Hotel booking demand

Group 1

By

Students ID	Students Name	Section
444008713	Haya Alnashwan	51S
444008721	Lubna Bin Taleb	51S
444008709	Fajer Alshuwier	51S
444008729	Shahad Ben-Alameer	51S
444008690	Baylasan khalid almuqati	51S

This project is part of Assessments Grades

for the Course DS312: Data Mining

CCIS, PNU

Riyadh, KSA

First Semester 1446 - 1447H

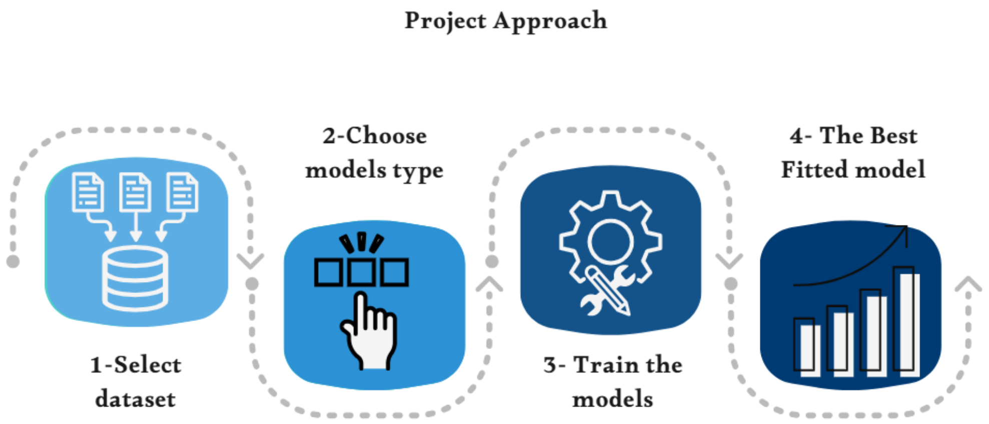
1. Introduction

Have you ever wondered what is the best time to book a hotel room? Or is there a duration to stay that could yield a better daily charged price? What if you wanted to predict whether a hotel is likely to experience an aversion to many special requests?

The booking dataset includes booking records of a city hotel and a resort hotel including some booking features such as when the reservation was made, the length of stay, number of adults, children, and infants, available parking spaces, among other features, that can be helpful to appraise how competitive hotel prices are based on factors similar to those for the hotel.

The project results can go a long way toward assisting travel agencies, clients, and hotel owners in reaching some informed decisions concerning booking. We will employ data-analytics techniques to explore price-driving factors and ascertain whether prices are competitive or not.

2. Project Approach



2.1 Step 1 – Data Preprocessing

The preprocessing steps in the document focus on cleaning the dataset by removing irrelevant features, identifying and potentially addressing missing values and duplicates, verifying data types, and calculating descriptive statistics. Each of these steps is crucial for preparing the data for effective analysis and modeling, ensuring that the resulting models are built on a solid foundation of clean and relevant data

2.1.1.Data Cleaning: Removing or correcting erroneous data entries.

2.1.2 Handling Missing Values: Filling in or removing missing data points to ensure a complete dataset e.g.(country column has 5 missing values)

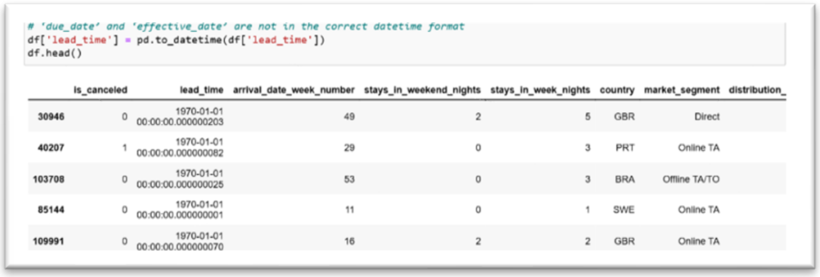
2.1.3 Encoding Categorical Variables: Converting categorical data into numerical format so that algorithms can process it

2.1.4 Feature Scaling: Normalizing or standardizing numerical features to ensure they contribute equally to distance calculations.

2.1.5 Removing Irrelevant Features: Certain columns that do not provide useful information for the analysis are removed. This helps in reducing noise and improving model performance.

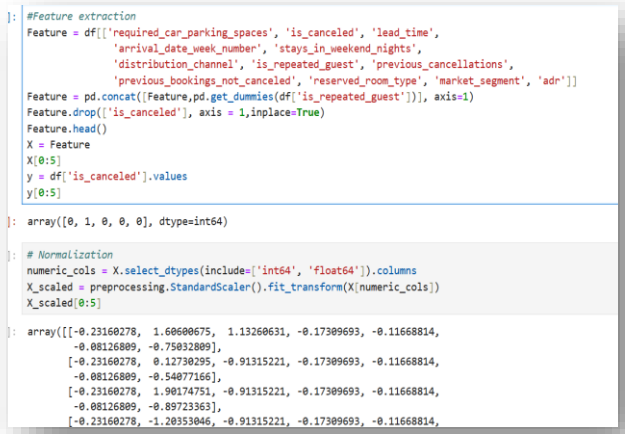
2.1.6 Checking for Duplicate Rows: Duplicate entries can skew the results of analyses and models, The Dataset contains 31 duplicate rows.

2.1.7 Descriptive Statistics Calculation: we use describe() method to calculate some basic statistics for numerical columns to identify potential outliers and help us with analysis.

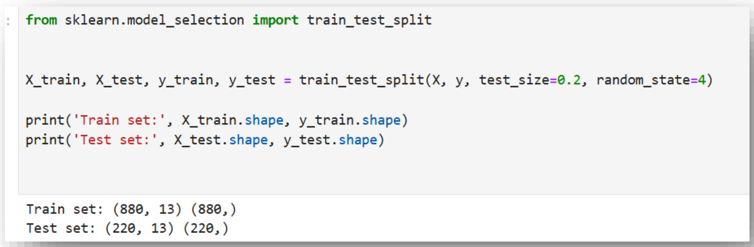


```
# 'due_date' and 'effective_date' are not in the correct datetime format
df['lead_time'] = pd.to_datetime(df['lead_time'])
df.head()
```

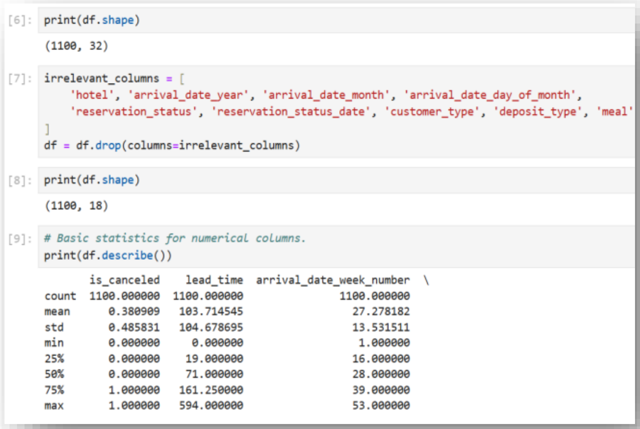
	is_canceled	lead_time	arrival_date_week_number	stays_in_weekend_nights	stays_in_week_nights	country	market_segment	distribution_
30946	0	1970-01-01 00:00:00.000000203	49	2	5	GBR	Direct	
40207	1	1970-01-01 00:00:00.000000082	29	0	3	PRT	Online TA	
103708	0	1970-01-01 00:00:00.000000025	53	0	3	BRA	Offline TA/TO	
85144	0	1970-01-01 00:00:00.000000001	11	0	1	SWE	Online TA	
109991	0	1970-01-01 00:00:00.000000070	16	2	2	GBR	Online TA	



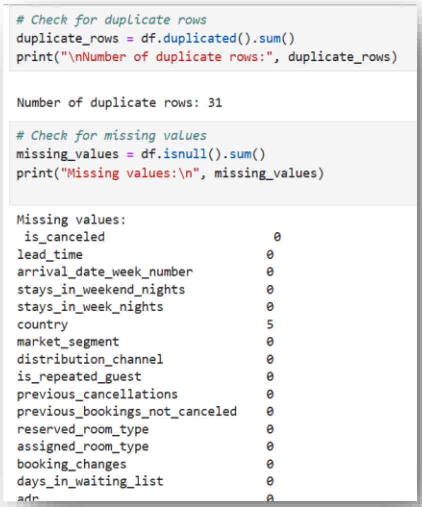
```
]: #Feature extraction
   Feature = df[['required_car_parking_spaces', 'is_canceled', 'lead_time',
                 'arrival_date_week_number', 'stays_in_weekend_nights',
                 'distribution_channel', 'is_repeated_guest', 'previous_cancellations',
                 'previous_bookings_not_canceled', 'reserved_room_type', 'market_segment', 'adr']]
   Feature = pd.concat([Feature,pd.get_dummies(df['is_repeated_guest'])], axis=1)
   Feature.drop(['is_canceled'], axis = 1,inplace=True)
   Feature.head()
   X = Feature
   X[0:5]
   y = df['is_canceled'].values
   y[0:5]

]: array([0, 1, 0, 0, 0], dtype=int64)

]: # Normalization
   numeric_cols = X.select_dtypes(include=['int64', 'float64']).columns
   X_scaled = preprocessing.StandardScaler().fit_transform(X[numeric_cols])
   X_scaled[0:5]

]: array([[-0.23160278,  1.60600675,  1.13260631, -0.17309693, -0.11668814,
           -0.08126809, -0.75032809],
          [-0.23160278,  0.12730295, -0.91315221, -0.17309693, -0.11668814,
           -0.08126809, -0.54077166],
          [-0.23160278,  1.90174751, -0.91315221, -0.17309693, -0.11668814,
           -0.08126809, -0.89723363],
          [-0.23160278, -1.20353046, -0.91315221, -0.17309693, -0.11668814,
```



```
: from sklearn.model_selection import train_test_split


  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4)

  print('Train set:', X_train.shape, y_train.shape)
  print('Test set:', X_test.shape, y_test.shape)

  Train set: (880, 13) (880,)
  Test set: (220, 13) (220,)
```



```
[6]: print(df.shape)

     (1100, 32)

[7]: irrelevant_columns = [
         'hotel', 'arrival_date_year', 'arrival_date_month', 'arrival_date_day_of_month',
         'reservation_status', 'reservation_status_date', 'customer_type', 'deposit_type', 'meal'
     ]
     df = df.drop(columns=irrelevant_columns)

[8]: print(df.shape)

     (1100, 18)

[9]: # Basic statistics for numerical columns.
     print(df.describe())

            is_canceled    lead_time  arrival_date_week_number  \
     count  1100.000000  1100.000000               1100.000000
     mean      0.380909   103.714545                 27.278182
     std       0.485831   104.678695                 13.531511
     min       0.000000     0.000000                  1.000000
     25%       0.000000    19.000000                 16.000000
     50%       0.000000    71.000000                 28.000000
     75%       1.000000   161.250000                 39.000000
     max       1.000000   594.000000                 53.000000
```



```
# Check for duplicate rows
duplicate_rows = df.duplicated().sum()
print("\nNumber of duplicate rows:", duplicate_rows)

Number of duplicate rows: 31

# Check for missing values
missing_values = df.isnull().sum()
print("Missing values:\n", missing_values)

Missing values:
 is_canceled                     0
lead_time                       0
arrival_date_week_number        0
stays_in_weekend_nights         0
stays_in_week_nights            0
country                         5
market_segment                  0
distribution_channel            0
is_repeated_guest               0
previous_cancellations          0
previous_bookings_not_canceled  0
reserved_room_type              0
assigned_room_type              0
booking_changes                 0
days_in_waiting_list            0
adr                             0
```

2.2 Step 2 – Models Training

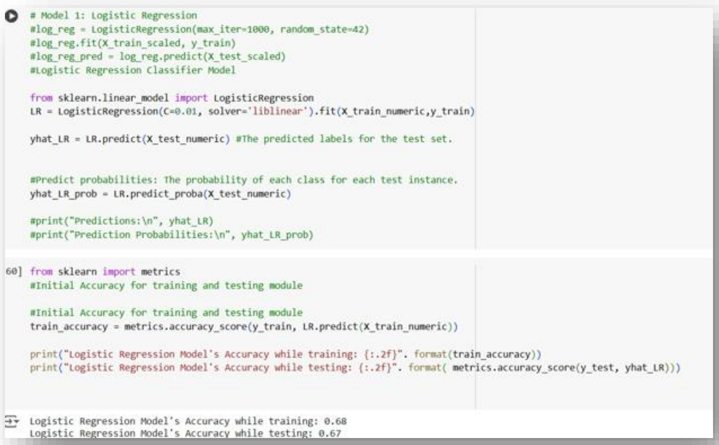
For this project, we aim to predict how likely it is for cancelation based on various attributes (e.g., lead time, market segment). The following classifiers are selected:

2.2.1 **Logistic Regression**: Logistic regression is a suitable model for binary classification (canceled vs. not canceled). It is computationally efficient and interpretable, making it a good choice for understanding the dataset's core predictive patterns.

2.2.2 **Random Forest**: Random Forest is selected for its robustness in handling feature interactions. Given the dataset's diversity (numerical and categorical features), Random Forest can help identify complex patterns in cancellation behavior. It also provides feature importance, which can help us understand key cancellation indicators.

2.2.3 **Support Vector Machine (SVM)**: SVM is chosen to handle high-dimensional spaces and to attempt optimal separation of bookings likely to be canceled. Kernel-based SVM can be particularly useful if the data has non-linear relationships.

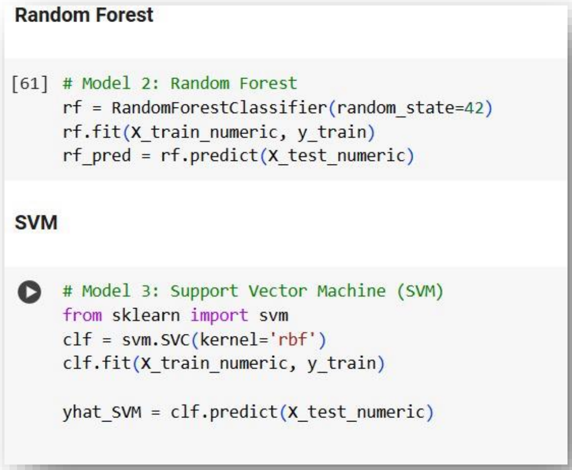
Each model was trained and tested using the Python scikit-learn library, using an 80-20 split between training and testing data to evaluate the model's performance. With 80% for training and 20% for testing.



```
# Model 1: Logistic Regression
#log_reg = LogisticRegression(max_iter=1000, random_state=42)
#log_reg.fit(X_train_scaled, y_train)
#log_reg_pred = log_reg.predict(X_test_scaled)
#Logistic Regression Classifier Model

from sklearn.linear_model import LogisticRegression
LR = LogisticRegression(C=0.01, solver='liblinear').fit(X_train_numeric,y_train)

yhat_LR = LR.predict(X_test_numeric) #The predicted labels for the test set.


#Predict probabilities: The probability of each class for each test instance.
yhat_LR_prob = LR.predict_proba(X_test_numeric)

#print("Predictions:\n", yhat_LR)
#print("Prediction Probabilities:\n", yhat_LR_prob)
```

```
60] from sklearn import metrics
    #Initial Accuracy for training and testing module

    #Initial Accuracy for training and testing module
    train_accuracy = metrics.accuracy_score(y_train, LR.predict(X_train_numeric))

    print("Logistic Regression Model's Accuracy while training: {:.2f}". format(train_accuracy))
    print("Logistic Regression Model's Accuracy while testing: {:.2f}". format( metrics.accuracy_score(y_test, yhat_LR)))
```

```
Logistic Regression Model's Accuracy while training: 0.68
Logistic Regression Model's Accuracy while testing: 0.67
```



Random Forest

```
[61] # Model 2: Random Forest
     rf = RandomForestClassifier(random_state=42)
     rf.fit(X_train_numeric, y_train)
     rf_pred = rf.predict(X_test_numeric)
```

SVM

```
# Model 3: Support Vector Machine (SVM)
from sklearn import svm
clf = svm.SVC(kernel='rbf')
clf.fit(X_train_numeric, y_train)

yhat_SVM = clf.predict(X_test_numeric)
```

2.3 Step 3 – Comparative Analysis

Models used: Random Forest, Support Vector Machine (SVM), and Logistic Regression

Overview:
Random Forest is an ensemble method that builds multiple decision trees to enhance accuracy and reduce overfitting. SVM identifies the optimal hyperplane to separate classes, excelling in high-dimensional spaces. Logistic Regression models the probability of binary outcomes using a logistic function.

Performance matrix:
Random Forest typically achieves high accuracy, especially with larger datasets, while SVM performs well but may need tuning. Logistic Regression is effective but struggles with non-linear relationships. In imbalanced datasets, Random Forest excels, with SVM also performing well when properly tuned. Logistic Regression may require adjustments for balance.

Training time:
Random Forest is usually fast to train due to parallel processing. SVM can be slow, especially with large datasets and non-linear kernels. Logistic Regression is generally the quickest to train.

Scalability:
Random Forest scales efficiently with data size, while SVM can struggle with large feature sets. Logistic Regression also scales well, particularly with linear relationships.

Parameter tuning:
Random Forest needs tuning for the number of trees and depth but is robust to overfitting. SVM requires careful tuning of parameters like kernel type. Logistic Regression is straightforward to tune, focusing on regularization.

Interpretability:

Random Forest is less interpretable than single trees but provides feature importance scores. SVM is interpretable with linear kernels, while Logistic Regression is highly interpretable due to its coefficients.

Use cases:

Random Forest is suited for complex datasets (e.g., image classification). SVM works well for smaller, high-dimensional data (e.g., text classification). Logistic Regression is best for binary classification with clear linear relationships.

Robustness to Outliers:

Random Forest is robust to outliers, while SVM and Logistic Regression are sensitive, potentially skewing results.

Conclusion:

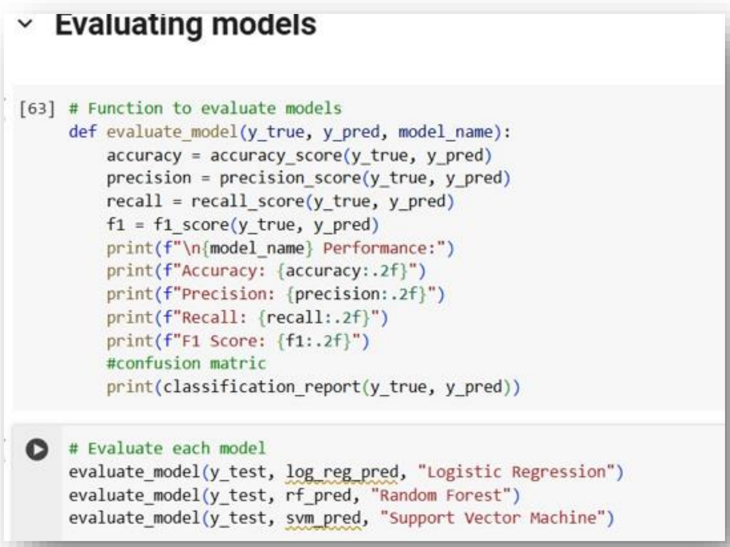
Random Forest is ideal for large, complex datasets; SVM excels in high dimensions; and Logistic Regression is preferred for interpretable models. The choice depends on dataset characteristics and analysis goals, with cross-validation recommended to find the best model.

2.4 Step 4 – Best Performing Models

To determine the best model, we considered the following criteria:

- **Accuracy**: Ensures the model performs well across all bookings, not just those prone to cancellation.
- **Precision and Recall**: Especially important as we want to accurately identify actual cancellations (high recall) without over-predicting them (precision).
- **F1-score**: Balances precision and recall to evaluate the model's overall robustness.
- **Feature Importance and Interpretability**: For models like Random Forest a, we looked at feature importance to interpret which factors most influence cancellations.

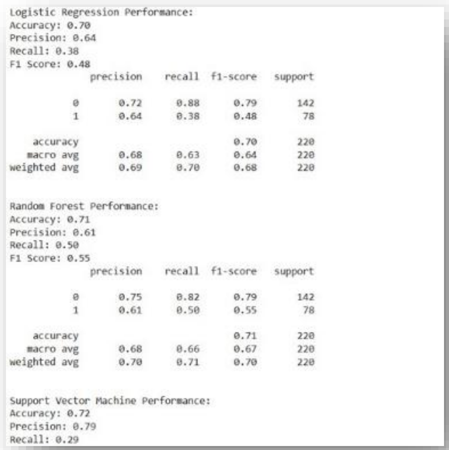
The model that best balances these metrics is selected as the best-performing model. The thresholds were set at for F1-score above 0.50, with precision and recall individually above 0.45. As well as we are comparing between the models themselves.
By comparing the models on these metrics, we identified which classifier most effectively predicts booking cancellations and informs hotel management on key drivers, enabling them to take actions to reduce cancellations.



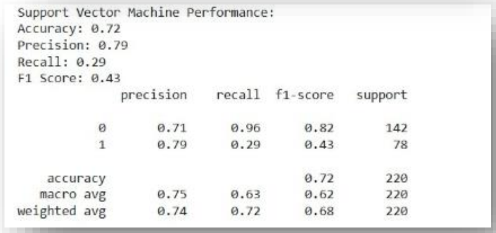
Evaluating models

```
[63] # Function to evaluate models
     def evaluate_model(y_true, y_pred, model_name):
         accuracy = accuracy_score(y_true, y_pred)
         precision = precision_score(y_true, y_pred)
         recall = recall_score(y_true, y_pred)
         f1 = f1_score(y_true, y_pred)
         print(f"\n{model_name} Performance:")
         print(f"Accuracy: {accuracy:.2f}")
         print(f"Precision: {precision:.2f}")
         print(f"Recall: {recall:.2f}")
         print(f"F1 Score: {f1:.2f}")
         #confusion matric
         print(classification_report(y_true, y_pred))
```

```
# Evaluate each model
evaluate_model(y_test, log_reg_pred, "Logistic Regression")
evaluate_model(y_test, rf_pred, "Random Forest")
evaluate_model(y_test, svm_pred, "Support Vector Machine")
```



```
Logistic Regression Performance:
Accuracy: 0.70
Precision: 0.64
Recall: 0.38
F1 Score: 0.48
              precision    recall  f1-score   support

           0       0.72      0.88      0.79       142
           1       0.64      0.38      0.48        78

    accuracy                           0.70       220
   macro avg       0.68      0.63      0.64       220
weighted avg       0.69      0.70      0.68       220


Random Forest Performance:
Accuracy: 0.71
Precision: 0.61
Recall: 0.50
F1 Score: 0.55
              precision    recall  f1-score   support

           0       0.75      0.82      0.79       142
           1       0.61      0.50      0.55        78

    accuracy                           0.71       220
   macro avg       0.68      0.66      0.67       220
weighted avg       0.70      0.71      0.70       220


Support Vector Machine Performance:
Accuracy: 0.72
Precision: 0.79
Recall: 0.29
```



```
Support Vector Machine Performance:
Accuracy: 0.72
Precision: 0.79
Recall: 0.29
F1 Score: 0.43
              precision    recall  f1-score   support

           0       0.71      0.96      0.82       142
           1       0.79      0.29      0.43        78

    accuracy                           0.72       220
   macro avg       0.75      0.63      0.62       220
weighted avg       0.74      0.72      0.68       220
```

3. Project Requirements

The dataset includes 119,390 hotel booking records with 32 different attributes, such as booking status, guest information, stay length, and reservation status. To make analysis easier, the code selects 1,100 random entries using a random state of 42. Instead of handling the entire dataset, which can be computationally heavy, this sampling method allow to work with a more manageable portion. This way, they can quickly explore and learn from the data without being overwhelmed by its size.

For analysis, there are two main tasks: predicting cancellations and estimating the average daily rate. For cancellation prediction, techniques like logistic regression, decision trees, random forests, and gradient boosting are suitable. On the other hand, predicting the average daily rate is a regression task that could use algorithms such as gradient boosting, random forests, and linear regression.

To measure performance, different indicators are used depending on the task. For classification tasks, metrics like F1 score, ROC-AUC, recall, accuracy, and precision are important. For regression tasks, R-squared, Mean Squared Error (MSE), and Mean Absolute Error (MAE) are the key metrics to consider.

3.1 Dataset Description

- **Is_canceled: (Integer)** 1 if canceled, 0 if not.
- **lead_time:(Integer)** Days between booking and arrival.
- **arrival_date_week_number: (Integer)** Week number of arrival.
- **stays_in_weekend_nights: (Integer)** Number of weekend nights in the stay.
- **stays_in_week_nights: (Integer)** Number of week nights in the stay.
- **country: (Categorical)** Guest's country of origin.
- **market_segment: (Categorical)** Booking market segment (e.g., online, corporate).

- **distribution_channel: (Categorical)** Booking channel (e.g., direct, agent).

- is_repeated_guest: (Integer) 1 if repeat guest, 0 if first-time.

- **previous_cancellations: (Integer)** Count of guest's prior cancellations.

.**- previous_bookings_not_canceled: (Integer)** Count of guest's non-canceled bookings

- **reserved_room_type: (Categorical)** Type of room originally reserved.

- assigned_room_type: (Categorical) Room type assigned at arrival.

- **booking_changes: (Integer)** Number of booking changes made.

- **days_in_waiting_list: (Integer)** Days on the waiting list before confirmation.

- **adr: (Float)** Average daily rate for the booking.

- **required_car_parking_spaces: (Integer)** Number of parking spaces needed.

- total_of_special_requests: (Integer) Count of special requests made by the guest.

3.2 Machine Learning Algorithms Description

Several machine learning methods are used in this project to examine and forecast traffic trends. Every algorithm has special advantages that make it applicable to various traffic data features. Here is a brief synopsis of each one's functions and ideal applications.

3.2.1 Random Forest

Random Forest is like an entire forest of decision-makers. Every "tree" in the forest focuses on a distinct area of the dataset since each one receives a random subset of the data. After that, each tree collaborates to produce a final forecast. This "team effort" method works well for managing complex data and is accurate. It is frequently used for identifying various traffic levels and predicting continuous values since it is quite dependable, especially when dealing with noisy data.

Pros: Able to handle missing values, maintains big datasets, and is highly accurate.

Cons: More difficult to understand than a single decision tree, and slower to train.

3.2.2 Support Vector Machine (SVM)

The Support Vector Machine (SVM) functions similarly to a tool for defining boundaries. Based on the data, it attempts to distinguish as clearly as possible between various groups (such as high vs. low traffic). SVM is excellent at handling complex patterns and works well with high-dimensional data. SVM can be a good option for traffic analysis when we need accurate classifications, but it does take more processing power, particularly when dealing with huge datasets.

Pros: Handles intricate patterns, performs well with high-dimensional data, and is less likely to overfit.

Cons: Requires a lot of memory and processing power; interpretation is more difficult.

3.2.3 Logistic Regression

A statistical technique for binary classification, logistic regression forecasts the likelihood of a result depending on input features. This function uses the logistic (sigmoid) function to produce numbers between 0 and 1.

Pros: Simplicity, Interpretability, Efficiency, Probabilistic Output

Cons: The concept of linearity It is assumed that the characteristics and the log-odds of the result have a linear connection, Restricted to Binary Outcomes, Outlier Sensitivity, Performance with Unbalanced Data.

3.3 Performance Metrics Description

3.3.1 Classification Metrics

- Classification Accuracy

This metric measures the proportion of correct predictions (both cancellations and non-cancellations) among all predictions made for hotel bookings.

$$\text{Accuracy} = (TP + TN) / (TP + FN + FP + TN)$$

- Precision

Precision indicates the accuracy of positive predictions, specifically how many of the predicted cancellations were actual cancellations. This is crucial for minimizing false alarms in booking cancellations, which can affect hotel operations.

$$\text{Precision} = TP / (TP + FP)$$

- Recall

Recall measures the model's ability to identify all actual cancellations. It highlights how effectively the model captures true positive instances, which is vital for hotel management to understand potential revenue loss due to cancellations.

$$\text{Recall} = TP / (TP + FN)$$

- F-Score

The F1 Score provides a balance between precision and recall, giving an overall measure of the model's accuracy in predicting cancellations. This is particularly useful in scenarios where both false positives and false negatives carry significant consequences for hotel management.

$$\text{F-Score} = 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall})$$

3.3.2 Regression Metrics

When estimating the average daily rate (ADR) for hotel bookings, the following regression metrics are commonly used:

- Mean Absolute Error (MAE)

  MAE measures the average magnitude of errors in predictions, providing insight into how far off the predicted ADR values are from the actual values without considering direction. It's crucial for understanding the overall accuracy of pricing models.

$$\mathrm{MAE} = \frac{\grave{1}}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

- Root Mean Square Error (RMSE)

RMSE provides a measure of the average error, giving more weight to larger
errors by squaring them before averaging. This metric helps in identifying significant prediction errors in ADR, which is critical for pricing strategies.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

- Coefficient of Determination (R^2)
- R^2 quantifies how well the predicted ADR values explain the variance in actual values. A higher R^2 indicates a better fit of the model, which is essential for evaluating the effectiveness of pricing strategies in the hotel industry.

$$R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2}$$

4. Project Analysis

4.1 Logistic Regression:

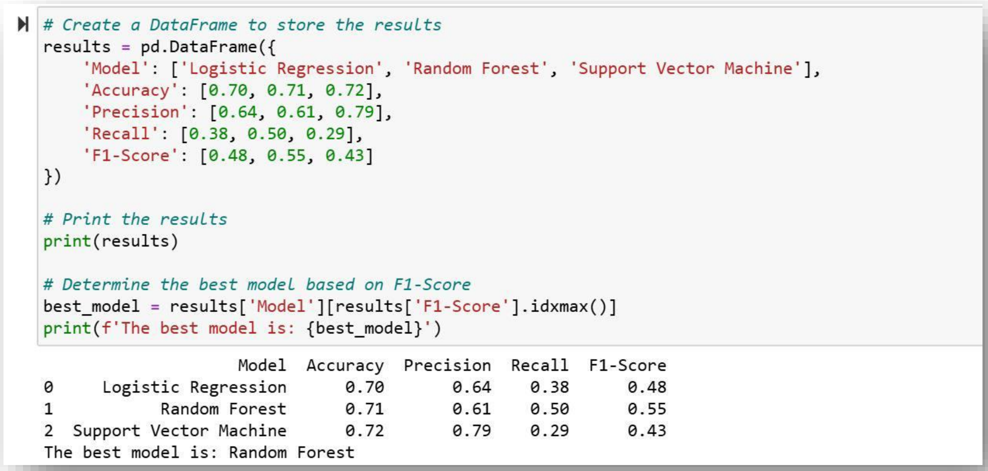
The accuracy of 70% means that the model was able to classify 70% of instances in the dataset correctly. It only detected a positive case of 64% which is relatively low (it has a precision of 0.64), but its recall is even lower with a score of 0.38 meaning it missed quite a number of actual positives! With an F1 score of 0.48 it indicates a reasonable balance between precision and recall.

4.2 Random Forest:

This model gave 71% accuracy which was still better than Logistic Regression. It has a precision of 0.61 meaning it correctly predicted positive predictions only 61% of the time. It has 0.50 as recall which shows that half of actual positives were captured. With an F1 score of 0.55, the balance here is far more favorable compared to Logistic Regression, meaning this would be a more attractive option for applications that require reliable predictions.

4.3 Support Vector Machine (SVM):

This means SVM succeeds in predicting instances of this dataset most accurately with an accuracy of 72%. A precision of 0.79 indicates that it correctly classifies a large number of true positives, but the much lower recall value of 0.29 reveals it is missing many actual positives. The F1 score of 0.43 shows the compromise due to high precision still not covering the low recall.



```python
# Create a DataFrame to store the results
results = pd.DataFrame({
    'Model': ['Logistic Regression', 'Random Forest', 'Support Vector Machine'],
    'Accuracy': [0.70, 0.71, 0.72],
    'Precision': [0.64, 0.61, 0.79],
    'Recall': [0.38, 0.50, 0.29],
    'F1-Score': [0.48, 0.55, 0.43]
})

# Print the results
print(results)

# Determine the best model based on F1-Score
best_model = results['Model'][results['F1-Score'].idxmax()]
print(f'The best model is: {best_model}')
```

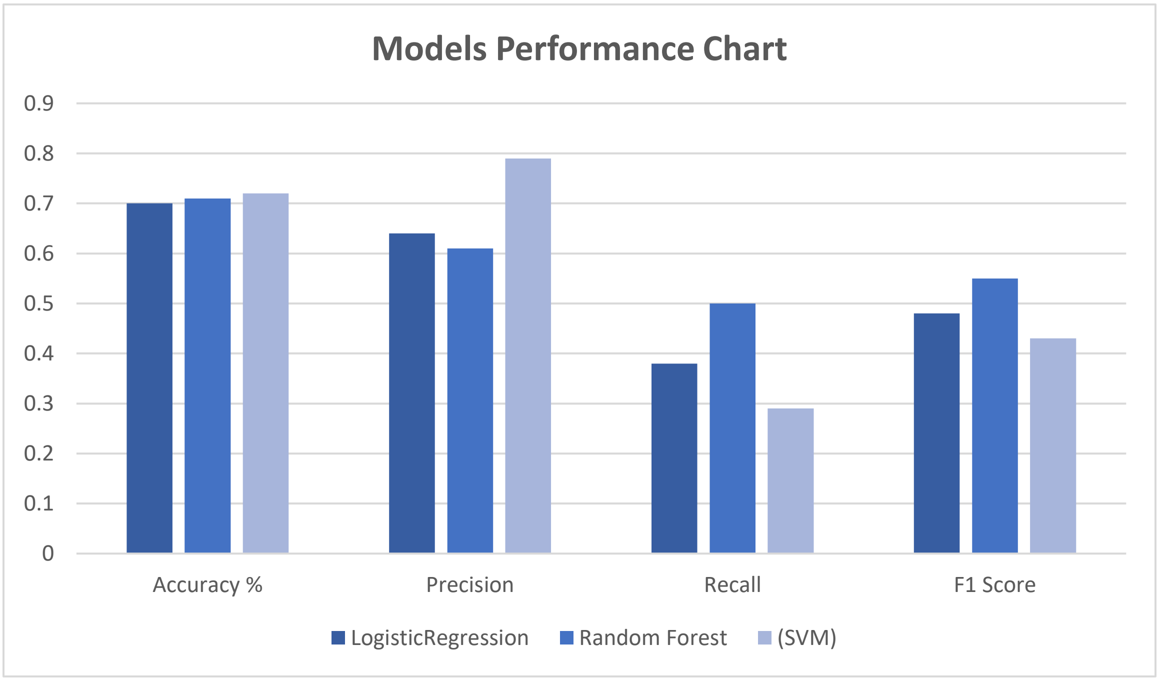
```
                    Model  Accuracy  Precision  Recall  F1-Score
0     Logistic Regression      0.70       0.64    0.38      0.48
1           Random Forest      0.71       0.61    0.50      0.55
2  Support Vector Machine      0.72       0.79    0.29      0.43
The best model is: Random Forest
```

4.4 Comparison

The results of our analysis are summarized in Table 4-1 below:

ML Algorithms	Models' Performance			
	Accuracy %	Precision	Recall	F1 Score
Logistic Regression	0.70	0.64	0.38	0.48
Random Forest	0.71	0.61	0.50	0.55
(SVM)	0.72	0.79	0.29	0.43

The table shows that SVM achieved the highest accuracy, while Random Forest had a better balance of precision and recall, making it the most practical model for predicting cancellations.



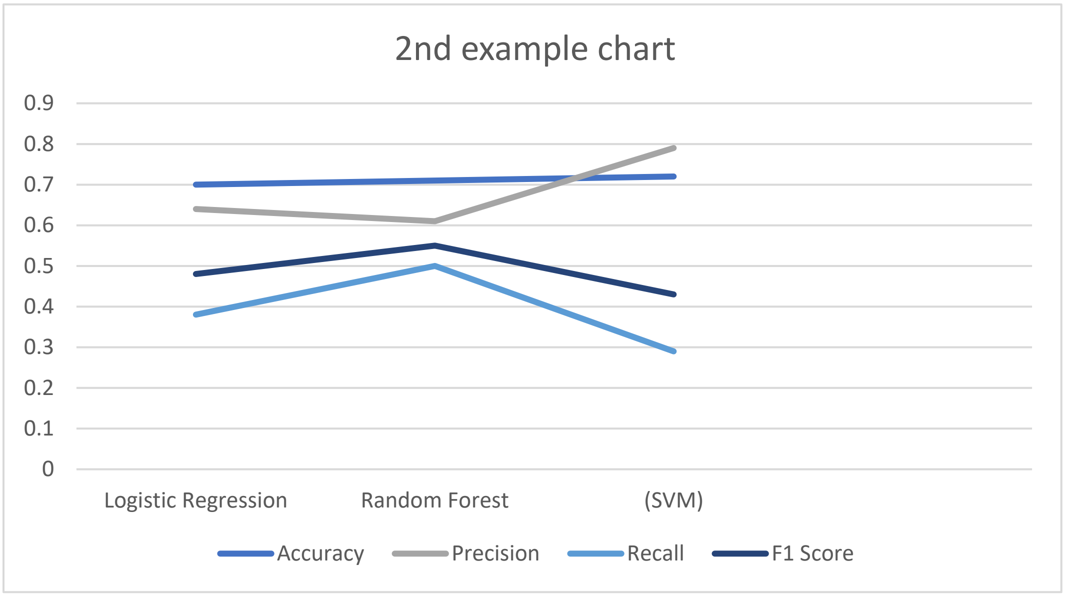
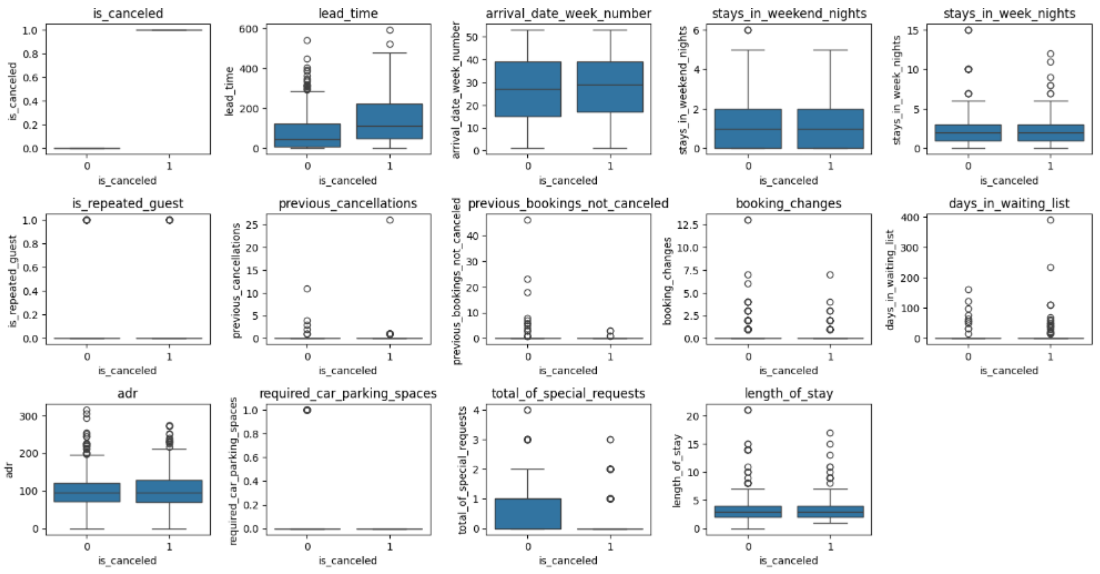
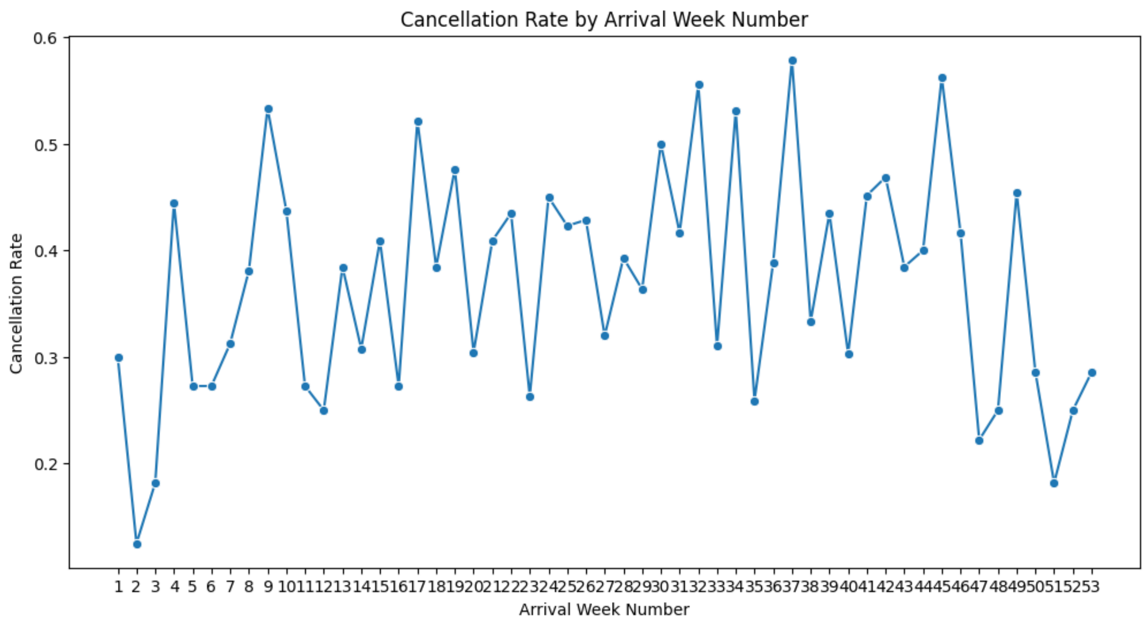
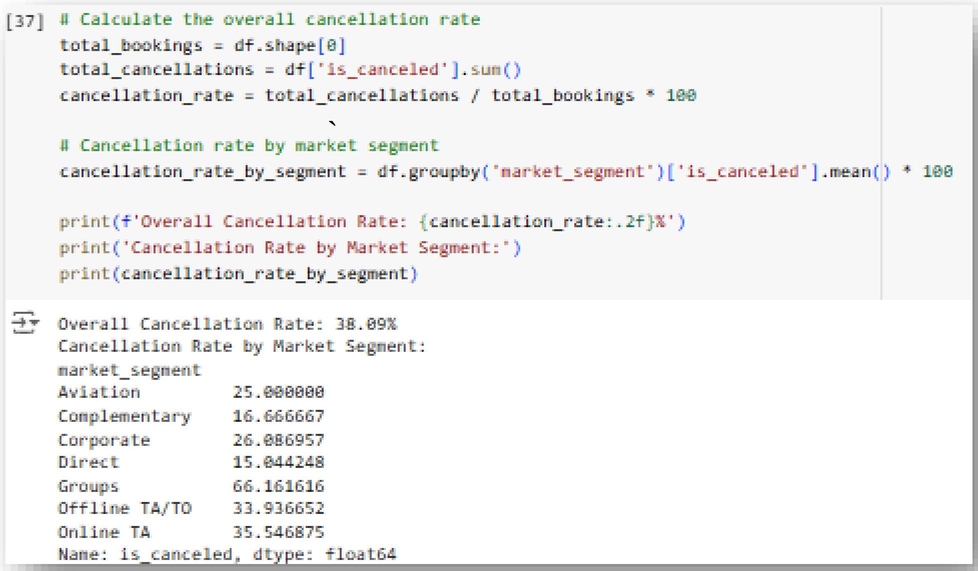


Figure 1: This is 2nd example






```
[37] # Calculate the overall cancellation rate
     total_bookings = df.shape[0]
     total_cancellations = df['is_canceled'].sum()
     cancellation_rate = total_cancellations / total_bookings * 100

     # Cancellation rate by market segment
     cancellation_rate_by_segment = df.groupby('market_segment')['is_canceled'].mean() * 100

     print(f'Overall Cancellation Rate: {cancellation_rate:.2f}%')
     print('Cancellation Rate by Market Segment:')
     print(cancellation_rate_by_segment)
```

```
Overall Cancellation Rate: 38.09%
Cancellation Rate by Market Segment:
market_segment
Aviation         25.000000
Complementary    16.666667
Corporate        26.086957
Direct           15.044248
Groups           66.161616
Offline TA/TO    33.936652
Online TA        35.546875
Name: is_canceled, dtype: float64
```

5. Conclusion

The analysis of the hotel booking and predictive models of hotel booking cancellations, revealed that among all models, the Random Forest classifier is the best. Evaluated procedures including Logistic Regression and Support Vector Machines (SVM) all had lower overall accuracy levels, precision, recall and F1 scores in comparison to this model which performed the best.

Hence, as implied by the overall performance of the Random Forest model, it is the best in understanding the intricacies of hotel booking data and estimating the probability of a cancellation occurring. It is therefore the best option to be used in practical scenarios, which involves booking hotel cancellation predictions.

Even though the Logistic Regression and SVM models did have the predicted performance, their metrics could not compete with those of the random forest classifier. In this perspective, strategies such as hyperparameter tuning or inclusion of more relevant features may prove to be beneficial; however, the random forest model still outperformed these other models on analysis.

References

[1] J.Mostipak, Hotel Booking Demand [online]. Avaliable: https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand .[Accessed: Nov.13 , 2024].

[2] Adam Shafi “Random forest Classification with Scikit-Learn”,Updated October 2024.

[3] Salvador García, Julian Luengo, Francisco Herrera “Data preprocessing in data mining”.

[4] Scikit-learn Developers, "Metrics and scoring: quantifying the quality of predictions," Scikit-learn Documentation. [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html. [Accessed: Nov. 15, 2024].

[5] C. M. Bishop, Pattern Recognition and Machine Learning. New York, NY, USA: Springer, 2006