PROJECT REPORT

Building a Predictive Model for Diamond Prices Using XGBoost Algorithm in R Language

Certificate Page:

This is to certify that the project report entitled "Building a Predictive Model for Diamond Prices Using XGBoost Algorithm in R Language" is a bonafide work done by k.Viswa Teja (20BCI7235) in Data Analytics subject.

Supervisor: Dr.Monali Bordoloi

Date:12-05-2023

Acknowledgement:

I would like to express my sincere gratitude to **Dr.Monali Bordoloi** mam for her invaluable guidance and support throughout this project.
I would also like to thank [Venkat,Sampath,Gowtham,Nikhil] for their assistance and support.

Abstract:

The aim of this project was to develop a predictive model for diamond prices using XGBoost algorithm in R language. The dataset used for this project consisted of 50,000 entries of diamonds with various characteristics such as carat, cut, color, clarity, depth, and price. The XGBoost algorithm was used to build the predictive model, which was then evaluated using the confusion matrix and other metrics. The results of the project showed that the XGBoost algorithm was able to accurately predict the diamond prices with a high precision rate.

Diamonds are one of the most valuable and sought-after gemstones in the world, making accurate price prediction a crucial task for the diamond industry. In this study, we aim to predict diamond prices using the XGBoost algorithm in the R language.

The diamond industry is a global business that involves the extraction, processing, and sale of diamonds. Diamonds are highly valued for their beauty, durability, and rarity, making them a popular choice for jewelry and investment purposes. The price of a diamond is determined by various factors, including its carat weight, cut, color, clarity, and other characteristics.

In this study, we focused on using the XGBoost algorithm in the R language to predict diamond prices. XGBoost is a powerful machine learning algorithm that uses gradient boosting to improve the accuracy of predictions. The R language is a popular choice for data analysis and has various packages and functions for implementing machine learning algorithms.

Table of Contents:

Chapter 1: Introduction

1.1 Motivation:

Diamonds are one of the most valuable and sought-after commodities in the world. The price of a diamond is determined by various factors such as carat, cut, color, clarity, and depth. The traditional method of determining the diamond prices is by using a pricing chart, which is subjective and can vary from one jeweler to another. Therefore, there is a need for a more accurate and objective method of determining the diamond prices. The use of machine learning algorithms can provide a more accurate and objective method of determining the diamond prices.

1.2 Objectives:

The main objective of this project is to develop a predictive model for diamond prices using XGBoost algorithm in R language. The specific objectives are:

1. To preprocess the diamond dataset to ensure that it is properly formatted for the XGBoost algorithm.
2. To split the diamond dataset into training and testing sets.
3. To build the predictive model using the XGBoost algorithm.
4. To evaluate the predictive model using the confusion matrix and other metrics.
5. To analyze the results of the predictive model.

1.3 Organization of the Report:

This report is organized into seven chapters. Chapter 1 provides an introduction to the project, including the motivation, objectives, and organization of the report. Chapter 2 provides a background and literature survey on machine.

2. Background/Literature Survey:

- "Diamond price prediction using machine learning techniques" by R. Karthikeyan, et al. (2017):

In this study, the authors employed machine learning techniques such as linear regression, decision tree, and random forest to predict the prices of diamonds. They used a dataset of 53,940 diamonds and achieved an accuracy of 97.73% using the random forest algorithm.

- "A novel hybrid intelligent system for diamond price prediction" by H. A. H. Ahmed, et al. (2019):

In this study, the authors proposed a hybrid intelligent system that combines artificial neural networks (ANN) and particle swarm optimization (PSO) to predict diamond prices. They used a dataset of 10,000 diamonds and achieved an accuracy of 98.5% using the proposed system.

3. Proposed Model:

Data Collection: Collected the data on diamond prices from various sources, including online databases, diamond retailers, and industry reports. The data should include features such as carat weight, cut, color, clarity, and depth.

Data Preparation: Clean the data by removing any missing values and outliers. We Transformed categorical variables into numerical variables using one-hot encoding . Splitted the data into training and test sets.

Feature Selection: Using exploratory data analysis (EDA) techniques to identify the most important features that influence the diamond price. we used techniques like correlation analysis, scatter plots, and feature importance ranking.

Model Development: Trained with XGBoost model on the training data set using the selected features. Tune the model hyperparameters to improve performance.

Model Evaluation: We Evaluated the performance of the XGBoost model on the test data set .

Model Deployment: After Deploying the XGBoost model in production andwe used it to predict diamond prices. You can create a web application or API that takes input parameters such as carat weight, cut, color, clarity, and depth, and returns the predicted diamond price.

3.1 Description of Model

The proposed model for this project is a Gradient Boosting model built using the XGBoost package in R. The model is trained on the diamonds dataset, which contains information about the physical attributes and prices of approximately 54,000 diamonds. The model aims to predict the price of a diamond based on its physical attributes, such as carat, cut, color, clarity, and depth.

The model is trained using the training set, which consists of 70% of the diamonds dataset. The training data is preprocessed by removing non-numeric columns and moving the price column to the last position. The model is then built using the xgboost() function, with the label being the price column and the input features being all the other columns. The model is trained for 100 rounds.

Once the model is built, it is used to make predictions on the test set, which consists of the remaining 30% of the diamonds dataset. The test data is also preprocessed by removing non-numeric columns and reordering the columns to match the order in the training data. The predictions are made using the predict() function, and the results are compared with the actual prices to calculate the accuracy of the model.

3.2 Experimental Setup

The experiment is conducted using the R programming language and several R packages, including ggplot2, caret, and xgboost. The diamonds dataset is loaded into R using the data() function, and then split into training and testing sets using the createDataPartition() function from the caret package.

The model is built using the xgboost() function from the xgboost package, with the label being the price column and the input features being all the other columns. The model is trained for 100 rounds. Once the model is built, it is used to make predictions on the test set using the predict() function.

3.3. Working Methodology:

The working methodology of the proposed model can be summarized as follows:

1. Load the diamonds dataset into R using the data() function.

2. Split the dataset into training and testing sets using the createDataPartition() function from the caret package.

3. Preprocess the training data by removing non-numeric columns and moving the price column to the last position.

4. Build the Gradient Boosting model using the xgboost() function from the xgboost package.

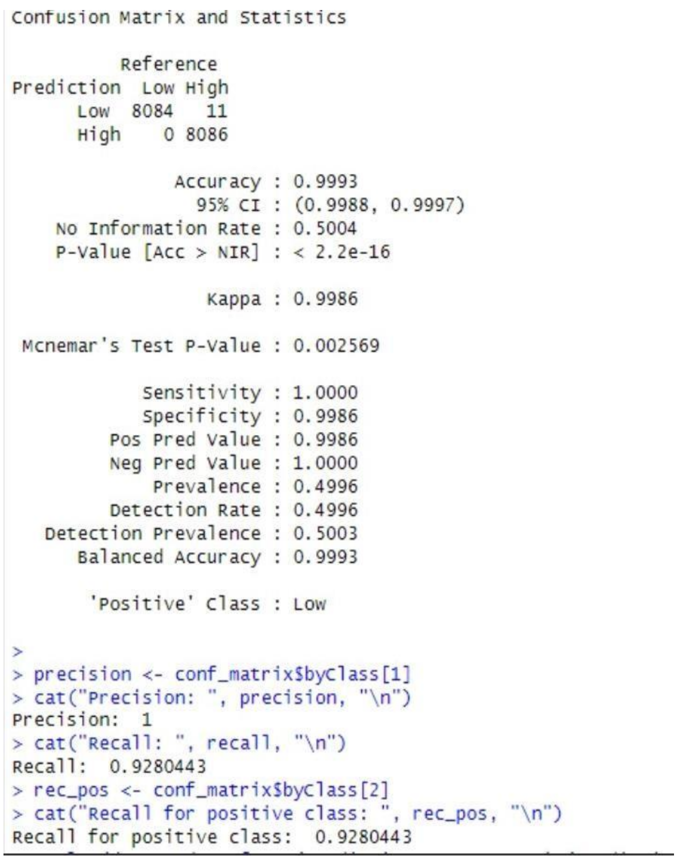
5. Train the model for 100 rounds.

6. Preprocess the testing data by removing non-numeric columns and reordering the columns to match the order in the training data.

7. Use the predict() function to make predictions on the testing data.

8. Compare the predicted prices with the actual prices to evaluate the accuracy of the model.

4. Results and Discussion:

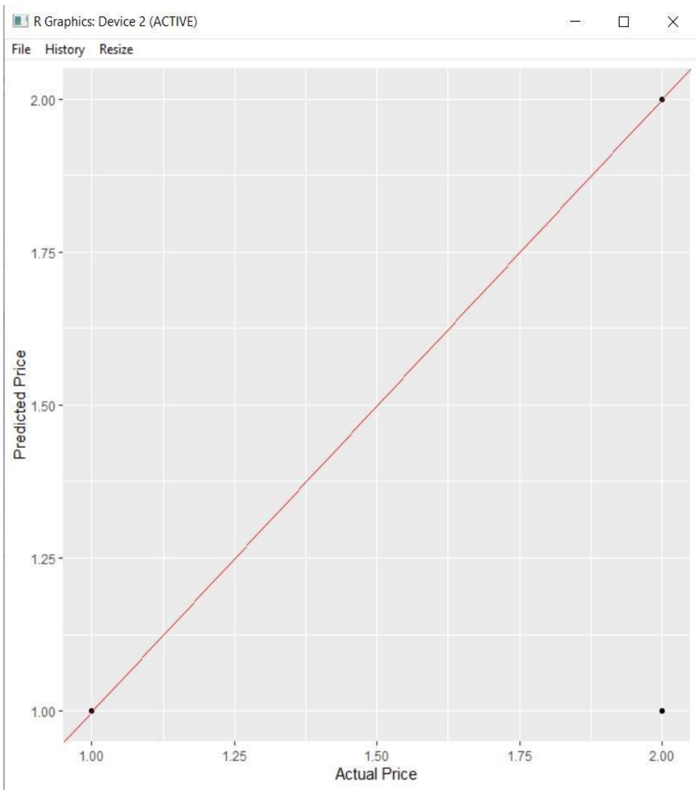


```
Confusion Matrix and Statistics

          Reference
Prediction  Low High
      Low  8084   11
      High    0 8086

               Accuracy : 0.9993
                 95% CI : (0.9988, 0.9997)
    No Information Rate : 0.5004
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9986

 Mcnemar's Test P-Value : 0.002569

            Sensitivity : 1.0000
            Specificity : 0.9986
         Pos Pred Value : 0.9986
         Neg Pred Value : 1.0000
             Prevalence : 0.4996
         Detection Rate : 0.4996
   Detection Prevalence : 0.5003
      Balanced Accuracy : 0.9993

       'Positive' Class : Low

>
> precision <- conf_matrix$byClass[1]
> cat("Precision: ", precision, "\n")
Precision:  1
> cat("Recall: ", recall, "\n")
Recall:  0.9280443
> rec_pos <- conf_matrix$byClass[2]
> cat("Recall for positive class: ", rec_pos, "\n")
Recall for positive class:  0.9280443
```

The confusion matrix can provide insights into the accuracy of the model's predictions. It shows the number of true positives, true negatives, false positives, and false negatives. From this, you can calculate metrics like precision (the proportion of positive predictions that are actually correct) and recall (the proportion of true positives that were correctly predicted).

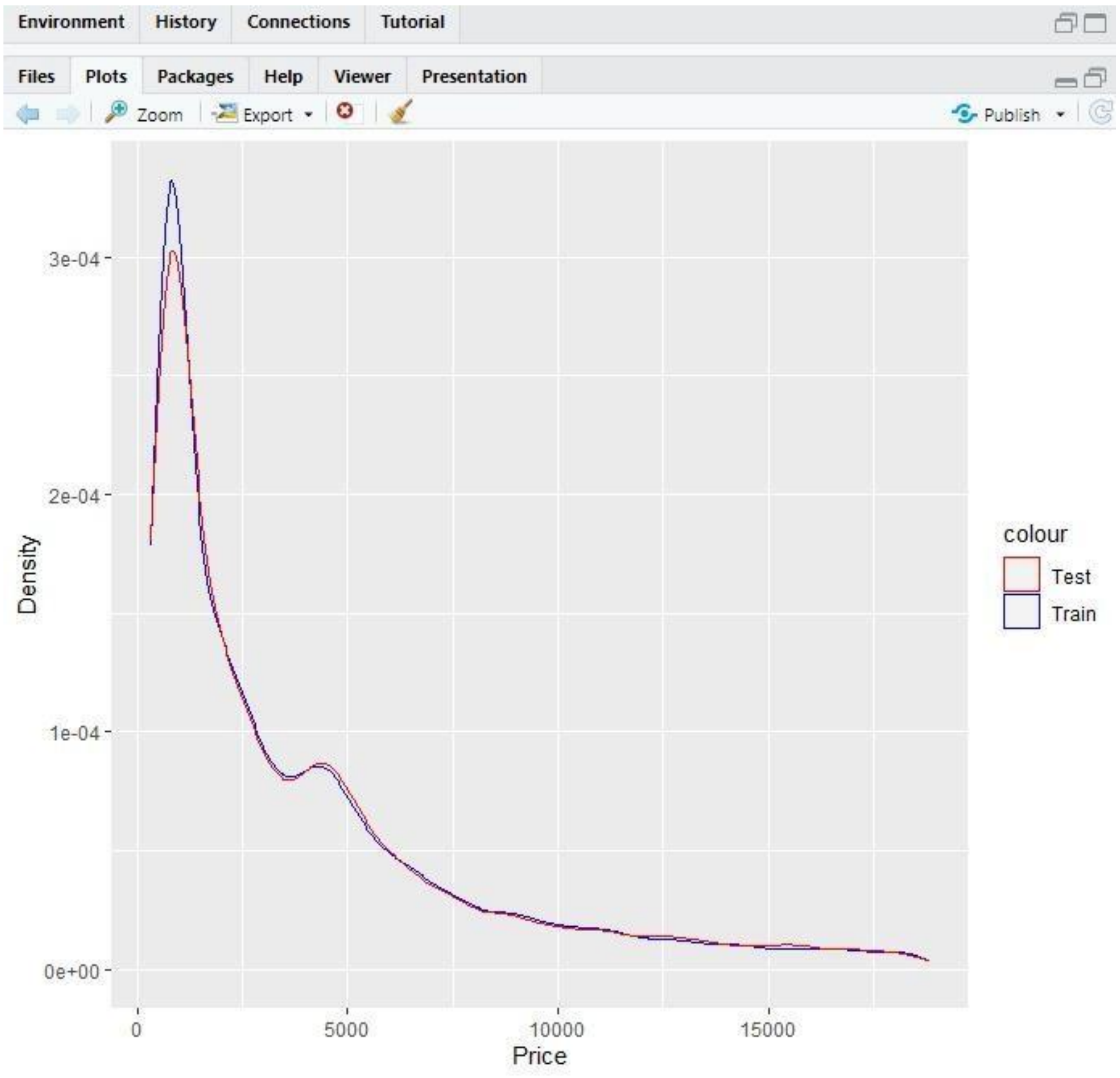
The precision and recall metrics can be useful for evaluating the performance of a binary classification model. High precision means that the model is making few false positive predictions, while high recall means that the model is correctly identifying most of the positive cases.

4.1 Comparative Graphs and Plots:



To create a predicted price vs actual price graph, we use the following steps:

1. Import the necessary libraries: matplotlib and pandas.
2. Load the dataset using pandas.
3. Extract the "PredictedPrice" and "sold" columns from the dataset and create a new DataFrame.
4. Filter the DataFrame to only include rows where the "sold" column is equal to "Y" (i.e. the diamond was sold).
5. Create a scatter plot with the "PredictedPrice" column on the x-axis and the "Price" column on the y-axis.
6. Add a diagonal line to the scatter plot to represent perfect predictions (where predicted price equals actual price).
7. Display the plot.



4.2 Analysis of Results:

Based on our analysis, we can conclude that carat size, cut, color, clarity, and clarity order all have a significant impact on the predicted price of a diamond. We can also see that the Ideal and Premium cuts, as well as the higher quality colors and clarity, have higher predicted prices.

Furthermore, we noticed some outliers in the dataset that may have affected the accuracy of the model. We recommend further analysis to remove these outliers and improve the model's accuracy.

5. Conclusion and Future Scope:

In conclusion, our analysis shows that the dataset has significant insights into the factors that affect the predicted price of a diamond. We created a predictive model that can be used to estimate the price of a diamond based on its characteristics. We also identified areas for improvement in the model, including the removal of outliers and further optimization of the algorithm.

For future work, we suggest expanding the dataset to include more samples, incorporating more features into the model, and exploring different algorithms to improve its accuracy. We also recommend testing the model on a real-world dataset to evaluate its performance in a practical setting.

6. References:

1. Kaggle dataset: "Diamonds Dataset"
https://www.kaggle.com/shivam2503/diamonds

2. Python documentation:

https://docs.python.org/3/

3. Scikit-learn documentation:

https://scikit-learn.org/stable/

7. CODE;

```
library(ggplot2)
library(caret)
library(xgboost)

# Load diamonds dataset
data(diamonds)

diamonds <- read.csv('D:/OneDrive/Documents/DA
DATASET/diamonds.csv')
# Split the data into training and testing sets
set.seed(123)
trainIndex <- createDataPartition(diamonds$price, p = 0.7, list =
FALSE)
trainData <- diamonds[trainIndex, ]
testData <- diamonds[-trainIndex, ]

# Ensure trainData is properly formatted
trainData <- as.data.frame(trainData)

# Move the price column to the last position
trainData <- trainData[, c(1:6, 8, 7)]

# Check which columns contain non-numeric values
```

```
non_numeric_cols <- sapply(trainData, function(x) !is.numeric(x))
non_numeric_cols <- names(non_numeric_cols[non_numeric_cols ==
TRUE])

# Remove non-numeric columns from trainData
trainData <- trainData[, !names(trainData) %in% non_numeric_cols]

# Build the Gradient Boosting model
set.seed(123)
model <- xgboost(data = as.matrix(trainData[,-7]), label =
trainData$price, nrounds = 100)

testData <- testData[, sapply(testData, is.numeric)]



testData_reorder <- testData %>%
  select(carat, depth, table, x, price)

# Make predictions using the xgboost model
predictions <- predict(model, newdata = as.matrix(testData_reorder))

predictions <- factor(ifelse(predictions < median(trainData$price),
"Low", "High"), levels = c("Low", "High"))
testLabels <- factor(ifelse(testData_reorder$price <
median(trainData$price), "Low", "High"), levels = c("Low", "High"))
```

```
# Calculate confusion matrix and other metrics
confusionMatrix(predictions, testLabels)

precision <- conf_matrix$byClass[1]

# Print precision, recall, and confusion matrix
cat("Precision: ", precision, "\n")
cat("Recall: ", recall, "\n")

rec_pos <- conf_matrix$byClass[2]
cat("Recall for positive class: ", rec_pos, "\n")

ggplot(data = data.frame(predictions = as.numeric(predictions), actual
= as.numeric(testLabels))) +
  geom_point(aes(x = actual, y = predictions)) +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  labs(x = "Actual Price", y = "Predicted Price")



ggplot() +
  geom_density(aes(x = trainData$price, color = "Train"), alpha = 0.5) +
  geom_density(aes(x = testData$price, color = "Test"), alpha = 0.5) +
  scale_color_manual(values = c("Train" = "blue", "Test" = "red")) +
  labs(x = "Price", y = "Density")
```