**Data Mining and Visualization**

**Final Project**

**Predictive Analysis of Hotel Reservation Cancellations: Optimizing Occupancy and Revenue Management**

**Yonatan Golan 208980888**

**Ziv Chaba 326681285**

**July 2024**

**Project Overview: Predicting Hotel Reservation Cancellations**

**Introduction:**

Hotel management frequently encounters issues with reservation cancellations, which can result in significant revenue loss and suboptimal resource utilization. Effective prediction of these cancellations is crucial for optimizing occupancy rates and enhancing profitability.

**Objective:**

This project involves analyzing a dataset of hotel reservations to develop a predictive model for identifying likely cancellations. By doing so, hotels can adopt more robust reservation strategies and improve their overall operational efficiency.

**Significance:**

* **Revenue Optimization:** Hotels will be able to minimize financial losses by accurately forecasting cancellations.
* **Resource Management:** Hotels will be able to allocate resources more efficiently based on predicted occupancy.
* **Strategic Planning:** Hotels will be able to implement proactive measures to counteract the impact of cancellations.

**Project Steps:**

1. **Data Analysis:** Examine and preprocess the dataset to understand key patterns and trends.
2. **Models Development:** Create machine learning models to predict reservation cancellations.
3. **Validation:** Test and validate the model to ensure its accuracy and reliability.
4. **Prediction:** Develop actionable insights for hotels to optimize their reservation management processes best on the best model.

By accomplishing these steps, this project aims to provide hotels with a valuable tool for mitigating the effects of reservation cancellations, thereby boosting both profitability and operational efficiency.

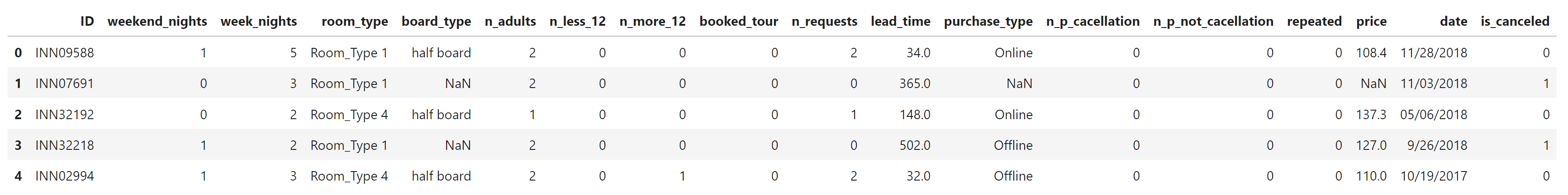
**Data Attributes**

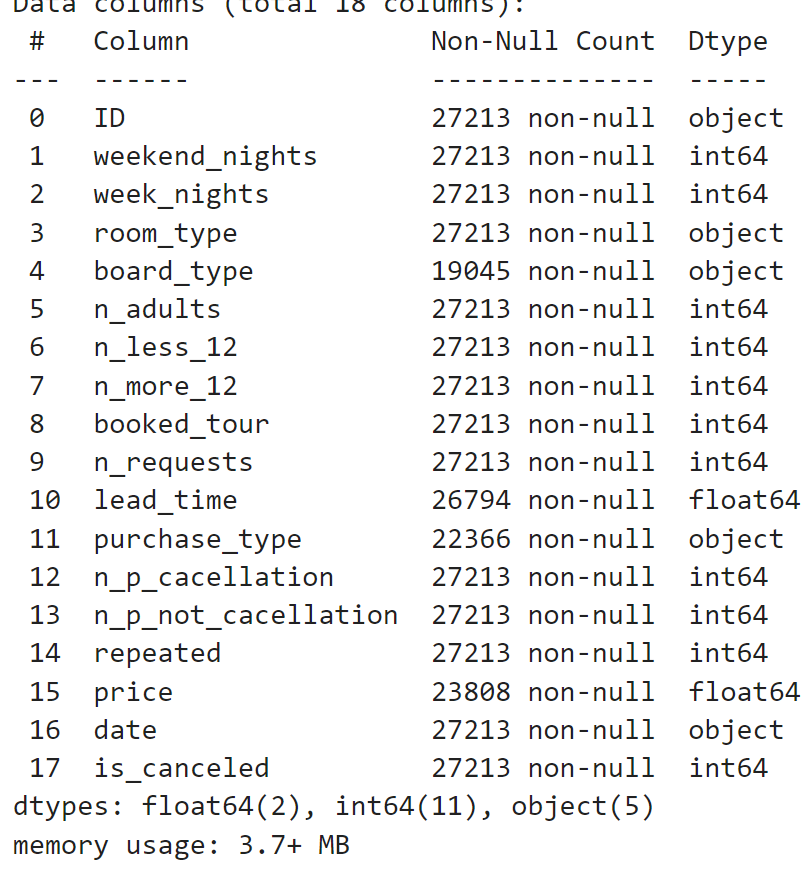
The data contains booking details, customer information, and reservation specifics as follows:

* ID: reservation id
* n\_adults: number of adults
* 'n\_less\_12': number of children aged less than 12
* 'n\_more\_12': number of children aged more than 12
* 'weekend\_nights': number of weekend nights
* 'week\_nights': number of week nights
* 'board\_type'
* 'booked\_tour': indicates whether a tour was included in the reservation
* 'room\_type'
* 'lead\_time': number of days between the reservation date and the arrival date
* 'purchase\_type'
* 'repeated': indicates whether the reservation is a repeat reservation
* 'n\_p\_cacellation': number of previous reservations that were canceled by the customer prior to the current reservation
* 'n\_p\_not\_cacellation': number of previous reservations not canceled by the customer prior to the current reservation
* 'price'
* 'n\_requests': number of special requests made by the guest
* 'date': date of the reservation
* 'is\_canceled': target value, 0 – not canceled, 1 – canceled

**Process description**

**1. Data information**

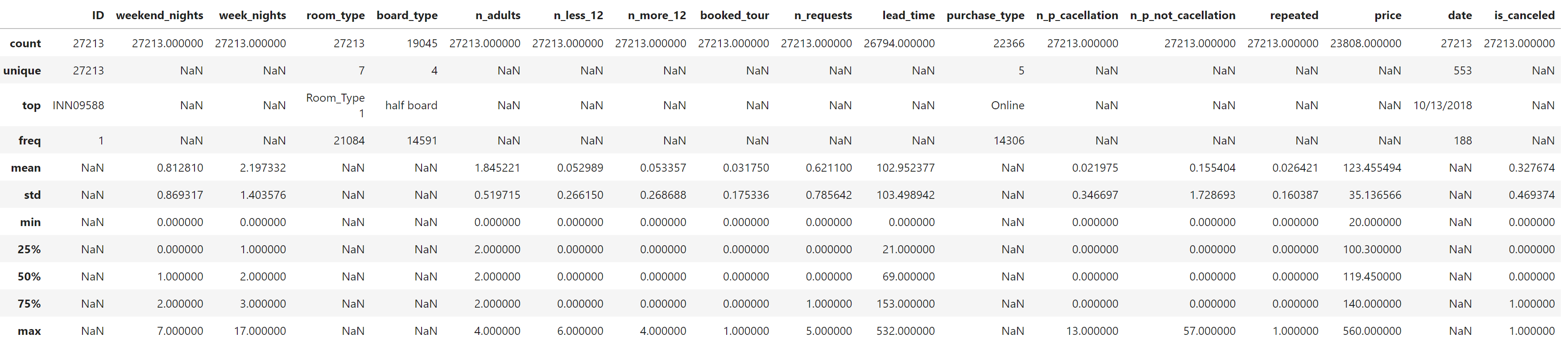
First we looked how our data looks like:

 From the information here we can learn about missing values; attributes type and unmatched data type.

* We have 27213 records with objective function ('is\_canceled'), each record has 18 features.
* Types of features:

5 features - a string, 2 features - a decimal number, 11 features - an integer.

* Missing values: for the attributes 'board\_type', 'lead\_time', 'purchase\_type', and 'price'.
* Unmatched data type: for the attribute 'room\_type' the data type is object and it should be a numeric for, we had to convert it to become numeric value.

We also looked at the described data:

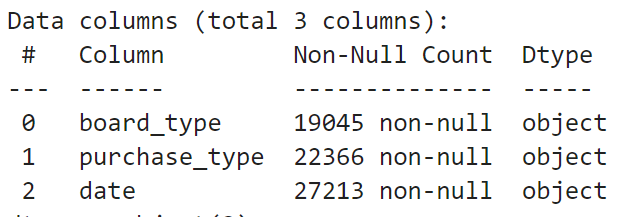
The next step we did in this section was to split to attributes and target and to split to nominal and numerical attributes.

**2. Data Statistics:**

First we looked at the Nominal data:

תמונה שמכילה טקסט, צילום מסך, תרשים, עלילה

התיאור נוצר באופן אוטומטי**תמונה שמכילה טקסט, צילום מסך, תרשים, עלילה

התיאור נוצר באופן אוטומטי**

**תמונה שמכילה טקסט, צילום מסך, גופן, מספר

התיאור נוצר באופן אוטומטי**

We can see the histograms of 'board\_type' and 'purchase\_type', the difference values and the count for each value. We can also see some statistics about the frequency, the top value and the total number of values.

In the second part we looked at the Numerical data:

For numerical data we would like to know the range of values, whether there are outliers, the distribution and other statistics, so first we will draw the histograms.

**Statistics information:**

We used the describe function to show statistic information for the numerical data:

תמונה שמכילה צילום מסך, עלילה, קו, תרשים

התיאור נוצר באופן אוטומטי**תמונה שמכילה טקסט, צילום מסך, מקביל, תרשים

התיאור נוצר באופן אוטומטיHistograms:**

From the histograms above it looks like 'week\_nights' and 'price' are normal distributed whereas the other attributes are not. Moreover, it seems the 'lead\_time' exponentially distributed and we can see that 'room\_type' has only few values with importance to the order (assumption) i.e. it's a categorical ordinal attribute. In this part we also computed the Skewness of the Numerical data and looked at the probability of each attribute.

תמונה שמכילה צילום מסך, תרשים, קו, עלילה

התיאור נוצר באופן אוטומטי

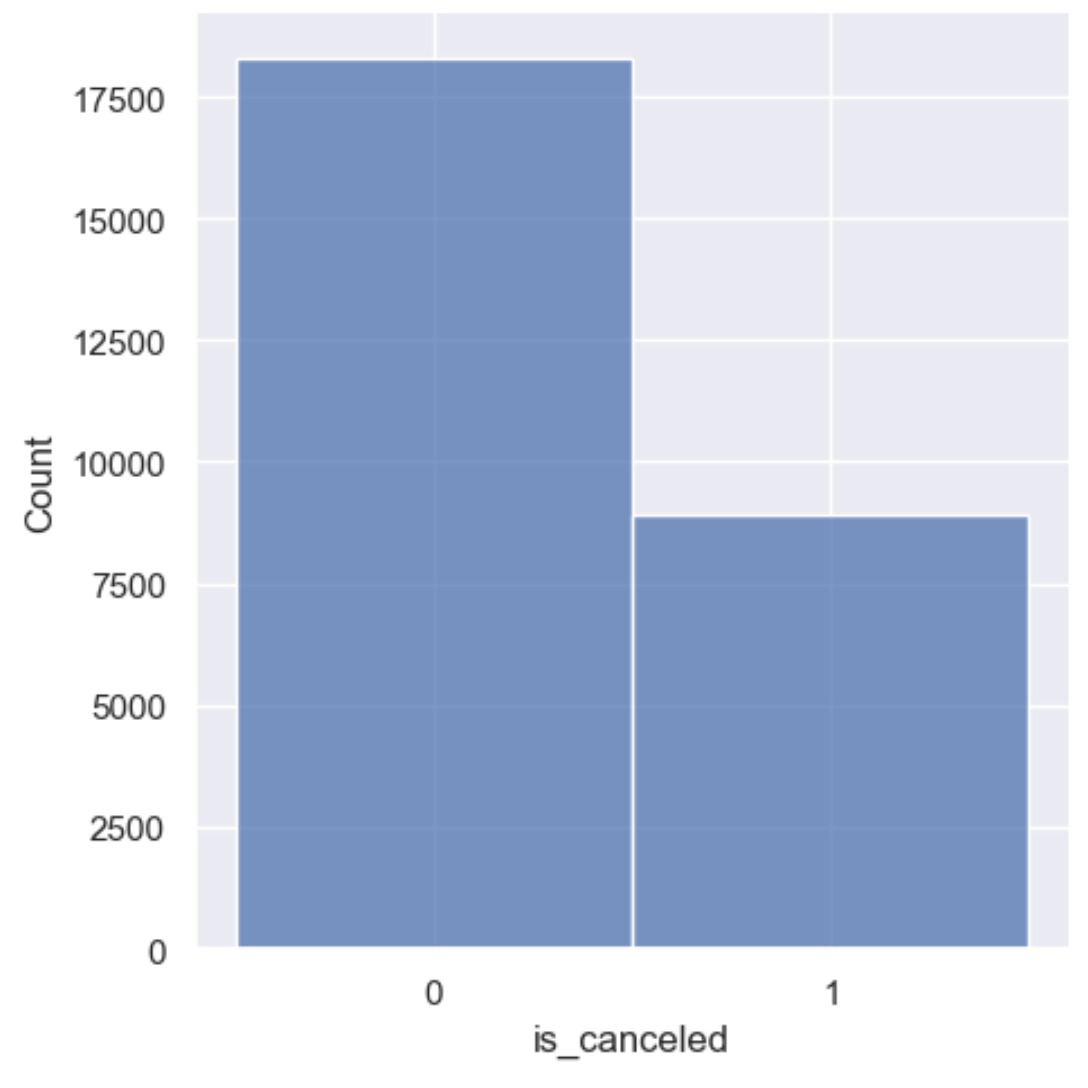
We did a boxplot for the price:

From this boxplot, we can conclude that most of the prices are concentrated between 100 and 140, with a few higher outliers. The median price is around 120. There is a notable spread in the data, with prices ranging from around 40 to 200 within the whiskers, and some extreme values extending up to 550.

More insights from the boxplot can be found in the notebook.

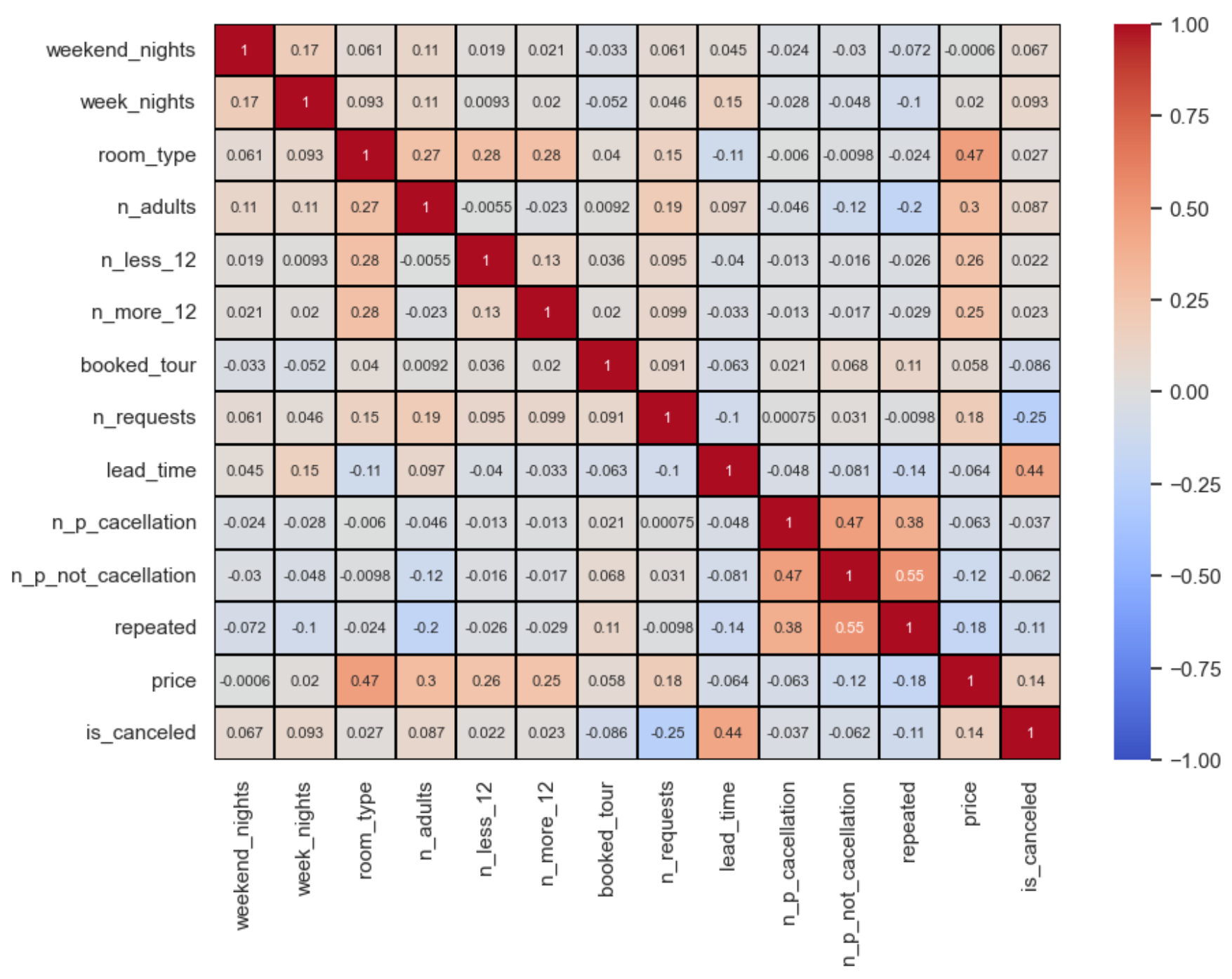
In the last part of this section, we've looked on our **Target** function distribution:

תמונה שמכילה עיגול, טקסט, צילום מסך, תרשים

התיאור נוצר באופן אוטומטי****

**3. Attributes Correlations**

We've calculated the correlations of the numeric data and visualized the correlation matrix:



We can see that the attribute with the strongest correlation to our target value 'is\_canceled' is 'lead\_time'.

Moreover, we can see strong correlation between 'room\_type' and 'price' which is obvious because some of the room types cost more than others. Other strong correlations we can see between 'n\_p\_cacellation' and 'n\_p\_not\_cacellation, and between 'n\_p\_not\_cacellation and 'repeated' (and smaller correlation between 'repeated' and 'n\_p\_cacellation').

Those three attributes describe:

• 'repeated': indicates whether the reservation is a repeat reservation

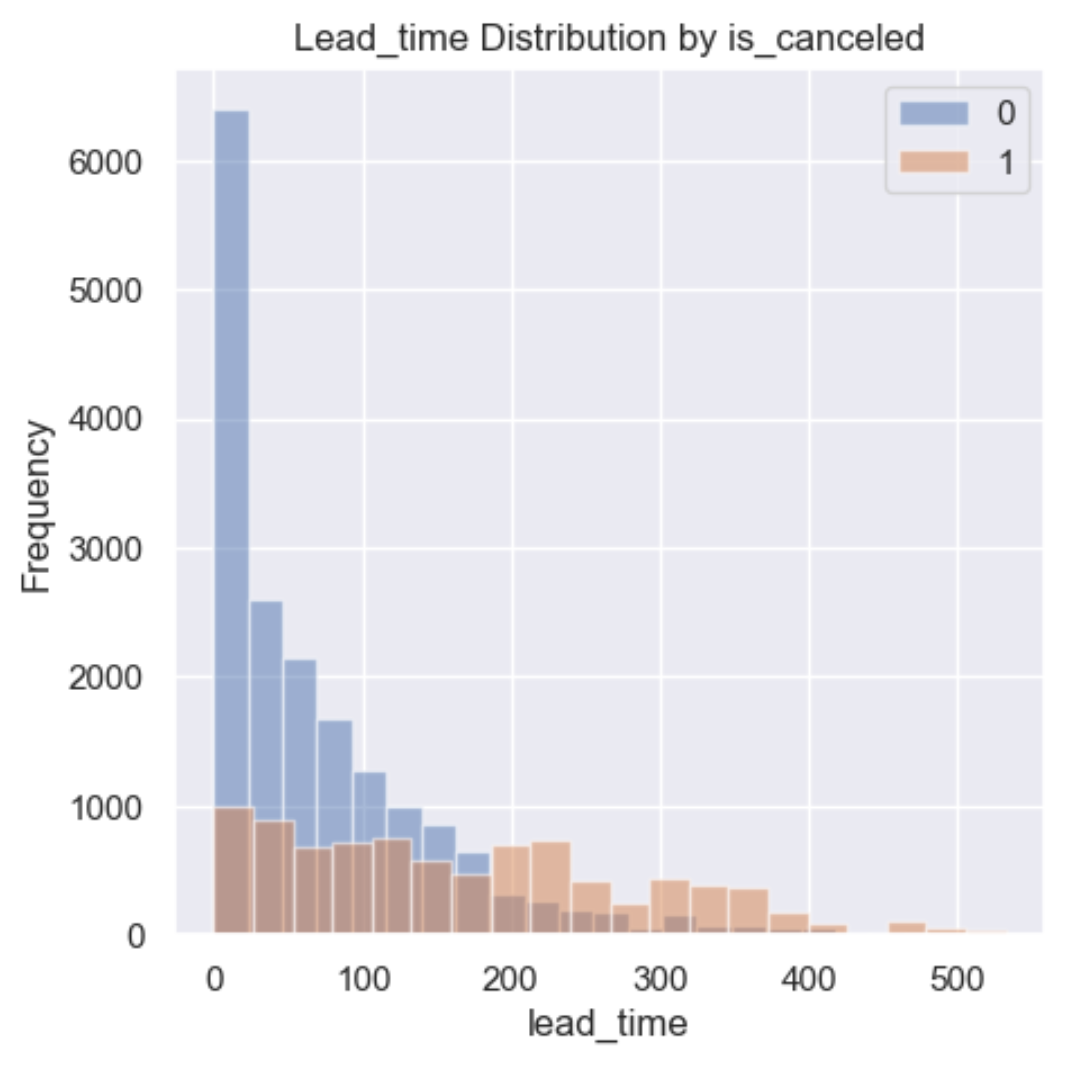
• 'n\_p\_cacellation': number of previous reservations that were canceled by the customer prior to the current reservation

• 'n\_p\_not\_cacellation': number of previous reservations not canceled by the customer prior to the current reservation

It's making sense those three have correlation because just if reservation is a repeat reservation there is a chance that the customer had prior reservation which might got canceled (and then increase the 'n\_p\_cacellation' or not and then increase the attribute 'n\_p\_not\_cacellation').

We can see other smaller correlation between 'n\_adults' and 'price' which also obvious because when you have more adults the price getting higher.

**4. Interesting insights from the data**

As been said in the previous note - the attribute with the strongest correlation (0.44) to our target value 'is\_canceled' is 'lead\_time'.

'lead\_time' represents the number of days between the reservation date and the arrival date. We can see from the above plot and the correlation that when the lead time is greater - there are more chances the reservation will get canceled.

תמונה שמכילה טקסט, צילום מסך, תרשים, מספר

התיאור נוצר באופן אוטומטי

It's also seeming that 'n\_requests' has a small negative correlation (-0.25) to the target value:

We can see here the negative correlation relation we've talked about - It's seeming like when the number of requests is higher there is lower probability the reservation will get canceled.

תמונה שמכילה טקסט, צילום מסך, תרשים, עלילה

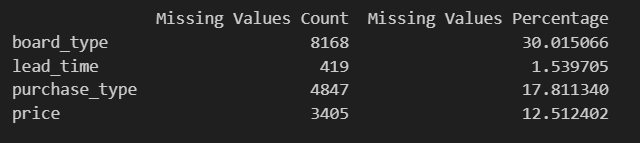
התיאור נוצר באופן אוטומטי

Another insight we can see is when the price is greater - there are more chances the reservation will get canceled.

**5. Data Cleaning**

We will now conduct an analysis of the provided dataset to identify potential issues, including missing values and inconsistencies.

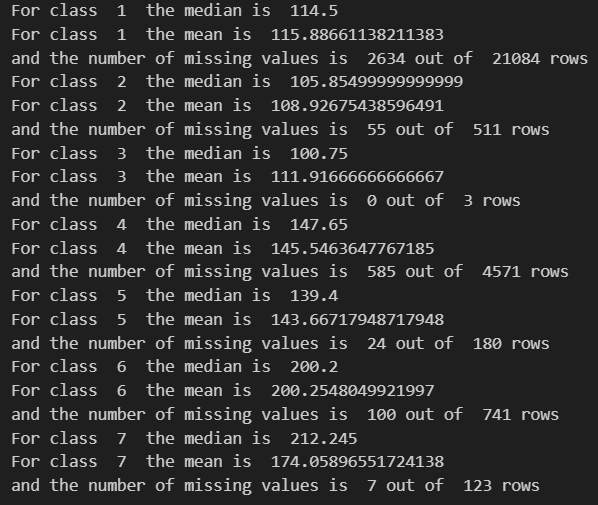
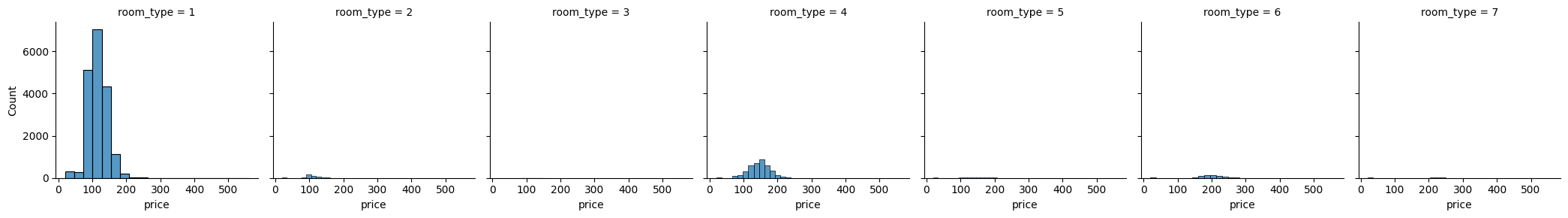
First, we detected the missing values, and see in which attributes are there missing values-



Upon examination of the training set, we have identified missing values in the attributes: board\_type, lead\_time, purchase\_type, and price. Notably, the attributes board\_type, purchase\_type, and price exhibit a substantial proportion of missing values, with board\_type being the most affected.

To address these deficiencies, we will implement the following imputation strategies:

1. Impute missing values for lead\_time using its median.
2. Impute missing values for price based on its quite strong correlation (0.47) with room\_type, as indicated by the correlation matrix.



1. Impute missing values for board\_type according to its probability distribution.
2. Impute missing values for purchase\_type using its mode.

Subsequently, we will convert categorical variables into numerical representations to facilitate further analysis.

After this process, we checked again, and saw that we have no missing values now, by completing them on reliable values.

**6. Add / Delete Attributes**

We noticed that the 'ID' attribute is just a generic string which give us no information, so we deleted this attribute.

To facilitate analysis of the 'date' attribute, we will decompose it into three new categorical attributes: 'day\_date', 'month\_date', and 'year\_date'. During this process, we have identified invalid dates, such as 2018-02-29, which is erroneous because February 29 only occurs in leap years, and 2018 is not a leap year.

These invalid dates will be treated as NaT (Not a Time) values. Given the infrequency of such anomalies, so there are not very important and the inherent difficulty in rectifying them accurately, we will exclude all rows containing NaT values from our dataset.

Then, we checked the data type of each attribute, and verify that it is indeed number, as we expect, and that we have no unnecessary duplicate columns, so that now we have no inconsistencies in the dataset.

**7. Data Transformation**

Following the completion of the previous steps, we proceeded with normalization. As we know, normalization is a crucial preprocessing step that standardizes the range of independent variables or features in the dataset. By applying the MinMaxScaler, we transform all features to a common scale, typically between 0 and 1. This ensures that each feature contributes equally to the model and prevents features with larger ranges from dominating those with smaller ranges. In previous part (the fifth part) we also encodes to numbers the categorical attributes.

The pre-processed training dataset is saved in the .csv file *hotels\_train\_pre\_processed.csv*.

**8. Implementation of the Pre-Process: Test Dataset**

Now, we implemented the pre-processing steps that we have done for the training dataset. Regard for the missing values, we saw that we have the same attributes of missing values like we had for the testing, and about the same percentage for each of the types, so we will treat them in the same way as we did for the training dataset. Additionally, we added and removed attributes as previously done and applied MinMaxScaler normalization to standardize the range of features, ensuring consistency with the training dataset.

The pre-processed test dataset is saved in the .csv file *hotels\_test\_pre\_processed.csv*.

**9. Splitting Data to Train and Validation Set and Using of Cross-Validation Method**

We've used the "train\_test\_split" function to split our training data for train and validation, we determine the seed to be 42 and the validation to be 0.25 of all the original training data.

We also defined here our Cross-Validation rate to 10 (since it's a large dataset).

**10. Evaluation Metrics**

**For each classifier we examined many evaluation metrics – accuracy, precision, recall,   
f1-score and AUC.**The most appropriate evaluation metrics depends on the context and the task - for example, in a case we want to classify patients with a particular disease we'll use the recall metric.

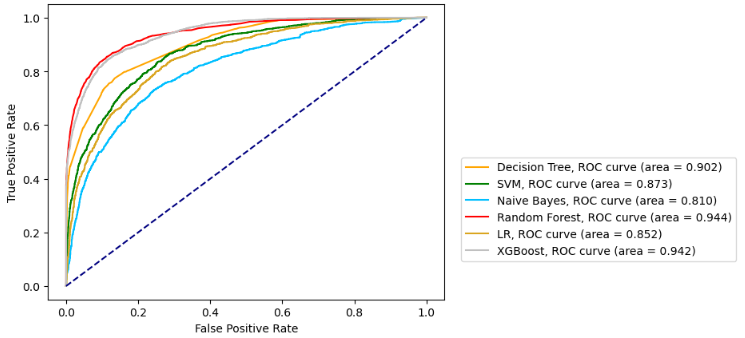
Therefore, choosing the most appropriate evaluation metric is dependent on the context of the task and on the business costs and consequences. For our problem we didn't knew if the cost of False positive or False negative is higher from a business point of view, and therefore we mainly focused on the accuracy and the AUC, and we have high recall values.

**11. Classifiers**

We trained 6 different classifiers and examined their performances.  
 we included two new classifiers that weren't covered in class – Logistic Regression and XGBoost.

The results we've got for each classifier before hyperparameters tuning**:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **AUC** | **f1-score** | **recall** | **precision** | **accuracy** | Classifier |
| 0.9021 | 0.88 | 0.90 | 0.87 | 0.843 | Decision Tree |
| 0.8726 | 0.86 | 0.91 | 0.82 | 0.807 | SVM |
| 0.8097 | 0.25 | 0.14 | 0.95 | 0.419 | Naive Bayes |
| 0.9439 | 0.92 | 0.94 | 0.89 | 0.885 | Random Forest |
| 0.8524 | 0.86 | 0.91 | 0.81 | 0.795 | Logistic Regression |
| 0.9417 | 0.91 | 0.93 | 0.89 | 0.876 | XGBoost |

ROC curve comparison:

**12. Hyperparameters Tuning**

We performed hyperparameters tuning for each of the models we've trained:

a. Decision tree – Decision Tree gave the great results, and his compute was fast so decided to do wide hyperparameter tuning on this classifier - We choose to use Grid search and gave wide range of hyperparameters because the model is fast. We choose to limit the 'max\_depth' in the range of [10,20] to avoid overfitting.

Best parameters: {'criterion': 'gini', 'max\_depth': 11, 'min\_samples\_leaf': 1, 'min\_samples\_split': 3, 'splitter': 'best'}

b. SVM - SVM gave a little bit lower result compared to the Decision tree and his running time was very long, so we did smaller hyperparameter tuning here. We focused on the parameter 'C' which plays a vital role in determining the trade-off between training error and margin width in the SVM algorithm. It controls the penalty for misclassified points during training, affecting the generalization performance and the potential for overfitting or underfitting. Furthermore, selecting an appropriate value for C is crucial for achieving a well-performing SVM model and therefore it was the only parameter we changed and after few attempts gave him range of [0.95 to 0.985].

Best parameters: {'C': 0.985, 'kernel': 'rbf', 'probability': True}

c. Random Forest- RF classifier gave us great result before the hyperparameter tuning so we tried to do fundamental, comprehensive and deep hyperparameter tuning. At the first step we did random hyperparameter tuning and at the second step we did grid search. Unfortunately, the hyperparameter here took a lot of time and at both options didn't improve much the accuracy (but AUC and some other metrics were improved).

Best parameters: {'bootstrap': False, 'ccp\_alpha': 0.0, 'class\_weight': None, 'criterion': 'gini', 'max\_depth': None, 'max\_features': 'auto', 'max\_leaf\_nodes': None, 'max\_samples': None, 'min\_impurity\_decrease': 0.0, 'min\_samples\_leaf': 1, 'min\_samples\_split': 5, 'min\_weight\_fraction\_leaf': 0.0, 'n\_estimators': 443, 'oob\_score': False, 'random\_state': 42, 'warm\_start': True}

d. Naive Bayes – this algorithm does not used parameters, so we didn't performed tuning for him (and

this algorithm gave poor results compared to the rest).

e. Logistic regression – this algorithm gave lower results compared to "Random Forest" and "XGBoost" so we did smaller hyperparameter tuning here.

Best Parameters: {'solver': 'liblinear', 'penalty': 'l1', 'max\_iter': 1000, 'C': 29.763514416313132}

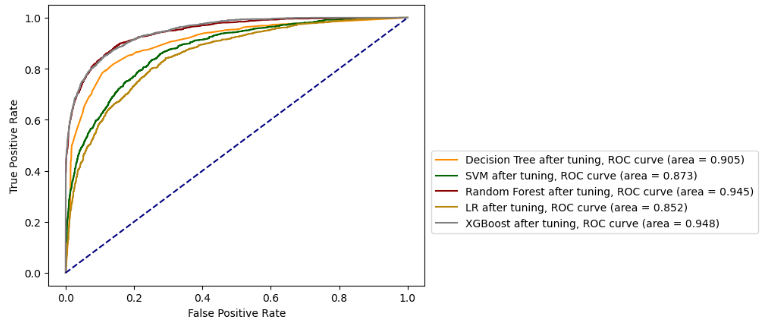
f. XGBoost – this classifier gave great result before the tuning so we thought he might be our best model and we did large Random grid search for the hyperparameters here.

Best Parameters: {'colsample\_bytree': 0.7, 'gamma': 0, 'learning\_rate': 0.05, 'max\_depth': 14, 'n\_estimators': 440, 'reg\_alpha': 0.1, 'reg\_lambda': 0.1, 'subsample': 0.9}

Results:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **AUC** | **f1-score** | **recall** | **precision** | **accuracy** | Classifier |
| 0.9048 | 0.89 | 0.89 | 0.89 | 0.857 | Decision Tree |
| 0.8731 | 0.86 | 0.91 | 0.82 | 0.807 | SVM |
| 0.9450 | 0.92 | 0.94 | 0.90 | 0.884 | Random Forest |
| 0.8522 | 0.86 | 0.90 | 0.82 | 0.797 | Logistic Regression |
| 0.9475 | 0.92 | 0.93 | 0.90 | 0.885 | XGBoost |

ROC curve comparison after hyperparameter tuning:



**13. Best model**

After the hyperparameter tuning our two best models were the Random Forest and the XGBoost, therefore, we conduct comparison between those two classifiers by Statistical Significance Tests - Two-sided test for the null hypothesis that two related or repeated samples have identical average (expected) values. We found that There is a significant difference between the two classifiers (Reject null hypothesis).

(This test conducted on the whole training dataset without the division into a validation set, with   
KFold = 10, as we saw in class, so the results are slightly different from the ones we got above) The average results we've got:

Classifier - **Random Forest:** **Mean Accuracy: 0.9678, Mean Precision: 0.9623, Mean AUC: 0.9869**

Classifier – XGBoost: Mean Accuracy: 0.8836, Mean Precision: 0.8497, Mean AUC: 0.9449

Therefore, our best model is the "Random Forest" and we choose him to classify the predications to the test. The test dataset with the prediction in its last column that was added – is\_canceled, *hotels\_test\_prediction.csv*, is attached.