HOTEL CANCELLATION

CAPSTONE PROJECT

CONTENTS


Project Background


Exploratory Data Analysis


Data Cleanse and Preprocessing


Model Implementation and Evaluation

PROJECT BACKGROUND

- Given a customer profile, is it possible to predict whether they're likely to cancel their upcoming bookings?

- If hotels know that a customer is likely to cancel, then they can better manage their bookings and waiting list. This will lead to reduced unplanned cost which is linked to higher cancellation rate. Furthermore, they can realign their strategy to the one that addresses their market trends. Strategy could include offer special deals and discounts for flexible customers.

PROJECT PROCESS

1. Exploratory Data Analysis

   To discover underlying trend that currently exists within the dataset, including information such as customer profile.

2. Data Cleanse and Preprocessing

   Transform raw data to a format suitable for the modelling process.

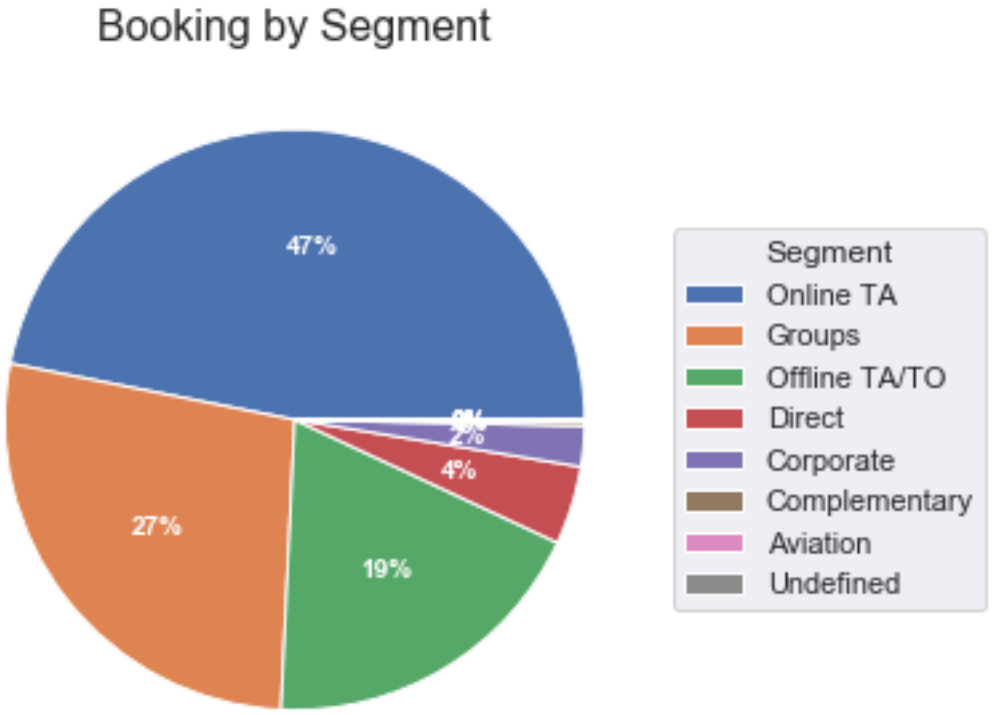
3. Develop Bi-Predict Model

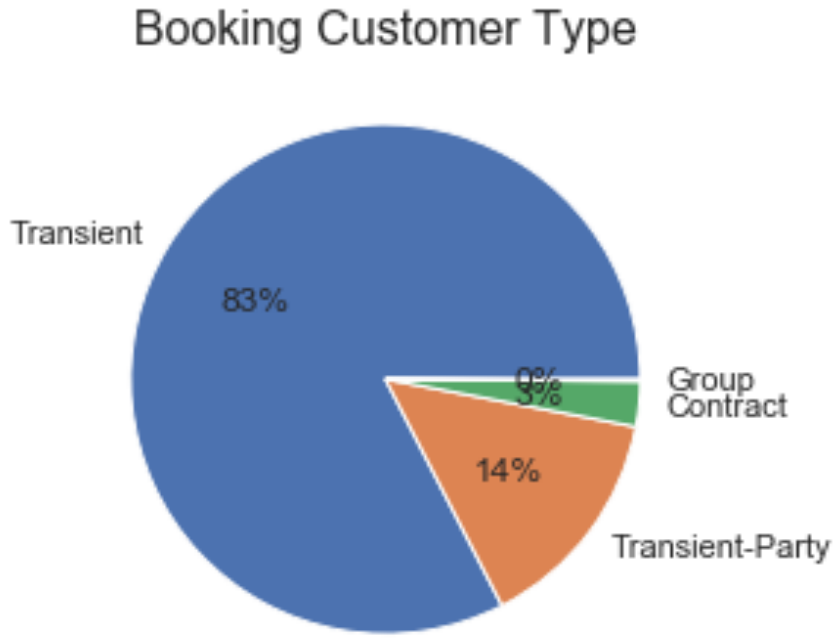
   Suitable model(s) would be selected and applied at this stage, evaluating their predictive power to detect cancellation.

EXPLORATORY DATA ANALYSIS

DISCOVER UNDERLYING TREND THAT CURRENTLY EXISTS WITHIN THE DATASET, INCLUDING INFORMATION SUCH AS CUSTOMER PROFILE.

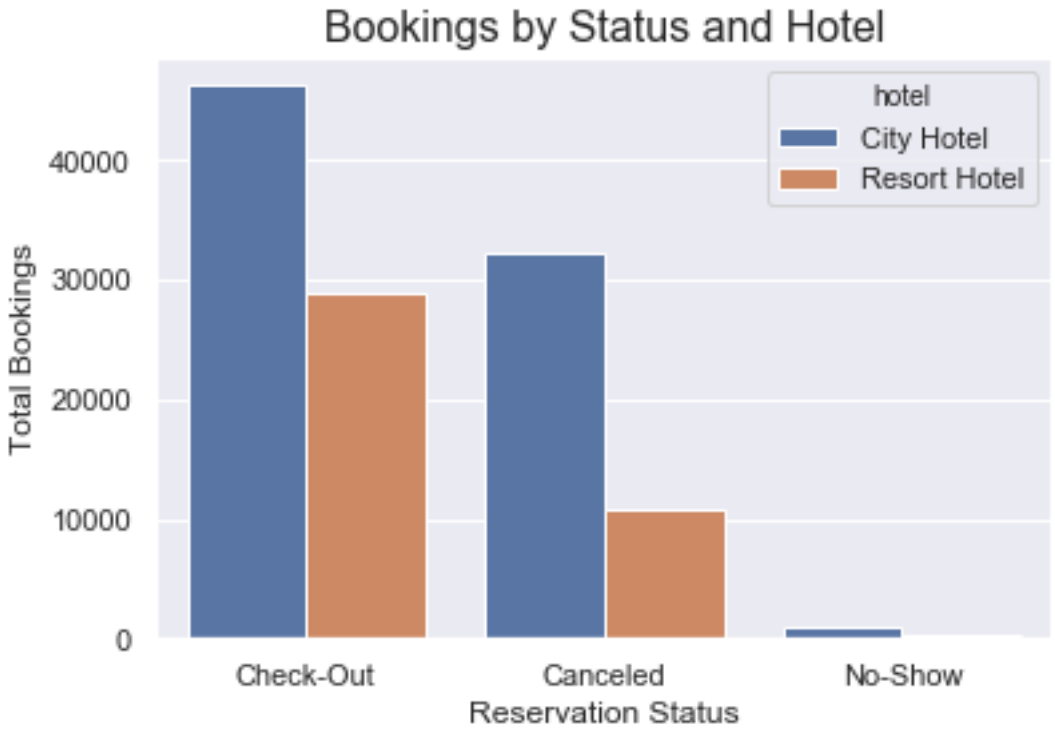
EXPLORATORY DATA ANALYSIS
BOOKINGS BY SEGMENT AND CUSTOMER TYPE

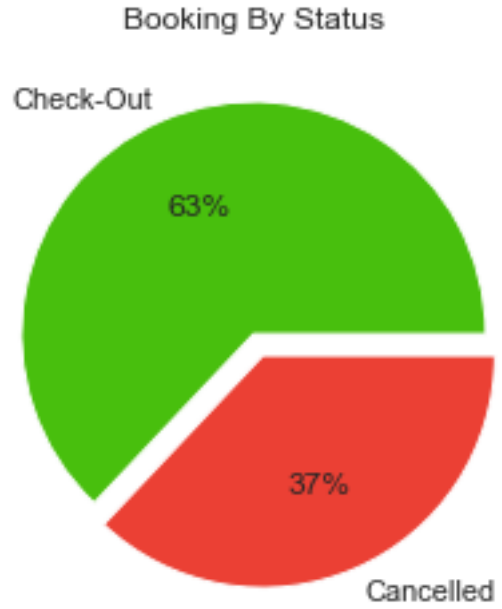




- Online travel agent accounts for the largest segment, claiming nearly half of overall bookings. Groups and offline travel agents are the other two largest segment, accounting for 27 per cent and 19 per cent of overall bookings, respectively.
- Transient customers are by far the largest customer type, with 4 out of 5 bookings being linked to this customer type.

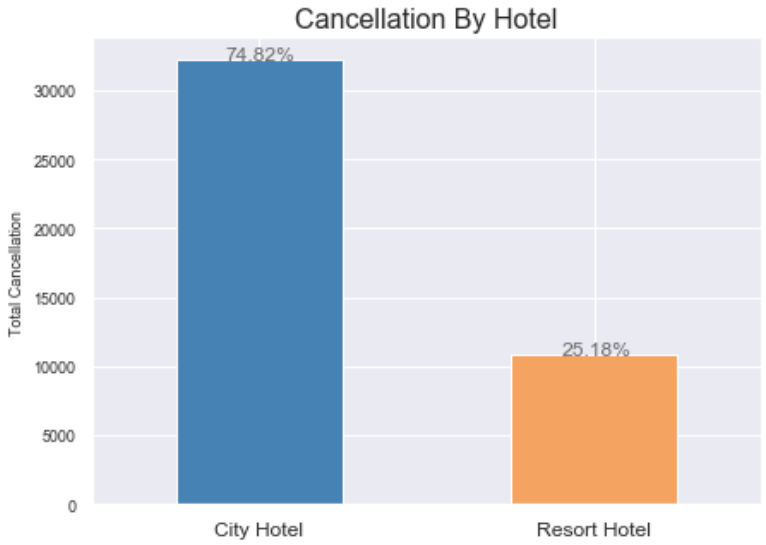
EXPLORATORY DATA ANALYSIS
BOOKINGS BY STATUS



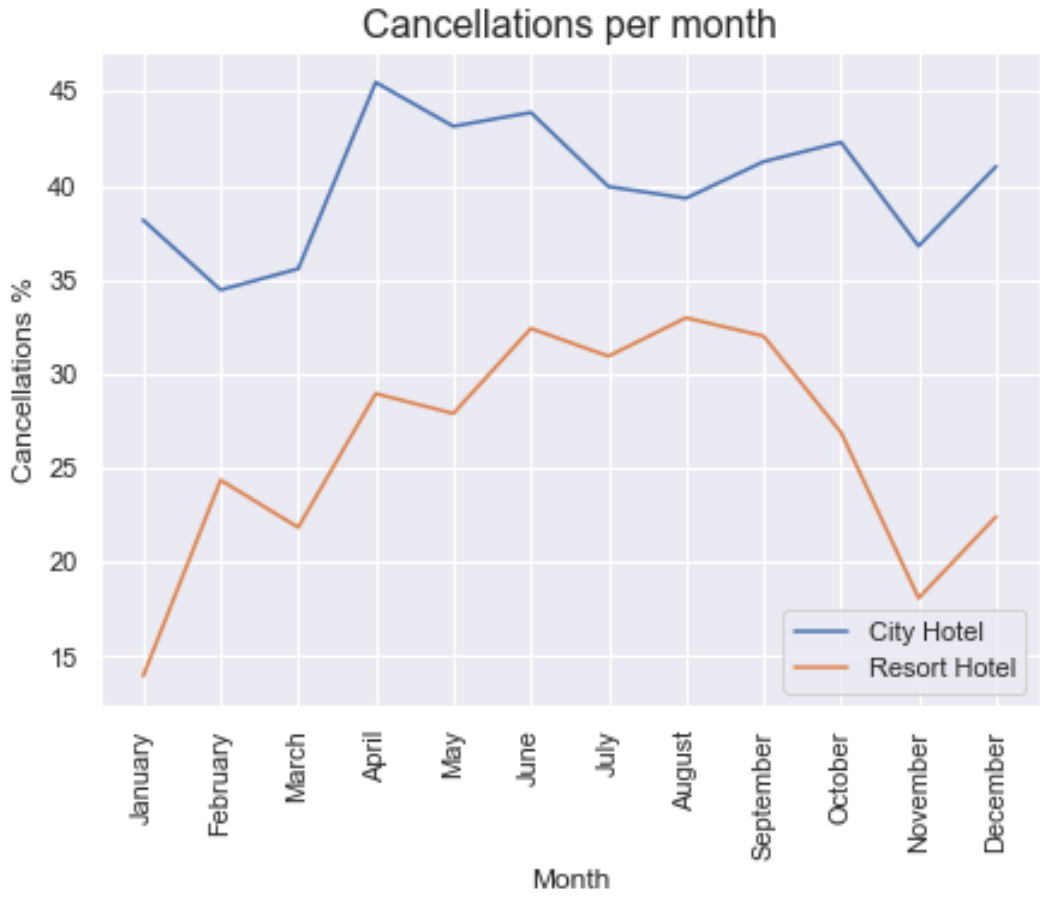


A reasonably large proportion of bookings resulted in cancellation (37%).
Majority of which were linked to City Hotel

EXPLORATORY DATA ANALYSIS
CANCELLATION BREAKDOWN

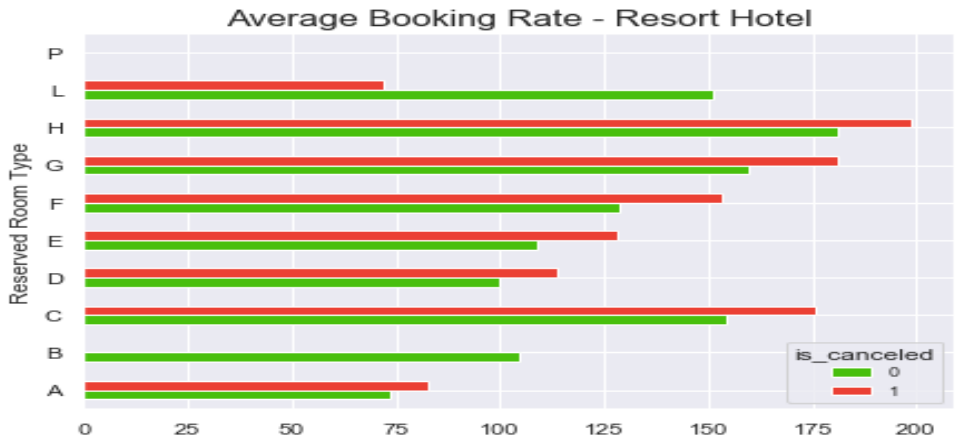


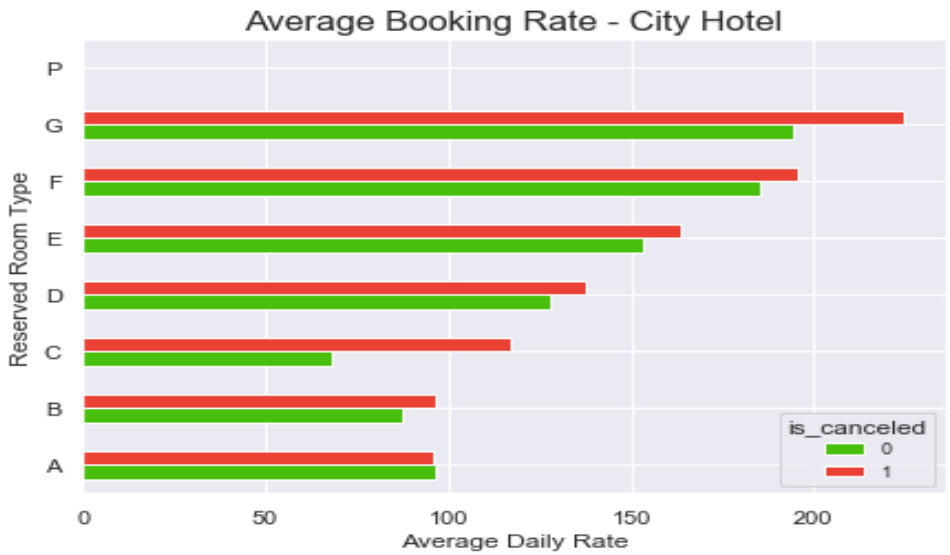
3 out of 4 cancellations are linked to City Hotel



Cancellation rate is at its peak during holiday season (between June and September) for Resort Hotel. Whereas cancellation rate for City Hotel peaks in April, then gradually drops over the summer period.

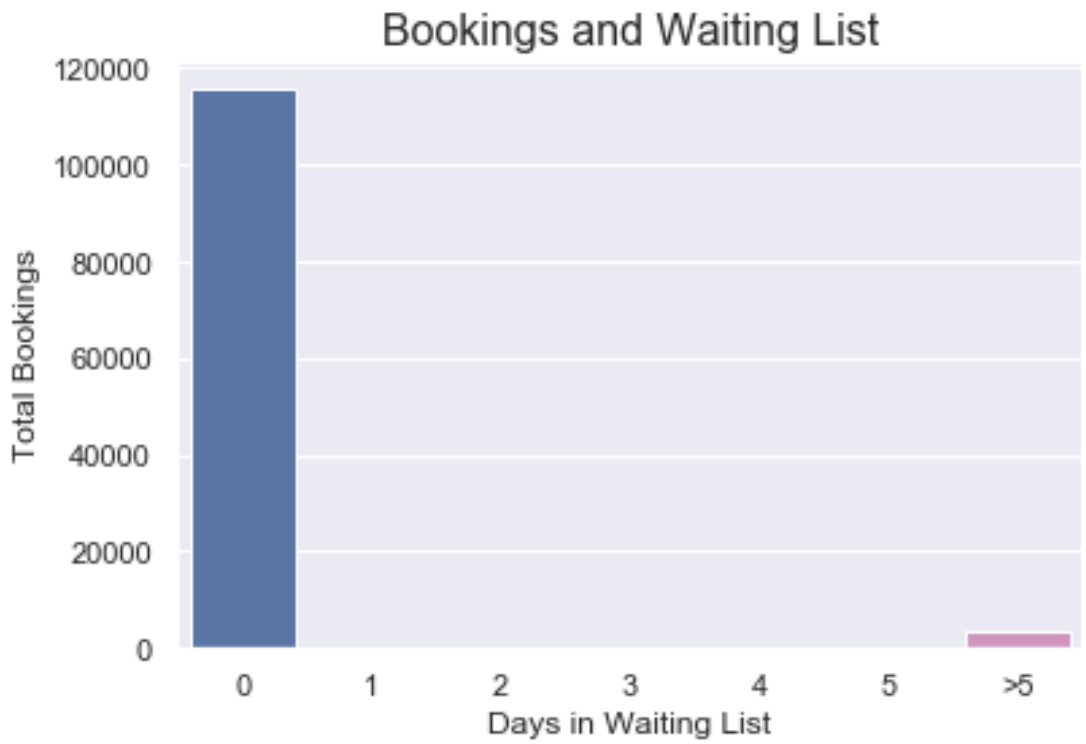
EXPLORATORY DATA ANALYSIS
BOOKING RATE & ROOM TYPE INFLUENCE





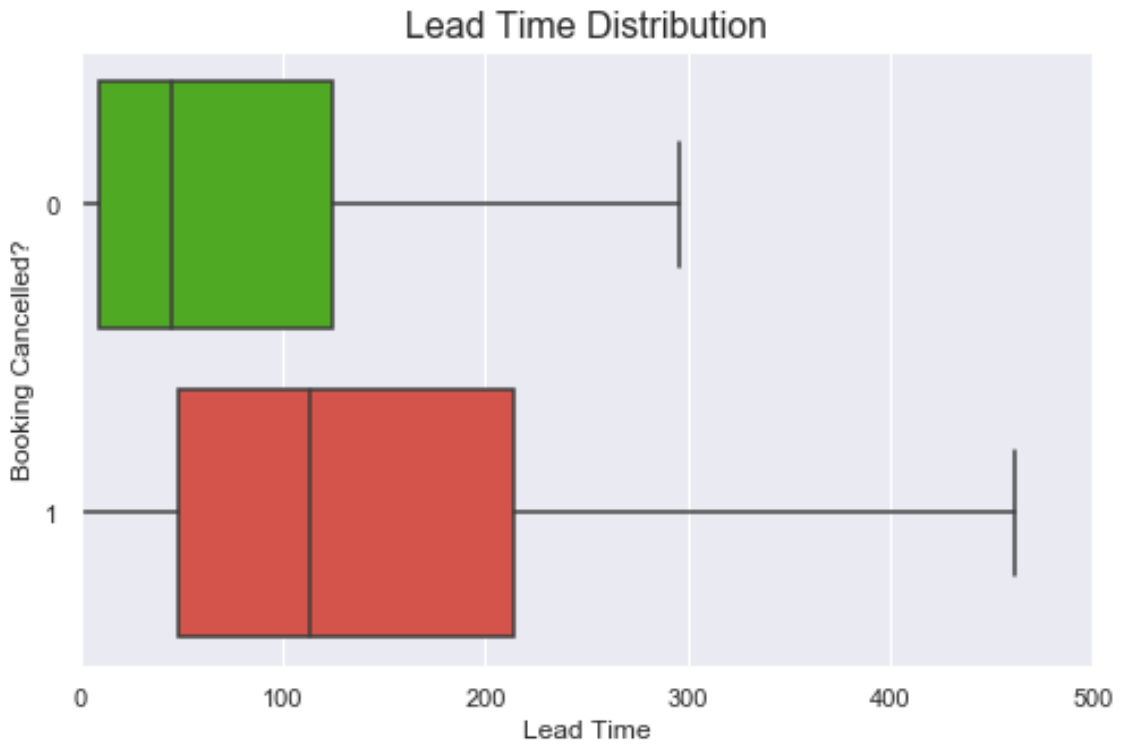
Bookings' average daily rate appears to be higher for bookings that resulted in cancellation across all but one room type (type A for City Hotel and Type L for Resort Hotel).

EXPLORATORY DATA ANALYSIS
BOOKINGS AND WAITING LIST



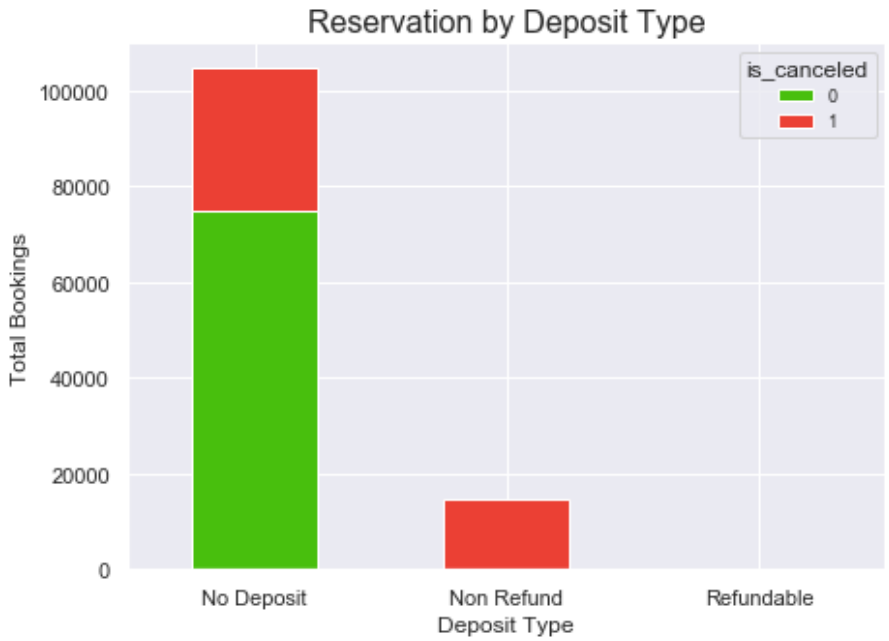
Just 3 per cent of bookings had to be placed on waiting list, with the minimum and maximum number of days in waiting list respectively being 39 and 183.

EXPLORATORY DATA ANALYSIS
LEAD TIME DISTRIBUTION (in days)



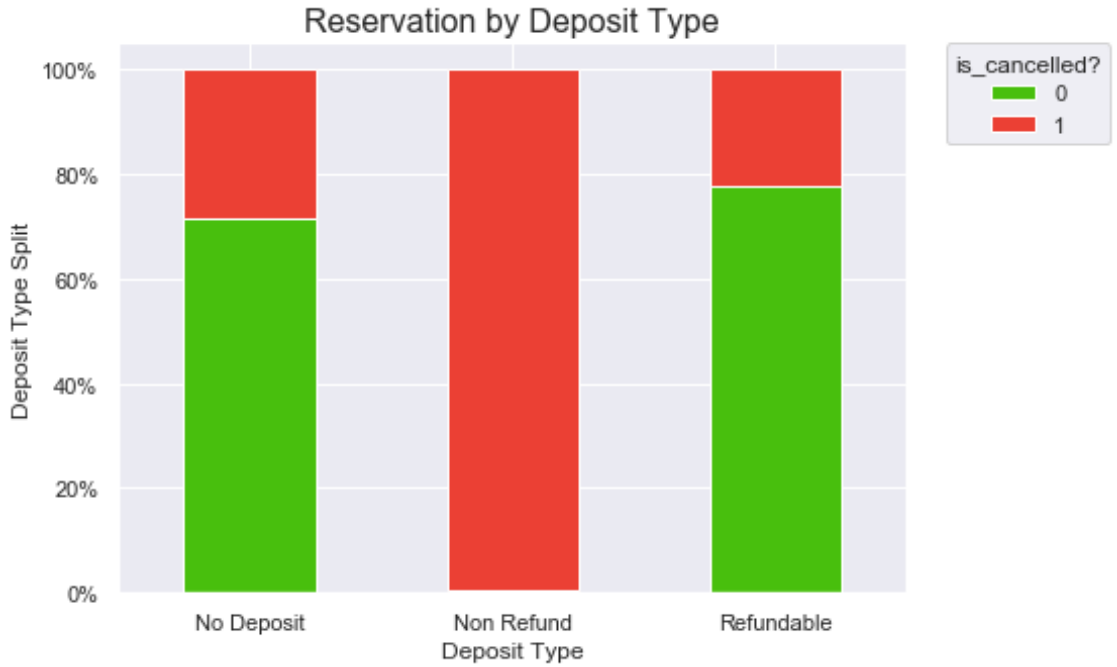
Lead time for cancelled booking is typically much higher compared to those of checked-in bookings.

EXPLORATORY DATA ANALYSIS
BOOKINGS BY DEPOSIT TYPE



'No Deposit' booking is the most common with approx. 87% of bookings associated with this deposit type, followed by 'Non refund' with 12% of the overall bookings. Refundable booking is very uncommon with just 0.1% (or 162) of bookings linked to this deposit type.

Nearly all 'Non refund' bookings result in cancellation (99.4%), Whereas, cancelled bookings account for just 28 per cent and 22 per cent of 'No deposit' and 'Refundable' booking types, respectively.



DATA CLEANSE AND PREPROCESSING

TRANSFORM RAW DATA TO A SUITABLE FORMAT FOR THE MODELLING PROCESS.

DATA CLEANSE AND PREPROCESSING
DEALING WITH OUTLIERS, MISSING AND IRRELEVANT VALUES

```
data.lead_time = np.where(data.lead_time > data.lead_time.quantile(0.95), data.lead_time.quantile(0.95),data.lead_time)
```

```
ata = data.drop(['arrival_date_year','arrival_date_week_number','arrival_date_day_of_month','company','agent'], axis=1)
```

```
data.required_car_parking_spaces = np.where(data.required_car_parking_spaces > 0,1,0)
data.groupby("required_car_parking_spaces")["required_car_parking_spaces"].value_counts()
```

```
required_car_parking_spaces  required_car_parking_spaces
0                            0                            111974
1                            1                              7416
Name: required_car_parking_spaces, dtype: int64
```

- Outliers in lead time were dealt by updating any value above 95th percentile with the cut-off (95th percentile) value.
- Simplified some features by grouping their data into binary options (Y/N): e.g. parking required?
- Irrelevant features (such as high % of missing data) were removed as they didn't provide any useful information.
- Created dummy variables for retained categorical data.

DATA CLEANSE AND PREPROCESSING
BALANCING, STANDARDIZING & SPLITTING DATASET

```
# First split data between cancelled and non-cancelled
cancelled_data_all = data_preprocessed[data_preprocessed.is_canceled == 1]
confirmed_data_all = data_preprocessed[data_preprocessed.is_canceled == 0]
```

```
# Next, determine the difference between the two subsets
to_remove = cancelled_data_all.shape[0] - confirmed_data_all.shape[0]
```

```
# Then, remove some data in non-cancelled data, equal to the difference between the two subsets
confirmed_data_all = confirmed_data_all[:to_remove]
```

```
print(confirmed_data_all.shape,confirmed_data_all.shape)
```

```
(44224, 57) (44224, 57)
```

```
# Finally, it's time to merge the subsets into a balanced dataset
balanced_dataset = pd.concat([cancelled_data_all, confirmed_data_all]
                             #, sort=True
                             #, ignore_index=True
                            )

# And confirm the dataset is balanced:
balanced_dataset.is_canceled.sum()/balanced_dataset.is_canceled.shape[0]
```

```
0.5
```

```
scaler = StandardScaler()
```

```
# Scaling just numerical features
scaler.fit(inputs_to_scale)
```

```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

```
scaled_inputs = scaler.transform(inputs_to_scale)
```

```
x_train, x_test, y_train, y_test = train_test_split(inputs, targets, #train_size = 0.8,
                                                    test_size = 0.2, random_state = 20)
```

- Balanced the dataset, to provide even split between cancelled and non-cancelled booking data.
- standardised numerical features in order to provide an similar scale across all features.
- Split the data into train & test sets, then shuffled the sets in random manner.

MODEL IMPLEMENTATION AND EVALUATION

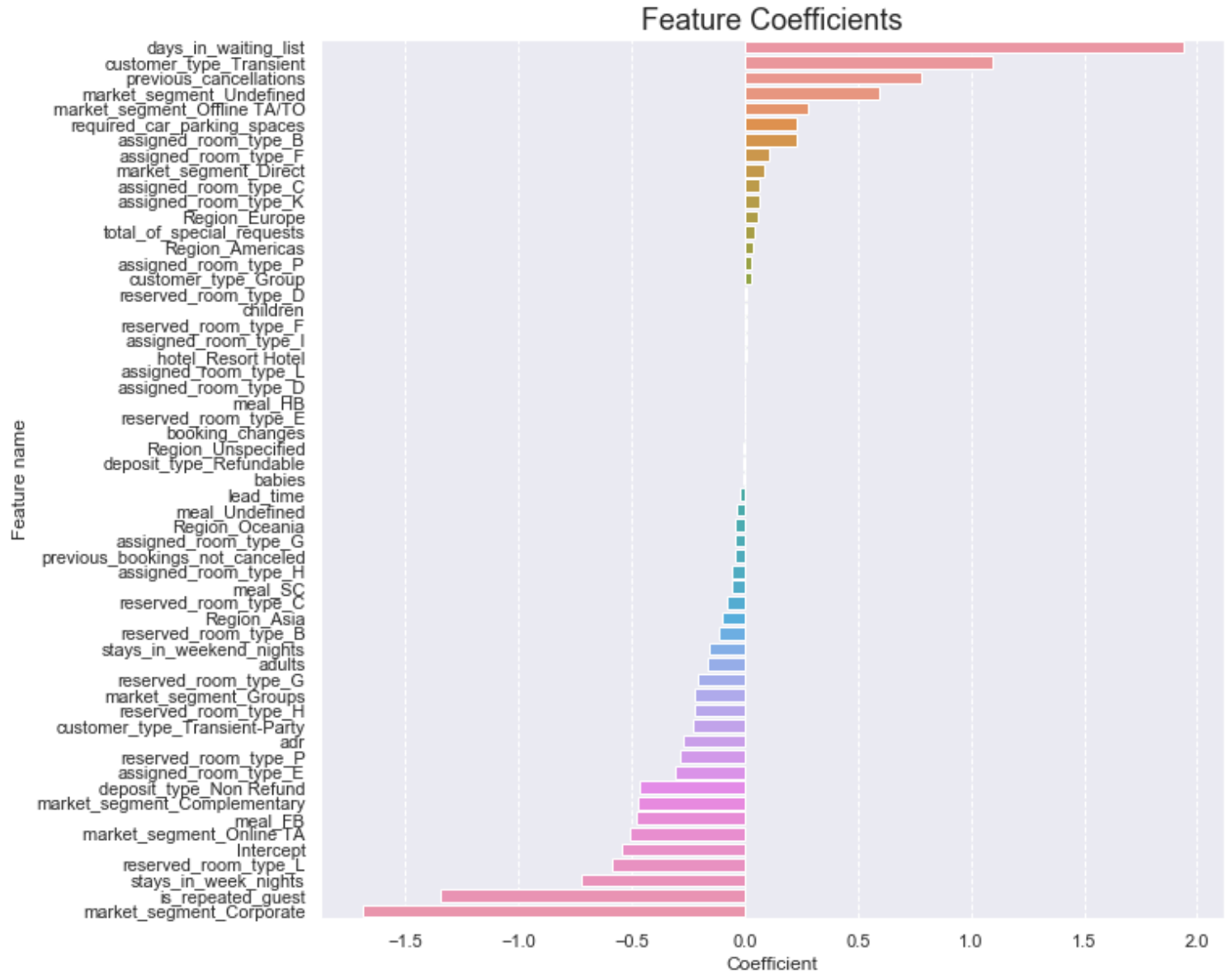
SUITABLE MODEL(S) WOULD BE SELECTED AND APPLIED AT THIS STAGE, EVALUATING THEIR PREDICTIVE POWER TO DETECT CANCELLATION.

MODEL IMPLEMENTATION AND EVALUATION

LOGISTIC MODEL

Given the context of this model, the charts shows the weight/importance of each feature, with values close to 0 (regardless of being positive or negative) weighting less.
In other words, the farther the feature coefficient is from zero, the more useful the feature is to the model.

*if coefficient is around 0, then the feature will have little or no impact at all on the output, as a weight of zero implies that no matter the feature value, it will be multiplied by 0 and the whole result will be unaffected by it.



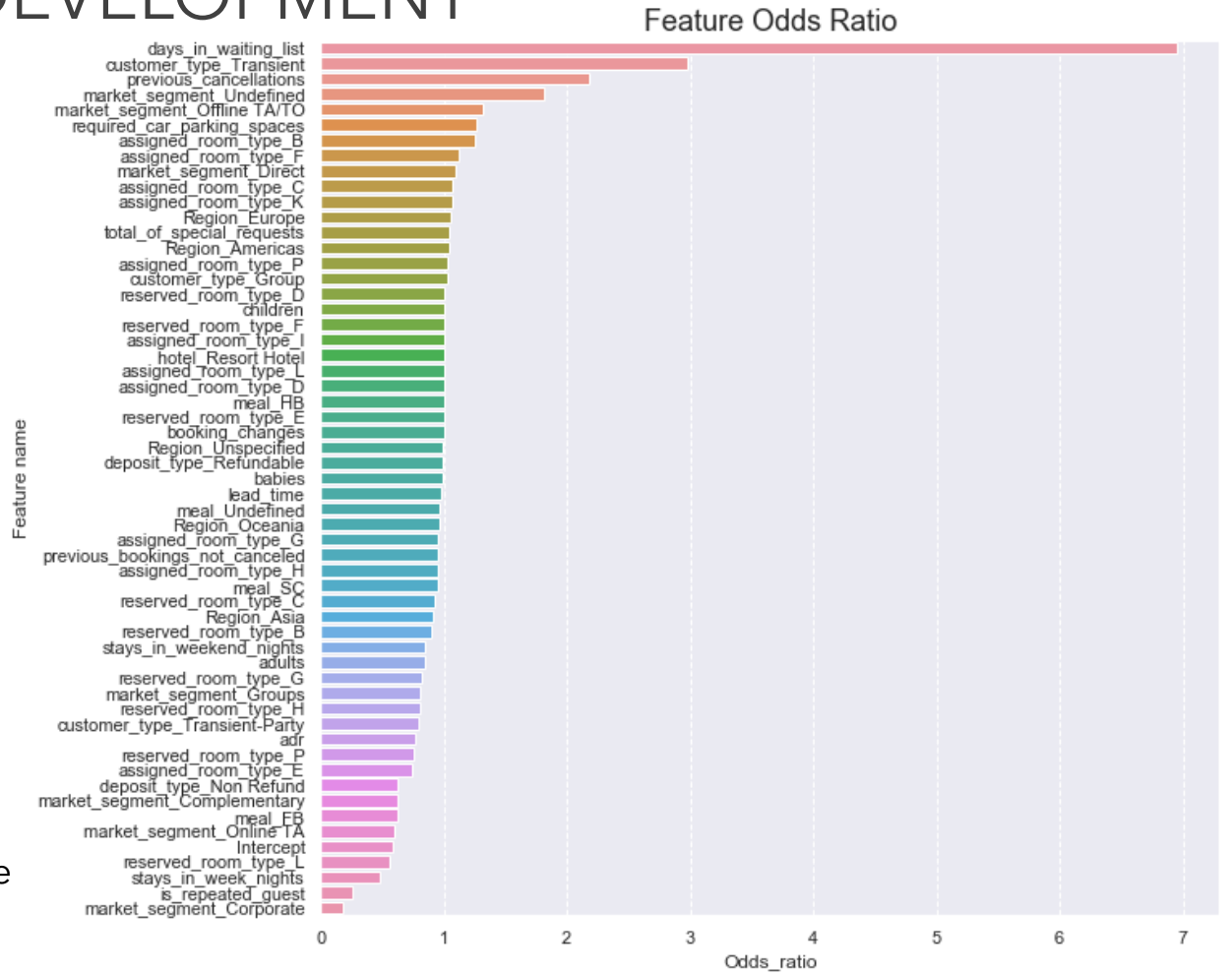
MODEL IMPLEMENTATION AND EVALUATION
LOGISTIC REGRESSION – MODEL DEVELOPMENT

Similar to coefficient, odds ratio (chart on right) shows the importance of each feature, wherein features with high odds ratio value weigh more.

The most important feature for this model is '**days spent in waiting list**'. There isn't a direct interpretability of it due to the fact that it's a numerical feature that has been standardised. Its odds ratio implies that for one standardised unit or for one standard deviation increase in days spent in waiting list, **it is 7 times more likely to cancel their bookings**. Which isn't much of a surprise as customer could and have the option to book elsewhere.

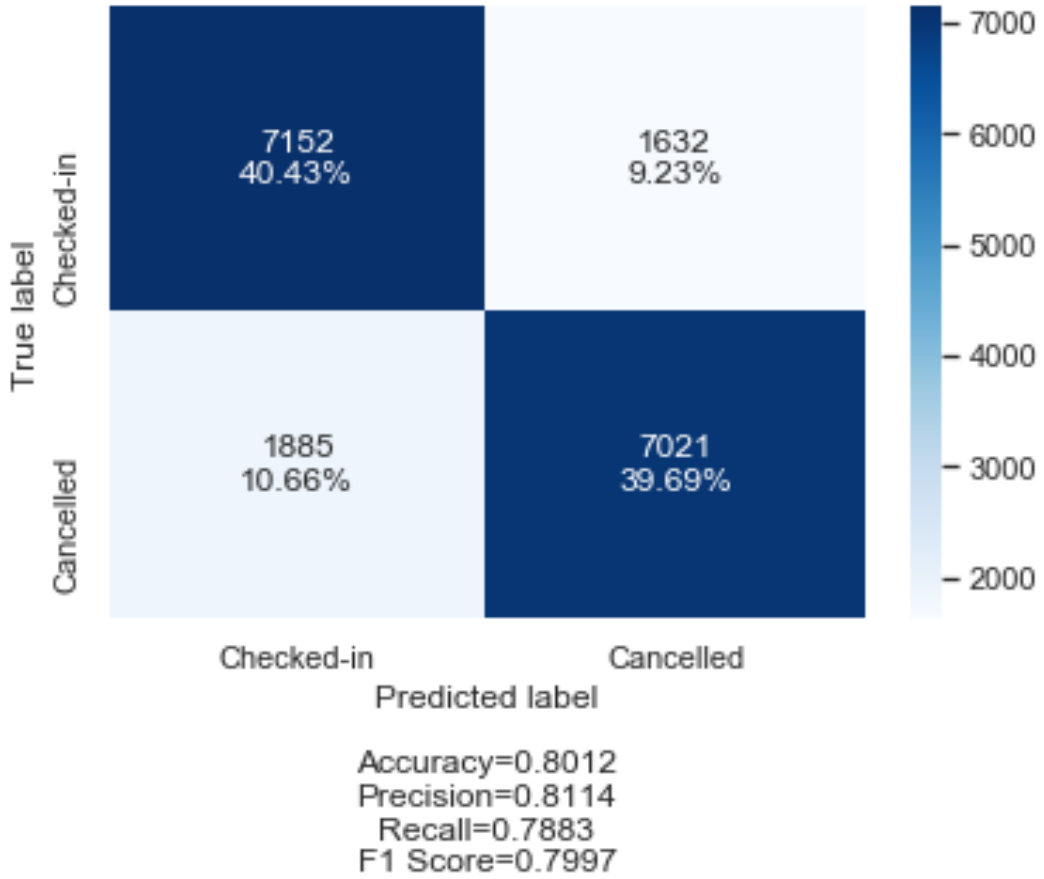
The second most important is '**Transient customers**', which is a binary feature, therefore there is a direct interpretability. Its odds ratio implies that a transient customer is approximately **3 times more likely to cancel their bookings**. Again, this isn't much of a surprise given the fact that 83% of bookings are associated to this customer type.

For bookings that include **weekend stays**, it can be said that for one standardised unit or for one standard deviation increase in weekend stay, the odds ratio are (1-0.85) or **15% lower than the base model (no weekend stay), i.e. less chance of cancelling**. This is because of its negative coefficient value.



*if odd ratio is around 1, then the feature will have little or no impact at all on the output. This is because for one unit change in the standardized feature, the odds increase by a multiple equal to the odds ratio. So if the odds ratio is one then the odds don't change at all.

MODEL IMPLEMENTATION AND EVALUATION
LOGISTIC REGRESSION – TESTING THE MODEL



- The logistic regression model has learned to accurately predict 80% of the observations. In other words, it was able to accurately determine whether a booking will eventually be cancelled in 4 out of 5 cases.

MODEL IMPLEMENTATION AND EVALUATION
NEURAL NETWORK – TRAINING AND TESTING THE MODEL

- Neural network was the second machine learning algorithm to be adopted for this project.
- To train the neural network model to predict cancellation, the following steps were taken:
  - Outlined the model.
  - Selected optimizer and loss function.
  - Fitted the outlined model to the dataset.
  - Fine tuned/adjusted the hyperparameters of the model in order to achieved the best validation accuracy.
- This was followed by testing the final prediction power of the model by fitting it on unseen (test) dataset.
- The final prediction power of NN model was 85.59 per cent, which was an improvement to the logistic regression model with its final prediction power of 80.12 per cent.

```
import tensorflow as tf

# Set the input and output sizes
input_size = 56
output_size = 2
# Use same hidden layer size for all hidden layers. Not a necessity.
hidden_layer_size = 800

# Defining the model for the problem
model = tf.keras.Sequential([
    # tf.keras.layers.Dense is basically implementing: output = activation(dot(input, weight) + bias)
    # it takes several arguments, but the most important ones for us are the hidden_layer_size and the activation funct
    tf.keras.layers.Dense(hidden_layer_size, activation='relu'), # 1st hidden layer
    #tf.keras.layers.Dense(hidden_layer_size, activation='relu'), # 2nd hidden layer
    tf.keras.layers.Dense(hidden_layer_size, activation='relu'), # 3rd hidden layer
    #tf.keras.layers.Dense(hidden_layer_size, activation='sigmoid'), # 4th hidden layer
    #tf.keras.layers.Dense(hidden_layer_size, activation='sigmoid'), # 5th hidden layer
    # the final layer is activated with softmax
    tf.keras.layers.Dense(output_size, activation='softmax') # output layer
])


### Choosing the optimizer and the loss function:

# preferred optimizer to use is Adaptive Moment Estimation (adam) , as it's one of the best optimiser available.
# the loss function,
# and the metrics we are interested in obtaining at each iteration
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])


### Training the model:

# set the batch size
batch_size = 100

# set a maximum number of training epochs
max_epochs = 100

# setting an early stopping mechanism, with patience=2, to be a bit tolerant against random validation loss increases
early_stopping = tf.keras.callbacks.EarlyStopping(patience=2)

# fit the model
# note that this time the train, validation and test data are not iterable
model.fit(train_inputs, # train inputs
          train_targets, # train targets
          batch_size=batch_size, # batch size
          epochs=max_epochs, # epochs that we will train for (assuming early stopping doesn't kick in)
          # callbacks are functions called by a task when a task is completed
          # task here is to check if val_loss is increasing
          callbacks=[early_stopping], # early stopping
          validation_data=(validation_inputs, validation_targets), # validation data
          verbose = 2 # getting essential information about the training process
          )
```

```
]: test_loss, test_accuracy = model.evaluate(test_inputs, test_targets)

8845/8845 [==============================] - 0s 27us/sample - loss: 0.3163 - accuracy: 0.8559

]: print('\nTest loss: {0:.2f}. Test accuracy: {1:.2f}%'.format(test_loss, test_accuracy*100.))

Test loss: 0.32. Test accuracy: 85.59%
```

CONCLUSION AND FINAL THOUGHT
SUMMARY

- The project began by identifying a problem/setting objective, and the objective here was to be able to predict a booking final outcome given a customer profile.
- Exploratory data analysis was conducted as the first process of the project, which allowed to gain a better understanding of the data by identifying the underlying trend within the data.
- This was followed by preprocessing or (feature engineering) step, with the aim of selecting appropriate features, and converting these features to a suitable format for the model(s) implementation.
- The final step was to select and apply Bi-predict models suitable for the problem of this project, which were Logistic Regression and Neural Network algorithms.
- The concluding observation following the application of the models in here was that Neural Network model performed slightly better in comparison to Logistic Regression model when applied to new and unseen (test) dataset, with prediction accuracy of just over 85 per cent for Neural Network, while the Logistic Regression provided a prediction accuracy of 80 per cent.