This tells the number of trees in the forest. We chose 100 as value for this parameter. We conducted various tests and concluded this value to be the most optimal.

b. **max_depth** : integer or None, optional (default=None)

   The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

   Since the number of features is less than 10, we decided to keep the maximum depth of the trees numerically low. After analysing the results with various values for max_depth, we concluded 3 as the most optimal max_depth.

c. **random_state** : int, RandomState instance or None, optional (default=None)

   If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random.

   We have chosen 10 as the random state for our model.

TEST METRICS:

We used sklearn's inbuilt score function to test our model's accuracy on test data. We split our data into the test, train data using an inbuilt train_test_split function that splits data randomly.

About score function: Returns the coefficient of determination R^2 of the prediction. The coefficient R^2 is defined as (1 - u/v), where u is the residual sum of squares ((y_true - y_pred) ** 2).sum() and v is the total sum of squares ((y_true - y_true.mean()) ** 2).sum(). The best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a R^2 score of 0.0.

RESULTS:

The score of Linear Regression:

1. Without Augmentation: 0.7137960420794098
2. With Augmentation: 0.7186527248412415

The score of Random Forest Regression Tree:

1. Without Augmentation: 0.898467744143873
2. With Augmentation: 0.8986201451869025

TEAM MEMBERS:

1. Akshi (IMT2016025)
2. Shreya Singh (IMT2016045)
3. Nimisha Garg (IMT2016082)