Model Building and Predictions

TEAM INVINCIBLE PREDICTORS

*APOORV PANSE

*DEEPTI CHAWDA

*SHAHBAZ KHAN

Recruit restaurant visitor forecasting project

Updated Data Merging after EDA

- We found out that HPG tables were not required to be merged at all as our training set contains actual visitors data and we do not have reservation data available for most of it.
- So we only merged air_store_info and date_info tables with train dataset to get final training dataset and added extra features as required.
- Overall we played around with only 3 CSV files for model training :
  train.csv, air_store_info.csv, date_info.csv
  and merged these to form our train_data dataframe.

Final Set of features for Model Training

#	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

#	Column	Non-Null Count	Dtype
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.

Training Testing Splits

```
#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)
```

*Train and test data was split in 80% and 20% respectively.
*air_store_id, visit_date, air_area_name, and longitude columns were dropped for training as they are not taken as features as they were not improving the score.
*Random split was done because we saw that there is not much difference in the average visitors year wise.
*Note that we have split train.csv into two parts here and test data is not from sample submission file.

Model Selection and evaluation metric

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)

```
# Create evaluation function (the competition uses Root Mean Square Log Error)
from sklearn.metrics import mean_squared_log_error

def rmsle(y_test, y_preds):
    return np.sqrt(mean_squared_log_error(y_test, y_preds))
```

Model 1 : Linear Regression

*We used scikit-learn library for implementing Linear regression model.

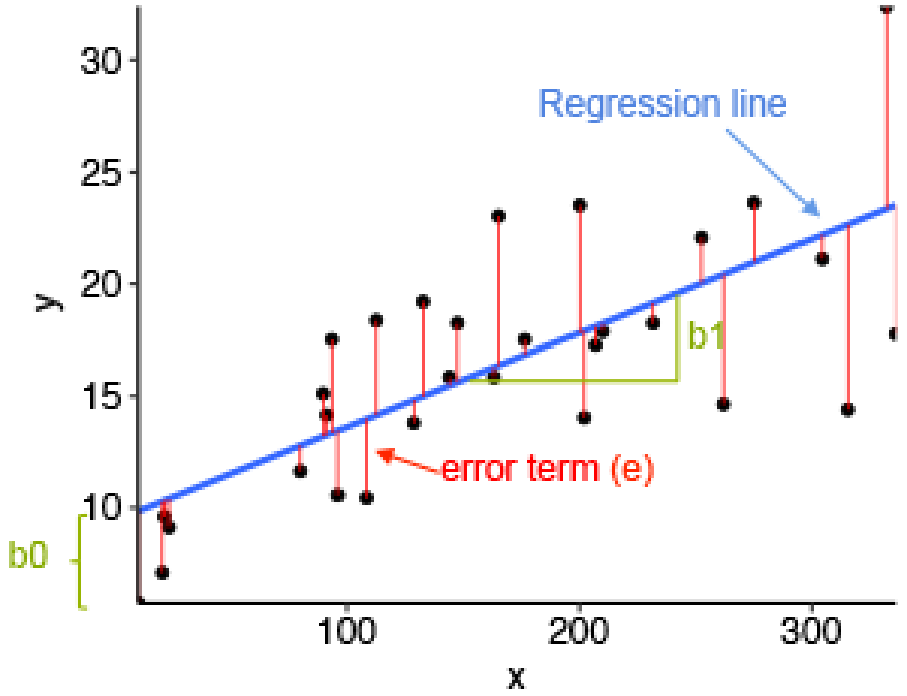
*It uses Ordinary least squares Linear Regression.

*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)
```

```
0.5358758809585568
```



Model 2 : KNeighbors Regression

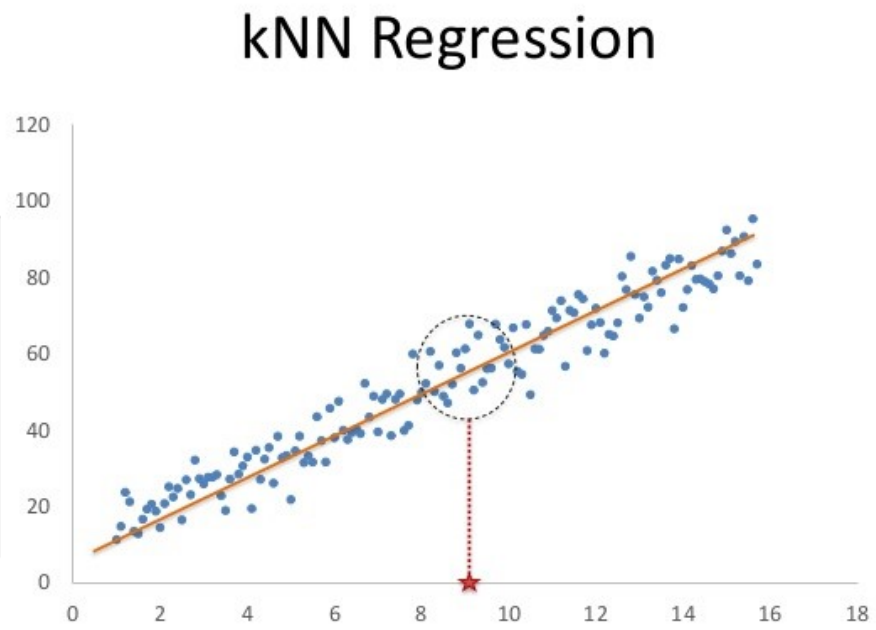
*Regression based on k-nearest neighbors.

*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)
```

```
0.5226004963298835
```



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.

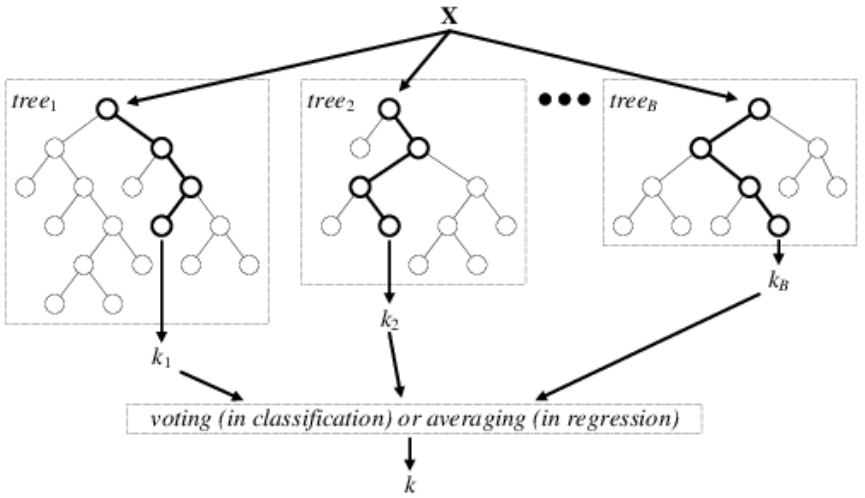
```
#Trying Random Forest Regressor Regression model

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)
```

```
0.5106684368905897
```



GridSearchCV : Hyperparameter Tuning

* We tried to tune the hyperparameters for random forest search and the result improved with the score as shown in previous slide.

```
from sklearn.model_selection import GridSearchCV

params_grid = { "n_estimators": [20],
                "n_jobs": [-1],
               "max_samples": [None],
              "min_samples_split": [1,5,10,15],
              "min_samples_leaf": [1,2,3,4,5],
              }
grid_search = GridSearchCV(rfrmodel, params_grid,
                           n_jobs=-1, cv=5,
                           verbose=-1, scoring='neg_mean_squared_log_error')
grid_search.fit(X_train, y_train)
```

```
grid_search.best_estimator_.get_params()
```

```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'criterion': 'mse',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 5,
 'min_samples_split': 15,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 20,
 'n_jobs': -1,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

Predictions

- For actual predictions of sample submission data, it was featured exactly how we did for training dataset. Now entire train dataset is our training data and featured Sample submission became the test dataset for Kaggle submission.

- Random Forest Regression gave the best result of RMSLE score of **0.51066** on training set and **0.53868** on Kaggle

- Average result of all models was RMSLE score 0.54234 and taking average did not improve the score.

What else we could have done?

- We could have used the time series forecasting models like ARIMA for better prediction.
- We could have used the Gradient boosting algorithms such as XGBoost and Light GBM.
- However the reason we did not pick up those, is because we wanted to learn more about how regression techniques work and not focus on the Kaggle score.

VIII. References

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

Books and Lectures:

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

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

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

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