Hotel Rating Predictive Analysis

By
Cole Gluth and Trevor DeBardeleben

Section 1: Problem Addressed and Data Collected

For our project, we decided that we wanted to build several models in order to create predictions based varying attributes of hotel reviews to see how accurate our prediction on a certain quality would be given the other attributes of a hotel review. In our case, we studied how accurately a model composed of all other attributes could make a prediction of a desired property of hotel reviews. Specifically, we performed studies on the overall_ratingsource of a hotel and also on the cleanliness of the hotel. In order to conduct any type of study for this project, we needed a large enough dataset. The dataset that we ended up using is composed of 3105 hotel reviews from 11 cities all across the globe.

We retrieved our data from https://github.com/kavgan/OpinRank. The data was originally stored in 10 separate csv files for each city represented by the dataset. We first wrote a script to merge all of the data into one csv, as our goal was not to predict cities, but rather look at a group of data and predict certain attributes (cleanliness and overall_ratingscore) from that data. After merging the files, the data was stored in csv with the following format: doc_id, hotel_name, hotel_url, street, city, state, country, zip, class, price, num_reviews, CLEANLINESS, ROOM, SERVICE, LOCATION, VALUE, COMFORT, overall_ratingsource. In preparing our data, it became immediately clear that some of the data's attributes were not necessary for our study so we dropped them. In the case of this project, we dropped doc_id, hotel_name, hotel_url, street, city, state, country, zip, class, price, and num_reviews as these attributes would not be useful for any type of predictive analytics to be done. Of the remaining data, we realized some cleanliness

scores were zero. This implies there was faulty data, so we decided to remove any rows where cleanliness scores scores were zero.

We conducted two separate studies for this project, one where we make predictions for overall_ratingsource from our dataset and an additional one where we make predictions on cleanliness based on the other attributes in our dataset. In preparations of our data split, we created two numpy arrays. The first was titled predict and stored all values for cleanliness.The second was our features array which stored all over useful values still in our dataset. These being, num_reviews, ROOM, SERVICE, LOCATION, VALUE. Our dataset was then split into two sets, a training and a validation set. We then used train_test_split to split the data into a training and validation set in which 75% of our dataset was dedicated to training the set with the remaining 25% to be used as a validation tool. It is important to use both a training and validation set to determine the accuracy of our model and ensure our model does not overfit the data we trained it with. The goal of this model was to make a prediction of cleanliness from the features that we stored in feature array.

The second of our models was very similar to our test of cleanliness except this time using overall_ratingsource as our predict array and our features array being composed of city, num_reviews, CLEANLINESS, ROOM, SERVICE, LOCATION. Again, We used train_test_split to split the data into a training and validation set in which 75% of our dataset was dedicated to training the set with the remaining 25% to be used as a validation tool. The target of this prediction was to guess the overall_ratingsource on a basis of the data contained in our features array.

Section 2: Models and Hyper-Parameters

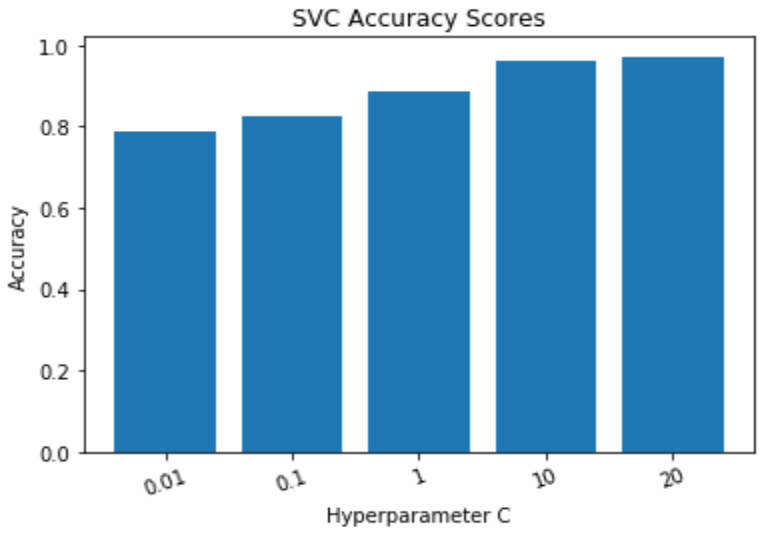
For both of our CLEANLINESS and overall_ratingsource, we used the same models and hyperparameters, so we are going to discuss them together. In the case of the models that we used, we used DecisionTreeClassifier with criterion gini, we used MultinomialNB to create a multinomial Naive Bayes classifier, we performed a Linear Regression on the data, we implemented a support vector machine with a C-Support Vector Classification estimator (SVC), and finally we used RandomForestClassifier. We used these models for both of our tests. Basically, we wanted to see how each of the models that we had studied in class would perform on our dataset and see which would be appropriate for similar datasets in the future. In the case of our SVC model, we used the hyperparameters, c = [.01,.1, 1, 10, 20]. Where the C parameter of the SVC model corresponds to how much we care about avoiding the misclassification of data, where a low C means we have a higher chance of misclassifying data. In the case of our Random Forests Model, we altered the hyperparameter of n_estimators for estimators=[20, 100, 500, 1000], where n_estimators indicates the number of trees in our random forest.

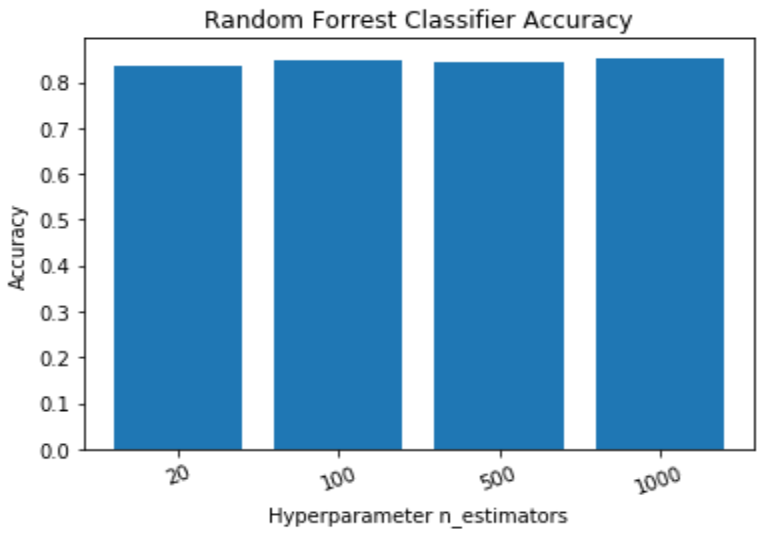
Section 3: Evaluation of Performance

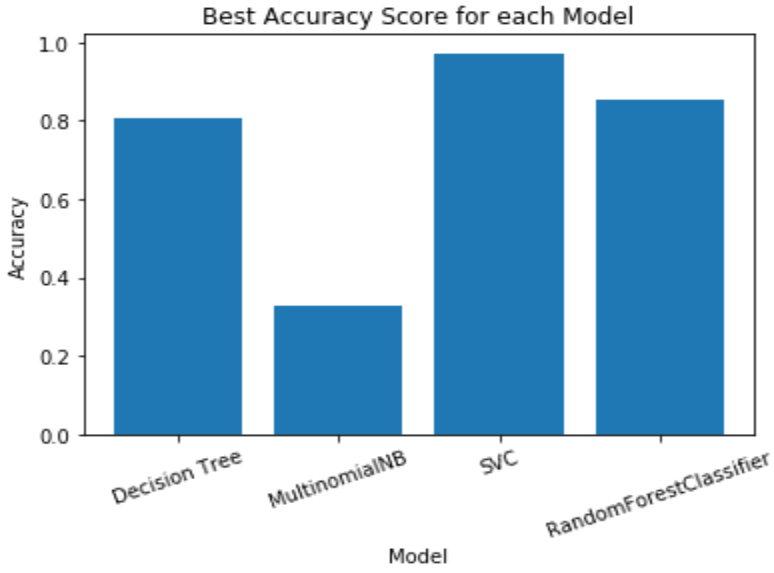
3.1 Cleanliness Predictions

Model	Accuracy
DecisionTreeClassifier	0.807365
MultinomialNB	0.328612
LinearRegression	0.896860

SVC with C = .01	0.786119
SVC with C = .1	0.827195
SVC with C = 1	0.883853
SVC with C = 10	0.963173
SVC with C = 20	0.971671
RandomForestClassifier with n_estimators = 20	0.837110
RandomForestClassifier with n_estimators = 100	0.847025
RandomForestClassifier with n_estimators = 500	0.845609
RandomForestClassifier with n_estimators = 1000	0.852691



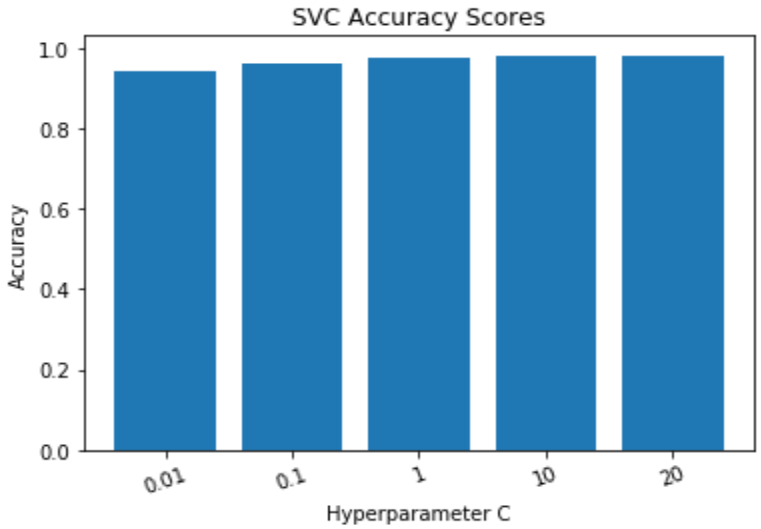


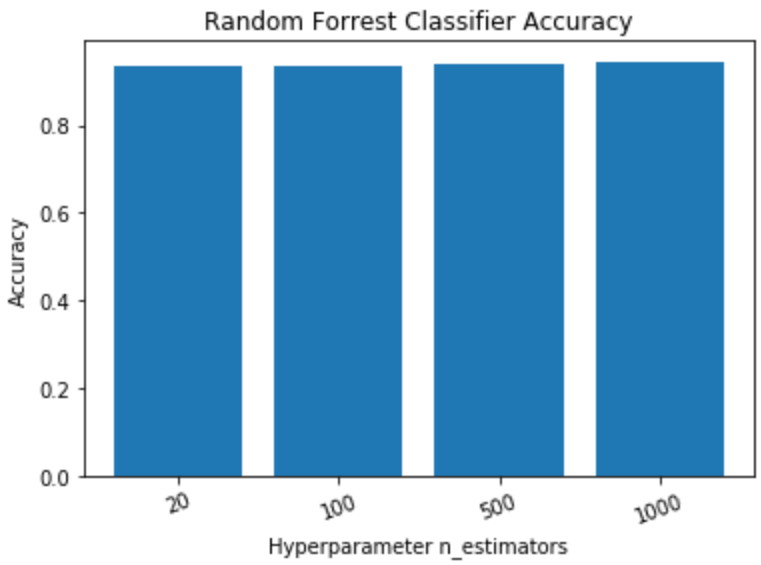


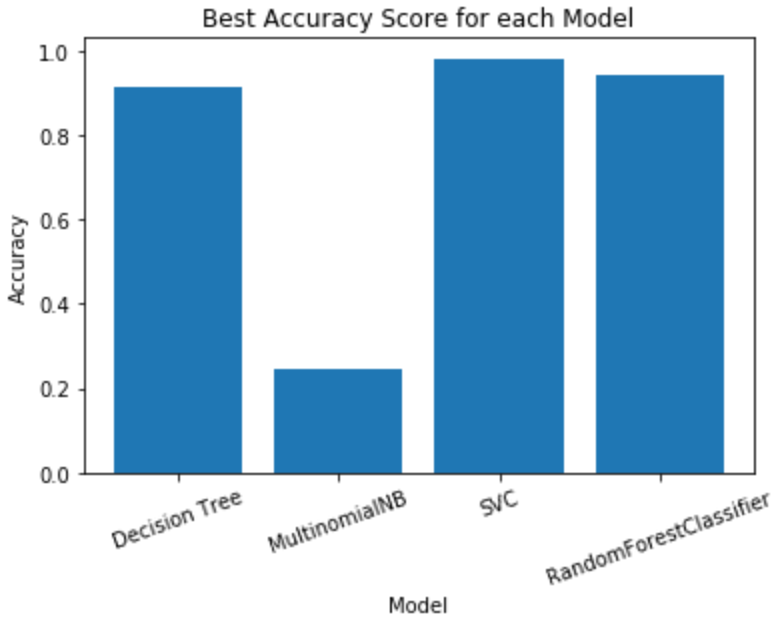
3.2 overall_ratingsource Predictions

Model	Accuracy
DecisionTreeClassifier	0.916431
MultinomialNB	0.246459
LinearRegression	0.867843
SVC with C = .01	0.943343

SVC with C = .1	0.963173
SVC with C = 1	0.977337
SVC with C = 10	0.981586
SVC with C = 20	0.981586
RandomForestClassifier with n_estimators = 20	0.936261
RandomForestClassifier with n_estimators = 100	0.934844
RandomForestClassifier with n_estimators = 500	0.940510
RandomForestClassifier with n_estimators = 1000	0.944759







Explanation of Performance Measurement for Both Tests

For our evaluation of performance, we used accuracy_score from sklearn.metrics except for the case where we performed a linear regression. This is because LinearRegression does not have an accuracy_score component so instead we metrics.explained_variance_score. The last bar chart

that is included excludes LinearRegression for this reason. We obtain the accuracy of our classification from accuracy_score as it returns the percentage of labels that are an exact match between the predicted values and the actual values.

Section 4: Discussion of Results

In general, the the varying models that we tried all performed decently with the exception being MultinomialNB. Upon investigating the matter, it became clear that using a Multinomial Naive Bayes classifier is incompatible with the type of study that we were performing. Typically a MultinomialNB is used for classification with discrete features such as words in a text. A more suitable usage for it would be if we were trying to predict whether or not a description of an event is of a category. Such as, whether the description “a very close game” belongs to a sports category. Otherwise, DecisionTreeClassifier, LinearRegression, SVC (with any hyperparameters), and RandomForestClassifier (with any hyperparameter), each performed pretty well with prediction accuracies greater than 80% for each. Since these models all have high accuracies, that would indicate that they are suitable choices for type of problem that we are testing here. The best performing model in the case of our cleanliness prediction was our Support Vector Classifier with hyperparameter C = 20 which had an accuracy_score of 0.85269. The best performing model in the case of our overall_ratingsource prediction was also our Support Vector Classifier with hyperparameters of C = 10 and C = 20 which both had an accuracy_score of 0.981586. One area of surprise is the lack of variance in our SVC model in comparison to what we observed in the homework assignment.

Overall, our project builds upon concepts taught in class and successfully demonstrates how various predictive models can be used on a dataset to accurately predict attributes within the dataset. For our tests, we discovered the Support Vector Classifier was the most accurate model for predicting both cleanliness and overal_ratingscore. Multinomial Naive Bayes Model ended up being the least accurate model for our dataset. With this information, in the future, we certainly would choose the Support Vector Classifier to make predictions on our dataset. However, this is not a one size fits all solution. Every dataset is different, and what works well for this dataset will not necessarily be best for other datasets. This project therefore provides a good example of steps that could be used in the future to determine the best model to use to make predictions on a given dataset.