Auction Price and Item Prediction

CP322 - Final Project

Stefan Bukarica

190563930

Dr. Sukhjit Singh Sehra

Table of Contents

Abstract…….2

Introduction……..3

Data Description……..4

- Data Collection……4
- Exploratory Data Analysis……...5

Methodology……6

Results and Visualization……9

Conclusion and Further Interpretation……11

Resources……12

Abstract

In this project, I explore two types of problems, a regression problem for predicting the price of an auction listing and a classification problem for predicting the item type of an auction listing. The data that was collected is from kaggle, where each record is a bid on an item, which shows us 9 features about the bid. After some feature manipulation and scaling (discussed in the Exploratory Data Section) I obtained two datasets, one where the target variable is Price for our regression, and one where the target variable is the Item Type for our classification. From this, I created 4 models for each problem to train and predict on a test set in which I was able to identify a high accuracy for both problems. The model which performed best on the Price prediction was a tuned Random Forest Regressor, and the model which performed best on the Item Type prediction was a tuned Random Forest Classifier. To evaluate amongst the other models R^2 Score, Mean Absolute Error, and Mean Squared Error was used for the regression, and Accuracy, Precision, Recall, and F1-Score were used for the evaluation of classification models. Although the regression model can predict price accurately and the classification model item type, respectively, there are some drawbacks which are discussed in the conclusion. Overall, the goal was to predict price and item type accurately which was achieved which brings me to see how these two models can work together in an auto bidder with personally collected auction data.

Introduction

The purpose of this project was to develop two models, one to predict auction price, and another to predict auction item types. The idea behind this project was to see if it is possible to accurately predict the price of an auction as well as accurately predict the type of item in an auction listing. Eventually, I would like to create an auto-bidder that can predict what price to bid on specific items. This project was an introduction to see if the above is possible, and more importantly, if it is of use.

The first part of the project consists of pre-processing the dataset, this included things such as dropping the bidder id and auction id columns, as well as creating a new column called 'bid no' which is essentially the number of that particular bid in the sequence for that item. After scaling the values which needed scaling (non-categorical), two datasets were created for training and testing.

The second part of the project was creating a set of models to test for the regression and classification each. For the regression problem, a set of classes were made for this. Each class follows a similar structure to the ones at the end of Assignment 3 in this course. The constructor assigns the predictors and target variable along with doing train/test split, followed by a fit method which includes an option for tuning, a method to return predictions of the fitted model, and finally a method which returns the respective scores of the model. The above is done in an identical style for the classification problem, with model classes following the same structure. The models used for the regression are: Linear Regression, KNNRegressor, RandomForestRegressor, and a Support Vector Regressor. The models used for the classification are: Logistic Regression, KNNClassifier, RandomForestClassifier, and a Support Vector Classifier.

The last part of the project is the evaluation of the above models in the main.py module, which displays the scores of the models on the test data along with the visualization of each model's predictions, from which the best model from each problem is chosen.

Data Description

Data Collection

Data was collected from Kaggle, the raw dataset consists of 9 features, with each record representing an individual bid on an auction listing. The following are a list of those 9 features:

- auctionid: unique identifier of an auction
- bid: the proxy bid placed by a bidder
- bidtime: the time in days that the bid was placed, from the start of the auction
- bidder: eBay username of the bidder
- bidderrate: eBay feedback rating of the bidder
- openbid: the opening bid set by the seller
- price: the closing price that the item sold for (equivalent to the second highest bid + an increment)
- item: auction item with 3 levels: (Cartier Wristwatch, Palm Pilot M515 PDA, Xbox Console)
- auction_type: The duration of auction in days, with 3 values (3,5,7 days)

The raw dataset has a shape of 10681 records with 9 features. In the next section I discuss how I modified this dataset to be prepared for model training and to complete the task that this project sets out to complete.

Exploratory Data Analysis

Since our dataset contains 9 features, the first step is to understand this dataset. I first realized that there was no "bid number" column indicating which bid this was on the listing. Since an auction can have multiple bids until the final bidder wins I created a column to track this.

```
#since the auction id column has dupes we can find the number of bids on an auction

auction['auctionid'].value_counts()
```

```
8214355679    75
8212629520    57
3023174478    54
3020532816    51
8212602164    50
              ..
3025639289     1
3023652961     1
3024121735     1
3025036419     1
3025035412     1
Name: auctionid, Length: 628, dtype: int64
```

```
auction["bid_no"] = auction.groupby("auctionid").cumcount()+1
auction.head(10)
```

The next step was removing the columns 'auctionid' and 'bidder' since these were only serving as ID's. After this I dropped the item type 'Palm Pilot M515 PDA', around half of the auction bids were for this item type (55%) so to shrink the dataset and reduce the training time of the models, specifically RF and SVM, this item type was dropped making this a binary classification for the item type. This is important as well because the item type was relatively unbalanced like, which would make accuracy a difficult way to interpret the success of the classification models. The next step was to scale the data that needed to be scaled. For the regression dataset I created dummy variables for the Item Type and the auction type was numerical but technically it was

categorical, so I kept that feature unscaled. The rest of the features are all on different scales, so it was necessary to scale the dataset before training, especially for distance based algorithms such as KNN. The classification dataset follows a similar story, where I scaled the same features except the item type and auction type. This was due in part again to using KNN as one of the models for training. Now that I have explained the data we can move on and take a look at how I train various models and evaluate them.

Methodology

As mentioned before, for each of the respective problems, regression and classification there is a set of 4 models used to train on. For the regression, I used Linear Regression, KNNRegressor, RandomForestRegressor, and a Support Vector Regressor (SVR). Each of these except the Linear Regression model had an option to tune. In the tuning parameters, I use a grid search to tune the hyperparameters of KNN, RF and SVR. For KNN, the number of neighbors, weights, and distance types are tested and scored by negative MSE. For RF I tune the number of estimators, depth and leafs. Finally, for the SVR, I tune different values of C, gamma and keep the kernel constant with a linear kernel. The results of the tuning for the default models and after are shown below:

Before:

```
R^2 Scores:

Linear Regression: 0.8245225891994606
KNN Regressor: 0.88721080187006
RF Regressor: 0.8977720437537272
Linear SVM Regressor: 0.7925729278164269

Mean Absolute Error:

Linear Regression: 0.24167873071048324
KNN Regressor: 0.14437825409047864
RF Regressor: 0.12223984631118713
Linear SVM Regressor: 0.20285637913939605

Mean Squared Error:

Linear Regression: 0.16003825555191115
KNN Regressor: 0.10286558498593378
RF Regressor: 0.09323356044321297
Linear SVM Regressor: 0.1891768668973168
```

After:

```
R^2 Scores:

Linear Regression: 0.8245225891994606
KNN Regressor: 0.8652828207005111
RF Regressor: 0.8970867242201719
Linear SVM Regressor: 0.7925729278164269

Mean Absolute Error:

Linear Regression: 0.24167873071048324
KNN Regressor: 0.17033963347805844
RF Regressor: 0.12093885554718896
Linear SVM Regressor: 0.20285637913939605

Mean Squared Error:

Linear Regression: 0.16003825555191115
KNN Regressor: 0.1228642608162872
RF Regressor: 0.0938585830153235
Linear SVM Regressor: 0.1891768668973168
```

We can see that the RF Regressor performs the best out of all of the 4 models for all 3 metrics, R^2 Score, MAE and MSE. You might notice that the SVM Regressor does not change throughout the tuning, this is because the default non-tuned SVR was the same as the best parameters used in a grid search of the tuning. This is because tuning using multiple kernels such as RBF, Polynomial, was taking very long in the grid search, so I opted to use only a linear kernel for both the regression and classification models.

Next, let's go over the procedure for the classification problem, while the structure is very similar to the regression problem. For this I also used a set of 4 models to train and predict, these are: a Logistic Regression, KNNClassifier, RandomForestClassifier, and a Support Vector Classifier (SVC). For the latter 3 models I tune the same hyperparameters, substituting the values appropriately for run time consistency. For the Logistic Regression, a gridsearch was composed of parameters including different solvers, an L2 penalty, and different c values along with a 10-fold CV, while the other models mentioned prior had a 5-fold CV for both classification and regression. I did this to see if my Logistic model could compete with some of the other ones, due to how well RF and KNN performed in the regression section I assumed the same would be for classification. Getting back on track, each of the grid searches for the classification models' tuning was scored on accuracy. Finally, we can see these are the results for before and after tuning of the hyperparameters for each model.

Before Tuning:

```
Accuracy:

Logistic Regression: 0.9221871713985279
KNN Classifier: 0.9348054679284963
RF Classifier: 0.9968454258675079
SVM Classifier: 0.9316508937960042

Classification report:

Logistic Regression:                    precision    recall  f1-sc

Cartier wristwatch       0.95      0.86      0.90       401
 Xbox game console       0.91      0.97      0.93       550

          accuracy                           0.92       951
         macro avg       0.93      0.91      0.92       951
      weighted avg       0.92      0.92      0.92       951

KNN Classifier:                  precision    recall  f1-score

Cartier wristwatch       0.95      0.89      0.92       401
 Xbox game console       0.92      0.97      0.94       550

          accuracy                           0.93       951
         macro avg       0.94      0.93      0.93       951
      weighted avg       0.94      0.93      0.93       951

RF Classifier:                   precision    recall  f1-score

Cartier wristwatch       1.00      1.00      1.00       401
 Xbox game console       1.00      1.00      1.00       550

          accuracy                           1.00       951
         macro avg       1.00      1.00      1.00       951
      weighted avg       1.00      1.00      1.00       951

SVM Classifier:                  precision    recall  f1-score

Cartier wristwatch       0.95      0.89      0.92       401
 Xbox game console       0.92      0.97      0.94       550

          accuracy                           0.93       951
         macro avg       0.93      0.93      0.93       951
      weighted avg       0.93      0.93      0.93       951
```

After Tuning:

```
Accuracy:

Logistic Regression: 0.9232386961093586
KNN Classifier: 0.961093585699264
RF Classifier: 0.9968454258675079
SVM Classifier: 0.9305993690851735

Classification report:

Logistic Regression:                    precision    recall  f1-sc

Cartier wristwatch       0.95      0.87      0.90       401
 Xbox game console       0.91      0.97      0.94       550

          accuracy                           0.92       951
         macro avg       0.93      0.92      0.92       951
      weighted avg       0.92      0.92      0.92       951

KNN Classifier:                  precision    recall  f1-score

Cartier wristwatch       0.95      0.96      0.95       401
 Xbox game console       0.97      0.96      0.97       550

          accuracy                           0.96       951
         macro avg       0.96      0.96      0.96       951
      weighted avg       0.96      0.96      0.96       951

RF Classifier:                   precision    recall  f1-score

Cartier wristwatch       1.00      1.00      1.00       401
 Xbox game console       1.00      1.00      1.00       550

          accuracy                           1.00       951
         macro avg       1.00      1.00      1.00       951
      weighted avg       1.00      1.00      1.00       951

SVM Classifier:                  precision    recall  f1-score

Cartier wristwatch       0.95      0.88      0.91       401
 Xbox game console       0.92      0.97      0.94       550

          accuracy                           0.93       951
         macro avg       0.93      0.92      0.93       951
      weighted avg       0.93      0.93      0.93       951
```

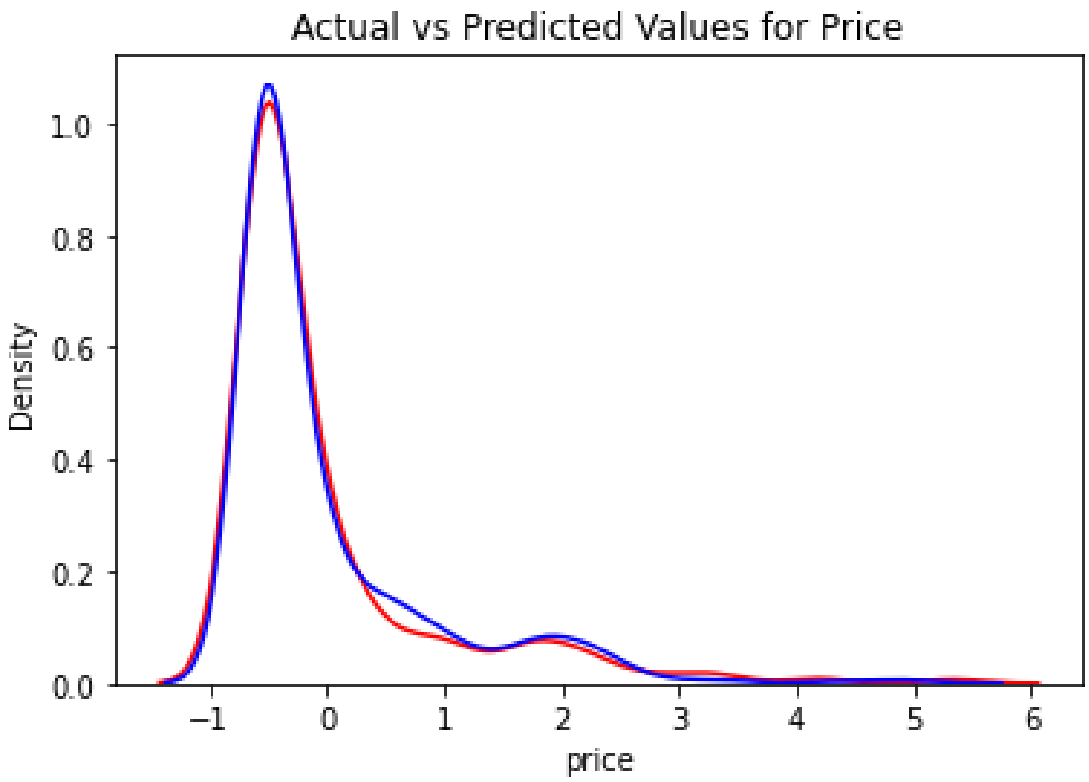
We can see that the RF classifier performs extremely well compared to the other models, and since we want to predict the auction type accurately I choose the Random Forest Classifier. I'd like to add that the same instance is happening here with the RF classifier as did with the SVR in the regression section, the tuned params are actually the same as the default params for the RF. Whereas here the SVC has a slight improvement after tuning.

Results

As shown above, I selected the best model from each situation with the Random Forest Regressor being the best for regression and the Random Forest Classifier being the best for the classification. This is a good outcome since Random Forests are generally easy to interpret. Most models suffer from interpretability and complexity as the accuracy increases or error is lowered, since we want to choose the most accurate model here, it is a good sign that we don't have to sacrifice interpretability in order to have a high performance model. Since we measured test error and accuracy on our test set, it is evident that the model is not overfitting despite having extremely high accuracy. Here are some visualizations of the models predictions:

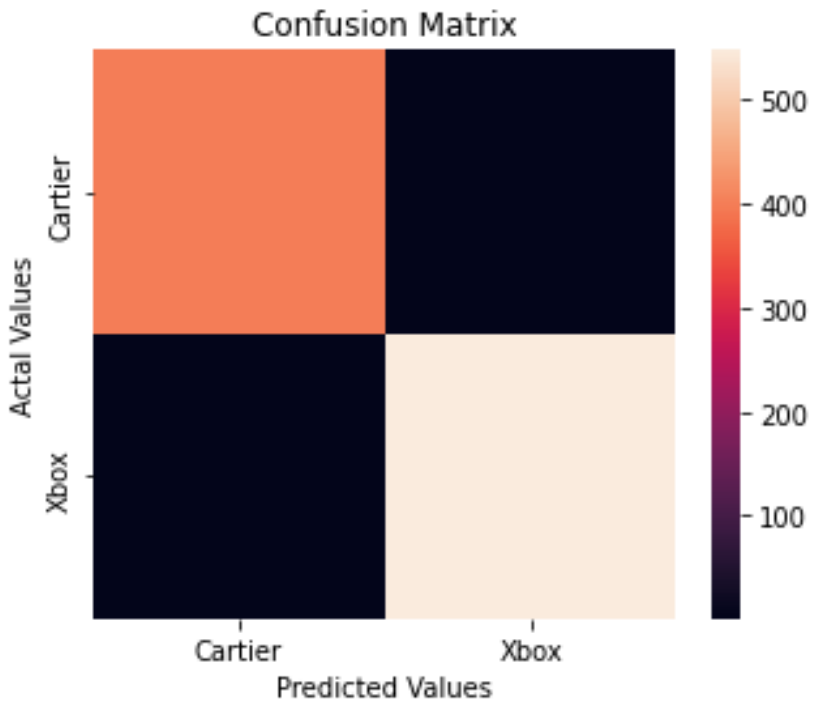
Regression Problem: (Red indicates the predictions)

```
RF:
```



Classification:

RF:



Another reason for the accuracy of the RFClassifier being so high, could be due to the simplicity of the problem. Since there are only two categories to predict in the item type, this accuracy could change drastically when we introduce many different item types. This is more of a 'real-life' scenario and something I want to do to test on my own with different auction data.

Conclusion and Further Interpretation

This project set out with the goal of understanding the feasibility of auction price and auction item prediction. By the end of this, I can happily say that it is definitely possible and that the problem would be very interesting to investigate with personally collected data. With that being said I'd like to go over some key takeaways and insight for future improvement. The dataset used was not ideal by any means. I removed one of the categories in the item type making it a binary classification problem. In reality, I would like to be able to predict from a wide array of item types, maybe more than 20 or so to be able to grab every brand within a type of product. Also, the dataset simply doesn't have enough features. There are many features that could have been included in an auction such as "Number of favourites", or "How many in watchlist", "Number of pictures shown", etc. With this, feature selection would be more appropriate for our models, which is why it was not used here. Next, I would like to test different models. As the dataset changes, the models chosen for testing here may not be applicable or may need to be tuned differently. Finally, I would like to explore if it is possible to predict both price and item type at the same time, ie, we have two targets, a labeled class and price. One way to do this would be to have a dataset where neither price or item type is used in the X training set, and then simultaneously run two models (one for price one for item type) that outputs both given our dataset. Another way would be latent class analysis which I am unfamiliar with, but would like to explore. All in all, I am happy with the outcome of this project and the possibility of further expansion of it to apply to my personal endeavors with auction bidding.

Resources

Auctions, M. O. (2016, November 13). Online auctions dataset. Kaggle. Retrieved April 6, 2023, from https://www.kaggle.com/datasets/onlineauctions/online-auctions-dataset

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). An Introduction to Statistical Learning: With applications in R. Springer Nature.