# Data 144 Final Project Presentation
# Hotel Daily Rates

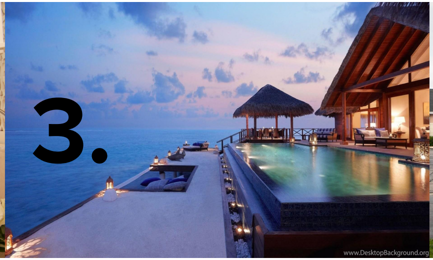
Alvin Lee, Jacqueline Yu, Jay Feng, Justin Cheung

# Presentation Content

***Hotel Average Daily Rate Prediction***


Project Introduction


Dataset & Cleaning



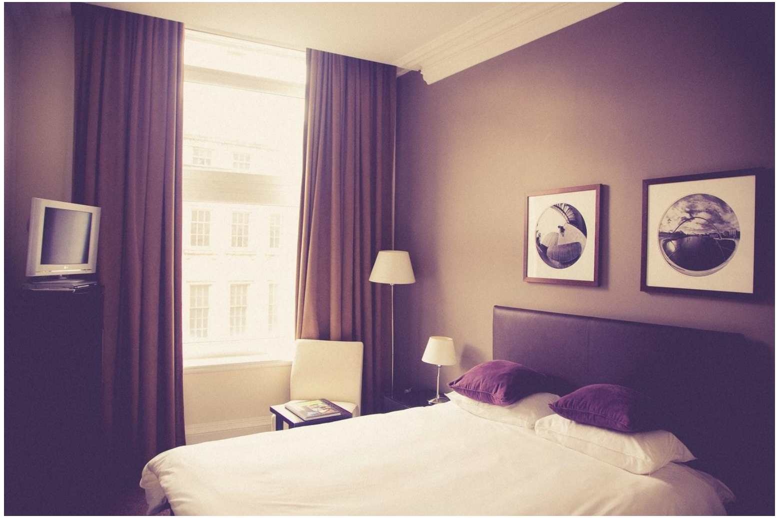

Methods


Conclusions

# Project Introduction

- Looking to book your next trip over winter break?
- How much would it cost to stay a day at a hotel?
- Would that change where you want to travel?
- Can we predict the average daily rate based on information of the booking?

# The Dataset



- Open hotel booking demand dataset from Antonio, Almeida and Nunes, 2019.
- 119,390 observations, 31 features
- **Target:** ADR (Average Daily Rate)

| | hotel | is_canceled | lead_time | arrival_date_year | arrival_date_month | arrival_date_week_number | arrival_date_day_of_month | stays_in_weekend_nights | stays_in_week_nights | adults | children | babies | meal | country | market_segment | distribution_channel |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Resort Hotel | 0 | 342 | 2015 | July | 27 | 1 | 0 | 0 | 2 | 0.0 | 0 | BB | PRT | Direct | Direct |
| 1 | Resort Hotel | 0 | 737 | 2015 | July | 27 | 1 | 0 | 0 | 2 | 0.0 | 0 | BB | PRT | Direct | Direct |
| 2 | Resort Hotel | 0 | 7 | 2015 | July | 27 | 1 | 0 | 1 | 1 | 0.0 | 0 | BB | GBR | Direct | Direct |
| 3 | Resort Hotel | 0 | 13 | 2015 | July | 27 | 1 | 0 | 1 | 1 | 0.0 | 0 | BB | GBR | Corporate | Corporate |
| 4 | Resort Hotel | 0 | 14 | 2015 | July | 27 | 1 | 0 | 2 | 2 | 0.0 | 0 | BB | GBR | Online TA | TA/TO |

# Data Cleaning

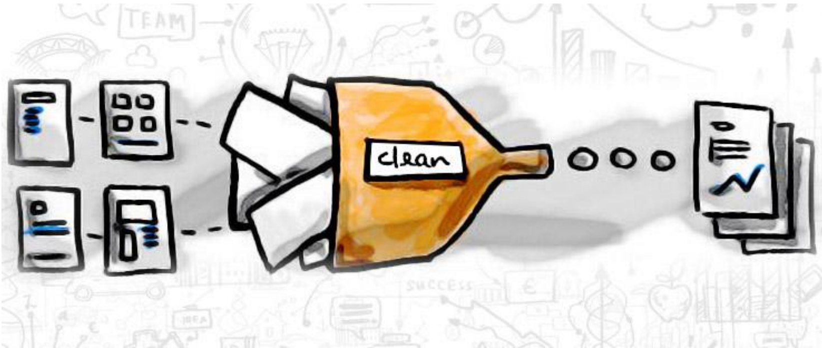
-Removed cancelled bookings

-Converted date data types

-OHE (year, month, week, country, meal, etc...)

-Separated into two datasets based on type of hotel (City & Resort)

-Normalized lead_time



# Feature Engineering

| percentage_of_booking_canceled |
|---|
| 0.500000 |
| 0.333333 |
| 0.250000 |
| 0.500000 |
| 0.333333 |
| ... |
| 0.500000 |
| 0.200000 |
| 0.166667 |
| 0.142857 |
| 0.125000 |

- Percentage of previous bookings cancelled per observation

# Cleaned Dataset

```
city_hotels.head()
```

| | lead_time | stays_in_weekend_nights | stays_in_week_nights | adults | children | babies | is_repeated_guest | previous_cancellations | previous_bookings_not_canc |
|---|---|---|---|---|---|---|---|---|---|
| **40060** | -0.831305 | 0 | 2 | 1 | 0.0 | 0 | 0 | 0 | |
| **40066** | -0.864689 | 0 | 3 | 1 | 0.0 | 0 | 0 | 0 | |
| **40070** | -0.419562 | 0 | 2 | 2 | 0.0 | 0 | 0 | 0 | |
| **40071** | -0.419562 | 0 | 2 | 2 | 0.0 | 0 | 0 | 0 | |
| **40072** | -0.419562 | 0 | 2 | 2 | 0.0 | 0 | 0 | 0 | |

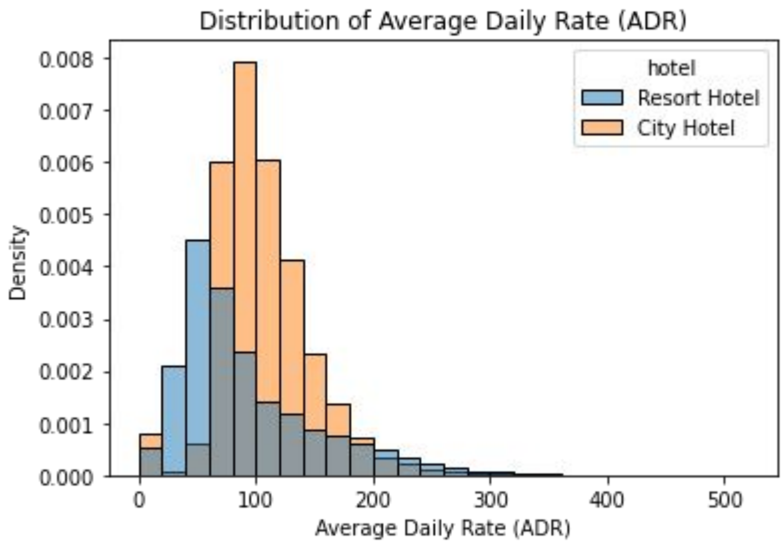
5 rows × 217 columns

```
resort_hotels.head()
```

| | lead_time | stays_in_weekend_nights | stays_in_week_nights | adults | children | babies | is_repeated_guest | previous_cancellations | previous_bookings_not_cancele |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2.828014 | 0 | 0 | 2 | 0.0 | 0 | 0 | 0 | |
| **1** | 7.072791 | 0 | 0 | 2 | 0.0 | 0 | 0 | 0 | |
| **2** | -0.771986 | 0 | 1 | 1 | 0.0 | 0 | 0 | 0 | |
| **3** | -0.707509 | 0 | 1 | 1 | 0.0 | 0 | 0 | 0 | |
| **4** | -0.696763 | 0 | 2 | 2 | 0.0 | 0 | 0 | 0 | |

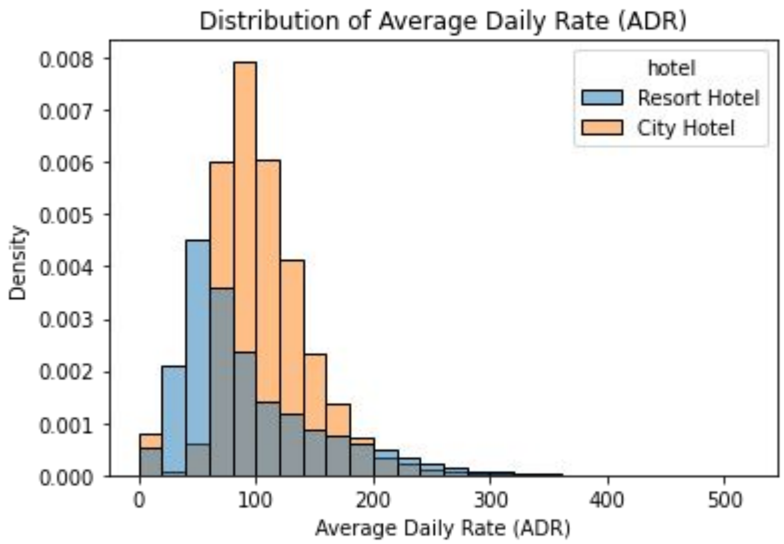
5 rows × 188 columns

# EDA




- Distribution of ADR for Resort Hotels and City Hotels

# EDA




- We can see from looking at our visualization that the Resort Hotel ADR values are centered at a lower value than City Hotel ADR values
- Both are right skewed

# EDA

```
[24] city_hotels["adr"].describe()
```

```
count    46228.000000
mean       105.745948
std         40.596109
min          0.000000
25%         80.000000
50%         99.900000
75%        126.000000
max        510.000000
Name: adr, dtype: float64
```

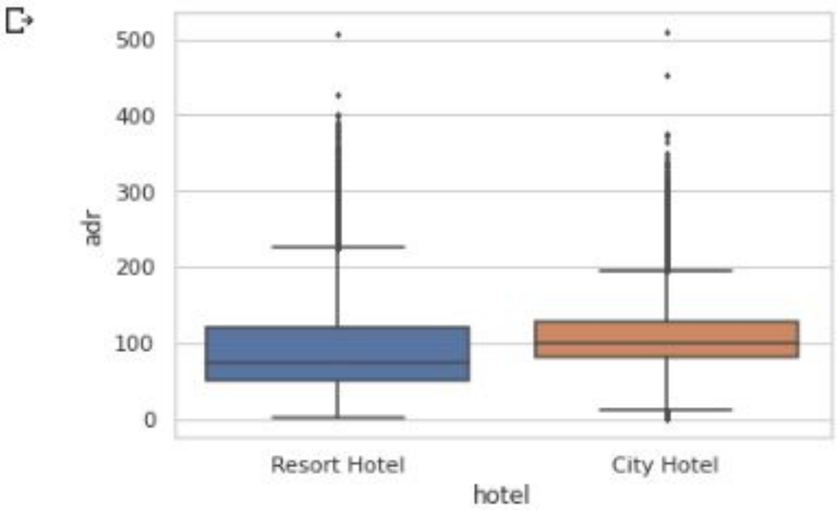
```
resort_hotels["adr"].describe()
```

```
count    28938.000000
mean        90.788971
std         59.320827
min         -6.380000
25%         48.000000
50%         72.000000
75%        118.185000
max        508.000000
Name: adr, dtype: float64
```

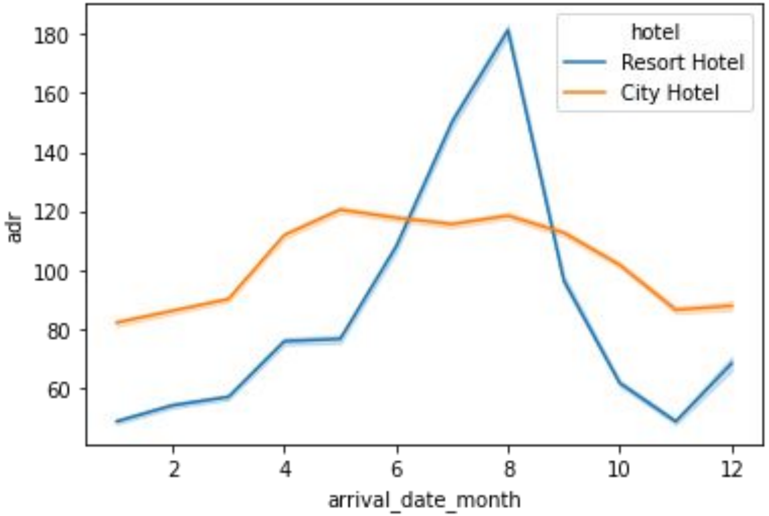
- We looked further into the distribution of ADR values for both groups
- City Hotel ADR has a mean of ~105.7 and a median of 99.9
- Resort Hotel ADR has a mean of ~90.7 and a median of 72.0
- This helped inform our decision to separate our data set based on type of hotel

# EDA

- The skewness observed on the histograms seemed to indicate that there might be outliers
- We looked further into this by creating boxplots
- Appears to be a lot of outliers in the higher ADR’s
- Will use MAE as our metric (when possible/no need for differentiability) since it is more robust to outliers

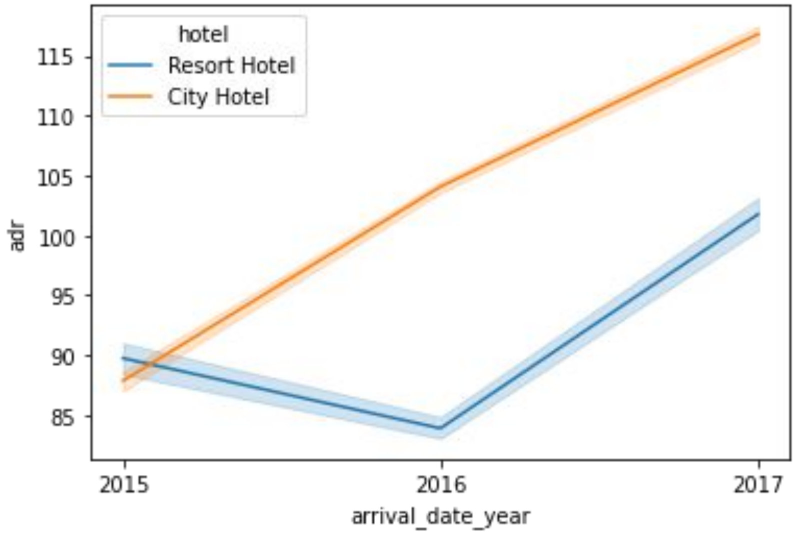


# EDA



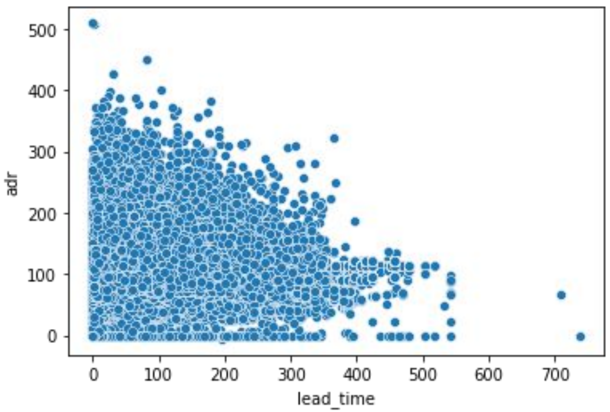

- Next, we looked at ADR by month
- We can see that for both city and resort hotels, ADR tends to be higher during the summer months
- The months of June to August are when Resort Hotel ADR surpasses City Hotel ADR

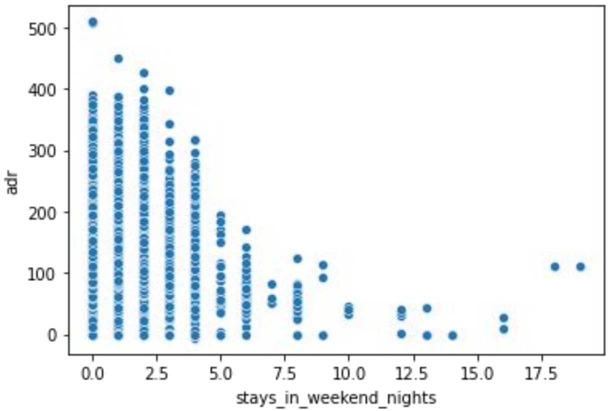
# EDA




- Then, we looked at how ADR tended to change across the years
- Our plot shows an increase in City Hotel ADR from 2015 to 2017
- Resort Hotel ADR decreased from 2015 to 2016 then increased from 2016 to 2017

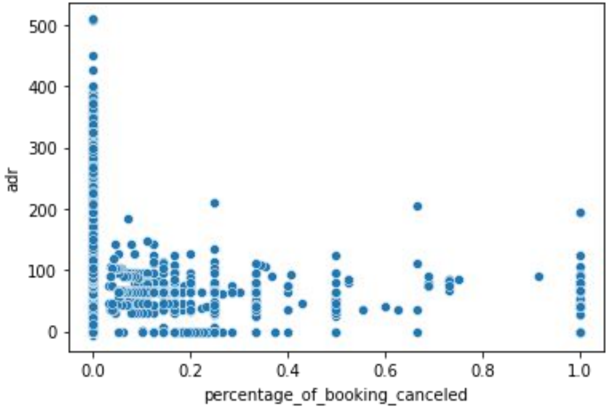
# EDA

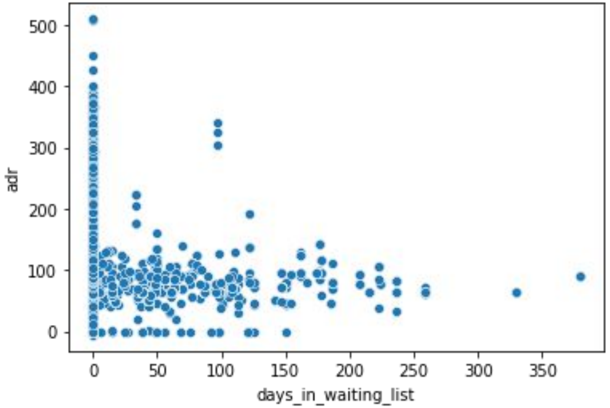



- Scatter plots to visualize the relationship between ADR and other variables
- **Lead_time:** days elapsed between entering date of booking and arrival date
- **Stays_in_weekend_nights:** weekend nights (Sat/Sun) the guest stayed/booked

# EDA




- Scatter plots to visualize the relationship between ADR and other variables
- **Percentage_of_booking_canceled**
- **Days_in_waiting_list:** days the booking was in the waiting list before it was confirmed to the customer
- Scatter plots did not seem to show any strong linear associations

# KMeans Exploration (City Hotels)

- We also decided to explore our data using clustering with KMeans
  - Defined a function to find best k
  - N_cluster = 5 has the best silhouette score
- Obtained the labels from the KMeans trained with the 5 clusters

```
[33] def best_k(df):

       # YOUR CODE HERE
       maximum = -1
       n = 2
       for i in np.arange(2, 9):
         kmeans = KMeans(n_clusters=i, random_state=42)
         kmeans.fit(df)
         score = silhouette_score(df, kmeans.labels_)
         if score > maximum:
           maximum = score
           n = i
       return n

     print(best_k(city_hotels))

     5
```

```
[38] best_KMeans = KMeans(n_clusters=5, random_state=42)
     best_KMeans.fit(city_hotels)

     best_KMeans.labels_

     array([3, 0, 0, ..., 2, 1, 1], dtype=int32)
```

# KMeans Exploration (City Hotels)

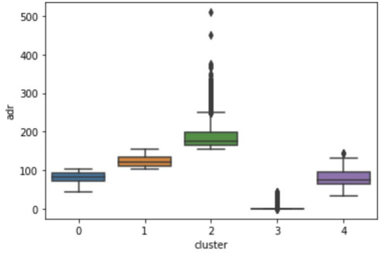
```
[39] city_kmeans = pd.DataFrame({'adr' : city_hotels.adr, 'cluster' : best_KMeans.labels_})
     city_kmeans
```

| | adr | cluster |
|---|---|---|
| 40060 | 0.00 | 3 |
| 40066 | 58.67 | 0 |
| 40070 | 86.00 | 0 |
| 40071 | 43.00 | 3 |
| 40072 | 86.00 | 0 |
| ... | ... | ... |
| 119385 | 96.14 | 0 |
| 119386 | 225.43 | 2 |
| 119387 | 157.71 | 2 |
| 119388 | 104.40 | 1 |
| 119389 | 151.20 | 1 |

46228 rows × 2 columns

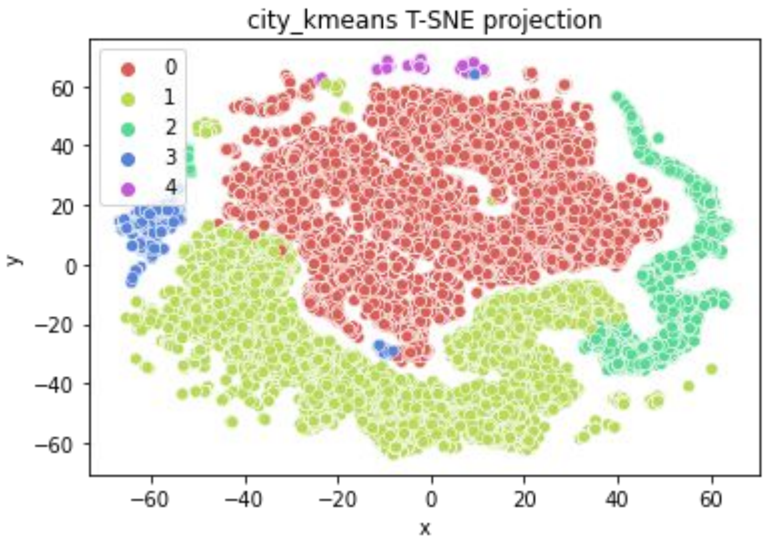
```
[42] sns.boxplot(x='cluster', y='adr', data=city_kmeans)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fc52050f7d0>




- We then isolated the adr and the cluster labels
- Cluster 2 seemed to have the highest average adr
    - Contains a lot of outliers with high adr
- Significant overlapping for clusters 0, 1, and 4
- Cluster 3 is also very close to 0 and is highly right skewed

# KMeans Exploration (City Hotels)




- Dimensionality reduction:
  - We then used T-SNE to visualize our five clusters in 2-D

# KMeans Exploration (Resort Hotels)

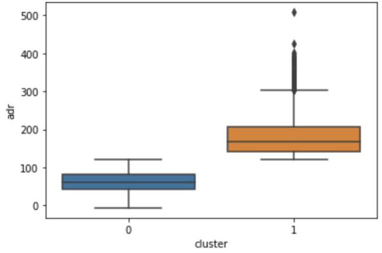
```
[45] resort_kmeans = pd.DataFrame({'adr' : resort_hotels.adr, 'cluster' : best_KMeans_resort.labels_})
     resort_kmeans
```

| | adr | cluster |
|---|---|---|
| 0 | 0.00 | 0 |
| 1 | 0.00 | 0 |
| 2 | 75.00 | 0 |
| 3 | 75.00 | 0 |
| 4 | 98.00 | 0 |
| ... | ... | ... |
| 40055 | 89.75 | 0 |
| 40056 | 202.27 | 1 |
| 40057 | 153.57 | 1 |
| 40058 | 112.80 | 0 |
| 40059 | 99.06 | 0 |

28938 rows × 2 columns

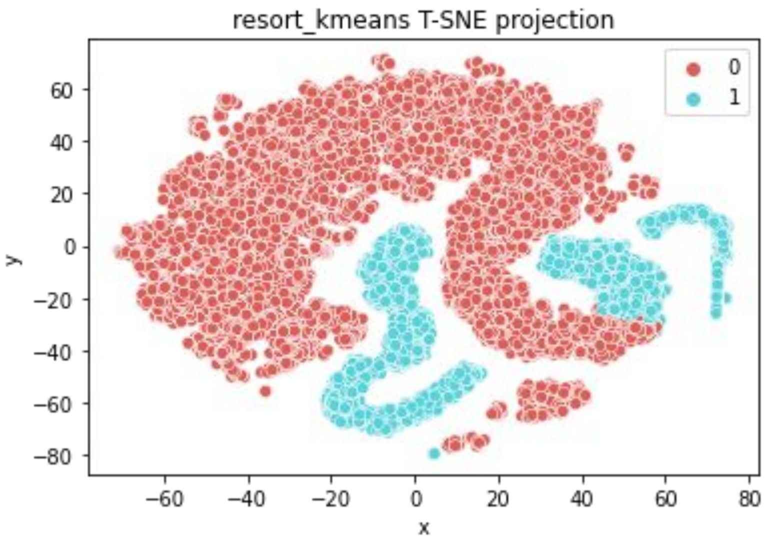
```
[46] sns.boxplot(x='cluster', y='adr', data=resort_kmeans)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fc51ffae910>




- 2 clusters seem to perform the best based on the best_k function that we defined
- There seems to be a significant difference between the distribution of adr of the two clusters.
- Adr of cluster 0 seems to be approximately normally distributed with a median of around 80, while cluster 1 is right skewed and contains outliers with high adr.

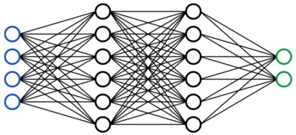
# KMeans Exploration (Resort Hotels)




- Dimensionality reduction:
    - We then used T-SNE to visualize our two clusters in 2-D

# Keras: Neural Net

- To prepare our data for training neural nets, we one hot encoded categorical variables and standardized them.
- Keras consists of calling the Sequential() model, then adding layers to the neural network, alongside with kernel_initializer and activation as hyperparameters
    - The hidden layers have 256 nodes, and the batch size is tuned to 64 after trial and error



# Neural Net (City Hotels Model)

- We have decided to add 3 hidden layers in the neural network
- Kernel initializer: normal
    - Initial weights are generated from a normal distribution
- Activation: “relu”
    - Output 0 if negative, keep if positive

```
[22] # Using Keras to build a Neural Net Model
     model = Sequential()

     # The Input Layer :
     model.add(Dense(128, kernel_initializer='normal',input_dim = X_train.shape[1], activation='relu'))

     # The Hidden Layers :
     model.add(Dense(256, kernel_initializer='normal',activation='relu'))
     model.add(Dense(256, kernel_initializer='normal',activation='relu'))
     model.add(Dense(256, kernel_initializer='normal',activation='relu'))

     # The Output Layer :
     model.add(Dense(1, kernel_initializer='normal',activation='linear'))

     # Compile the network :
     model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mean_absolute_error'])
     model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 dense (Dense)                (None, 128)               38528

 dense_1 (Dense)              (None, 256)               33024

 dense_2 (Dense)              (None, 256)               65792

 dense_3 (Dense)              (None, 256)               65792

 dense_4 (Dense)              (None, 1)                 257

=================================================================
Total params: 203,393
Trainable params: 203,393
Non-trainable params: 0
_________________________________________________________________
```

# Neural Net (City Hotels Model)

- ModelCheckpoint
  - Used to save weights of a epoch,
  - loaded later to recreate optimized model
- The val_loss metric would be mean absolute error as mentioned

```
[23] # Creates the checkpoint that has the best metric, we can then use that model to train it on our test set
     checkpoint_name = 'Weights-{epoch:03d}--{val_loss:.5f}.hdf5'
     checkpoint = ModelCheckpoint(checkpoint_name, monitor='val_loss', verbose = 1, save_best_only = True, mode ='auto')
     callbacks_list = [checkpoint]
```

# Neural Net (City Hotels Model)

- The trained model with the best weights are loaded
- MAE on the training data is 10.99962
- MAE on test data is 11.20755

```
wights_file = 'Weights-049--10.99962.hdf5' # choose the best checkpoint
model.load_weights(wights_file) # load it
model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mean_absolute_error'])

results = model.predict(X_test).flatten()
results
```

```
array([138.04451, 118.54774, 111.92947, ...,  91.57425, 117.2184 ,
        62.59509], dtype=float32)
```

```
len(results)
```

```
11557
```

```
city_df = pd.DataFrame({'actual' : Y_test, 'predicted_NN' : results})
city_df
```

| | actual | predicted_NN |
|---|---|---|
| 88671 | 121.50 | 138.044510 |
| 77174 | 147.00 | 118.547737 |
| 116323 | 112.67 | 111.929466 |
| 78320 | 75.00 | 73.092651 |
| 105514 | 96.99 | 129.117645 |
| ... | ... | ... |
| 94048 | 125.00 | 116.550201 |
| 103833 | 70.00 | 71.792160 |
| 87183 | 90.95 | 91.574249 |
| 106466 | 121.33 | 117.218399 |
| 76294 | 62.00 | 62.595089 |

11557 rows × 2 columns

```
city_mae = sum(abs(city_df.predicted_NN - city_df.actual))/11557
city_mae
```

```
11.207552652093439
```

# Neural Net (Resort Hotel Model)

```
# Using Keras to build a Neural Net Model
model2 = Sequential()

# The Input Layer :
model2.add(Dense(128, kernel_initializer='normal',input_dim = X_train.shape[1], activation='relu'))

# The Hidden Layers :
model2.add(Dense(256, kernel_initializer='normal',activation='relu'))
model2.add(Dense(256, kernel_initializer='normal',activation='relu'))
model2.add(Dense(256, kernel_initializer='normal',activation='relu'))

# The Output Layer :
model2.add(Dense(1, kernel_initializer='normal',activation='linear'))

# Compile the network :
model2.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mean_absolute_error'])
model2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_5 (Dense)              (None, 128)               34816

dense_6 (Dense)              (None, 256)               33024

dense_7 (Dense)              (None, 256)               65792

dense_8 (Dense)              (None, 256)               65792

dense_9 (Dense)              (None, 1)                 257

=================================================================
Total params: 199,681
Trainable params: 199,681
Non-trainable params: 0
_________________________________________________________________
```

```
[26] # Creates the checkpoint that has the best metric, we can then use that model to train it on our test set
     checkpoint_name = 'Weights-{epoch:03d}--{val_loss:.5f}.hdf5'
     checkpoint = ModelCheckpoint(checkpoint_name, monitor='val_loss', verbose = 1, save_best_only = True, mode ='auto')
     callbacks_list = [checkpoint]
```

```
[27] model2.fit(X_train, Y_train, epochs=100, batch_size=32, validation_split = 0.2, callbacks=callbacks_list)
```

```
[29] wights_file = 'Weights-018--10.74073.hdf5' # choose the best checkpoint
     model2.load_weights(wights_file) # load it
     model2.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mean_absolute_error'])

     resort_df = pd.DataFrame({'actual' : Y_test, 'predicted' : model2.predict(X_test).flatten()})
     resort_df
```

| | actual | predicted |
|---|---|---|
| **29577** | 71.20 | 53.307625 |
| **35751** | 147.00 | 138.484741 |
| **4829** | 54.50 | 55.013214 |
| **28963** | 0.00 | -0.114785 |
| **33494** | 60.00 | 68.091583 |
| **...** | ... | ... |
| **21046** | 39.00 | 37.368851 |
| **23799** | 39.00 | 38.278496 |
| **430** | 107.00 | 105.235878 |
| **583** | 121.01 | 116.315926 |
| **33785** | 53.14 | 45.391735 |

7235 rows × 2 columns

```
resort_mae = sum(abs(resort_df.predicted - resort_df.actual))/11557
resort_mae
```

```
6.448530729289678
```

Train MAE: 10.74073
Test MAE: 6.44853

# CatBoost

- 12/31 of our features are categorical -- CatBoost (Categorical Boosting) can take in categorical features as well as numerical features
    - helps prevent loss of information through other simple encoding methods
- CatBoost, like XGBoost, can take in null/missing values without additional preprocessing by treating them as minimum or maximum values for node splits
    - this helps avoid losing data or using imputation methods which may reduce the effectiveness of the feature

# CatBoost: City Hotels Model

- CatBoostRegressor model on the City Hotels data
- Used grid search and 3-fold cross validation to identify the best hyperparameters
- We were limited to tuning a few important hyperparameters due to model training time, even with GPU training and only training on a fraction of the data

```
city_hotels_sample = city_hotels.sample(frac=0.5)

Y = city_hotels_sample.adr
X = city_hotels_sample.drop(['adr'], axis=1)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=42)
```

```
cbr_cv = CatBoostRegressor(cat_features=cat_features,
                           task_type="GPU",
                           devices='0:1',
                           loss_function='MAE',
                           iterations=1000,
                           verbose=False)

grid = {'learning_rate': [0.05, 0.1],
        'depth': [6, 8, 10],
        'l2_leaf_reg': [1, 4, 7, 10]}

randomized_search_result = cbr_cv.grid_search(grid,
                                              X=X_train,
                                              y=Y_train,
                                              plot=True,
                                              verbose=False,
                                              cv=3)
```

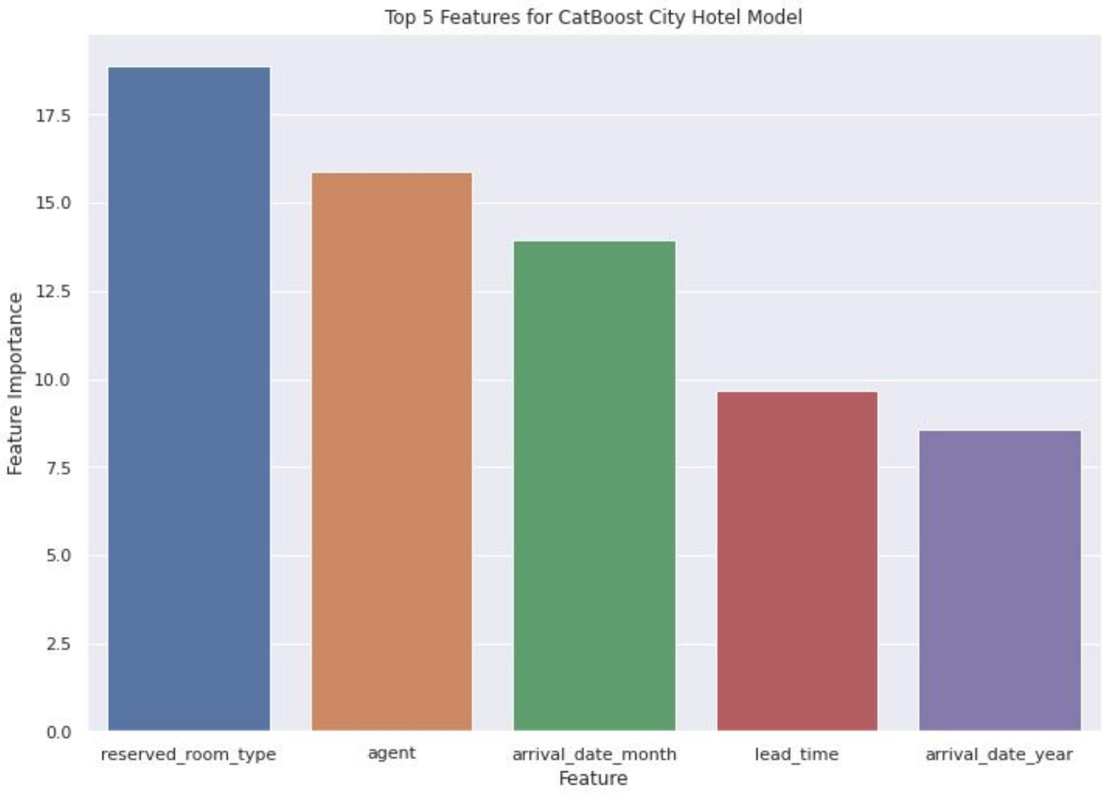
# CatBoost: City Hotels Model

- Achieved a test MAE of 8.435 using the best model from grid search + cross validation
- Much better than Neural Net models with minimal tuning

```
Y_train_pred = cbr_cv.predict(X_train)
print("Train MAE:", mean_absolute_error(Y_train, Y_train_pred))

print("\n")

Y_test_pred = cbr_cv.predict(X_test)
print("Test MAE:", mean_absolute_error(Y_test, Y_test_pred))
```

```
Train MAE:  6.353846889890939 5

Test MAE:  8.43425699116862
```

# CatBoost: City Hotels Model



# CatBoost: Resort Hotels Model

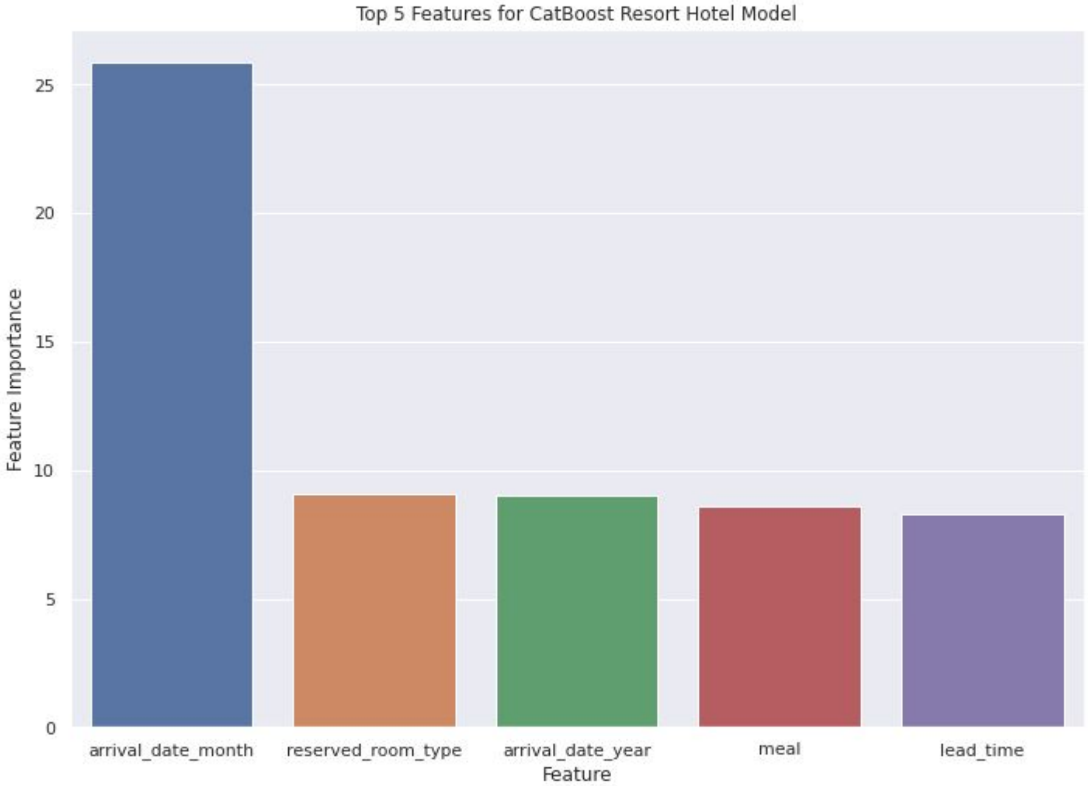
```
resort_hotels_sample = resort_hotels.sample(frac=0.5)

Y = resort_hotels_sample.adr
X = resort_hotels_sample.drop(['adr'], axis=1)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=40)
```

```
cbr_cv_2 = CatBoostRegressor(cat_features=cat_features,
                             task_type="GPU",
                             devices='0:1',
                             loss_function='MAE',
                             iterations=1000,
                             verbose=False)

grid = {'learning_rate': [0.05, 0.1],
        'depth': [6, 8, 10],
        'l2_leaf_reg': [1, 4, 7, 10]}

randomized_search_result = cbr_cv_2.grid_search(grid,
                                                X=X_train,
                                                y=Y_train,
                                                plot=True,
                                                verbose=False,
                                                cv=3,
                                                partition_random_seed=1)
```

```
Y_train_pred = cbr_cv_2.predict(X_train)
print("Train MAE:", mean_absolute_error(Y_train, Y_train_pred))

print("\n")

Y_test_pred = cbr_cv_2.predict(X_test)
print("Test MAE:", mean_absolute_error(Y_test, Y_test_pred))
```

```
Train MAE: 6.568920165896925


Test MAE: 8.947529707459086
```

# CatBoost: Resort Hotels Model



# Results & Conclusions

- Findings
  - The models created do a good job of predicting hotel costs, with CatBoost performing better than neural nets on both city and resort hotel contexts. This is likely because neural nets require substantial tuning for good prediction scores, while CatBoost is better with minimal tuning. CatBoost can also easily identify nonlinear relationships, such as specific holiday seasons.
- Conclusions
  - Given the input features, we can predict both city hotel and resort hotel prices with under 10 mean absolute error, meaning the predictions will be less than 10 dollars from the true price, on average
- Implications
  - Can use a few features to best determine what is a good daily rate is in order to meet supply & demand and to best increase revenue
  - Hotel booking demands
- What needs to be solved before it's ready for the real world?
  - More classification/segmentation problems
  - Predict which future bookings would be cancelled
  - Time series analysis

# Thank you!