Predicting diamond prices

Zicheng Hu[1,*], Dajun Luo[2,*], Zhe Wang[3,*], Wenhan Xia[4,*]

[1]) qz298@cornell.edu [2]) dl938@cornell.edu

[3]) zw273@cornell.edu [4]) byw4@cornell.edu

∗) These authors contributed equally.

Abstract

Diamond is a popular commodity in every culture around the world. Predicting the prices of diamonds can be valuable not only for customers but also for companies. Using statistical learning methods including linear regression, polynomial regression, random forests, and XGBoost, together with a data set from Kaggle that contains diamond prices and different predictors, we have developed a method to predict diamond price according to features such as carat, cut, clarity, and diamond sizes. The best test mean square error using our method is xxx.

1 Introduction

Diamonds have been of significant interests to the general public for their durability and aesthetic value. While the diamonds market seems like a mystery and it is nearly impossible to judge the value of the diamonds solely by their appearance, predicting diamonds' price can be helpful to trades and personal investment [1]. Suppose a diamonds bider is about to purchase a great number of diamonds, predicting the diamonds' value can give a reference price of how much the bider should invest. Diamonds are also popularly purchased for engagement rings. By applying machine learning methods to analyze diamonds' price, we can come up with diamonds' price calculators to help customers make their best

purchase without getting ripped off. Our project aims to analyze diamonds' price by their cut, color, clarity, carat and other attributes.

2 Exploratory Data Analysis

To better understand the structure of data [2], we conducted an exploratory data analysis (EDA). The dataset includes 53940 observations and we randomly split them into a training set (ratio needed), containing 43152 observations, and a test dataset, containing the remaining observations. The EDA only evaluates the training set because we do not want to comprise any information in the test dataset until the final stage.

The training set contains 43152 rows and 10 columns. 6 out of 9 features take on numerical values and the rest are categorical features, as is shown in Fig. 1. We noticed that the response variable, price, only takes on integer values, but the number of distinct levels is very large. Naturally, a supervised regression model would be more appropriate for this study.

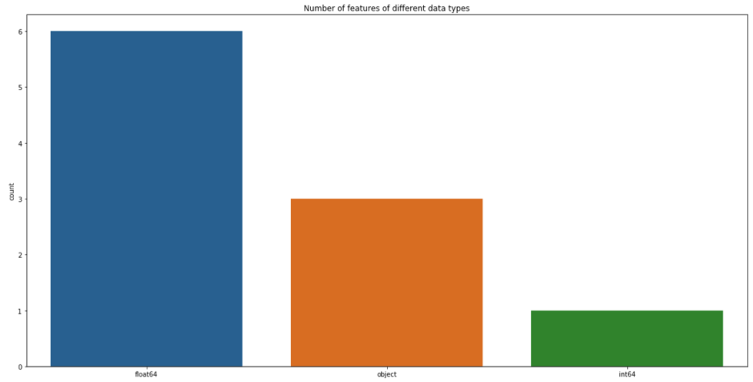

Figure 1: Features of the diamond dataset. Among all 9 features, 6 of them take on numerical values and the rest are categorical features. The green bar indicates the response variable, price.

The next step is to assess the quality of the dataset. Fortunately, the dataset is completely free of missing value, which saves us a lot of effort and time of imputation. Next we examined the box plot of each numerical feature to discover outliers and high leverage points, as is shown in Fig. 2. Those box plots highlight a few zero points in feature x, y and z which might have been corrupted during data collection. Besides, there is one observation with an extremely large value in y. Since these data fall more than 3 standard deviations from the norm, we tend to identify them as anomalies. To get

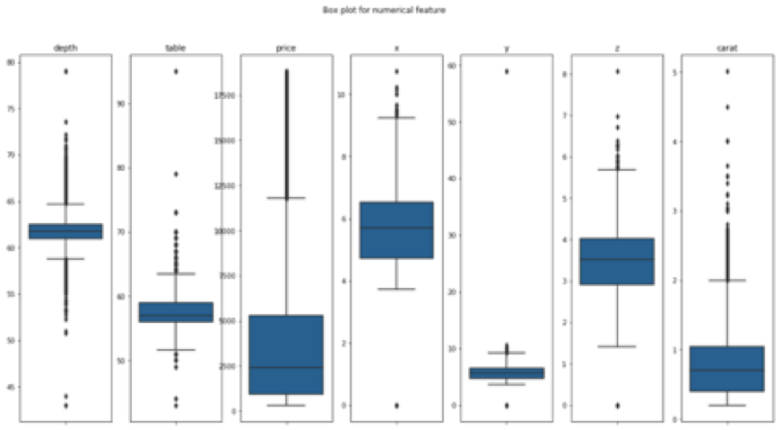

Figure 2: Box-plot of the 6 numerical feature and the response variable price.

rid of those anomalies that might lower the quality of our model, we simply filtered 17 samples with before mentioned traits. Notice that although a lot of diamonds has a price higher than 3 standard errors, we found it is reasonable because the price of commodities usually follows the power law distribution and there is no abnormal gap in the data. For categorical features, the bar plot was adapted to demonstrate the discrete distribution. Apparently, none of the three categorical features are well balanced. However, every minority class has at least 1000 observations, which means they are well represented and we cannot omit them in the analysis. Another interesting discovery that contradicts common sense is that most diamonds have a cut level better than Good, as is shown in Fig. 3. We suspect this phenomenon might be caused by overselling as there is no official rating for the cut of diamonds. Therefore, we may lower our expectation for the importance of this feature.

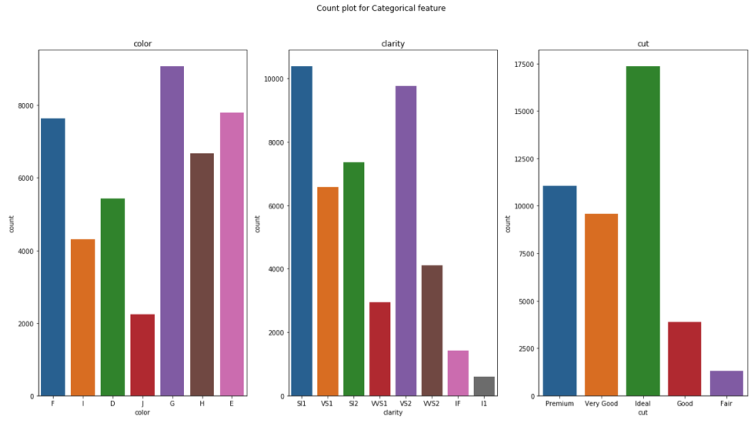

Figure 3: Bar plot of the 3 categorical feature.

After cleaning, categorical features in the dataset were encoded by one-hot encoding schema (reference). Figure. 4 shows the corresponding heat map for the Pearson correlation coefficient between any pair of features. In Fig. 4, a brighter color indicates a higher correlation. We can observe that the price, carat and features associated with the size of the diamonds are strongly correlated to each other. This is reasonable since usually the larger the diamond, the higher the price will be. Other attributes, such as color and clarity, seems to have much weaker correlations to the price. Hence, we may infer that the size/weight might be the most important features in determining the price among all other features.

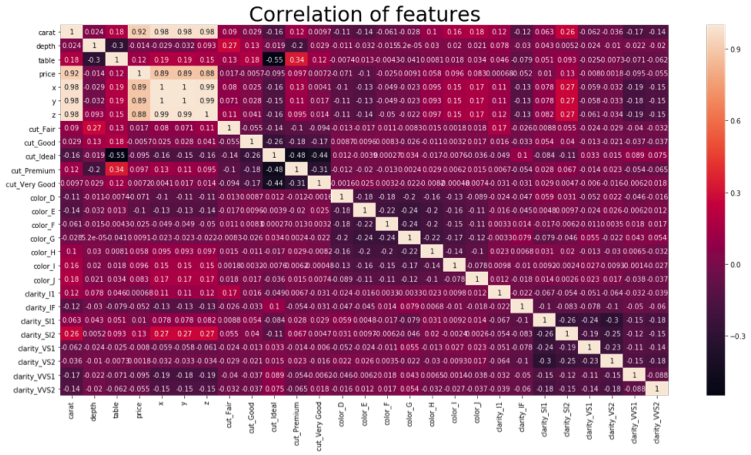

Figure 4: The Pearson correlation coefficient between any pair of features.

To have a better idea about how different attributes affect pricing, we plotted the relationship between price and carat, color, clarity. The relationship between carat and the price is not strictly linear, but closer to an exponential relationship, as is shown in Fig. 5. Besides, we found that the average

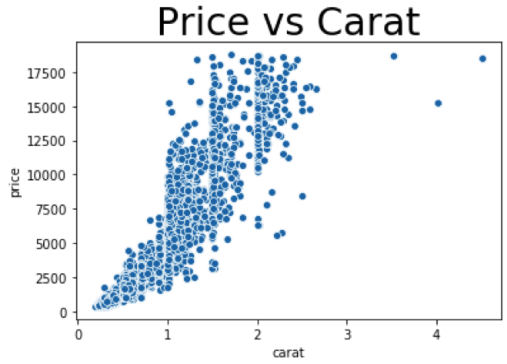

Figure 5: The relationship between price and Carat.

price between diamonds with different color can be up to 2000 dollars, as is shown in Fig. 6, which might mean a number of customers are willing to pay more for a better color. On the other side,

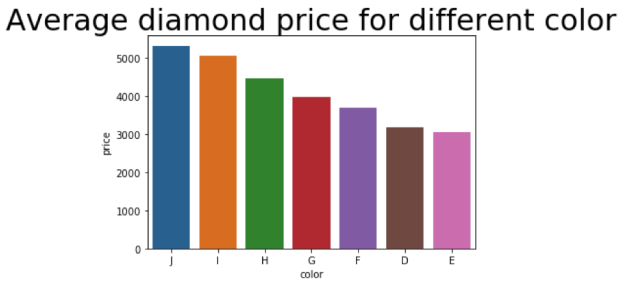

Figure 6: Average price of the diamond as a function of color.

the average price for diamonds with different cut and clarity is quite counterintuitive. In Fig. 7, the average price of fairly cut diamonds is even higher than ideal cut diamonds, and the average price of diamonds with the best level of clarity, IF level, is only 3/5 of the average price of diamonds with mediocre clarity level, SI2. Those seemingly strange behavior can be explained by the imbalance of

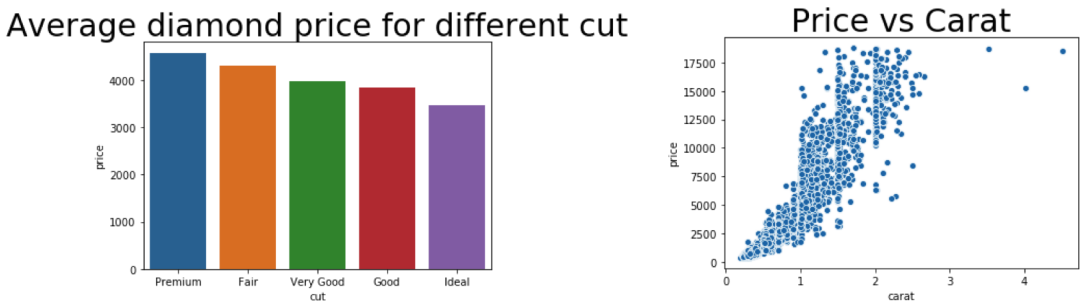

Figure 7: Diamond price for different cut and clarity.

the dataset.

The strikingly high correlation between feature x, y and z might also indicate the existence of collinearity, as shown in Fig. 8. Therefore, we can expect that those four features will have similar predictive power for the price.

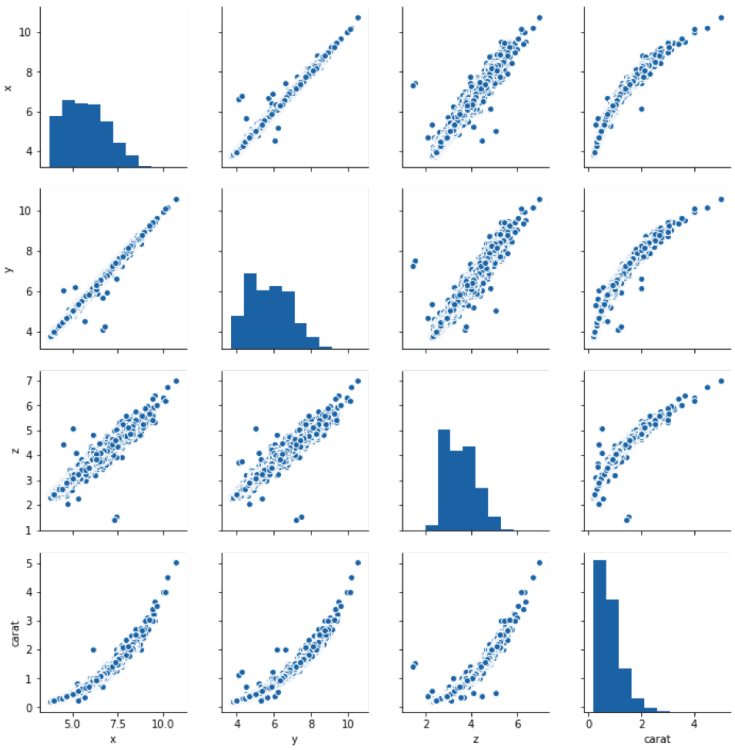

Figure 8: Correlation between the predictors x, y, and z.

3 Models and Performance

3.1 Linear model

As a baseline price prediction model for the *Diamond* data set, our team starts with basic linear regression on the data set. We can build the initial linear model simply by applying the linear model function (lm) in R on the training set. In this first step no additional change is made to the training set except for the general data preprocessing such as removing outliers and unused columns. To examine how well the model performs, we look at the summary (Fig. 9) and calculate mean squared error (MSE) on the test set.

```
Residual standard error: 1124 on 43110 degrees of freedom
Multiple R-squared:  0.9198,    Adjusted R-squared:  0.9197
F-statistic: 2.149e+04 on 23 and 43110 DF,  p-value: < 2.2e-16

>
> #MSE for baseline lineal model
> MSE_baseline = mean((predict(train.lm, test) - test$price)^2)
> MSE_baseline
[1] 1883173
```

Figure 9: Summary of the linear model.

According the summary, the adjusted R^2 is 0.9198, meaning 91.98% of the data variability can be explained by our basic linear model, and MSE is 1883173. These two statistics form the baseline of our model. We will improve our linear model in this section and use more advanced methods in the next few sections to achieve better prediction accuracy.

During our initial data analysis, we found strong collinearities between some dimensional features of the *Diamond* data, so our first improvement on the current linear model is to throw out unnecessary predictors with high collinearity scores. In this case we compute the variance inflation factor (VIF) for multiple times, and each time we remove the feature with the highest score until we get normal scores no higher than 5 or 10 for all the remaining features. This is an effective way to assess multicollinearity, and the methodology is introduced by Gareth James and his coauthors in An Introduction to Statistical Data Mining with Applications in R [3]. Based on the correlation check in RStudio, the dimensional features x, y, z and the feature carat are highly correlated ($> 98\%$) to each other. By applying *vif* function in R, we threw out x and z in the first two steps. Then in the third step we chose to remove y instead of carat despite carat has slightly higher VIF score. This is because removing carat would significantly reduce adjusted R^2 of the linear model to 84.61%, while removing y has only moderate effect on the model and adjusted R^2 would be 91.52%. After throwing out x, y, and z the remaining features end with normal VIF scores. Using the current model on the the test set, we gained an MSE of 1336384 which is considerably better than our original linear model.

In addition to wiping out undesirable collinearities, our team also considered interactions between features. We checked through all possible combinations of the two-feature interactions and found out that adding the interaction between carat and clarity leads to the best effect on the model, as is shown in Fig. 10. As shown above, the adjusted R^2 reached 92.89% and MSE was reduced to 1134352.

Finally, to see if this linear model can be further improved, we also attempted on ridge regression based our current best model with the interaction term. The summary is shown in Fig. 11. Since we have already removed unnecessary features and added the appropriate interaction effect in the previous steps, ridge regression does not provide further progress for our predictions and even has negative effect on the test error. The MSE went back up to 1502276, which is only better than the primitive

```
> train.lm.chosen =  lm(price~.-x-z-y+carat*clarity,train)
> summary(train.lm.chosen)

Residual standard error: 1058 on 43106 degrees of freedom
Multiple R-squared:  0.9289,    Adjusted R-squared:  0.9289
F-statistic: 2.087e+04 on 27 and 43106 DF,  p-value: < 2.2e-16

> #calculate MSE on test set for the model with chosen interaction terms
> MSE_interaction = mean((predict(train.lm.chosen, test) - test$price)^2)
> MSE_interaction
[1] 1134352
```

Figure 10: Summary of the linear model with the interaction term between carat and clarity included.

```
> # apply second ridge regression on the best lambda and calculate MSE
> ridge.mod = glmnet(x_ridge,y_ridge,alpha=0,lambda=bestlam)
> cv.out = cv.glmnet(x_ridge,y_ridge,alpha=0,nfolds=5)
> ridge.pred = predict(ridge.mod,s=bestlam,newx=x_ridge_test)
> MSE_ridge = mean((ridge.pred-test_ridge$price)^2)
> MSE_ridge
[1] 1502276
```

Figure 11: Summary of the ridge regression.

linear model. Therefore, if a linear model is designated for predicting diamond prices, our team would suggest removing x, y, z and add an interaction between carat and clarity to achieve best results.

3.2 Polynomial model

For the polynomial regression method, we use degrees up to 5. The reason for not going above 5 degrees is to prevent possible overfitting of the training data. Using k-fold validation, the MSE as a function of the polynomial degree is shown in Fig. 12, where the red cross indicates the polynomial degree that is corresponding to the lowest MSE. We see that as the degree of the polynomial increases, the MSE firstly decreases and then increases. The minimum MSE is obtained at the degree of 4. Part of the summary of the polynomial fit using the degree 4 is shown in Fig. 1. We see that the R^2 is ~ 0.93 for the polynomial model with the degree 4.

We then performed another polynomial fit with degree up to 5 using only the predictors *carat*,

Table 1: Part of the summary of the polynomial fit using the degree 4

residual standard error: 1026 on 43092 degrees of freedom	
Mulitple R-squared: 0.9332	Adjusted R-squared: 0.9331
F-statistic: 1.468e+4 on 41 and 43092 DF	p-value: $\leq$ 2.2e-16

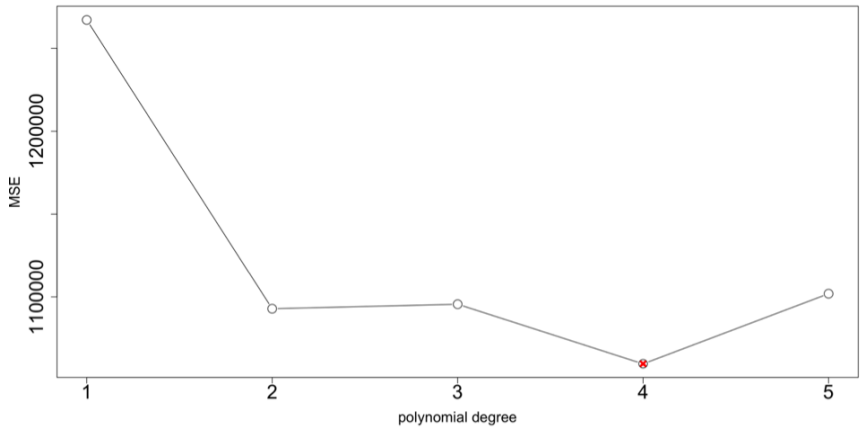

Figure 12: MSE as a function of the degree used for the polynomial model. The red cross indicates that the model using a degree of 4 yields the lowest MSE.

y, and *clarity*. This is because these three predictors are the predictors given by a decision tree model. We find that again the best polynomial fit is the one with degree 4. And the MSE and R^2 is similar with the model using all predictors.

3.3 Random Forests

As one of the most representative tree based methods, random forest usually outperforms linear model in terms of accuracy in large sample learning. The exploratory data analysis has shown that the true model is non-linear. A non-parametric model such as random forest should be flexible enough to capture more correlation and archive lower MSE. To get the best out of our model, we tried random forest regressor with randomized search cross validation to tune hyper-parameters.

Instead of using a single tree to divide the sample space, random forest grows a number of trees. Each tree is trained by using a random subset of the original observations and predictors to remove correlation between trees. The final prediction is decided by the majority vote. This divide and vote strategy effectively increase the accuracy with the expense of interpretability.

The most essential hyper-parameter for random forest is the number of trees to grow and the number of random predictors each tree should see. In the tuning process, we predefined a possible range for each parameter and used a randomized schema to search the parameter space. The popular python

package Scikit-learn provides consistent API and concurrency support for both random forest and randomized parameter search. Our code runs on a 4 core Intel Core i5 machine took 5 mins to iterate 100 combination of parameters.

The best model used 398 trees with max depth of 40 and max feature number of 9. The overall out-of-bag R square is 0.98, outperformed linear methods, as expected.

The feature importance plot, as is shown Fig. 13, also validates some intuition we obtained at the

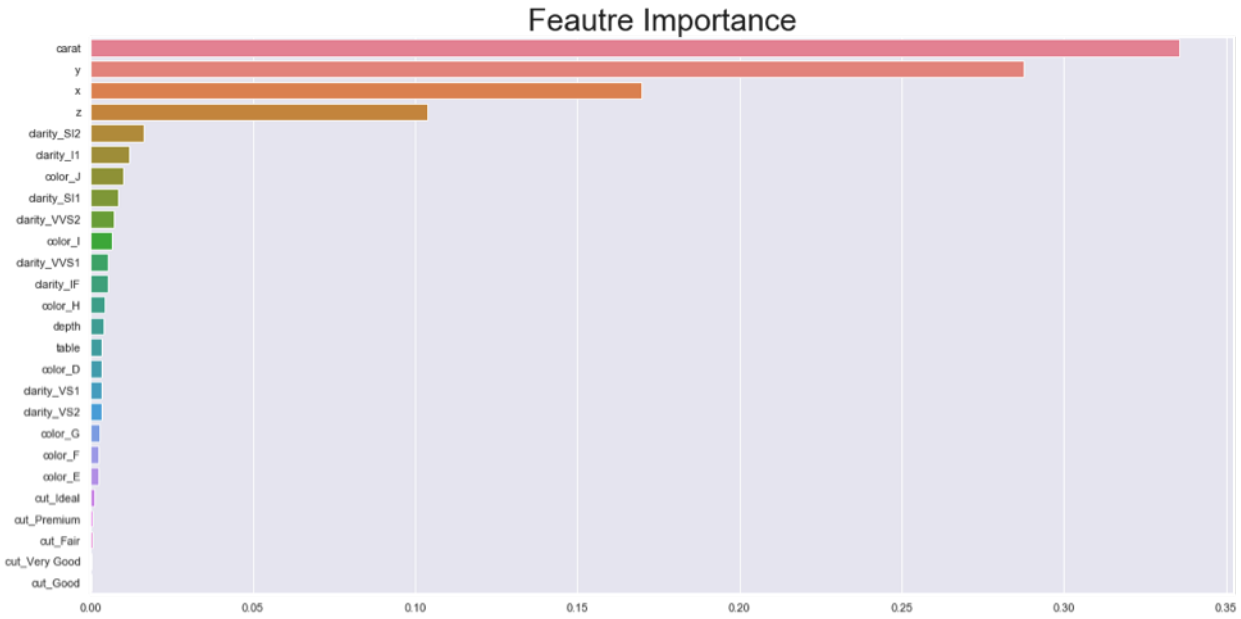

Figure 13: Feature importance plot of the random forest model.

exploratory data analysis stage. Carat is the most important feature, contributing around 34% percent of prediction power, followed by three highly correlated feature about the size of a diamond. Clarity, color and cut can help, but only to a limited extend.

3.4 XGBoost

XGBoost, short for extreme gradient boosting, has become the ultimate choice for tough predictive modeling problems for its power to deal with data irregularities. We implemented an XGBoost regression model for predicting diamond prices. Similar to the way we tuned random forests, we optimized the XGBoost regressor's hyperparameters with randomized search cross validation.

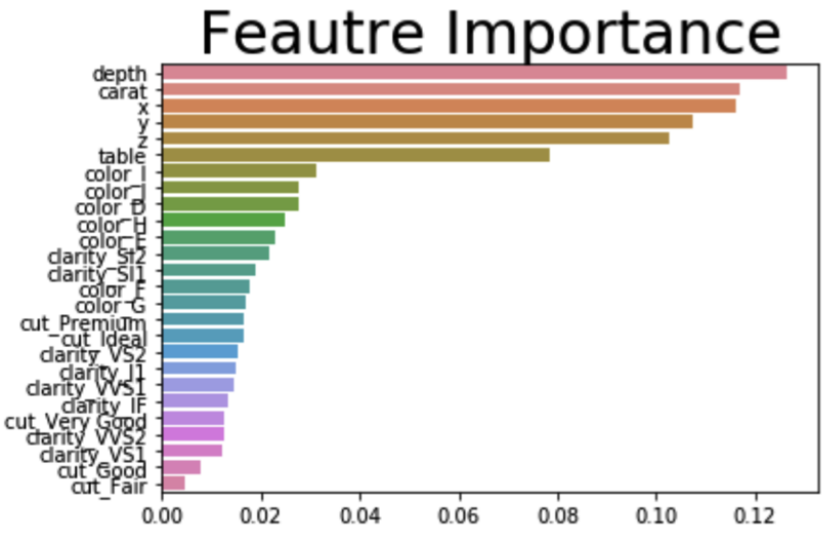

Figure 14: Feature importance obtained from the XGBoost model.

We referred to the parameter tuning procedure introduced in [4] to select the model that produces the highest cross-validation R^2 score. First, we fixed the learning rate and found the optimal number of trees using cross-validation. Then, we tuned tree-based parameters such as `max_depth` and `min_child_weight` by finding the optimal parameters within a range through randomized search cross validation. Later, we tuned subsample and `colsample_bytree`, regularization parameters, and learning rate in a similar way.

As a result, our optimal XGBoost regression model gave cross-validation R^2 score of 0.98204, and a test accuracy of xxx? , which outperformed the other models we have tried. The feature importance produced by XGBoost is displayed in Fig. 14 below, where depth, carat, and dimensional sizes play important roles in the prediction of diamond prices.

4 Summary

To summarize, we analyzed the given *diamond* data set from Kaggle and found that the carat of a diamond is highly correlated with its size. Furthermore, we find that the average price of fairly cut diamond is even higher than that of ideal cut. Using statistical learning methods including linear regression, polynomial model, random forests, and XGBoost model, we have predicted the price of a diamond using the given predictors to a high accuracy. Our study and model can find its application not only for ordinary diamond customers but also for related industries to set a price for diamonds in

a scientific and systematic way.

References

[1] Wikipedia. Diamonds as an investment.

[2] shivamagrawal in Kaggle. Original diamond data set from Kaggle.

[3] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[4] Aarshay Jain. Complete Guide to Parameter Tuning in XGBoost (with codes in Python). 2016.