RECRUIT RESTAURANT VISITOR FORECASTING

Team Name: Invincible Predictors

Apoorv Panse
MT2020013
apoorv.panse@iiitb.org

Deepti Chawda
MT2020050
deepti.chawda@iiitb.org

Shahbaz Khan
MT2020160
Shahbaz.khan@iiitb.org

Abstract—**This is the detailed project report on our work on predicting the number of visitors for the set of restaurants based on the dataset given to us. This report contains various phases of analysis and prediction starting from initial dataset analysis, Exploratory data analysis, Feature engineering, Model selection and training, and finally choosing the best model for predictions.**

I. Introduction and Problem Statement

Recruit holdings owns Hot Pepper Gourmet (a restaurant review service), AirREGI (a restaurant point of sales service), and Restaurant Board (reservation log management software). We were challenged to use reservation and visitation data to predict the total number of visitors to a restaurant for future dates. This information would help restaurants be more efficient in resource management and it will allow them to create much more effective dining experience for their customers.

II. Data set

The data comes from two separate sites:

- Hot Pepper Gourmet (hpg): like Yelp, here users can search restaurants and make a reservation online
- AirREGI / Restaurant Board (air): like Square, a reservation control, and cash register system.

The datasets contain daily and hourly observations. This makes it a Time Series Forecasting problem.

Air_reserve.csv
This file contains reservations made in the air system.

- air_store_id - the restaurant's id in the air system
- visit_datetime - the time of the reservation
- reserve_datetime - the time the reservation was made
- reserve_visitors - the number of visitors for that reservation

hpg_reserve.csv
This file contains reservations made in the hpg system.

- hpg_store_id - the restaurant's id in the hpg system
- visit_datetime - the time of the reservation
- reserve_datetime - the time the reservation was made
- reserve_visitors - the number of visitors for that reservation

air_store_info.csv
This file contains information about select air restaurants. Column names and contents are self-explanatory.

- air_store_id
- air_genre_name
- air_area_name
- latitude
- longitude

hpg_store_info.csv
This file contains information about select hpg restaurants. Column names and contents are self-explanatory.

- hpg_store_id
- hpg_genre_name
- hpg_area_name
- latitude
- longitude

store_id_relation.csv
This file allows you to join select restaurants that have both the air and hpg system.

- hpg_store_id
- air_store_id

train.csv
This file contains historical visit data for the air restaurants.

- air_store_id
- visit_date - the date
- visitors - the number of visitors to the restaurant on the date

sample_submission.csv
This file shows a submission in the correct format, including the days for which you must forecast.

- id - the id is formed by concatenating the air_store_id and visit_date with an underscore
- visitors- the number of visitors forecasted for the store and date combination

date_info.csv
This file gives basic information about the calendar dates in the dataset.

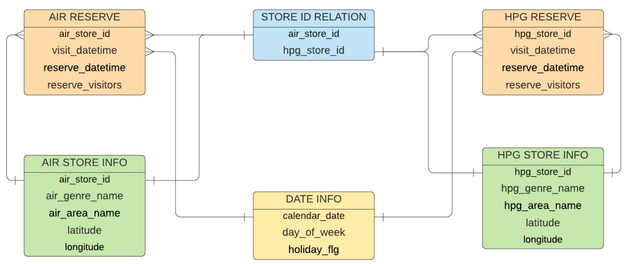
- calendar_date
- day_of_week
- holiday_flg - is the day a holiday in Japan

III. Exploratory Data analysis

First, we observed the given data set without performing any merge operation to get the idea of kind of data that we have. Hence, we call it pre-exploratory analysis. Below were the findings:

- We have total 92378 rows in air_reserve table.
- We have at least 1 visitor for all rows.
- Maximum visitor count was much higher than mean visitors count.
- There were no null values in any of the tables.

How the metadata was linked:



Now after doing the pre analysis, we got the idea how to merge the data sets and perform the exploratory analysis.

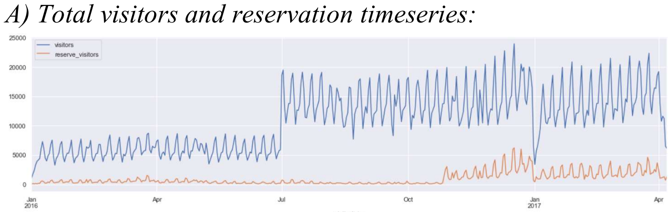
Merged data contained following columns which were used for data analysis:

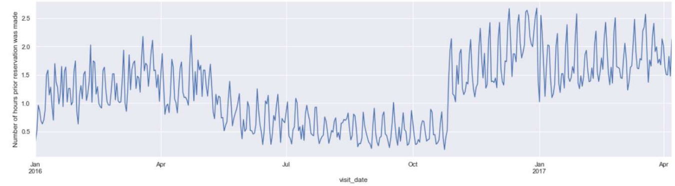
#	Column	Non-Null Count	Dtype
0	air_store_id	239673 non-null	category
1	visit_date	239673 non-null	datetime64[ns
2	visitors	239673 non-null	int64
3	air_genre_name	239673 non-null	category
4	air_area_name	239673 non-null	category
5	reserve_visitors	239673 non-null	float64
6	Time_Difference	239673 non-null	float64
7	visit_time	239673 non-null	float64
8	reserve_time	239673 non-null	float64
9	visit_year	239673 non-null	int64
10	visit_month	239673 non-null	int64
11	visit_weekday	239673 non-null	int64
12	city	239673 non-null	category
13	ward	239673 non-null	category
14	neighborhood	239673 non-null	category
15	holiday_flg	239673 non-null	int64

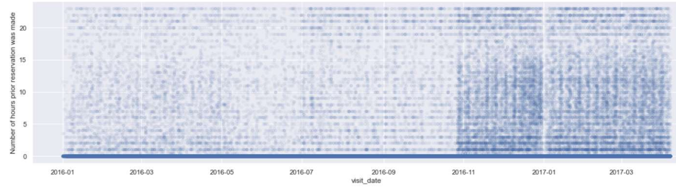
Date column was broken down into visit year, visit month and visit weekday columns. Area information was broken down into city ward and neighborhood for better EDA.

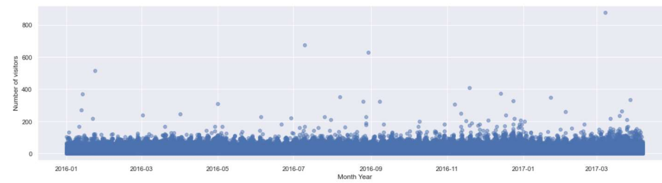
Few important observations that we extracted from data is as follows:

A) Total visitors and reservation timeseries:





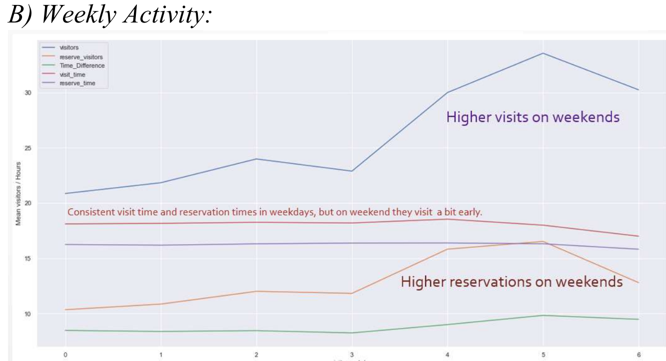




Insights derived:

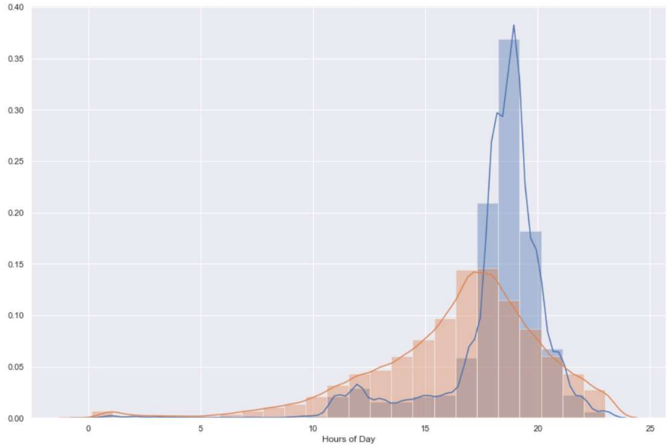
- Most of the restaurants have no prior reservations.
- Starting from November 2016, number of reservations started to grow. Which implies that new restaurants were onboarded to take reservations in advance.
- The dip during new year is caused because people prefer to spend new year with their families at home. First week of January sees less visitors.
- The reason of sudden increase in number of visitors in July 2016 is because many new restaurants were
- added in the database
- Higher number of reservations are seen after November 2016 and hence we can clearly see that people need to book the restaurants much earlier than previously. People tend to reserve less during July to October. This can be a seasonal thing.
- For most of the restaurants we do not have reservations data. Hence most of the entries are zero. Which means many restaurants do not provide reservation facility.
- After November, prior reservations were made even 24 hours before in many restaurants.
- We see the outliers here, where in a single day restaurant have more than 400 visitors. We can filter out outliers if needed and focus more on restaurants that have less than 200 visitors

B) Weekly Activity:



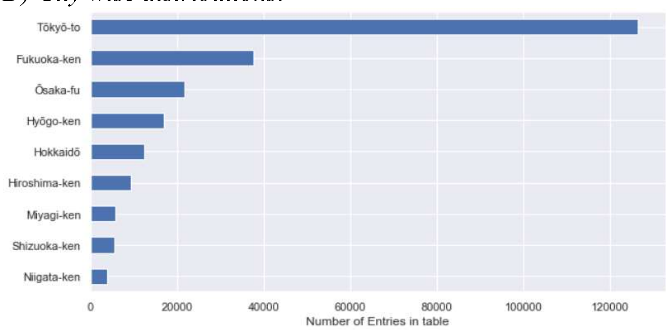
- Fridays and Saturday sees a greater number of visitors on average.
- Monday has lowest visitors.
- There seems to be high number of visitors when prior reservations were high.
- Day of week contributes significantly to the number of visitors.
- More reservation means people must book earlier to get the seat.

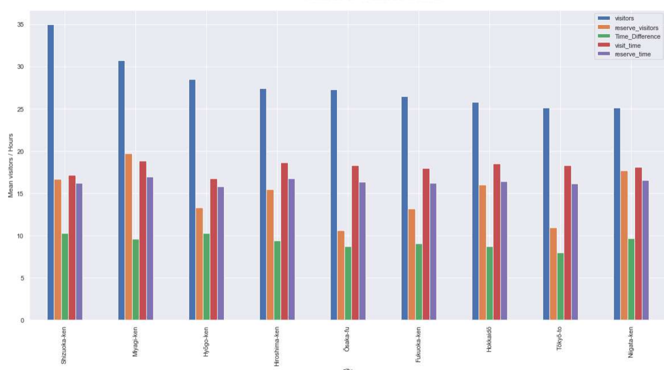
C) Hourly visit and visitor distribution:



- People tend to visit the restaurants in the evening between 6 PM to 8 PM
- Most reservations are made between 3 PM and 7 PM
- Hence most people tend to have dinner rather than lunch.

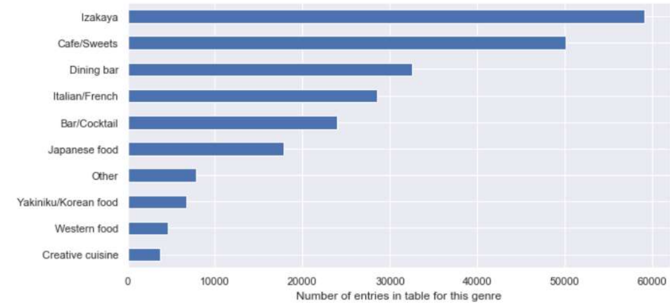
D) City wise distributions:

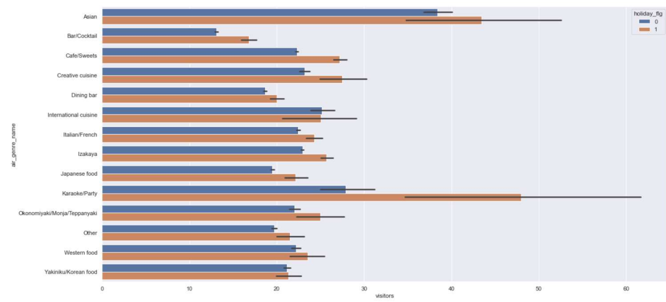


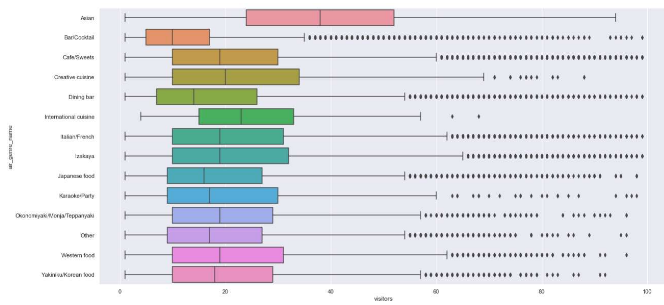


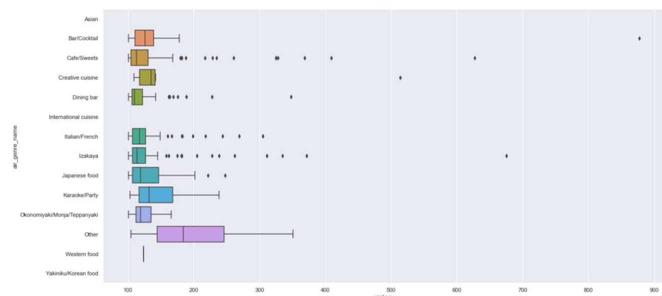
- Tokyo is the most popular city followed by Fukaoka and Osaka.
- These areas have highest number of restaurant and factually these areas are populous
- Earlier we saw Tokyo, Fukuoka and Osaka were most populous cities. But Shizuoka, Myagi, Hyogo have higher number of mean visitors. This implies that Shizuoka have a smaller number of restaurants but more visitor capacity.
- People of Myagi seems to reserve more before the visit.
- Tokyo must be having a greater number of small restaurants as mean visitor count is less.
- Visit time is almost consistent in all the cities, that is in the evening.

E) Genre wise distributions:



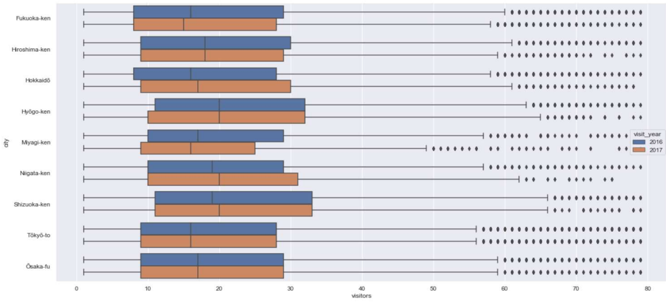


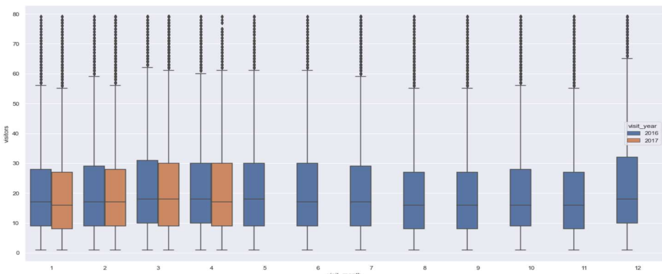




- Izakaya, café and Dining bars are the most popular genre and number of restaurants serving this genre is high.
- There are more visits on holidays compared to normal days.
- Asian and Karaoke Party genre seems to have more average visitors, especially on holidays.
- Our earlier question about restaurants being closed or open on holidays, is resolved. Restaurants are open on holidays Asian genre seems to have more average visitors, especially on holidays.
- Even though Izakaya has the greatest number of entries in table, we have comparatively less visitors which implies that restaurants serving Izakaya must have small visitor capacity.
- Many big restaurants throw Karaoke Parties, Serve Japanese food.
- Even some cafes accommodate more visitors.
- Almost all of the restaurants serving Asian / Korean / Western food / International cuisine have small accommodation capacity.
- Size of restaurant is directly proportional to number of visitors, hence Genres they serve play important role.

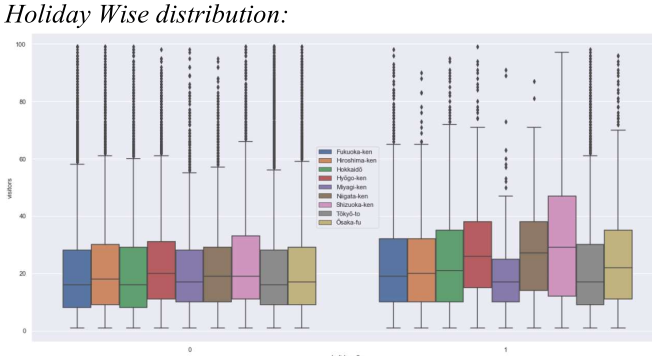
F) Year Wise distribution:

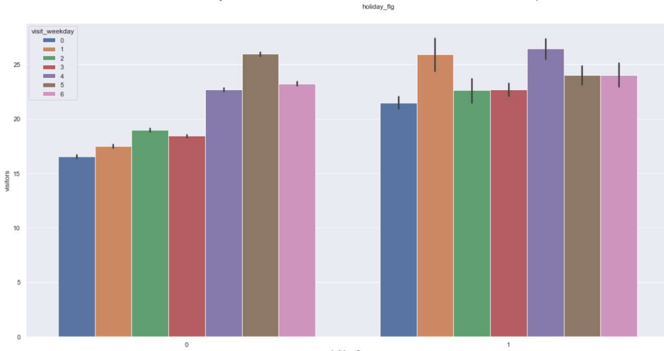




- Here we can see Fukuoka and Myagi saw less visits compared to previous year.
- Whereas Hokkaido, Shizuoka, Nigata saw slight increase in number of visitors.
- Tokyo, Osaka, Hiroshima, Hyogo saw consistent activity the average visitors seems to be concentrated between the range 10 to 30. So now we can expect more predictions in this range.
- We have excluded the outliers and considered max visitors as 80 for this chart.

G) Holiday Wise distribution:





- Shizuoka is most active on holidays.
- Myagi seems to be less active on holidays.
- We can still see higher activity on holidays on average
- We can see when there is no holiday, there are less visitors on weekday.
- When there are holidays, even Monday, Tuesday have more visitors.

IV. Feature Engineering

Now that we explored our data in and out, it was easy to figure out which features we want to include in our training set.

We could see that Weekday, Holiday, Genre, Area played an important role in the number of visitors. Hence, we added related features to our training data from existing columns. Final set of features looked as below:

#	Column	Non-Null Count	Dtype
0	air_genre_name	225227 non-null	int32
1	latitude	225227 non-null	float64
2	visit_year	225227 non-null	int64
3	visit_month	225227 non-null	int64
4	visit_weekday	225227 non-null	int64
5	city	225227 non-null	int32
6	ward	225227 non-null	int32
7	neighborhood	225227 non-null	int32
8	holiday_flg_0	225227 non-null	uint8
9	holiday_flg_1	225227 non-null	uint8
10	mean_visitors	225227 non-null	float64
11	median_visitors	225227 non-null	float64
12	min_visitors	225227 non-null	float64
13	max_visitors	225227 non-null	float64

- Label encoding was used for categorical features having more than two categories.
- One hot encoding was used for categorical features having binary categories.
- Mean, median, min, max visitor features were added for better prediction possibility. These aggregations were done by grouping columns air_store_id and visit_weekday.

V. Model selection and model training

Now that we have our final set of features, we can decide how to split the set. Firstly, for training purpose, we decided to split the data into 80%-20% train-test ratio.

```
#train test split
from sklearn.model_selection import train_test_split
X = train_data.drop(["air_store_id","visit_date","visitors","air_area_name","longitude"], axis=1)
y = train_data["visitors"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1)
```

- air_store_id, visit_date, air_area_name, and longitude columns were dropped for training as they are not taken as features.
- Random split was done because we saw that there is not much difference in the average visitor year wise.
- Note that we have split train.csv into two parts here and test data is not from sample submission file.

As predicting the number of visitors is the regression problem, we have tried few regression models :

* Simple Linear Regression
* KNeighbors Regression
* Random Forest Regression

Evaluation Metric Used was RMSLE: Root mean squared logarithmic error. The RMSLE is calculated as

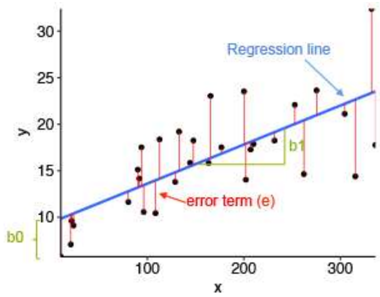
$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i+1)-\log(a_i+1))^2},$$

where:

n is the total number of observations
p_i is your prediction of visitors
a_i is the actual number of visitors
$\log(x)$ is the natural logarithm of x

Model 1: Linear Regression.

- We used scikit-learn library for implementing Linear regression model.
- It uses Ordinary least squares Linear Regression.
- Linear Regression fits a linear model with coefficients to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.



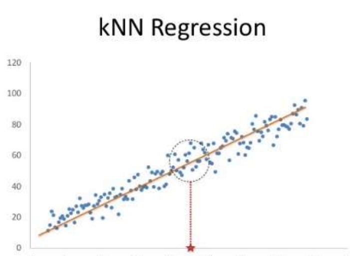

```
#Trying simple Linear Regression model

from sklearn.linear_model import LinearRegression
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
y_preds=lr_model.predict(X_test)
rmsle(y_test, y_preds)
```

```
0.5369255002063672
```

Model 2: KNeighbors Regression.

- Regression based on k-nearest neighbors.
- *The target is predicted by local interpolation of the targets associated of the nearest neighbors in the training set.



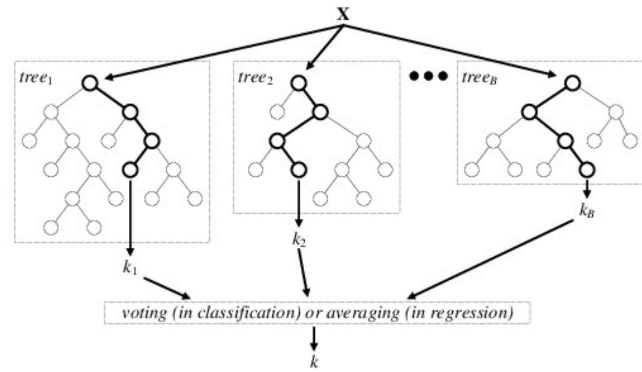

```
#Trying KNeighbors Regression model

from sklearn.neighbors import KNeighborsRegressor
knr_model = KNeighborsRegressor(n_jobs=-1, n_neighbors=10)
knr_model.fit(X_train, y_train)
y_preds=knr_model.predict(X_test)
rmsle(y_test, y_preds)
```

```
0.5398689672131062
```

Model 3: Random Forest Regression.

- A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.




```
#Trying Random Forest Regressor Regression model

from sklearn.ensemble import RandomForestRegressor

rfrmodel = RandomForestRegressor(n_estimators=200, min_samples_leaf=5,
                                 min_samples_split=15,
                                 max_features=1, n_jobs=-1,
                                 )

rfrmodel.fit(X_train, y_train)
y_preds=rfrmodel.predict(X_test)
rmsle(y_test, y_preds)
```

```
0.5289086869919264
```

Model 4: XGboost Regression

- XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible
- Implements machine learning algorithms under the Gradient Boosting framework.
- XGBoost provides a parallel tree boosting.

```
#Trying XGBoost Regression model

from xgboost import XGBRegressor
xgbmodel = XGBRegressor(
                        max_depth =16,
                        learning_rate=0.1,
                        n_estimators=20,
                        subsample=0.4,
                        colsample_bytree=0.8,
                        seed=5
                       )
xgbmodel.fit(X_train,y_train)

y_preds_xgb=xgbmodel.predict(X_test)
rmsle(y_test,y_preds)
```

```
0.5188255921668121
```

RMSLE ON VARIOUS ALGORITHMS

1 Linear Regression	0.536
2 KNeighbors Regression	0.539
3 Random Forest Regression	0.528
4 XGBoost Regresssion	**0.518**

As XGBoost gave better results, we selected that model for final training. For final training, the entire train data set was passed for model training and Submission dataset was converted in the same format by feature engineering, and Submission data became the test set.

The final submission score in Kaggle was: **0.526**

VI. Conclusion

With the given dataset we think the result is satisfactory. We could have included external data set such as weather data for better prediction. Also, we could have used Time series forecasting methods such as ARIMA for better prediction. However, from curriculum perspective and to learn various regression methods, we preferred Standard regression models that were available in Scikit learn library.

VII. Acknowledgements

We would like to thank our Teaching Assistant Arjun Verma for guiding us regarding what to expect from the project and what steps we need to take for successfully implementing the project. Especially his advice to use Gradient Boosting models to improve our score. Special thanks to Raghavan Sir and Neelam mam for teaching us the basic concepts of Machine Learning. The lectures were detailed and helped us understand how the regression models work.

VIII. Reference

Websites:

[1] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

[2] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html

[3] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

[4] https://www.kaggle.com/c/recruit-restaurant-visitor-forecasting/overview

[5] https://xgboost.readthedocs.io/en/latest/

Books and Lectures:

[6] *Learning from Data*, *A short course* by Yaser S Abu. Mostafa, Malik-Magdon Ismail, Hsuan-Tein Lin.

[7] *Mathematics for Machine Learning* by Marc Peter Deisenroth, A. Aldo Faisal, and Cheng Soon Ong.

[8] *Class lecture videos and slides* by Prof. Raghavan and Prof. Neelam, IIIT Bangalore.

[9] *Sample Project Reports* by Tejas Kotha and Arjun Verma.