Pricing policy for hotels

Shalom Rosh (ID. 308154418), Alexander Shachor (ID. 204698724), Avi Miletzky (ID. 203043401)

Submitted as final project report for Practical Topics in Machine Learning, BIU, 2021

1 Introduction

The modern world is intense, so despite the Coronavirus pandemic, people are still likely to enjoy their holidays away from home, ideally in a hotel**.** When guests stay in high-rank hotels, the facilities, comfort, and tasty meals are among the benefits they enjoy. The staff also does its best to meet the needs of every guest. When ordering a hotel room, there are several factors to take into consideration. In addition to those who have mentioned, the price of a hotel is another core consideration. Although luxury hotels are luxurious and enjoyable, one cannot afford to stay in them if the cost is too high.

In fact, the better the hotel's experience is, the higher the check-in price will be. We would, however, like to improve the prices for booking hotels by analyzing their booking data, understanding their pricing policies, and predicting the best prices using discount codes. Allowing people to save money while enjoying their stay in a fine and enjoyable hotel.

The project examines the pricing policies of hotels in New York. In the dataset itself, there are listed hotels and the general information concerning them, such as their name and rating, check-in date, number of days to stay, original price, discount price, as well as a snapshot date and ID indicating when the data was taken.

In this study, we are looking to identify the hotels' pricing policies through the analysis of different techniques**.** Discounts in hotel prices - There are four discount codes (ranging from one to four), each giving a certain amount of discount. For this objective, the best discount code is determined by using **classification** algorithms which predict the code offering the best discounts. To identify groups of hotels with similar pricing policies, **clustering** algorithms are used. As a result, it is possible to predict the price of hotels without big data, using comprehensive data about other hotels in the same group and similar pricing policies.

There is a data cost - Some datasets cannot be obtained for free, and sometimes, to obtain material that is worthwhile, a substantial sum of money is required. An **Active Learning** algorithm **[1]** can help in this case by learning about which queries we are the most uncertain about **[2]** (uncertainty sampling - uncertain about sample price), and for those queries, we will query and pay for their labeled price. Therefore, we will not be paying for every query, but the most uncertain ones only. By calculating a hotel's check-in price based on the paid data, we can predict the price of queries we are more certain about, thus saving money on some less certain and more vital queries for the prediction of the rest.

2 Solution

2.1 General approach

To find out the pricing policies of the hotels, we used classification and clustering. Using classification, we attempt to determine what the best discount code is (the one that will give us the cheapest discount in price). Through clustering, we can find hotels with similar pricing standards. If free data is not available, we can use active learning to predict the price of part of the data and pay for the rest. As a first step, we prepared the data as for classification, clustering, and active learning as explained in the Experimental results part.

After preparing the data, we decided to use several techniques and algorithms for the classification part. Decision Tree (type Gini and Entropy), Random Forest, XGBoost, Naive Bayes, Neural network, Extra Trees Classifier, and K Neighbors Classifier are among the classification algorithms we use. As well as getting the best result, we use various classification algorithms to examine the differences between their techniques as it relates to our data.

The Decision Tree is the basic algorithm based on trees, to begin with. It can be looked at as an anchor to see how the other tree-based algorithms function concerning it. Our decision to use a Random Forest is based on the following reasons: One, Random Forests are considered a highly accurate and robust method due to the number of decision trees participating in the process. The overfitting problem doesn't affect it much. By averaging all predictions, it reduces biases. Moreover, as for Random Forests, they combine predictions from a large number of decision trees. The logic is that even a single model made up of many mediocre ones will still be better than a single great one.

It also makes sense to use XGboost, since it is the most widely used algorithm for machine learning **[3]**, While Random Forest is what we call "bagging applied to decision trees" (reduce variance through bagging), which keeps the model relatively stable even if the data changes. XGBoost is based on **Boosting** which reduces variance and bias. The use of multiple models (bagging) reduces variance. It reduces bias too by training the subsequent model with the information of what errors the previous models made (the boosting part). The Extra Trees Classifier is also used because it uses a slightly different technique from the Random Forest. Random Forest chooses the optimum split while Extra Trees chooses it randomly. However, once the split points are selected, the two algorithms choose the best one from all the subset of features. Therefore, **Extra Trees adds randomization but still has optimization.**

The Naive Bayes method is also used. The algorithm works quickly and saves a considerable amount of time. Naive Bayes is a suitable prediction algorithm for multi-class problems. If its assumption of independence of features is true, it can perform better than other models and require less training data. We believe our data does not fit this assumption of independence (we believe the features are interdependent), but we still decided to use this algorithm. The K-Nearest Neighbor Classifier is used because of the speed at which its training phase can be accomplished compared to other classification algorithms. There is no need to train a model for generalization. Therefore, KNN is known **[4]** as a simple and instance-based learning algorithm.

A neural network was also built and used for classification. The neural network has five layers, two of which are input-output layers, and the others are inner layers. In the first layer (input), we have six neurons as the number of features. In the output layer, there are five features because we used the to_categorical function to turn the target class into a vector of five variables. We selected the number of neurons in the inner layers after trying out different settings for hyperparameters to see which provided the best results.

As part of the classification process, **[5]** we also up-sampled our data to allow us to take into consideration the fact that most of the time the information we gather is heavily imbalanced. What is an imbalanced dataset? The training samples are not equally distributed across the target classes. For instance, if we take the case of the personal loan classification problem, it is effortless to get the 'not approved' data, in contrast to, 'approved' details. As a result, the model is more biased towards the class which has many training instances which degrades the model's prediction power. We did not run all the classification algorithms after resampling but only the best, worst, and neural network algorithms.

As part of the clustering process (after preparing the data for clustering), we use several clustering algorithms to obtain the best results and to compare each algorithm's efficiency on our dataset. We used hierarchical clustering algorithms (using the methods: 'ward', 'complete', 'average', 'single') because it is easy to understand and easy to do and gives the best result in some cases **[6]**. Agglomerative Hierarchical clustering, this algorithm works by grouping the data one by one based on the nearest distance measure of all the pairwise distances between the data points. Distance between the data points is recalculated but which distance to consider when the groups have been formed? For this, there are many available methods. Some of them are: 'ward', 'complete', 'average', 'single'.

We also use K-Means and Gaussian Mixture clustering algorithms. K-Means clustering is a method of vector quantization, originally from signal processing, that aims to partition observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. K-Means clustering is simple to implement, scales to large data sets, guarantees convergence, can warm-start the positions of centroids, and easily adapts to new examples. Gaussian Mixture Models (GMMs) assume that there are a certain number of Gaussian distributions, and each of these distributions represents a cluster. Hence, a Gaussian Mixture Model tends to group the data points belonging to a single distribution together **[7]**. A GMM can also fit and return overlapping clusters, whereas K-Means necessarily imposes a hard break between clusters. During the clustering process, we use Principal Component Analysis (known as PCA) with two components - before and after, using the best clustering algorithm and plot the resulting clusters on graphs. **[8]** Doing PCA before clustering analysis is also useful for dimensionality reduction as a feature extractor and to visualize/reveal clusters. Doing PCA after clustering can validate the clustering algorithm.

In the Active learning process, we analyzed the top 40 hotels with the most data, encoded the string columns with LabelEncoder, and produced a train set based on the query: snapshot date, hotel name, discount code, where each valid row is within 30 days of the snapshot date (more

than that, it becomes more and more difficult to predict prices correctly). The learning process itself uses a query strategy called uncertainty sampling (we sample the queries we are not confident about, saving money on the queries we are confident about), as described in the introduction section. Active Learner's estimator is a classifier, such as Decision Tree, Random Forest, and XGBoost. As a start, we randomly selected five lines from the train set, and we then asked the model which lines to select. We stop running the model after its learning score or R2 score reaches 90% or once we have scanned a quarter of the data set.

2.2 Design

Google Colab notebooks serve as the platform. Sklearn libraries are used for clustering and classification. Modal library for active learning models. The neural network is built using Keras. To validate the results of the classification, we use a CV and various statistical methods. Using the silhouette score, we evaluate the clustering results. For some classification models (Neural network, XGBoost, Random Forest), the training process takes days (Once in a while, the internet goes down and the program must start over). In choosing the best parameters for the classification models, we faced a challenge that we managed to overcome. It took days for the regular Search grid to run and sometimes it gave poor results (since it was not possible to test all the parameters) so we came up with a 'Search grid' of our own, a faster version that even passes on more parameters, and it worked for us. The active learning process uses a very small sample set of the dataset. When we filter the dataset by snapshot time, even within 30 days from the query time, we still got 44 records at most. Therefore, the full power of advanced classifiers cannot be demonstrated properly (more on that in the discussion section).

3 Experimental results

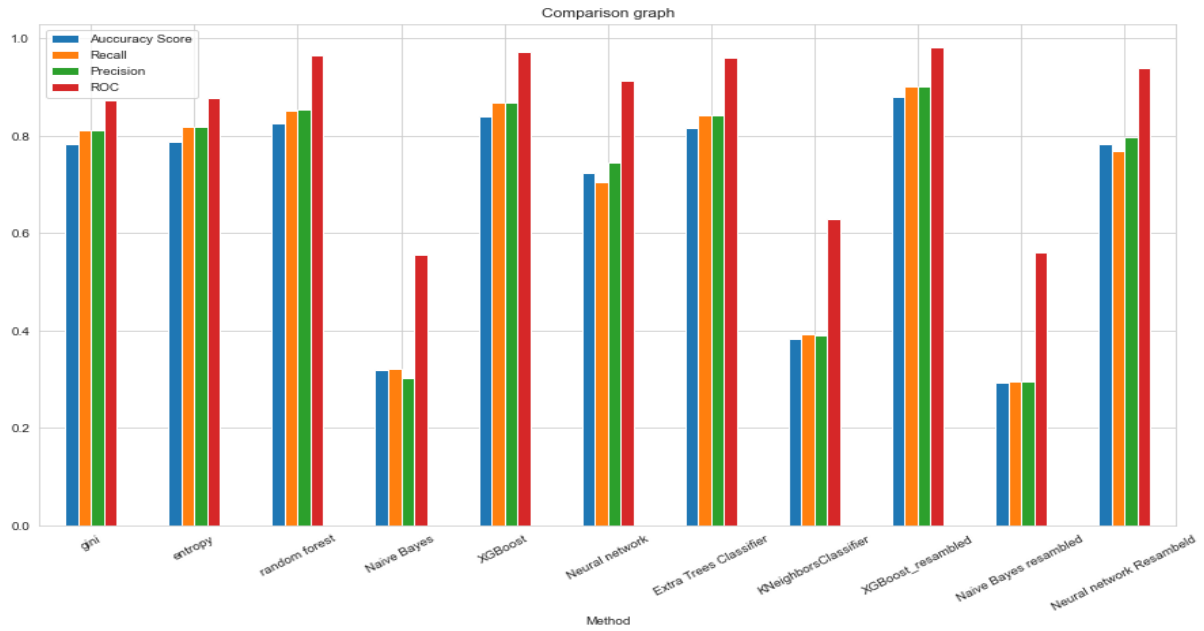
Data Preparation - The original dataset includes the following features: Snapshot - sample ID indicates the ID of records from the same snapshot time. Snapshot Date - indicates when the sample was delivered. Checkin Date - the check-in date of the hotel. Days - duration of stay at the hotel. Original Price - the original price of the booking in dollars. Discount Price - the cost of the booking after discount (the discount depends on the type of discount code). Discount Code - a number between 1 and 4, each having a different effect on the amount of the discount. Rooms Available - the number of rooms available on this date (a value of '-1' indicates that the number of rooms could not be obtained). Hotel Name - the name of the hotel. Hotel Stars - the hotel's rating.

As a first step, we added a few additional features, which can help improve learning model results. The first feature is DayDiff - it shows how many days have passed since the Snapshot Date and Check-In Date. The second feature is WeekDay, which indicates the day of the week when the check-in took place. Third feature is DiscountDiff - which measures the amount of discount in dollars (the difference between Original Price and Discount Price). The last feature is DiscountPerc - which shows the percentage discount. Having gathered all the desired

features, we have cleaned up the dataset and removed the invalid or outlier rows. Then, we wanted the features to follow a normal distribution, so we checked the distribution of each feature and normalized it if necessary. As a last step, we cleaned the outliers to minimize "noise" that affects the performance of the algorithms. Following this, we used BoxPlot and Scatter Matrix to visualize the data distribution, and a log function to normalize discount columns. Lastly, we removed outliers from the data. As can be seen in the new BoxPlot and Scatter Matrix, the preparation has improved the distribution. All the graphs we produced during the preparation can be found here.

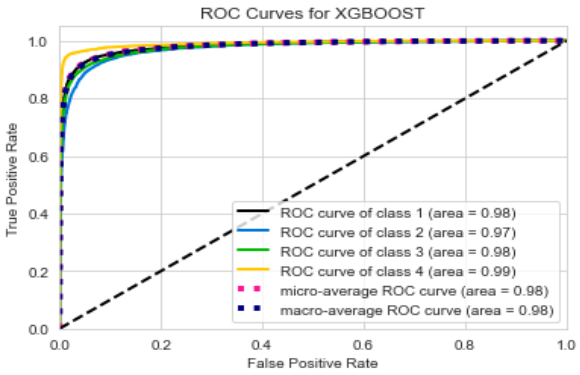
Classification - To prepare the data for classification, the maximum discount code was used to group the data. In the next step, we encoded string data (hotel name, week day, snapshot date, check-in date) using Label Encoder. Finally, we resampled the data in order to achieve a balanced representation of discount codes. After the preparation process, the data were split into a train set (0.7) and a test set (0.3), then the data were cross-validated (5) to determine the best classifier (with the highest accuracy score). As part of this process, we run each classifier using our 'Search grid' to improve the results via the best parameters for each classifier. Our methodology for assessing each classifier (listed in section 2.1) includes visualizing and expressing the **confusion matrix**, and **ROC** graph, also we print the **important features** (the features that influence the classifier the most). To compare the performance score of the classifiers, we used several different metrics such as **accuracy**, **recall**, **precision**, and **ROC.** We resampled the data to improve the performance and repeated the above processes with XGBoost, Naive Bayes, and a Neural Network.

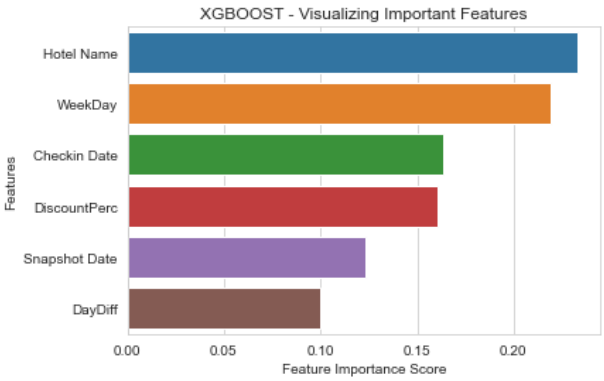
In the end, we produced the following comparison graph that allows us to understand the score of each classifier and method:



The resampled Naive Bayes is the worst classifier as demonstrated in the graph. This is its graphs.

The resampled XGBoost is the best classifier as demonstrated in the graph. This is its ROC and the feature influence graphs:

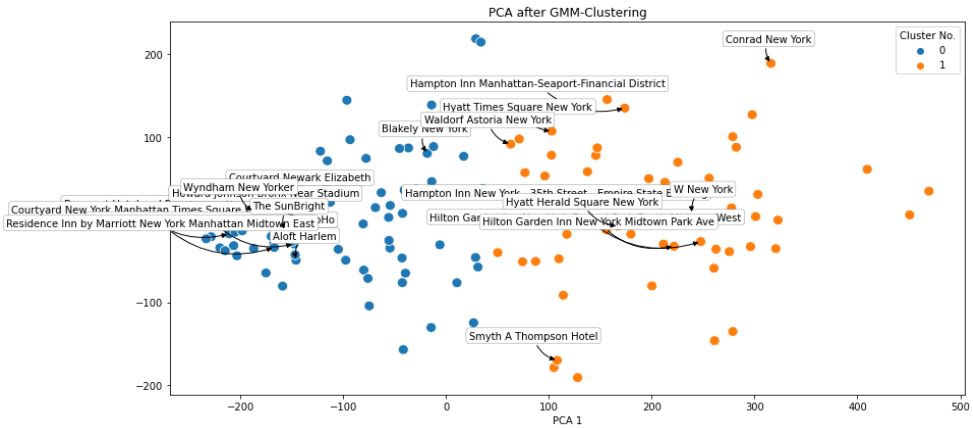




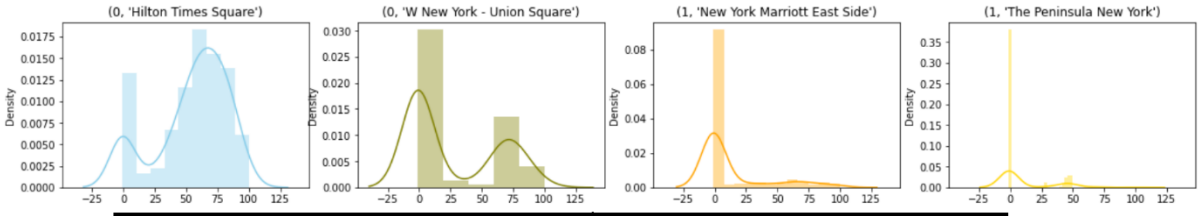
Clustering - In preparation for clustering, we filtered the most appearing hotels in the data (top 150). After that, we filtered check-ins with the highest appearances (top 40), providing enough data to cluster. As a last step, we normalized the price data. Following the preparation phase, we used various clustering models to create the data set - agglomerative with the following methods: ward, complete, average, and single (we produced a dendrogram for each, it can be found here: Dendrogram), as well as K-Means and Gaussian Mixture. We used the Silhouette Score and the number of clusters to compare the model's performance (**as shown in the table above**).

	Model Name	Silhouette Score	Clusters	Hotels per cluster
0	ward	0.195748	2	[96, 53]
1	complete	0.205876	2	[118, 31]
2	average	0.211684	2	[39, 110]
3	single	0.229274	2	[148, 1]
4	KMeans	0.198121	2	[91, 58]
5	GaussianMixture	0.198802	2	[92, 57]

We also try PCA before and after. These are the clusters with PCA after GMM:

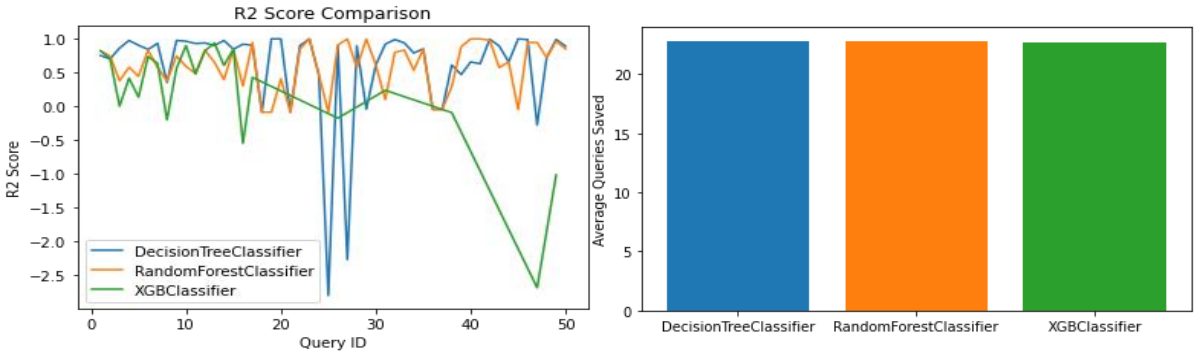


Below is the price distribution for 4 random hotels (2 of each cluster), as can be seen each cluster has distinctive prices:



Cluster	0		1
Hotel Name	Hilton Times Square	W New York - Union Square	New York Marriott East Side	The Peninsula New York
mean	54.917169	27.140908	12.292717	10.399218

Active Learning - Data (Hotels_data_changed) from the prepared dataset was used as the base dataset, and only the top 40 hotels with the most data were analyzed. Our active learner model was fitted with Decision Tree, Random Forest, and XGBoost classifiers as estimators with a query strategy of **uncertainty sampling** in which we sample the queries we are least certain about, saving money on more certain queries. Based on the **learner score** and **R2 score**, the results from this study were collected and stored in an excel file (Active Learning Results). Following that, we calculated how many queries were saved by each algorithm. In the next two graphs, we display only the top 50 queries for display purposes. As can be seen, although the



R2 score has shown a difference between the models (as will be explained in the discussion section) The average queries saved by each model are about the same.

NOTE: Because of space constraints, we only presented the best result, but all the produced graphs can be found as follows: Data Preparation, Clustering, Active Learning, Classification (Classifier results are organized by before resampling and after resampling, with each classifier having its own subfolder).

4 Discussion

Using a variety of clustering models, we attempted and succeeded in finding hotels with similar pricing standards. The GMM model was the best clustering model for our data, followed by the K-Means model. By using a PCA to analyze the data and printing statistical functions on two hotels from each cluster, it becomes evident that the average price of hotels differs among clusters. There are two main pricing policies indicated by the clustering graphs, and one group of hotels is cheaper than the other (as shown by the average price we took on samples from each cluster).

In addition, when we looked at the original dataset after taking the two samples from each cluster, we discovered that the two samples with the lowest average belonged to 3-star rated hotels, while the two samples with higher averages belonged to 5-star rated hotels. In other words, the division we received based on prices reflects the reality of hotels with high and low ratings and the price reflects the rating. There are probably some four-star hotels that are more similar in price to five-star hotels and some that are more similar to three-star hotels, and these are the overlap points in the two clusters (in the graph we printed). The points that are far from each other probably refer to hotels with three or five stars, not four.

Using classification, we determine what is the best discount code. As shown in the results, the best classification model was XGBoost. Also, after resampling, the results of the best model (XGBoost) got better, and the results for the worst model (Naive Bayes) have become less good. This can be explained, in our opinion, very simply. The best algorithms comprehend the data from the beginning, so the answers are almost always correct. The data is more balanced after resampling, thus providing better results. Conversely, the worst algorithms are not suitable for this type of data in the first place and the reason this gets worse is that with more balanced target classes, the chances of giving a correct answer decreases because if it was unbalanced then it might fall on a right answer by chance.

According to us, the naive assumption of the Naive Bayes algorithm (its name derives from this assumption), that there is no dependence between the properties/features of the classified objects once their classification is already known. **This assumption is incorrect in our case. There is a connection between the different features** and this assumption causes the algorithm to give us bad results. Despite its poor results, its training phase was the fastest of all - for data that suits it, it would be beneficial to use it. There are other algorithms whose results were not the best but their runtime was very fast (as we expected, according to the general approach).

Based on Active Learning results, the Decision Tree and Random Forest classifiers scored about the same while XGBoost trails behind. Although there were different scores, the average number of queries saved by each classifier was the same (20). There were not many data points, which can explain the difference between the models' performances. The query that produced the biggest train set yielded 44 lines, while the others yielded even fewer. As XGBoost, unlike the former, cannot be demonstrated fully where there is insufficient data to train it.

5 Code

This code is divided up into six separate Colab notebooks. Each of them will be briefly described and linked here. **DataPreparation.ipynb** - Here is the first step in preparing the data, and understanding what the data looks like, as well as identifying and cleaning outliers in the data. **ClusteringPreparations.ipynb** - Prepares a CSV file for clustering based on the problem definition. **Clustering.ipynb** - This is where clustering and PCA are done, along with the graph prints. **ClassificationPreparations.ipynb** - Creates two CSV files for classification, one before and one after resampling. **Classification.ipynb** - This is where classification is done, along

with the graph prints. **ActiveLearning.ipynb** - This module attempts to model prices by sampling the most uncertain queries, creating a CSV file holding the results, along with prints of relevant graphs.

References

[1] **A modular active learning framework**

[2] **Uncertainty sampling**

[3] **XGBoost Algorithm for Classification and Regression in Machine Learning**

[4] **KNN Classification using Scikit-learn**

[5] **Handling Imbalanced Data- Machine Learning, Computer Vision, NLP**

[6] **Data Clustering Algorithms - Hierarchical clustering algorithm**

[7] **Build Better and Accurate Clusters with Gaussian Mixture Models**

[8] **How would PCA help with a k-means clustering analysis? - Cross Validated**