DePaul University
Jarvis College of Computing Digital Media

DSC 478 Final Project Report

Project Title:

Potential Revenue Loss and Gain Prediction for Hotel Reservations (A Data Analysis Project)

Team Members:

Bramhashree Manoharan, Hoda Masteri, Goutham Selvakumar

Fall 2022

Introduction:

One of the major reasons for revenue loss in the hotel industry is booking cancellations. The goal of this project was to analyze the data and predict whether future bookings would get canceled or not. However, not all cancellations lead to revenue loss, as some of the reservations are non-refundable. We further engineered the data to provide even more insights on whether these cancellations would result in revenue loss or gain.

We utilized various classification algorithms (single and ensemble models) and by means of grid search and other algorithms we found the best parameters for each model. Finally, we analyzed the accuracies of the various classifiers to select the best one that can predict the profit or loss for each reservation with the highest accuracy. The model also provides insights into which variables play a major role in determining whether the booking results in a profit or loss.

Dataset:

We obtained the hotel bookings data from the Kaggle page linked below. This dataset has 119390 rows and 32 columns. The columns include various aspects of hotel reservations.
https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand

Exploratory Data Analysis:

We performed EDA to understand the dataset's main characteristics; which included looking at the five-number summary and the correlation matrix, and some univariate and multivariate plots to understand the variables and their relations better. We performed a short EDA again after generating a target, Revenue, to observe the relationships of independent variables with the target variable as well. Among the information we collected was that a higher lead-time means a higher probability that a reservation gets canceled and thus higher probability of revenue loss for a hotel; or guests with higher number of special requests were less likely to cancel a reservation; or deposit type can predict whether the hotel would make profit from a reservation or not; or the fact that reservation with market-segment type of online were more likely to cause a revenue loss for the hotels.

Pre-processing:

1.Data cleaning & discretization:

1.1 Checking the columns for Null Values:

The four columns namely ‘children’, ‘country’, ‘agent’, and ‘company’ had null values.

Data Quality Assessment and Justification for checking the soundness of values and finding mismatched data types:

1. There are only 4 out of 119390 instances that contain Null in the **Children** column. Thus, dropping the 4 rows seems to be the best.
2. **Country** has 488 null values. Replacing them with "unknown" appears to be the best option. In addition, the country code PRT seems to be very distinctive from the rest, because it has a higher cancellation rate compared to others, hence the rest can go into a single bin, the "other".
3. **Agent** column has 16340 null values. Replacing them with -1 appears to be the best option Agent 1 seems to be very distinctive from the rest, because it has a higher cancellation rate compared to others. Thus, the rest can go into a single bin, the "other".
4. **Company** columns have nearly 94% missing values in them. Since no company stands out among others, and since 112593 out of 119390 rows have a null company value, the best action is to drop the company column entirely.
5. We look at the data types (float64 - 4, int64 – 16, and object - 12) and at the first glance, there are 12 categorical variables. However, the variables 'is_canceled', 'is_repeated_guest', and 'agent' are categorical variable by default but are represented here as numbers. Furthermore, the last column, 'reservation_status_date' is an ordinal attribute, but in most of the cases it is identical to the 'arrival date'.

Feature Engineering:

We further engineered the data to create a target column that reflects the profit or loss for each reservation rather than the reservation status. To do so, we computed the "difference" between the reservation status dates (dates of the latest status updates) and the arrival dates. Then we used a query to select only a portion of the rows based on the following rules:

- If a reservation is canceled and either the deposit is non-refundable or the reservation is canceled 200 (an arbitrary number we selected) days prior to arrival, then the hotel gains profit by forfeiting the deposit.
- or if the guest has already checked out, then it is recorded as profit.
- or if a reservation is a no-show and the deposit is non-refundable, then it is recorded as profit.

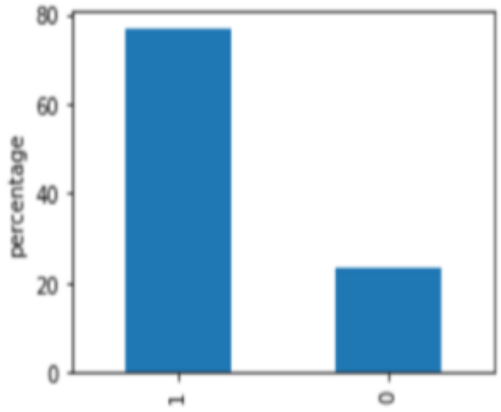
After selecting specific rows that matched the query and saving them in a profit data frame, we used the unselected rows to form a loss data frame. We then created a new column "Revenue" for each of the profit and loss data frames and populated those columns with 1's and 0's, respectively. Then we (outer) joined the two profit and loss data frames to form the final one, hotel_2. We then removed some of the temporarily made columns to avoid data leakage.

2.Data Preparation & Transformation:

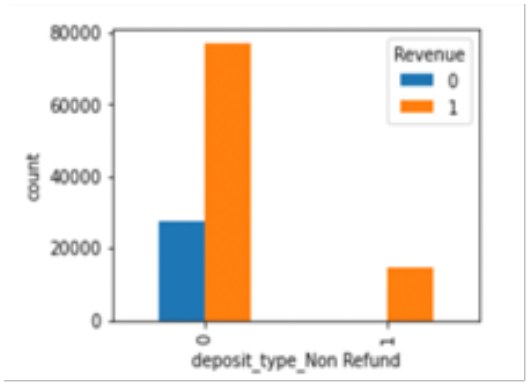
- We first used the 80-20 rule to split the data into train and test sets, then trained a minmax scalar on train set only to normalize it; then we used the same minmax scalar to transform the set-aside test data.
- Since KNN and SVM are distance-based classifiers, the use of min-max normalization is a necessity for these classifiers.
- We have used the normalized data for other classifiers too, such as Naïve Bayes, LDA, & Random Forest, though they do not require the data to be normalized.

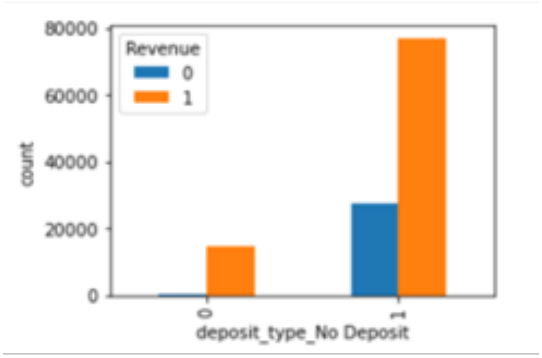
- As we wanted to look at the Decision tree graph and obtain meaningful insights from it, we have used the unnormalized data to train decision trees.

Visualizations:



Bar graph of class counts of profit and loss





Cross-tabulations for target variable '**Revenue'** with that of the 'deposit_type_Non Refund', 'deposit_type_No Deposit', & 'markey_segment_Online TA'.

Classification:

KNN:

KNN is a supervised learning classifier and it relies on data similarities and distance metrics in order to generate accurate predictions. Furthermore, we used KNeighborsClassifier from sci-kit-learn library to implement the k-nearest neighbors' vote. Since the knn follows only one parameter, this makes the parameter tuning to be easier. We performed a grid search for

parameters of values [k=1, 3, 5, 7, & 9] and cross-validation of 5. The model accuracy that we got was increased slightly after the hyperparameter tuning by 4%. The optimal value of n_neighbors was 7, with which the **overall (cross-validation), the train, and the test accuracies as 0.80(+/- 0.01), 0.78%, and 0.98% respectively.**

Decision Tree:

We used tree.DecisionTreeClassifier from scikit-learn library to create a classifier. We implemented the grid search for the decision trees on the unnormalized data. The optimal parameters generated were: 'criterion': 'entropy', 'max_depth': 11, 'min_samples_leaf': 17, 'min_samples_split': 4}, with which we were able to get the **overall (cross-validation),** the **train,** and the **test accuracies** as **0.82 (+/- 0.02)**, **84.1%, and 82.7%**, respectively. There was not much difference in the accuracies after the hyperparameter tuning. Decision Trees were appropriate for our dataset as it allowed us to create easily interpretable outcomes and pick the best possible solution. It helped us to explore and understand the data relationship in a tree hierarchy structure. To visually represent the decision tree, we used the graph viz.

Random Forest:

We used ensemble.RandomForestClassifier from the scikit-learn library to create our classifier. We implemented the grid search for the Random Forest and with the generated optimal parameters: 'criterion': 'entropy', 'max_depth': 20, 'n_estimators': 600, We were able to get the **overall (cross-validation)**, the **train**, and the **test accuracies** as **0.85 (+/- 0.02), 95.8%, and 99.54%,** respectively. The test accuracy significantly increased after the hyperparameter tuning. Random Forest is an ensemble model which can detect non-linearity and it resulted in better accuracy as we had a huge dataset.

SVM:

We used SVC from sci-kit-learn library to create an SVM classifier. We implemented grid search to optimize C, gamma, and kernel parameters. The SVM classifier was initialized with the balanced class weights. Setting up the parameters for our model and initializing the grid search with the cross-fold validation of 3. We are isolating the target variable as the 'Revenue'. Fitting the 3 folds for each of the 50 candidates, totaling 150 fits. Next, is to predict the classes of test data and print the classification report for SVM accordingly. Moreover, the SVM did not perform well with large datasets as the complexity of SVM is highly dependent on the size of the dataset. SVM is most effective in high dimensional spaces. The performance of the hyperparameter tuning did not show any significant performance improvement as from the classification report we can get the result of an **overall accuracy being 0.79 (+/- 0.01), the training and testing data accuracies being 0.95%, and 0.98%** respectively.

Naive Bayes:

We used naive_bayes.GaussianNB from scikit-learn library to create a Gaussian Naïve Bayes classifier. We found the accuracy of this classifier on the train and test sets and performed 10-fold cross-validation to find an average accuracy to report for our classifier. We tested if parameter tuning would increase the accuracy of this classifier. In the hyperparameter tuning step, we tried different values for the var_smoothing parameter, from 1e-9 to 1e-2, and used 10-

fold cross-validation. After finding the best var_smoothing from grid-search, we created the best Naïve Bayes model and found the train, cross-validation, and test accuracies, which were only slightly better than the classifier we created initially using the default parameter values. The reason can be that GaussianNB assumes the distributions of independent variables are normal; however, this assumption does not hold for the categorical, mostly binary dummy variable, which can explain the low prediction accuracy. The **overall (cross-validation), train, and test accuracies for the best Naïve Bayes model were 0.53 (+/- 0.03), 52.97%, and 54.56%**, respectively.

Linear Discriminant Analysis (LDA):

We used Linear Discriminant Analysis from scikit-learn library to create a linear discriminant-based classifier and trained it using the train set and compared the train, test, and cross-validation accuracies. We tried to increase the accuracy by doing a grid search and finding the best model parameters. In the hyperparameter tuning step, we first tried different solvers like svd, and lsqr. We found that svd (the default solver) is the best one to use. Then we performed another grid search to see which of the two solvers lsqr, and eigen performs better when the shrinkage parameter is set to auto. We did not include svd in the second grid search because svd only works with shrinkage=None. Eventually, we compared the accuracies from the two grid searches and found out the best setting is the default setting (the svd solver), so we did not manage to increase the accuracy further. We believe the reason may be that Linear Discriminant Analysis (LDA) is more suited to continuous numerical variables rather than dummy variables with few unique values. The **overall (cross-validation), the train, and the test accuracies for the LDA model were 0.78 (+/- 0.02), 78.66%, and 77.90%**, respectively.

Clustering:

We performed clustering for unsupervised knowledge discovery and detecting patterns in the dataset. We wrote a function that used cluster. KMeans from scikit-learn library to accomplish this task. We repeated the clustering process for various numbers of clusters, k, from 2 to 20 and plotted the mean silhouette, completeness, and homogeneity scores against k to find an appropriate value for k. Among different values of k, k=2, k=5, and k=7 were the ones with higher scores and silhouette mean. We picked k=2 because it enables us to compare only two centroids and to see how similar they are to the two classes (Revenue=1 and Revenue=0). The patterns we detected using centroid observations are as follows:

Centroid 1: has characteristics of revenue loss (Revenue=0) class:
higher lead-time, lower stays_in_weekend_nights, lower number of children, higher previous_cancellations, higher days_in_waiting_list, lower number of total_of_special_requests, higher agent_1, etc.
Centroid 0: has characteristics of a profit (Revenue=1) class:
lower lead-time, higher stays_in_weekend_nights, higher number of children, lower previous_cancellations, lower days_in_waiting_list, higher number of total_of_special_requests, low agent_1, etc.

We cannot however conclude that the above findings hold for most of the data because the homogeneity and completeness scores are not high.

Conclusion:

Below are the Overall Accuracies for each classifier:

- Random Forest: 0.85 (+/- 0.02)
- Decision Trees: 0.82 (+/- 0.02)
- KNN: 0.80(+/- 0.01)
- SVM: 0.79 (+/- 0.01)
- LDA: 0.78 (+/- 0.02)
- Naïve Bayes: 0.53 (+/- 0.03)

Based on our analysis of each classifier, we conclude that **Random Forest**, being an ensemble model, outperformed the simple classifiers with an overall accuracy of **85%**. Furthermore, the random forest is useful in handling larger datasets and can provide a higher level of accuracy.

In addition, the results, and observations from the unsupervised knowledge discovery (clustering) helped us in pattern recognition in the sense that the findings agreed with the information we had previously obtained from exploratory data analysis and visualizations.

Appendix:

ipynb file:

Profit- Loss final version 11-18.ipynb

html version of the notebook:

Profit- Loss final version 11-18-5.html

Data Samples:

hotel_bookings.csv

Tree output:

tree.dot

Tree Image: