IST 597

Project Report

-Ayush Tiwari

Abstract

The aim for my project will be identifying the factors that affect customers to give a better satisfaction ratings to hotels using data from hotels at locations across the globe. To follow along with steps performed, the python notebook is attached in the same file. For our task, we picked the features of ‘location_grade’, ‘discount_perc’, and ‘price_min’. These namely are chosen as our desirable traits that we want to look for before picking a hotel- how good the location is, how much discount Is possible, where do the prices start from.

Motivation

The motivation for it was seeing how travel requires so much research before actually making huge transactions for hotel and acoomadation, there should be a logical way to quantify the services we can get as customers. So, using random forest regression and the dataset given about hotel ratings, we can pick our desired features we want highly rated, and then book our accomodations accordingly.

Research Questions

- Will the location affect how many customers leave good ratings for the hotels they stayed at?
- Do prices/discounts create a sub-conscious effect on customers and their ratings?
- Will availability of good amenities help customers pick the hotel?

Data

For this endeavor, we utilized a dataset of 4599 reviews of the top hotels in Rome from TripAdvisor. This dataset contained the list of amenities, the price ranges, location ratings, awards given to the hotels, etc. along with the final ratings given to them by previous customers.

Analysis

So our analysis began by first cleaning the data as it contained some unrequired columns and also several rows with no reviews or even entries of NA in desired columns.

```
# Drop the 'Unnamed: 0' column
df = df.drop(columns=['Unnamed: 0'])
```

```
# Now we clean the data by removing missing values

data = df[df['views']!=0]


# Here, we have the same dataset as before
# except for any entries whose views had a value of zero views entered. As for these entries no analysis can be done,
#so they have gotten reomoved.
```

```
# Remove columns we will not be needing for our analysis
data = data.drop(columns=[ 'views_binary', 'hotel_url'])
```

```
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler

# Handle infinite values
data.replace([np.inf, -np.inf], np.nan, inplace=True)

# Drop rows with NaN (if any)
data = data.dropna()
```

This was decided when I was trying to find results with this dataset but encountered errors along the way. So, this made our dataset clean and suitable for analysis.

Now we moved onto separating our data into testing and training data. For this we split 20% of the remaining data (from the original 4599 rows) as test data, and rest as our training data. We performed the random forest regression on this training data.

```
# Define predictor variables (features) and target variable
X = data_final[['views', 'location_grade',  'discount_perc', 'price_min']]
y = data_final['score_adjusted']


# split data into train and test sets
seed = 42
val_size = 0.1
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=seed)
```

First, to fine tune our hyperparameters, we used grid search to find the best random fortest parameters to work with for our dataset.

```
# Define the parameter grid
param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Create the base model
rf_base = RandomForestRegressor(random_state=42)

# Create GridSearchCV
grid_search = GridSearchCV(estimator=rf_base, param_grid=param_grid, cv=5, scoring='neg_mean_squared_error', n_jobs=-1)

# Fit the model to the data
grid_search.fit(X_train, y_train)

# Print the best parameters and corresponding score
print("Best Parameters: ", grid_search.best_params_)
print("Best Negative Mean Squared Error: ", grid_search.best_score_)
```

```
Best Parameters:  {'max_depth': None, 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 150}
Best Negative Mean Squared Error:  -0.16302071213880007
```

Then we finally applied these sets of parameters.

```
# Now we go ahead and train our random forest model using the best parameter foudn by grid search

rf_model = RandomForestRegressor(**grid_search.best_params_, random_state=42)
rf_model.fit(X_train, y_train)
```

```
RandomForestRegressor(min_samples_leaf=4, min_samples_split=10,
                      n_estimators=150, random_state=42)
```

Results

And finally we found how far we are from our predictor data.

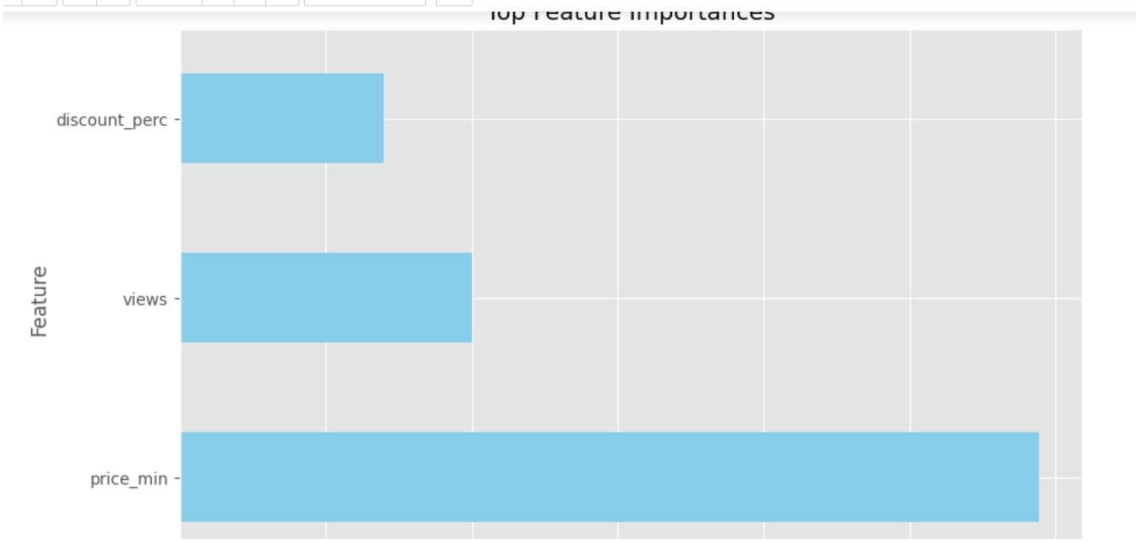
In [123]:
```
# Now we make predictions on our testing dataset
y_pred = rf_model.predict(X_test)
```

In [124]:
```
# Evaluating our model by findin gthe mean squared error (MSE)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 0.08292815462333239
```

Since the mean squared error turns out very close to 0, hence, we have found the best possible result.

We also ende up doing feature analysis just to be sure./


Top Feature Importances
Feature
discount_perc
views
price_min