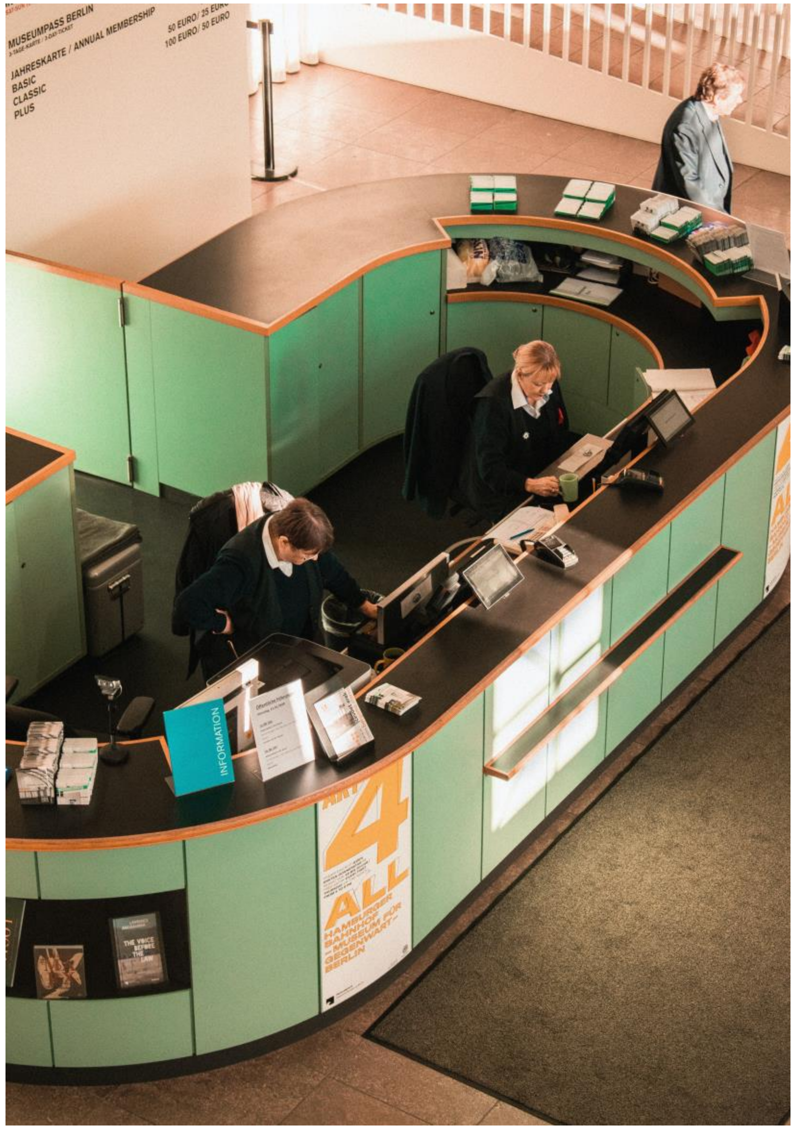
Hotel Reservation Cancelation

ISTANBUL DATA SCIENCE ACADEMY

Burak Arslan

Introduction



Motivation :

- As Hotel we will overbook once we know our safe margin.

Objective :

- Try to predict the ones who will cancel reservations ?

Goals :

- Try to find best algorithm and predict the which reservations will be canceled.

Methodology

Data :

- Hotel booking demand datasets : Kaggle

Tools :

  - Database : Postgresql
  - Data Cleaning : pandas, numpy
  - Visualizations : matplotlib, seaborn
  - ML : sklearn, K-fold
  - Development Environment : Jupyter

- Roadmap
  - Data Fetch & Clean (Postgresql -> dataframe)
  - Eda
  - Feature Engineering
  - Model Selection with K-Fold
  - Model tuning

EDA

Feature Data Types:

- 1 x Category
- 4 x float64
- 1 x int32
- 23 x int64
- 8 x object

Reservation Info:

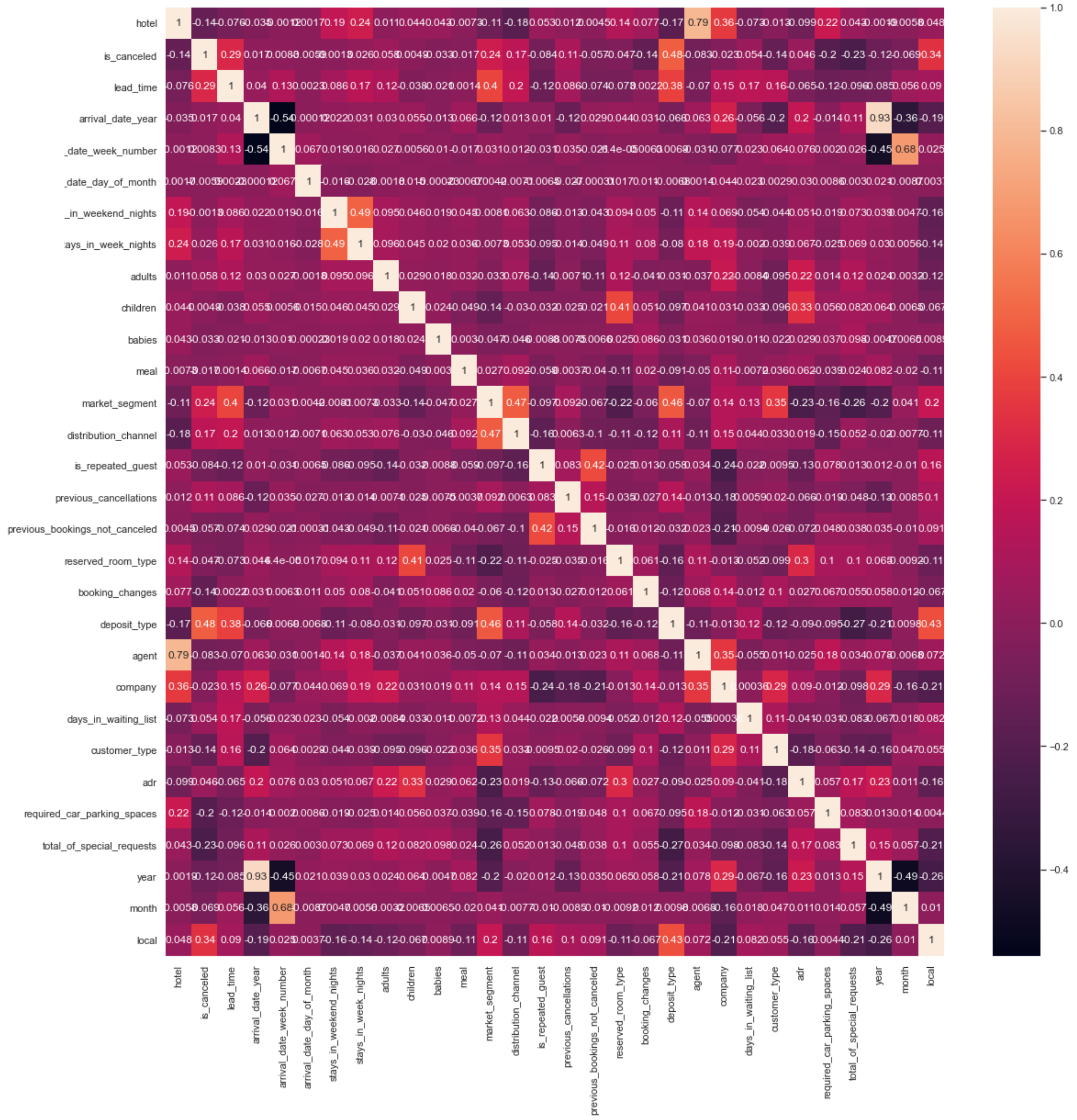
Lead time, Reservation date, Booking length,
Weekday stay length, weekend stay length,
Deposit, parking spot request, room type, booking channel,
Room type, daily rate, total number of special requests, agent,
Number of booking changes, number of days in waiting list

Customer Info

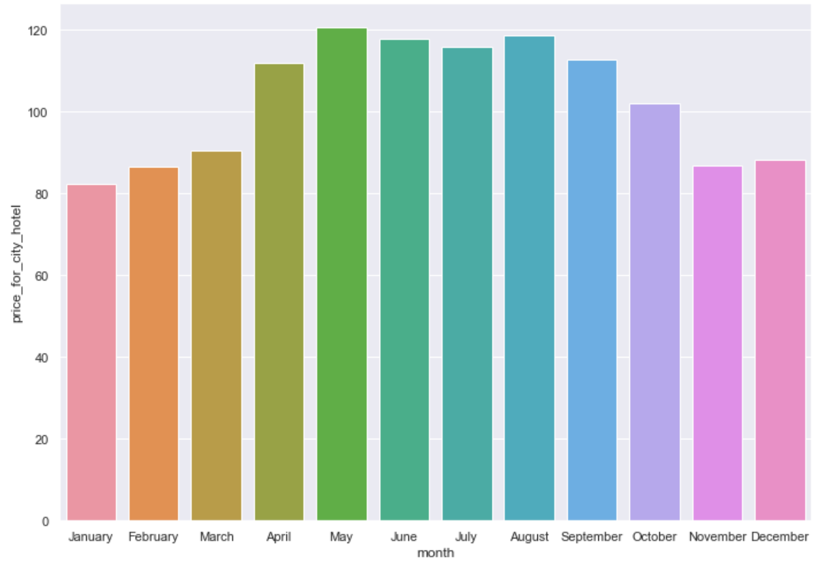
Previous cancelations, previous booking without cancel, repeated customer, number of adults, number of children, number of infants, business or individual,

Hotel Info

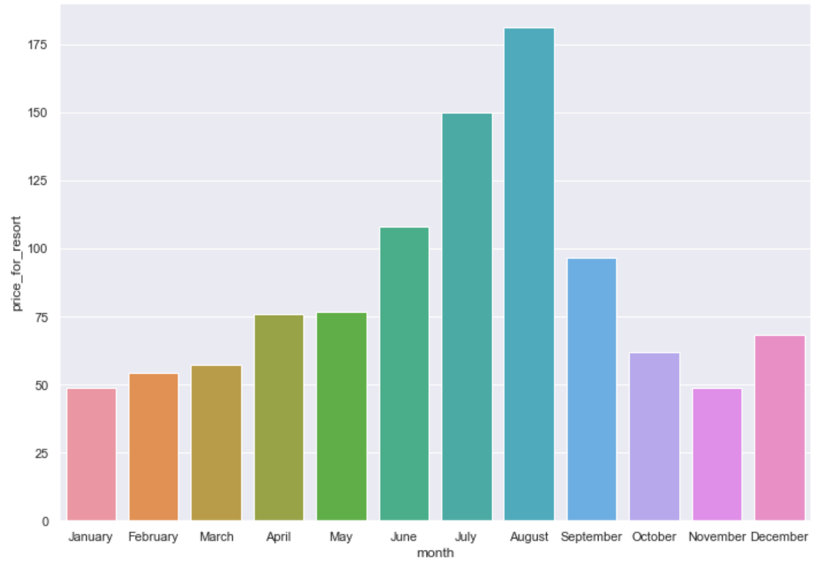
City or Resort, meal, parking space, average daily rate



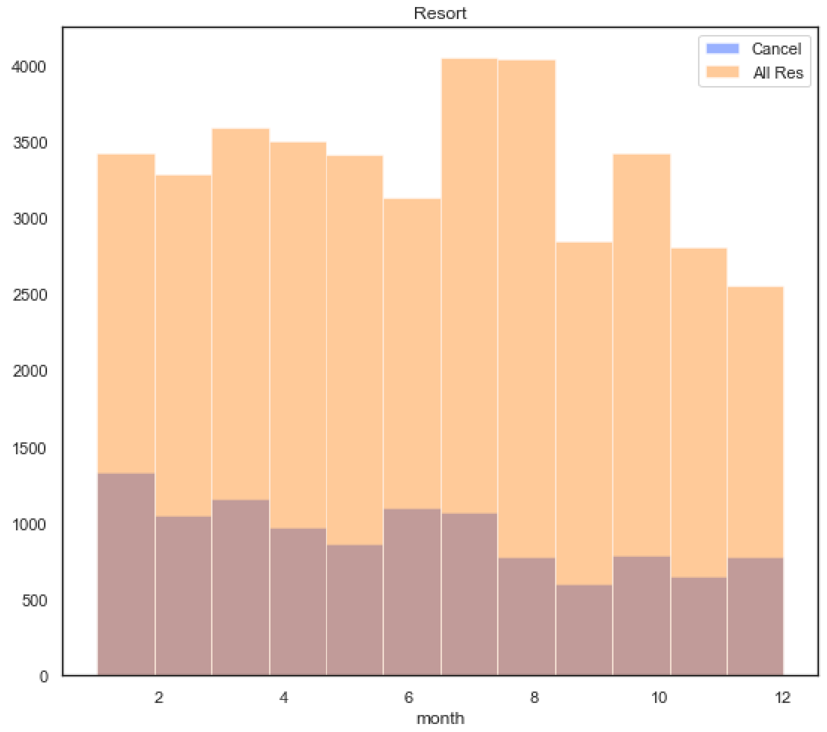
EDA



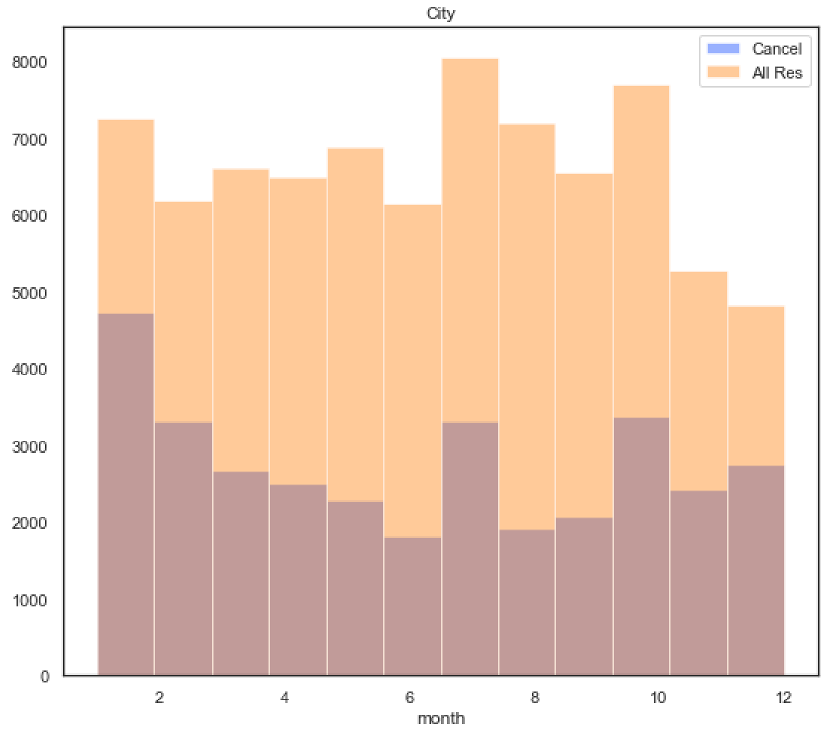
price_for_city_hotel
0
20
40
60
80
100
120
January
February
March
April
May
June
July
August
September
October
November
December
month



price_for_resort
0
25
50
75
100
125
150
175
January
February
March
April
May
June
July
August
September
October
November
December
month

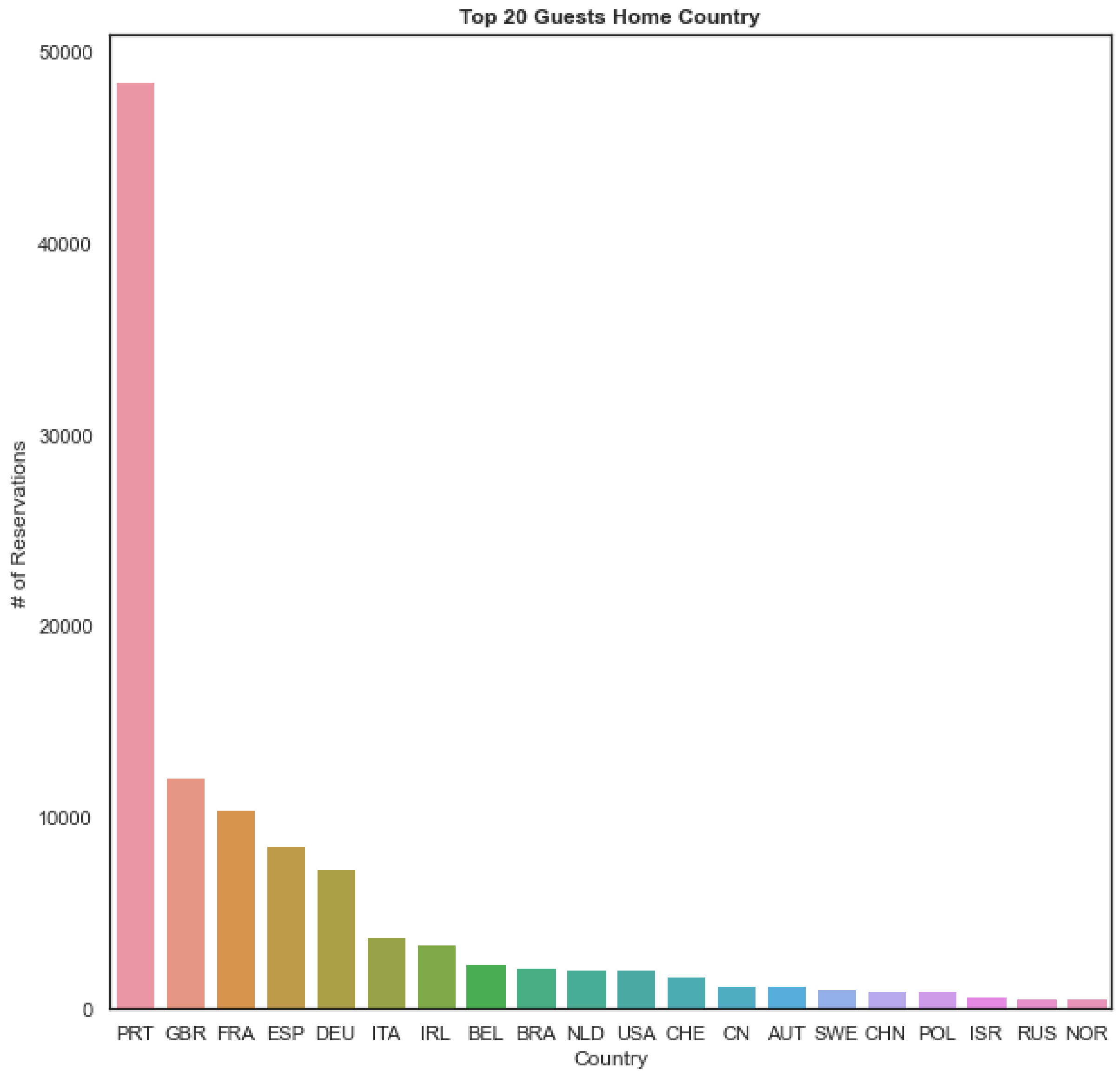


Resort
Cancel
All Res
0
500
1000
1500
2000
2500
3000
3500
4000
2
4
6
8
10
12
month

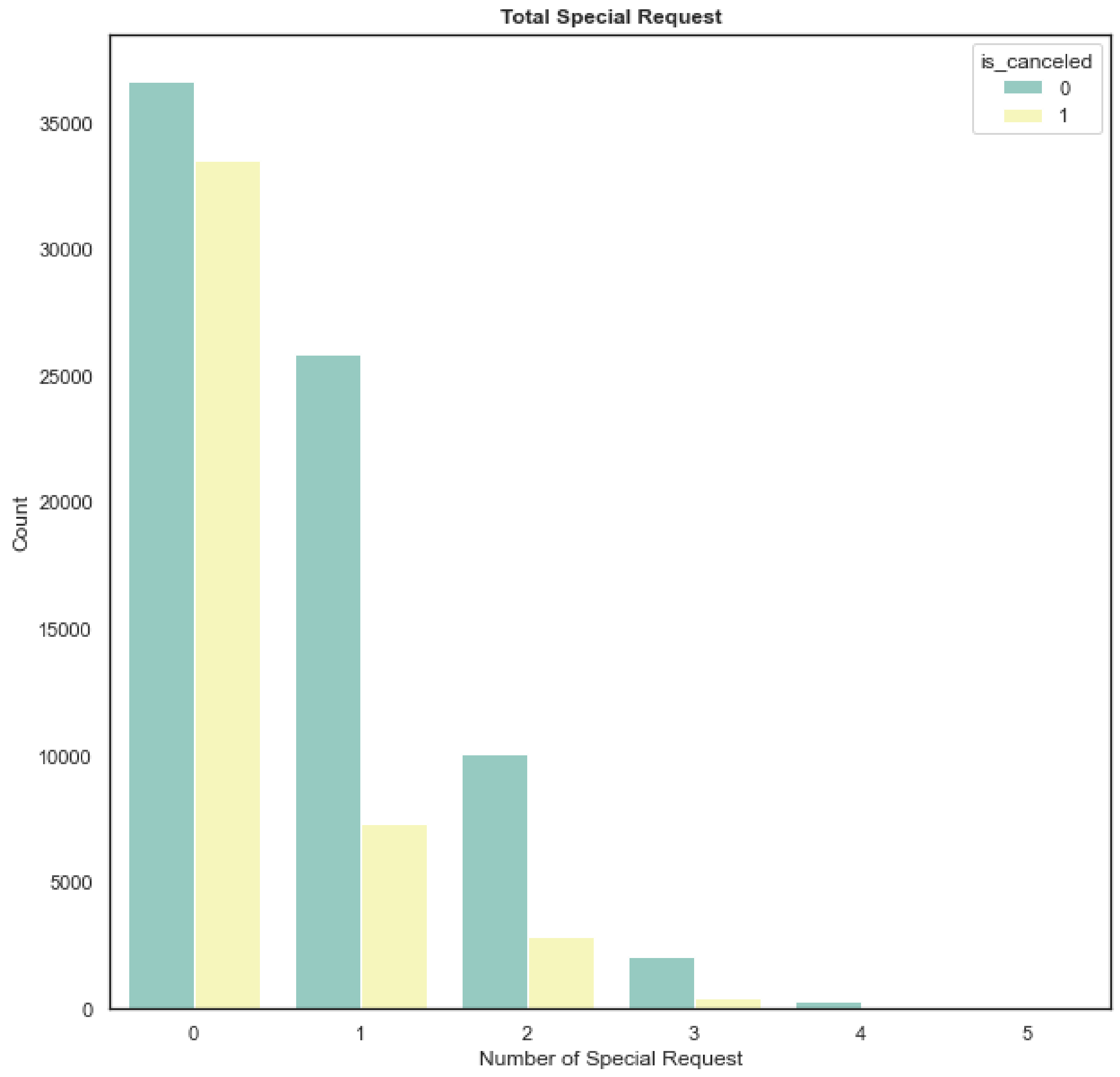


City
Cancel
All Res
0
1000
2000
3000
4000
5000
6000
7000
8000
2
4
6
8
10
12
month

EDA

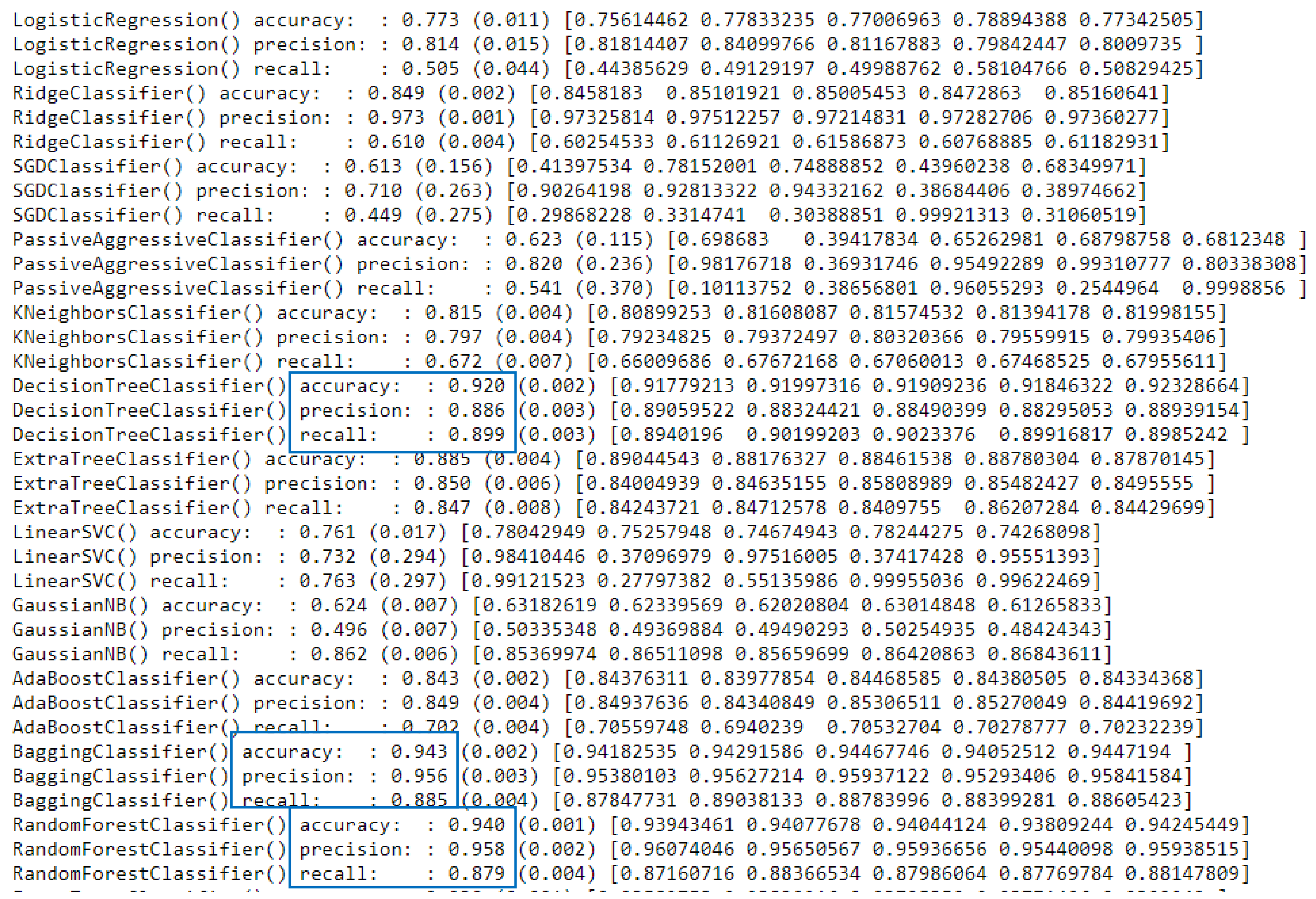


Top 20 Guests Home Country
# of Reservations
0
10000
20000
30000
40000
50000
PRT GBR FRA ESP DEU ITA IRL BEL BRA NLD USA CHE CN AUT SWE CHN POL ISR RUS NOR
Country

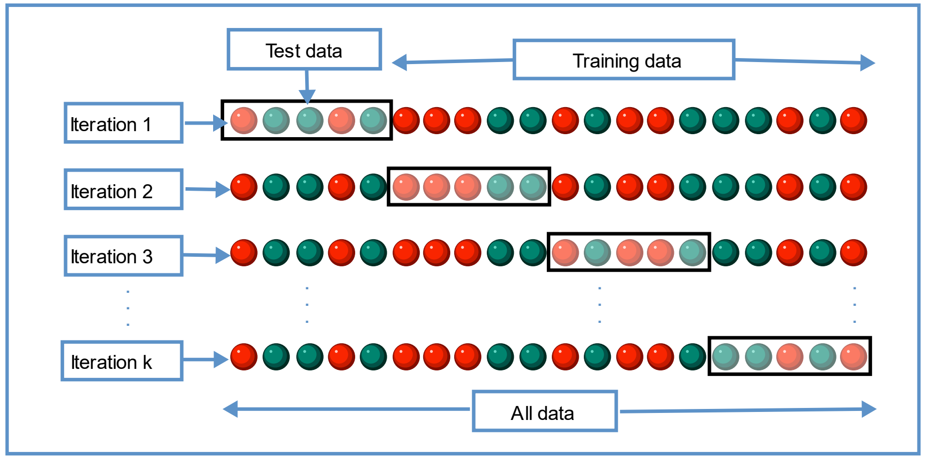


Total Special Request
is_canceled
0
1
Count
0
5000
10000
15000
20000
25000
30000
35000
0
1
2
3
4
5
Number of Special Request

Model Selection with Cross Validation



```
LogisticRegression() accuracy:  : 0.773 (0.011) [0.75614462 0.77833235 0.77006963 0.78894388 0.77342505]
LogisticRegression() precision: : 0.814 (0.015) [0.81814407 0.84099766 0.81167883 0.79842447 0.8009735 ]
LogisticRegression() recall:    : 0.505 (0.044) [0.44385629 0.49129197 0.49988762 0.58104766 0.50829425]
RidgeClassifier() accuracy:  : 0.849 (0.002) [0.8458183  0.85101921 0.85005453 0.8472863  0.85160641]
RidgeClassifier() precision: : 0.973 (0.001) [0.97325814 0.97512257 0.97214831 0.97282706 0.97360277]
RidgeClassifier() recall:    : 0.610 (0.004) [0.60254533 0.61126921 0.61586873 0.60768885 0.61182931]
SGDClassifier() accuracy:  : 0.613 (0.156) [0.41397534 0.78152001 0.74888852 0.43960238 0.68349971]
SGDClassifier() precision: : 0.710 (0.263) [0.90264198 0.92813322 0.94332162 0.38684406 0.38974662]
SGDClassifier() recall:    : 0.449 (0.275) [0.29868228 0.3314741  0.30388851 0.99921313 0.31060519]
PassiveAggressiveClassifier() accuracy:  : 0.623 (0.115) [0.698683   0.39417834 0.65262981 0.68798758 0.6812348 ]
PassiveAggressiveClassifier() precision: : 0.820 (0.236) [0.98176718 0.36931746 0.95492289 0.99310777 0.80338308]
PassiveAggressiveClassifier() recall:    : 0.541 (0.370) [0.10113752 0.38656801 0.96055293 0.2544964  0.9998856 ]
KNeighborsClassifier() accuracy:  : 0.815 (0.004) [0.80899253 0.81608087 0.81574532 0.81394178 0.81998155]
KNeighborsClassifier() precision: : 0.797 (0.004) [0.79234825 0.79372497 0.80320366 0.79559915 0.79935406]
KNeighborsClassifier() recall:    : 0.672 (0.007) [0.66009686 0.67672168 0.67060013 0.67468525 0.67955611]
DecisionTreeClassifier() accuracy:  : 0.920 (0.002) [0.91779213 0.91997316 0.91909236 0.91846322 0.92328664]
DecisionTreeClassifier() precision: : 0.886 (0.003) [0.89059522 0.88324421 0.88490399 0.88295053 0.88939154]
DecisionTreeClassifier() recall:    : 0.899 (0.003) [0.8940196  0.90199203 0.9023376  0.89916817 0.8985242 ]
ExtraTreeClassifier() accuracy:  : 0.885 (0.004) [0.89044543 0.88176327 0.88461538 0.88780304 0.87870145]
ExtraTreeClassifier() precision: : 0.850 (0.006) [0.84004939 0.84635155 0.85808989 0.85482427 0.8495555 ]
ExtraTreeClassifier() recall:    : 0.847 (0.008) [0.84243721 0.84712578 0.8409755  0.86207284 0.84429699]
LinearSVC() accuracy:  : 0.761 (0.017) [0.78042949 0.75257948 0.74674943 0.78244275 0.74268098]
LinearSVC() precision: : 0.732 (0.294) [0.98410446 0.37096979 0.97516005 0.37417428 0.95551393]
LinearSVC() recall:    : 0.763 (0.297) [0.99121523 0.27797382 0.55135986 0.99955036 0.99622469]
GaussianNB() accuracy:  : 0.624 (0.007) [0.63182619 0.62339569 0.62020804 0.63014848 0.61265833]
GaussianNB() precision: : 0.496 (0.007) [0.50335348 0.49369884 0.49490293 0.50254935 0.48424343]
GaussianNB() recall:    : 0.862 (0.006) [0.85369974 0.86511098 0.85659699 0.86420863 0.86843611]
AdaBoostClassifier() accuracy:  : 0.843 (0.002) [0.84376311 0.83977854 0.84468585 0.84380505 0.84334368]
AdaBoostClassifier() precision: : 0.849 (0.004) [0.84937636 0.84340849 0.85306511 0.85270049 0.84419692]
AdaBoostClassifier() recall:    : 0.702 (0.004) [0.70559748 0.6940239  0.70532704 0.70278777 0.70232239]
BaggingClassifier() accuracy:  : 0.943 (0.002) [0.94182535 0.94291586 0.94467746 0.94052512 0.9447194 ]
BaggingClassifier() precision: : 0.956 (0.003) [0.95380103 0.95627214 0.95937122 0.95293406 0.95841584]
BaggingClassifier() recall:    : 0.885 (0.004) [0.87847731 0.89038133 0.88783996 0.88399281 0.88605423]
RandomForestClassifier() accuracy:  : 0.940 (0.001) [0.93943461 0.94077678 0.94044124 0.93809244 0.94245449]
RandomForestClassifier() precision: : 0.958 (0.002) [0.96074046 0.95650567 0.95936656 0.95440098 0.95938515]
RandomForestClassifier() recall:    : 0.879 (0.004) [0.87160716 0.88366534 0.87986064 0.87769784 0.88147809]
```

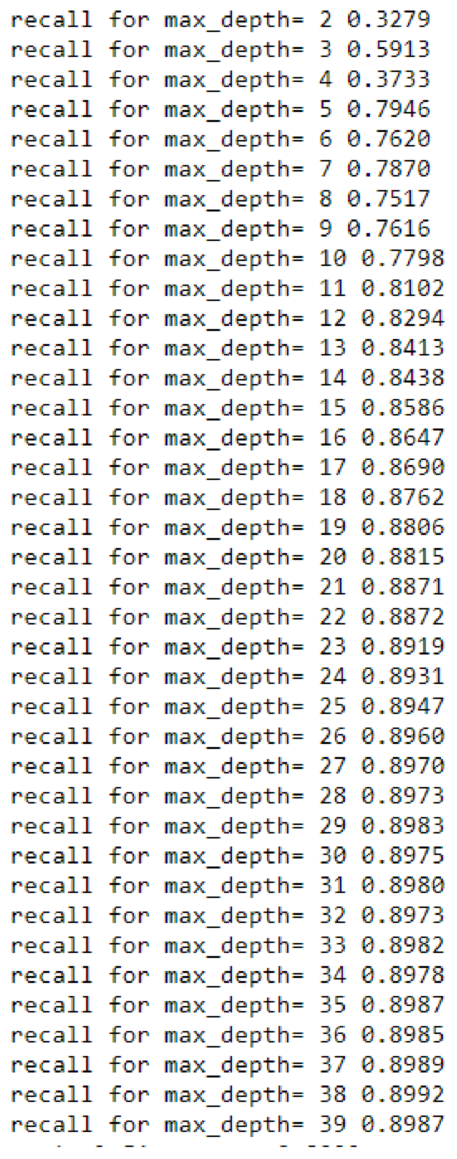


Optimize Decision Tree

```
All Data
DecisionTreeClassifier() accuracy: 0.920 (0.002) [0.91779213 0.91997316 0.91909236 0.91846322 0.92328664]
DecisionTreeClassifier() precision:0.886 (0.003) [0.89059522 0.88324421 0.88490399 0.88295053 0.88939154]
DecisionTreeClassifier() recall:   0.899 (0.003) [0.8940196 0.90199203 0.9023376 0.89916817 0.8985242 ]
```

```
Resort type Hotel
DecisionTreeClassifier() accuracy: 0.921 (0.005) [0.92846442 0.91822722 0.92421026 0.91909102 0.91497066]
DecisionTreeClassifier() precision:0.850 (0.008) [0.85932722 0.85733573 0.85065502 0.84492689 0.8380035 ]
DecisionTreeClassifier() recall: : 0.866 (0.012) [0.88124157 0.84738956 0.87567568 0.86857663 0.85873938]
```

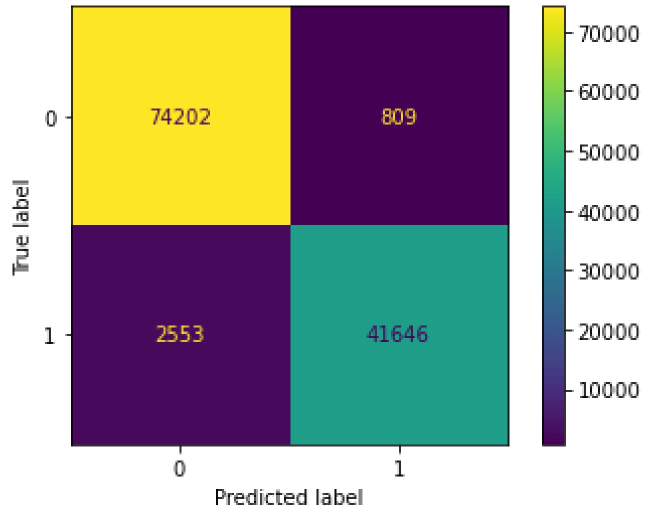
```
City type Hotel
DecisionTreeClassifier() accuracy: 0.918 (0.003) [0.91745089 0.92174572 0.91302975 0.91668772 0.91883527]
DecisionTreeClassifier() precision:0.895 (0.007) [0.8968254 0.90344515 0.88404508 0.89929777 0.89347696]
DecisionTreeClassifier() recall: : 0.908 (0.003) [0.91107458 0.9065534 0.9089525 0.90272782 0.91076876]
```

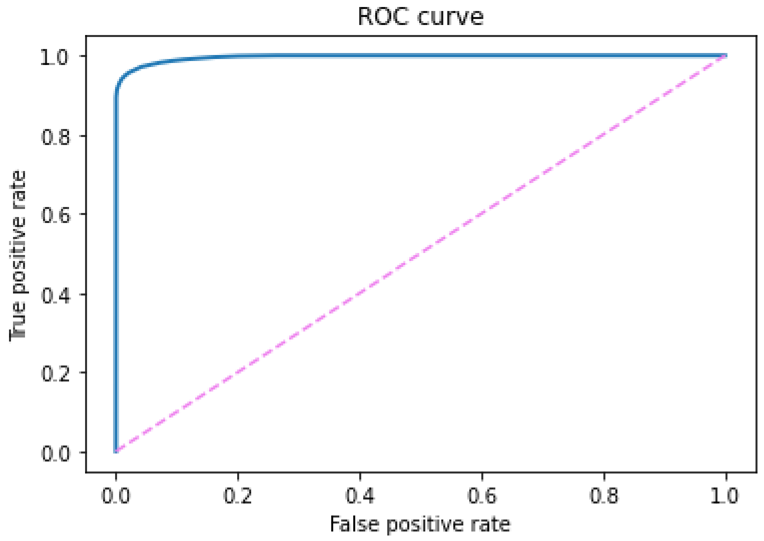


```
recall for max_depth= 2 0.3279
recall for max_depth= 3 0.5913
recall for max_depth= 4 0.3733
recall for max_depth= 5 0.7946
recall for max_depth= 6 0.7620
recall for max_depth= 7 0.7870
recall for max_depth= 8 0.7517
recall for max_depth= 9 0.7616
recall for max_depth= 10 0.7798
recall for max_depth= 11 0.8102
recall for max_depth= 12 0.8294
recall for max_depth= 13 0.8413
recall for max_depth= 14 0.8438
recall for max_depth= 15 0.8586
recall for max_depth= 16 0.8647
recall for max_depth= 17 0.8690
recall for max_depth= 18 0.8762
recall for max_depth= 19 0.8806
recall for max_depth= 20 0.8815
recall for max_depth= 21 0.8871
recall for max_depth= 22 0.8872
recall for max_depth= 23 0.8919
recall for max_depth= 24 0.8931
recall for max_depth= 25 0.8947
recall for max_depth= 26 0.8960
recall for max_depth= 27 0.8970
recall for max_depth= 28 0.8973
recall for max_depth= 29 0.8983
recall for max_depth= 30 0.8975
recall for max_depth= 31 0.8980
recall for max_depth= 32 0.8973
recall for max_depth= 33 0.8982
recall for max_depth= 34 0.8978
recall for max_depth= 35 0.8987
recall for max_depth= 36 0.8985
recall for max_depth= 37 0.8989
recall for max_depth= 38 0.8992
recall for max_depth= 39 0.8987
```

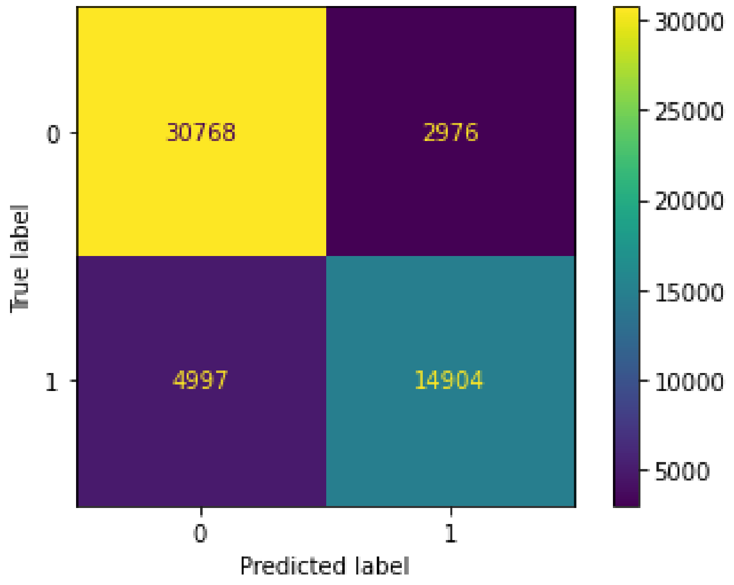
Decision Tree Results

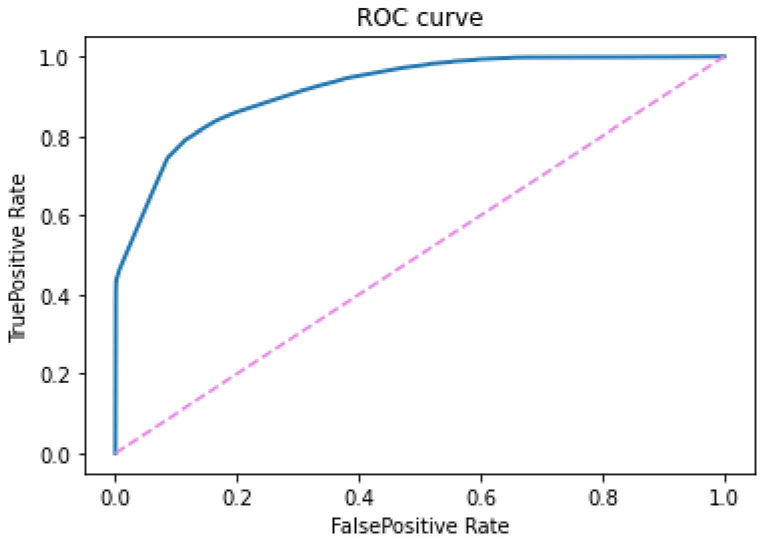
model.fit(X,y)





model.fit(X_test,y_test) Test Data Size : %0.45





Future Work

- A web application for hotel staff
- Different models for different sessions
- Different models for different personas