# HOTEL BOOKING PREDICTION

Meliane Ayebah
Matricola: A04402

# SOMMARIO

# INTRODUZIONE

L’argomento trattato per il progetto finale è hotel booking prediction.

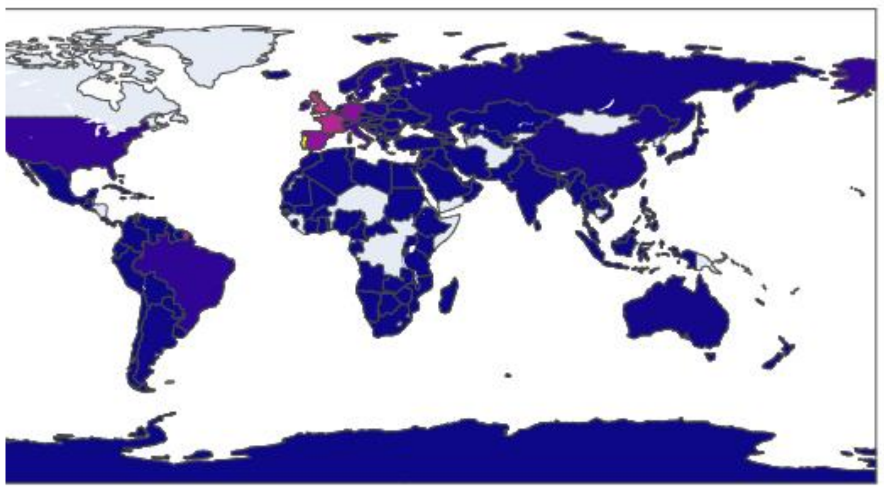
L’obiettivo è quello di creare degli stimatori significativi a partire dal dataset a disposizione e selezionare il modello migliore per predire la prenotazione confrontandolo con i punteggi di accuratezza di diversi modelli di Machine Learning.

# EXPLORATION DATA ANALYSIS (EDA)

Nella fase di exploration si cercherà a rispondere ai seguenti quesiti:

- Da dove proviene la maggior parte degli ospiti?
- Quanto pagano gli ospiti a notte per una camera ?
- Come varia il prezzo a notte di una camera durante l'anno?

# DA DOVE PROVIENE LA MAGGIOR PARTE DEGLI OSPITI?



La maggior parte degli ospiti soggiornano nei due hotel (Hotel City e Resort Hotel) e sono tutti provenienti per la maggior parte dal Portogallo e dagli altri paesi europei.

In [498]:
```
country_wise_guests = df[df['is_canceled'] == 0]['country'].value_counts().reset_inc
country_wise_guests.columns = ['country', 'No of guests']
country_wise_guests
```

Out[498]:

| | country | No of guests |
|---|---|---|
| 0 | PRT | 20977 |
| 1 | GBR | 9668 |
| 2 | FRA | 8468 |
| 3 | ESP | 6383 |
| 4 | DEU | 6067 |
| ... | ... | ... |
| 161 | AIA | 1 |
| 162 | SYC | 1 |
| 163 | MRT | 1 |
| 164 | MLI | 1 |
| 165 | LCA | 1 |

166 rows × 2 columns

# COME VARIA IL PREZZO A NOTTE DI UNA CAMERA DURANTE L'ANNO E QUANTO GLI OSPITI PAGANO UNA CAMERA A NOTTE?

Questa tabella mostra chiaramente che i prezzi del Resort Hotel sono molto più alti durante l'estate mentre i prezzi dell'Hotel city variano di meno e sono più alti durante la primavera e l'autunno.

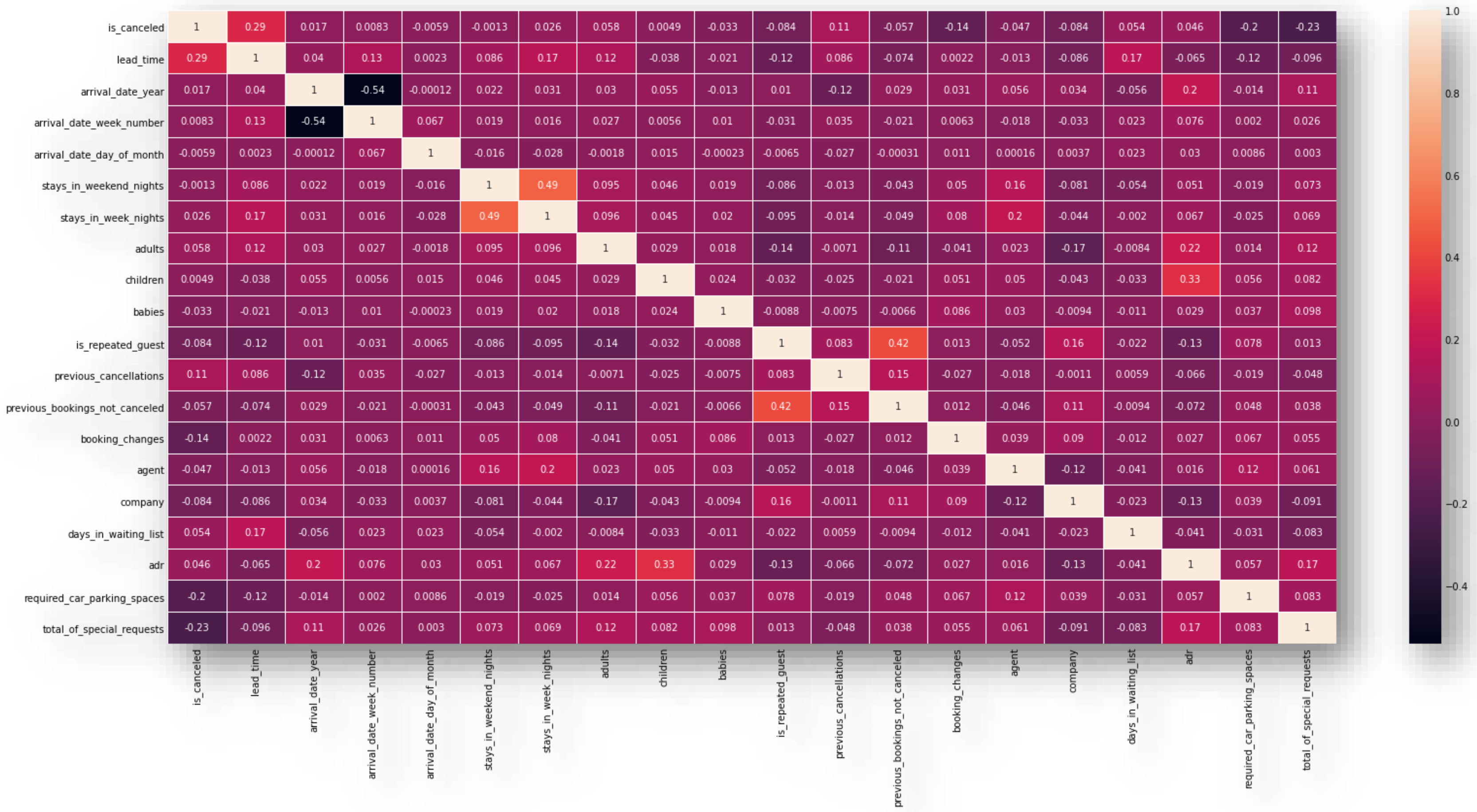
In [505]:
```
final_hotel = resort_hotel.merge(city_hotel, on = 'arrival_date_month')
final_hotel.columns = ['month', 'price_for_resort', 'price_for_city_hotel']
final_hotel
```

Out[505]:

| | month | price_for_resort | price_for_city_hotel |
|---|---|---|---|
| 0 | April | 75.867816 | 111.962267 |
| 1 | August | 181.205892 | 118.674598 |
| 2 | December | 68.410104 | 88.401855 |
| 3 | February | 54.147478 | 86.520062 |
| 4 | January | 48.761125 | 82.330983 |
| 5 | July | 150.122528 | 115.818019 |
| 6 | June | 107.974850 | 117.874360 |
| 7 | March | 57.056838 | 90.658533 |
| 8 | May | 76.657558 | 120.669827 |
| 9 | November | 48.706289 | 86.946592 |
| 10 | October | 61.775449 | 102.004672 |
| 11 | September | 96.416860 | 112.776582 |

# DATA PRE-PROCESSING

In questa slide osserviamo che le features con alta correlazione sono tutte collocate in corrispondenza della diagonale.



# MODELS BUILDING

Per poter valutare il modello migliore e scegliere quello efficiente si andrà ad implementare i seguenti modelli:

- Logistic Regression
- Decision Tree Classifier
- Random Forest
- KNN
- Ada Boost Classifier
- Gradient Boosting Classifier

# LOGISTIC REGRESSION

```
In [530]: lr = LogisticRegression()
          lr.fit(X_train, y_train)

          y_pred_lr = lr.predict(X_test)

          acc_lr = accuracy_score(y_test, y_pred_lr)
          conf = confusion_matrix(y_test, y_pred_lr)
          clf_report = classification_report(y_test, y_pred_lr)

          print(f"Accuracy Score of Logistic Regression is : {acc_lr}")
          print(f"Confusion Matrix : \n{conf}")
          print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of Logistic Regression is : 0.8108100550848643
Confusion Matrix :
[[21287  1246]
 [ 5520  7710]]
Classification Report :
              precision    recall  f1-score   support

           0       0.79      0.94      0.86     22533
           1       0.86      0.58      0.70     13230

    accuracy                           0.81     35763
   macro avg       0.83      0.76      0.78     35763
weighted avg       0.82      0.81      0.80     35763
```

# DECISION TREE CLASSIFIER

```
In [531]: dtc = DecisionTreeClassifier()
          dtc.fit(X_train, y_train)

          y_pred_dtc = dtc.predict(X_test)

          acc_dtc = accuracy_score(y_test, y_pred_dtc)
          conf = confusion_matrix(y_test, y_pred_dtc)
          clf_report = classification_report(y_test, y_pred_dtc)

          print(f"Accuracy Score of Decision Tree is : {acc_dtc}")
          print(f"Confusion Matrix : \n{conf}")
          print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of Decision Tree is : 0.9416156362721249
Confusion Matrix :
[[21490  1043]
 [ 1045 12185]]
Classification Report :
              precision    recall  f1-score   support

           0       0.95      0.95      0.95     22533
           1       0.92      0.92      0.92     13230

    accuracy                           0.94     35763
   macro avg       0.94      0.94      0.94     35763
weighted avg       0.94      0.94      0.94     35763
```

# RANDOM FOREST

```
In [532]: rd_clf = RandomForestClassifier()
          rd_clf.fit(X_train, y_train)

          y_pred_rd_clf = rd_clf.predict(X_test)

          acc_rd_clf = accuracy_score(y_test, y_pred_rd_clf)
          conf = confusion_matrix(y_test, y_pred_rd_clf)
          clf_report = classification_report(y_test, y_pred_rd_clf)

          print(f"Accuracy Score of Random Forest is : {acc_rd_clf}")
          print(f"Confusion Matrix : \n{conf}")
          print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of Random Forest is : 0.9526605709811816
Confusion Matrix :
[[22346   187]
 [ 1506 11724]]
Classification Report :
              precision    recall  f1-score   support

           0       0.94      0.99      0.96     22533
           1       0.98      0.89      0.93     13230

    accuracy                           0.95     35763
   macro avg       0.96      0.94      0.95     35763
weighted avg       0.95      0.95      0.95     35763
```

# KNN

```
In [533]: knn = KNeighborsClassifier()
          knn.fit(X_train, y_train)

          y_pred_knn = knn.predict(X_test)

          acc_knn = accuracy_score(y_test, y_pred_knn)
          conf = confusion_matrix(y_test, y_pred_knn)
          clf_report = classification_report(y_test, y_pred_knn)

          print(f"Accuracy Score of KNN is : {acc_knn}")
          print(f"Confusion Matrix : \n{conf}")
          print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of KNN is : 0.8922629533316556
Confusion Matrix :
[[21752   781]
 [ 3072 10158]]
Classification Report :
              precision    recall  f1-score   support

           0       0.88      0.97      0.92     22533
           1       0.93      0.77      0.84     13230

    accuracy                           0.89     35763
   macro avg       0.90      0.87      0.88     35763
weighted avg       0.90      0.89      0.89     35763
```

# ADA BOOST CLASSIFIER

```
In [534]: ada = AdaBoostClassifier(base_estimator = dtc)
          ada.fit(X_train, y_train)

          y_pred_ada = ada.predict(X_test)

          acc_ada = accuracy_score(y_test, y_pred_ada)
          conf = confusion_matrix(y_test, y_pred_ada)
          clf_report = classification_report(y_test, y_pred_ada)

          print(f"Accuracy Score of Ada Boost Classifier is : {acc_ada}")
          print(f"Confusion Matrix : \n{conf}")
          print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of Ada Boost Classifier is : 0.9393507256102676
Confusion Matrix :
[[21414  1119]
 [ 1050 12180]]
Classification Report :
              precision    recall  f1-score   support

           0       0.95      0.95      0.95     22533
           1       0.92      0.92      0.92     13230

    accuracy                           0.94     35763
   macro avg       0.93      0.94      0.94     35763
weighted avg       0.94      0.94      0.94     35763
```

# GRADIENT BOOSTING CLASSIFIER

```
In [535]: gb = GradientBoostingClassifier()
          gb.fit(X_train, y_train)

          y_pred_gb = gb.predict(X_test)

          acc_gb = accuracy_score(y_test, y_pred_gb)
          conf = confusion_matrix(y_test, y_pred_gb)
          clf_report = classification_report(y_test, y_pred_gb)

          print(f"Accuracy Score of Ada Boost Classifier is : {acc_gb}")
          print(f"Confusion Matrix : \n{conf}")
          print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of Ada Boost Classifier is : 0.9066633112434639
Confusion Matrix :
[[22340   193]
 [ 3145 10085]]
Classification Report :
              precision    recall  f1-score   support

           0       0.88      0.99      0.93     22533
           1       0.98      0.76      0.86     13230

    accuracy                           0.91     35763
   macro avg       0.93      0.88      0.89     35763
weighted avg       0.92      0.91      0.90     35763
```

# MODELS COMPARISON

## Models Comparison

In [539]:
```
models = pd.DataFrame({
    'Model' : ['Logistic Regression', 'KNN', 'Decision Tree Classifier', 'Random Forest Classifier','Ada Boost Classifier',
              'Gradient Boosting Classifier',],
    'Score' : [acc_lr, acc_knn, acc_dtc, acc_rd_clf, acc_ada, acc_gb]
})


models.sort_values(by = 'Score', ascending = False)
```

Out[539]:

| | Model | Score |
|---|---|---|
| 3 | Random Forest Classifier | 0.952661 |
| 2 | Decision Tree Classifier | 0.941616 |
| 4 | Ada Boost Classifier | 0.939351 |
| 5 | Gradient Boosting Classifier | 0.906663 |
| 1 | KNN | 0.892263 |
| 0 | Logistic Regression | 0.810810 |

# CONCLUSIONI

Come possiamo vedere dalla slide precedente, l'algoritmo Random Forest ha prodotto la migliore accuratezza.

Questo punteggio è buono e ideale in quanto riesce a predire con sicurezza la prenotazione di una camera in un hotel .