Restaurant Visitor Forecasting

DS 5220: Supervised Machine Learning and Learning Theory

Nandavardhan Chirumamilla chirumamilla.n@northeastern.edu

Saketh Dathrika dathrika.s@northeastern.edu

Abstract

Our objective is to predict the number of customers a restaurant would have on any given day in the future. We have worked on a dataset containing visitor and reservation information of restaurants in Japan generated from two online reservation portals. After Exploring, Feature Engineering, and preprocessing the data, we have evaluated the performance of several supervised models such as K-Neighbors Regressor, Decision Tree Regressor, Random Forest Regressor, Light Gradient Boosted Machine Regressor, XG Boost Regressor and Long Short-Term Memory (LSTM). An ensemble of XG Boost and Light Gradient Boost regressor algorithms ended up as the best performing model among all of them, where Root Mean Square Logarithmic error has been chosen as the evaluation criteria.

1. Statement of Contribution:

Saketh Dathrika : Performed EDA, Feature Engineering, preprocessing and built Decision Tree Regressor, Light Gradient Boosting Regressor and LSTM Model and optimized the models built. Prepared presentation and report

Nandavardhan Chirumamilla : Performed EDA, Feature Engineering and built K-Neighbors Regressor, XG Boost Regressor and Random Forest Regressor and optimized the models built. Prepared presentation and report

2. Introduction:

2.1 Background and Goal of the Project

Running a thriving neighbourhood restaurant isn't always as smooth as it appears on the surface. Unexpected problems frequently arise, posing a threat to the company's success. A lot of factors influence the number of customers that a restaurant has on any given day, ranging from the day of the week to the weather conditions. This estimate is very important for restaurants as they need to know how many guests they may expect for a given day in order to successfully purchase products and arrange staff members and make other arrangements accordingly. The goal of the project is to forecast the total number of people who will visit a given restaurant on a given day in the future. This will assist restaurants in optimizing their processes and will be able to function as efficiently as possible, allowing them to focus on providing a pleasant dining experience for their patrons. We employ the use of several Supervised Machine Learning algorithms to deal with the problem at hand. By being able to estimate, the number of visitors at a restaurant using the given dataset, we would also be able to identify the key features that result in attracting more customers which may aid both existing and upcoming restaurants make suitable business decisions.

2.2 Dataset

The Dataset is taken from Kaggle: https://www.kaggle.com/c/recruit-restaurant-visitorforecasting/data

The dataset used in this project provides information on the visitations and reservations at various restaurants along with other information to anticipate restaurant visitor totals for a given day. The data comes from two different sources: Hot Pepper Gourmet (hpg): Like Yelp, this site allows users to

browse for restaurants and make reservations online. AirREGI / Restaurant Board (air): a reservation and cash register system similar to Square. The dataset spans the months of 2016 through April of 2017. The dataset has been divided into train, validation and test sets as seen in **Fig 1**. The training data contains 2,092,698 records, and the test data contains 32,020 records.

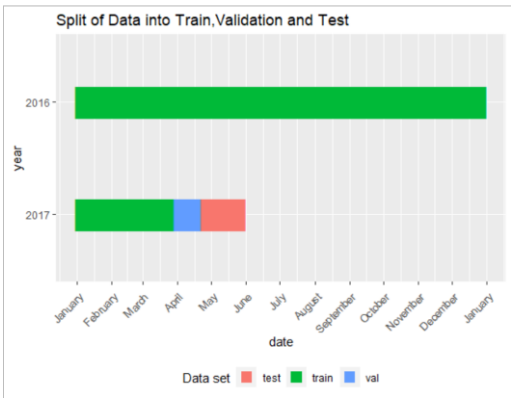

Fig 1: Visualization of the Train,Validation and Test splits in the dataset

The main attributes of the datasets and their descriptions are:

- air_store_id: The restaurant's unique id in the air store system
- hpg_store_id: The restaurant's unique id in the HPG system
- visit_datetime: The date and time of the reservation
- air_genre_name: Type of the restaurant (Example: Japanese, Italian, etc.) in the air store data
- air_area_name: Name of the area where the restaurant is located in the air store system
- hpg_genre_name: Type of the restaurant (Example: Japanese, Italian, etc.) in the HPG system
- hpg_area_name: Name of the area where the restaurant is located in the HPG system
- calendar_date: Date
- day_of_week: Day of the week
- holiday_flg: Shows if it's holiday or not for the given date
- reserve_datetime: The date and time when the reservation was made
- reserve_visitors: The number of visitors for that reservation
- latitude: The latitude of the area to which the store belongs to
- longitude: The longitude of the area to which the store belongs to

3. Exploratory Data Analysis:

Exploratory Data Analysis (EDA) helps us in analysing the data sets to visually summarize their characteristics. It helps us to see what the data can tell us beyond the formal modelling or statistical analysis task. Here, we have performed univariate and bivariate analysis to understand and interpret the different kinds of patterns that exist within the data. Throughout this section, we will be talking about some of the important trends and patterns discovered.

Let us begin with the analysis of the weekly and monthly trends present in the number of visitors that visit a restaurant. From **Fig 2.**, we can observe that the median number of visitors increases on Friday, Saturday and Sunday as compared to other days of the week. This insight is clearly inline, with the general intuition that people tend to visit restaurants more during the weekend than the weekdays, which are usually workdays. In **Fig 3.**, where the median number of customers for each month is presented, we can observe the monthly trend that the median number of customers is higher during

the months of December, March, April and May as compared to the other months throughout the year.

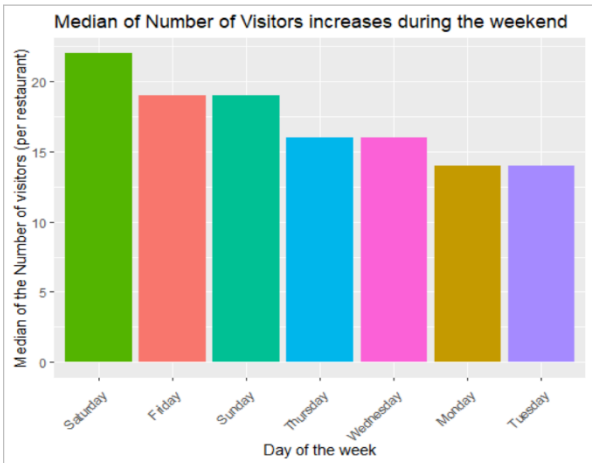

Fig 2: A plot showing the Median Number of Visitors that visit a restaurant for each day of the week

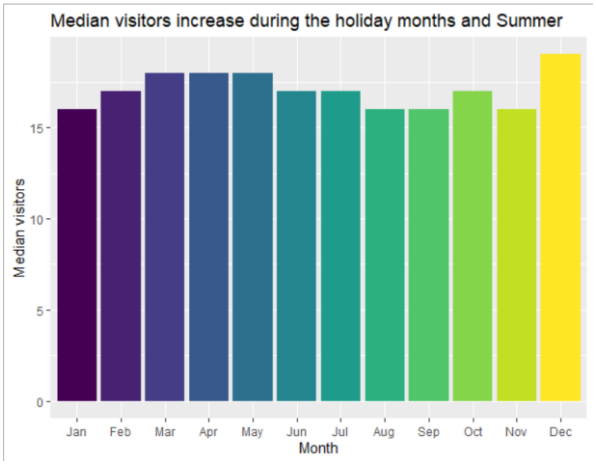

Fig 3: A plot showing the median of the average number of visitors that visit a restaurant for each month in a calendar year

In **Fig 4.**, we can see that average number of visitors per restaurant increases significantly when it's a holiday as compared to non-holidays for all days of the week except Saturday. Similar insights have been generated throughout the EDA process. We have also noticed trends within genre and areas as well. While most genre types had a consistent distribution of visitors, some had slight positive and negative slopes in their plots suggesting increase and decrease in popularity respectively. We have also noticed an impact of the number of restaurants per genre per area influence the number of visitors at those restaurants.

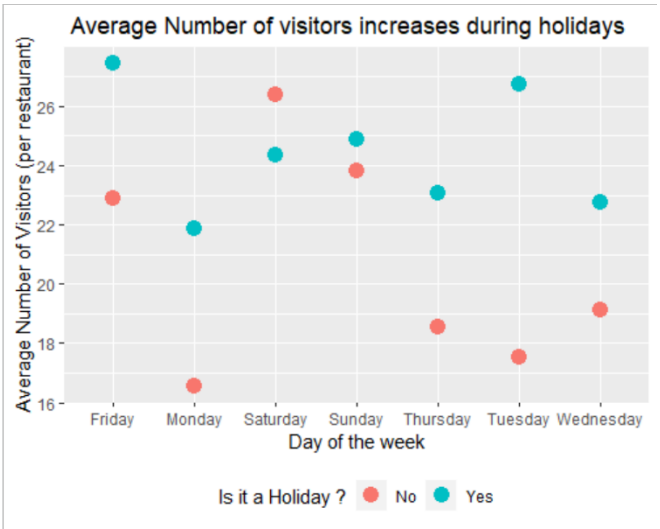

Fig 4: Plot showing the difference between the average number of visitors during holidays and non-holidays for each day of the week

4. Feature Extraction and Pre-processing:

The step of feature extraction involves the extraction, selection and construction of new features. The goal is to be able to produce new features using existing features that helps better model the variation in the data. We have produced 9 new features during this stage in the process. After performing visual analysis and exploration of data, we have introduced new statistical features based on the number of visitors for a given day of the week at a restaurant. After which, we have introduced features based on intuitional logic and through more statistical analysis.

List of features introduced :

1. Mean visitors: Mean of the total number of visitors for a given day of the week
2. Median visitors: Median of the total number of visitors for a given day of the week

3. Minimum visitors: Minimum of the total number of visitors for a given day of the week
4. Maximum visitors: Maximum of the total number of visitors for a given day of the week
5. Count of visitors: Count of the total number of visitors for a given day of the week
6. Number of restaurants per genre per area
7. Sum of the number of visitors through reservations for each restaurant Id and date is calculated for both air store reservation and HPG system reservation data.
8. Mean of the number of visitors through reservations for each restaurant Id and date is calculated for both air store reservation and HPG system reservation data
9. Time interval between reservation booking and visitation of the restaurant (in hours)

After introducing the new features, we have performed quantitative analysis of all features w.r.t to the target variable and each other, to identify and select variables with the highest correlation to the target variable. The quantitative analysis involved performing Chi-square tests to quantify the correlation between continuous and categorical values. We have also computed the Pearson's correlation matrix to quantify the correlation between two continuous values, as seen in **Fig 5**.

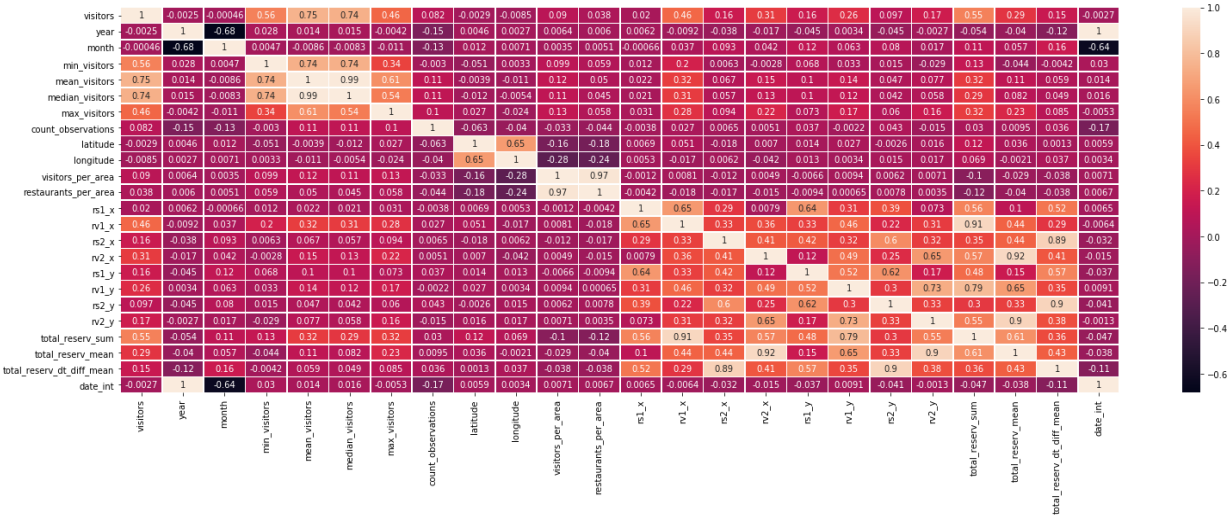

Fig 5: Pearson Correlation matrix between the continuous features in our data

Following the feature selection, we have pre-processed the data to make it suitable for modelling.

It involved the following transformations:

- Label encoding of all categorical variables
- Transforming data into the corresponding correct data types
- Checking for and impute missing values

5. Models, Hyperparameter tuning and Experimentation:

The goal of our project is to predict the number of visitors for a given restaurant in the future, for which we have curated 6 regression modelling techniques. Regression analysis is a predictive modelling technique that examines the relationship between the target or dependent variable and the independent variables. For most of the algorithms, there are multiple hyperparameters to choose from and identifying the optimal hyperparameters to attain an optimal model is a challenge. We have performed grid search for each of the algorithms limited to certain specific hyperparameters rather than extensive grid search on all hyperparameters and all combinations due to hardware and time constraints. ***GridSearchCV*** aids us in iterating over all the specified combinations of specified

hyperparameters and fits the estimator on the training data. We can obtain the optimal set of hyperparameters from the above function itself.

The 6 models which we implemented are:

5.1 K-Neighbors Regressor :

The K-Neighbours algorithm is a non-parametric Supervised algorithm. The KNN regressor algorithm is used to predict a continuous target variable for a data sample by using the mean of the nearest K neighbours to the given data sample. The KNN algorithm involves the fine tuning of several hyperparameters. For example, there are several distance metrics to choose from to calculate the K closest samples like the Euclidean, Minkowski, Chebyshev, or Manhattan distance. The distance functions used in KNN regression are the same as those used in KNN classification. The KNN algorithm involves the fine tuning of several hyperparameters. After implementing grid search, we achieved the best performance with number of neighbors as 11 with minkowski as the distance metric using the ball tree algorithm. **Fig 6** shows the impact of the change in number of neighbours on RMSLE.

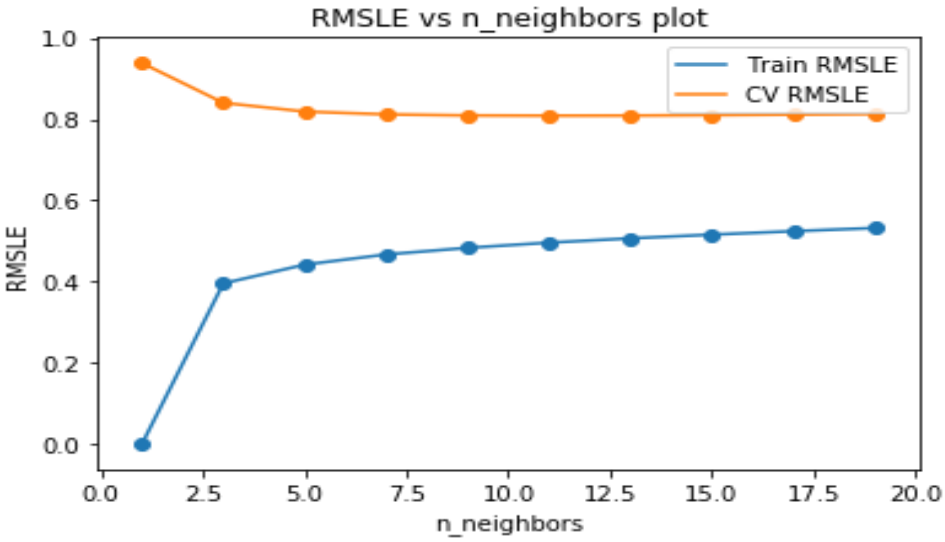

Fig 6: Plot of RMSLE of the train and cross validation data vs the K-neighbors

After fitting the model with the optimal set of hyperparameters, we evaluated the performance of the model on validation and test data, whose performance is shown in **Table 1**. As it can be observed in **Table 1**, model has achieved RMSLE of 0.56 for the test data.

Metrics	RMSLE	RMSE	MSE	MAE
Train data	0.496	10.717	114.85	6.731
Validation data	0.586	13.073	170.92	8.201
Test data	0.563	-	-	-

Table 1: Evaluation of the performance of Best performing KNN model

5.2 Decision Tree Regressor:

Decision tree builds a regressor is in the form of a tree structure. It incrementally cuts down a dataset into smaller and smaller sections while also developing an associated decision tree. A tree with

decision nodes and leaf nodes is the final result. A decision node can have two or more branches, each of which represents a value for the attribute being checked. A decision on the numerical target is represented as a leaf node (e.g., Visitors forecast). The root node is the topmost decision node in a tree that corresponds to the best predictor. We ensure that the best possible split is made at each node using the Gini impurity. Both categorical and numerical data can be handled by decision trees.

After using *GridSearchCV,* we got the optimal model with maximum depth of tree as 10 and the minimum number of samples to split a node at 500 which achieved an RMSLE score of 0.525 on the test data. **Fig 7** shows the input features in the order of importance according to the trained decision tree regressor. The features are obtained from the score which is based on impurity or gini index. As shown in **Fig 7**, *mean_visitors* and *date_int* are the two most important input features in making predictions about the number of visitors at a restaurant. **Table 2** shows evaluations of the optimal model on train, validation and test data.

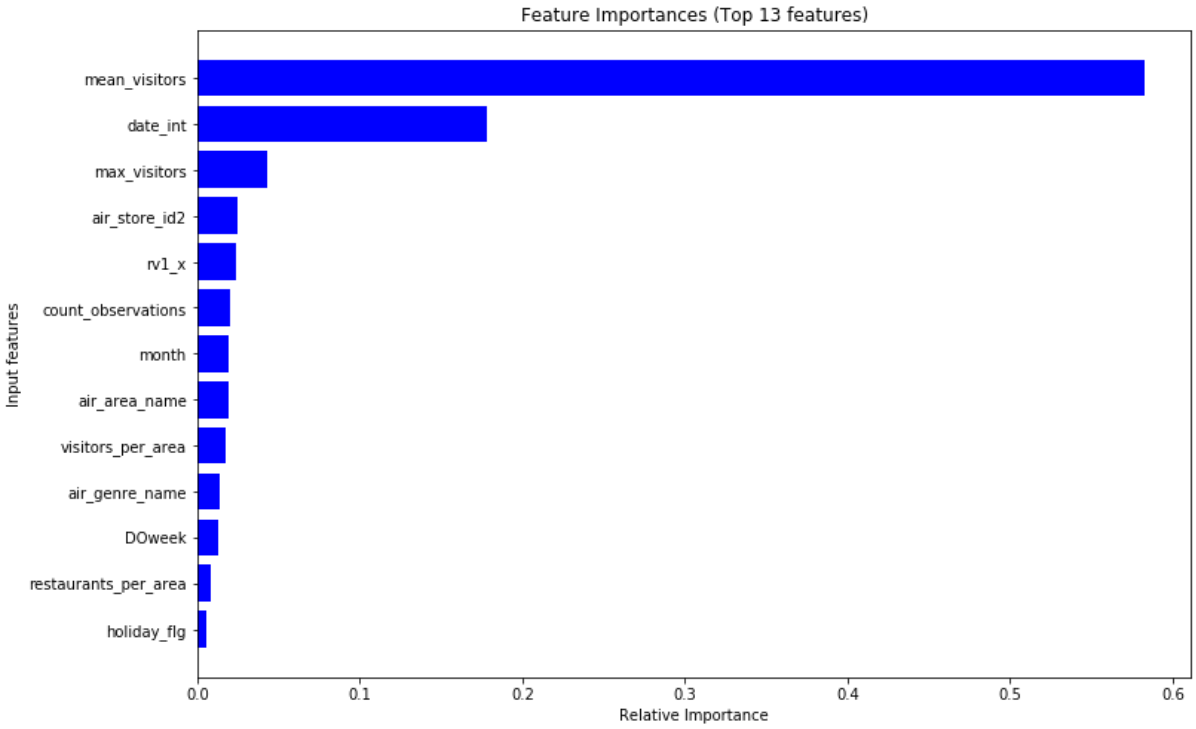

Fig 7: Plot showing the importance of each input feature as per the trained decision tree regressor model

Metrics	RMSLE	RMSE	MSE	MAE
Train data	0.509	11.03	121.722	6.901
Validation data	0.508	11.760	138.314	6.971
Test data	0.525	-	-	-

Table 2: Evaluation of the performance of Best performing Decision Tree Regressor model

5.3 Random Forest Regressor :

Random Forests is an ensemble algorithm that is based on the concept of bagging. It is based on the intuition that a large number of uncorrelated models, will outperform any of the individual models.

The ensemble learning method combines predictions from several machine learning algorithms to get a more accurate forecast than any single model. During training, it constructs numerous decision trees and outputs the mean of the predictions of all the decision trees.

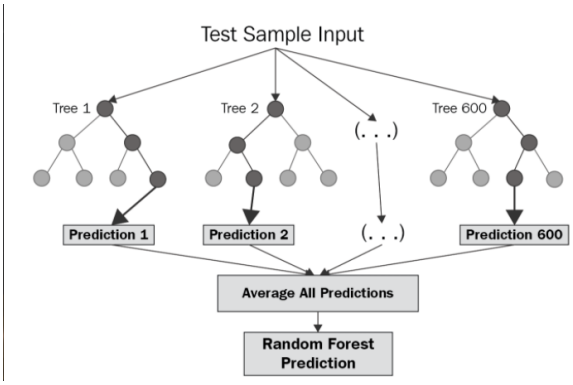

Fig 8: Sample working of the Random Forest Regressor Algorithm

Since *GridSearchCV* was taking too long to compute for the random forest regressor model, we have used ***HalvingGridSearchCV***. In this method, all parametric combinations are evaluated with a small number of resources in the first iteration. Only some of these candidates are selected for the next iteration, which will be allocated more resources and so on. Using this function, we have attained the optimal model with maximum depth of each decision tree set to 5, the minimum number of samples to split a node to 5 and 200 as the total number of individual decision trees. Our best model has achieved a RMSLE of 0.513 on validation data and 0.53 on test data, as seen in **Table 3**.

Metrics	RMSLE	RMSE	MSE	MAE
Train data	0.519	11.182	125.055	7.0659
Validation data	0.513	11.837	140.123	7.060
Test data	0.530	-	-	-

Table 3:Evaluation of the performance of the Best performing Random Forest Regressor model

5.4 XG Boost Regressor:

Extreme Gradient Boosting (XGBoost) uses decision tree-based ensemble Machine learning algorithm that provides an efficient and effective implementation of the gradient boosting algorithm. XGBoost is an optimized gradient boosting algorithm through parallel processing, tree-pruning, handling missing values and regularization to avoid overfitting/bias. It is a perfect combination of software and hardware optimization techniques to yield superior results using less computing resources in the shortest amount of time. **Fig 9** shows the parameters that have given as us the optimal model, for which we have attained an RMSLE of 0.511 for both validation and test data as seen in **Table 4**.

Tuned Hyperparameters :

- *Colsample_bytree: It is the fraction of features (randomly selected) that will be used to train each tree*
- *Learning_rate: It is the shrinkage you do at every step you are making*
- *Max_depth: It is the maximum height to which each tree is allowed to grow*
- *Min_child_weight: The building process will stop dividing if the tree partition step results in a leaf node with a sum of instance weight less than min child weight*

- *Subsample: It helps to specify the percentage of rows used per tree building iteration*

```
xgb_reg_cv.best_params_
```

```
{'colsample_bytree': 0.5,
 'learning_rate': 0.1,
 'max_depth': 4,
 'min_child_weight': 0.8,
 'subsample': 0.5}
```

Fig 9: Hyperparameters that give us the optimal XG Boost regressor model

Metrics	RMSLE	RMSE	MSE	MAE
Train data	0.486	10.522	110.717	6.559
Validation data	0.511	11.693	136.734	6.947
Test data	0.511	-	-	-

Table 4: Evaluation of the performance of the Best performing XG Boost Regressor model

5.5 Light Gradient Boosted Machine Regressor:

It is also a type of gradient boosting algorithm which includes adding a type of automatic feature selection as well as focusing on boosting examples with larger gradients. This can result in a dramatic speedup of training and improved predictive performance. It is different from the rest of the tree algorithms as it grows the tree vertically i.e., leaf wise whereas many other algorithms grow horizontally. **Fig 10** shows the parameters that have given as us the optimal model, for which we have attained an RMSLE of 0.51 for both validation and test data as seen in **Table 5**.

```
lgb_reg_cv.best_params_
```

```
{'colsample_bytree': 0.5,
 'learning_rate': 0.1,
 'min_child_weight': 0.001,
 'subsample': 0.5}
```

Fig 10: Hyperparameters that give us the optimal LGBM model

Tuned Hyperparameters:

- *Colsample_bytree: It is the fraction of features (randomly selected) that will be used to train each tree*
- *Learning_rate: It is the shrinkage you do at every step you are making*
- *Min_child_weight: The building process will stop dividing if the tree partition step results in a leaf node with a sum of instance weight less than min child weight*
- *Subsample: It helps to specify the percentage of rows used per tree building iteration*

Metrics	RMSLE	RMSE	MSE	MAE
Train data	0.496	10.807	116.656	6.738
Validation data	0.511	11.557	133.569	6.877
Test data	0.513	-	-	-

Table 5: Evaluation of the performance of the Best performing LGBM Regressor model

5.6 LSTM:

Long short-term memory networks are a type of recurrent neural network(RNN) that can learn the time sequence data in prediction challenges, where the outputs of previous time steps influence the output at the current time step. RNN uses feedback loops which helps to retain the information in it and gets output from the last step that is fed back to the model as input in the next iteration. An advantage of LSTM over RNNs is that, LSTMs overcome the long term memory retention challenge.

```
# create and fit the LSTM network
def create_LSTM(X,y):
    model = Sequential()
    model.add(LSTM(100, input_shape=(1, X.shape[2])))
    model.add(Dense(10))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
    model.fit(X, np.log1p(y['visitors']), epochs=100, batch_size=1000, verbose=2)
```

Fig 10: Architecture of best performing LSTM Network

It was quite a challenge to come up with the optimal LSTM network for the problem at hand. A common trend that we noticed was that the test error was increasing significantly as the model complexity increased. Hence, we proceeded with a simpler model. Mean-squared-error was used to compute the loss during training. We have trained the LSTM for about 100 epochs. The results obtained using LSTM can be seen in **Table 6**.

Metrics	RMSLE	RMSE	MSE	MAE
Train data	0.806	17.511	306.650	12.105
Validation data	0.830	19.325	373.469	12.825
Test data	0.8353	-	-	-

Table 6: Evaluation of the performance of the Best performing LSTM model

6. Evaluation Metrics:

For evaluating the models, we chose four regression evaluation metrics. They are:

- **RMSLE**: It is the root mean squared error of log of actual values and log of predicted values.
- **RMSE**: It is the root of average squared distance between the real and predicted variables.
- **MSE**: It is the average squared distance between the actual and predicted ones.
- **MAE**: It is the average of absolute value of the difference between the forecasted value and the actual value.

7. Results:

Github Link: https://github.com/Nandu960/Restaurant-Visitor-Forecasting

In Kaggle, Submissions are evaluated only using the root mean squared logarithmic error on test data. Hence, we only have RMSLE scores for the sample_submission i.e., test data. The comparative results for the all the different models are shown in **Table 7.**

Model	RMSLE
K Neighbors	0.563
Decision tree	0.525
Random forest	0.530
XG boost	0.511
LGBM regressor	0.513
LSTM	0.835

Table 7: Evaluation of the performance of the Best performing LSTM model

8. Conclusion:

- Among all the different algorithms that were used, XGboost Regressor is the best performing model.
- LGBM Regressor performed well but with high training and testing time.
- Decision tree also executed moderately well.
- LSTM has the high error rate when compared with the other algorithms.

9. Future Work:

- Adding weather data to the input data may help get better predictions as we would be factoring in more relevant features.
- Implementing ARIMA and SARIMA model for each restaurant so that it would be able to predict the number of visitors for any given future dates of any restaurant in the data
- Implementing ARIMA and SARIMA model for each genre so that it would be able to predict the number of visitors for any given future dates of any restaurant based on the genre

References:

1 *Recruit Restaurant Visitor Forecasting*. Kaggle. (n.d.). Retrieved December 15, 2021, from https://www.kaggle.com/c/recruit-restaurant-visitor-forecasting/data .

2 Patil, P. (2018, May 23). *What is Exploratory Data Analysis?* Medium. Retrieved December 13, 2021, from https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15.

3 Wikipedia. (2021, November 15). *Exploratory Data Analysis*. Wikipedia. Retrieved December 13, 2021, from https://en.wikipedia.org/wiki/Exploratory_data_analysis

4 Nelson, D. (2019, August 8). *Gradient Boosting Classifiers in Python with Scikit-Learn*. Stack Abuse. Retrieved December 13, 2021, from https://stackabuse.com/gradient-boosting-classifiers-in-python-with-scikit-learn/.

5 Yiu, T. (2021, September 29). *Understanding Random Forest*. Medium. Retrieved December 13, 2021, from https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

6 Aliyev, V. (2020, October 7). *Gradient Boosting Classification Explained through Python*. Medium. Retrieved December 13, 2021, from https://towardsdatascience.com/gradient-boosting-classification-explained-through-python-60cc980eeb3d.

7 Gupta, P. (2017, November 12). *Decision Trees in Machine Learning*. Medium. Retrieved December 13, 2021, from https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052.

8 Abhigyan. (2021, July 26). *Different types of hyper-parameter tuning.* Medium. Retrieved December 14, 2021, from https://medium.com/analytics-vidhya/different-types-of-hyper-parameter-tuning-3d99ca624baa.

9 Bakshi, C. (2020, June 9). *Random Forest regression*. Medium. Retrieved December 14, 2021, from https://levelup.gitconnected.com/random-forest-regression-209c0f354c84.

10 Brownlee, J. (2021, July 6). *A gentle introduction to long short-term memory networks by the experts*. Machine Learning Mastery. Retrieved December 14, 2021, from https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/.

11 Decision tree regression. (n.d.). Retrieved December 14, 2021, from https://www.saedsayad.com/decision_tree_reg.htm.

12 *K-Nearest Neighbors Algorithm: KNN Regression Python*. Analytics Vidhya. (2020, May 25). Retrieved December 14, 2021, from https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/.

13 Mandot, P. (2018, December 1). *What is LIGHTGBM, how to implement it? how to fine tune the parameters?* Medium. Retrieved December 14, 2021, from https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc .

14 Morde, V. (2019, April 8). *XGBoost algorithm: Long may she reign!* Medium. Retrieved December 14, 2021, from https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d .

15 Raj, A. (2021, June 11). *A quick and dirty guide to random forest regression*. Medium. Retrieved December 14, 2021, from https://towardsdatascience.com/a-quick-and-dirty-guide-to-random-forest-regression-52ca0af157f8.