VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

PROJECT REPORT

DIAMOND PRICE PREDICTION

Machine Learning

(CSE4036)

SLOT-G1

Team Members

20MIA1031 - Sanjay.M

20MIA1039 - Sriganth.R

20MIA1117 - Pilaram Manoj

Email:

sanjay.m2020a@vitstudent.ac.in

sriganth.r2020@vitstudent.ac.in

pillaram.manoj2020@vitstudent.ac.in

Phone number:

Sanjay.M- 7993169693

Sriganth.R- 7812817472

Pilaram Manoj- 9392022271

TABLE OF CONTENTS

Abstract:

Precious metals like diamond are in high demand due to their financial rewards. Diamond is one of the most powerful and valuable naturally occurring carbon compounds. However, unlike gold and silver, determining the price of a diamond is extremely difficult because many factors must be considered. In order to forecast diamond prices quickly and accurately, a variety of techniques are required. Using a variety of regression techniques, including linear regression, KNN, decision trees and random forests, we hope to forecast the prices of diamonds in this report. The goal of this project is to develop the most efficient algorithm for predicting diamond prices and also we are going to create user interface to know the price of diamonds.

Keywords:

Diamonds, Price, Prediction, Algorithms and User interface.

Introduction:

This project shows how diamond price can be predicted based on their cut, colour, clarity & other attributes. However, the prediction process is difficult due to the wide variation in the diamond stones sizes and characteristics. several machine learning algorithms are used to help in predicting diamond price.

Diamonds are one of the world's most valuable gems. It is also one of the most expensive gems, so its price is extremely volatile. The value of a diamond is determined by its structure, cut, inclusions (impurities), carats, and a variety of other factors. Diamonds have numerous applications, including those in industry, where they are effective in cutting, polishing, and drilling.

Diamonds are extremely valuable, so they have been traded across borders for centuries, and this trade is only growing. Color, cut, clarity, and carat and other attributes are used to grade and certify diamonds. Color, clarity, cut, and carat weight are all important factors to consider. These are the only metrics used to determine the quality of diamonds and the price of the diamond. This metric allows people all over the world to buy diamonds with a common understanding, facilitating trade and providing a fair price for what is purchased.

The idea that carat and price have a strong correlation is the more obvious one, but it has been noticed that this trend no longer seems to hold true, which has increased the volatility of the diamond price. This machine learning model analyses more than 4 features, which enables it to deliver a result that is more precise.Plotting the price chart results in a number of formations, including bottoms and tops. These formations are frequently used in the diamond market, among other trading markets, as well as the currency markets. The applications used here are Jupyter Notebook (also known as Anaconda Navigator), NumPy, a package for array processing, Pandas, a tool for data manipulation, Scikit Learn, a library for machine learning and Seaborn, tools for data visualization.

Background study:

Diamonds are one of nature's wonders. They are specially valued for their rarity, their optical properties and because they represent wealth in its most concentrated form (Cardoso & Chambel,2005) Although cut diamond unit prices may be influenced by multiple factors, carat, colour, clarity and cut, commonly known in the trade as the 4C's, are the best-known diamond properties used in these gems' valuation (Liddicoat, 1993). Diamonds differ from semi-precious metals which have a fixed price per gram that fluctuates with demand. Originally valued for their weight, or 'carat', many other qualities play into diamond values today (Stier-Moses& Zeevi,2008).

In his 1983 study, Verma, Manohar Lall, found that the diamond exporters and the diamond industry had a number of issues, the import of rough diamonds (raw materials) being one of the most significant. He found that this industry had a number of issues because the cost of raw materials was so high, as was the cost of finished goods. He also noted that this industry relied on labour incentives and made use of at least modern machinery and technology.

He also noted that there was a constant lack of skilled labour because India lacks professional schools or training facilities for this industry. He also noted that diamonds are one of the nation's top exports and a major source of foreign exchange received through remittances from abroad. According to his analysis of the theses, during the five years from 1979 to 1983, the percentage of diamonds imported by the USA from India increased to 48%, while the percentage of diamonds imported by Japan from India rose to 50%.

Purushothaman, Nair C. N. (1992): They highlighted the potential exports as well as the diamond industry's potential problems. He stated that the diamond industry is considered a traditional industry group, and that the diamond industry in India has made larger contributions to our export earnings year after year in value terms, beginning with Rs. 44.80 crores in 1970-1971 and rising to 618.40 crores in 1980-1981.

He went on to say that neither India had a source for diamonds nor did it have a balanced budget for its consumption. Only the diamond-processing industry makes India famous. He discussed a number of issues facing the diamond industry, including the supply of low-quality stones, inadequate bank facilities, the labour issue, and arbitrary pricing, with particular reference to Kerala. He concluded by proposing some appropriate policy implications to address the issues the diamond industry has encountered.

Sevdermish, Menahem, Alan R. Miciak, and Levinson6 (1998): He said that jewellery industry is significantly impacted by the diamond cutting sector. They discovered that the weight of polished diamonds increased by about 85% and their value by about 250%, respectively, in the modern era. At the moment, India controls over 90% of polished diamonds and about 10% of its waste.

Amita Mital (2008): She has analyzed issues pertaining to the competition that is present as well as issues pertaining to the demand and supply conditions for India's gems and jewellery products. To encourage the export of gems and jewellery products, which are also covered in her study, the government has taken steps to create regulations.

The demand for gems and jewellery is influenced by a number of variables, including economic growth, employment levels, income levels, saving levels, tax rates, banking facilities, credit availability, and competition based primarily on the carat weight, quality, quantity, availability of designs, shapes, and pricing of the product. She also analyzed how some individuals hold diamonds and jewellery as investments and do so for extended periods of time.

Alok Kala (2009): He talked about how the Indian gem and jewellery industry performed in terms of exports for the fiscal year 2008–09. However, this industry saw overall growth, primarily as a result of rising prices for both raw materials and finished goods. On the other hand, the study shows that exports of cut and polished diamonds and coloured gem stones have decreased, but demand for gold is growing due to the belief that it is a significant form of savings.

Methodology:

- We have considered the classic Diamonds dataset (from Kaggle) which contains the prices and other attributes of almost 54,000 diamonds.
- We will be conducting the following steps
    - Import Required Packages
    - Load the dataset
    - Perform the exploratory data analysis (EDA)
    - Prepare the dataset for training
    - Create Linear regression, KNN, Decision tree and Random Forest model.
    - Train the model to fit the data
    - Make predictions using the trained model.

Implementation:

The meaning of each column in the data is as below:

price: The price of the Diamond

carat: The carat value of the Diamond

cut: The cut type of the Diamond, it determines the shine

color: The color value of the Diamond

clarity: The carat type of the Diamond

depth: The depth value of the Diamond

table: Flat facet on its surface the large, flat surface facet that you can see when you look at the diamond from above.

x: Width of the diamond

y: Length of the diamond

z: Height of the diamond

The Software used here are Jupyter Notebook (also known as Anaconda Navigator), NumPy, a package for array processing, Pandas, a tool for data manipulation, Scikit Learn, a library for machine learning and Seaborn, tools for data visualization. import missingno as msn is used to find whether missing values is there or not in a dataset.

For user interface, Tkinter is the standard GUI library and the combination of Python and Tkinter makes it quick and simple to develop GUI applications. An effective object-oriented interface for the Tk GUI toolkit is provided by Tkinter. Tkinter module import.

A Python UI library called CustomTkinter, which is based on Tkinter, offers fresh, contemporary widgets that can be completely customised. Like standard Tkinter widgets, they can be made, used, and combined with other standard Tkinter elements.we have used prediction() module, It will predict the label of a new set of data.

Results and Discussion:

Below we displayed our dataset. It consists of 53940 rows and 11 columns and after that we have checked whether missing values is there or not. For this dataset has no missing values and we have described the dataset.

DIAMOND PRICE PREDICTION

Machine Learning
(CSE4036)

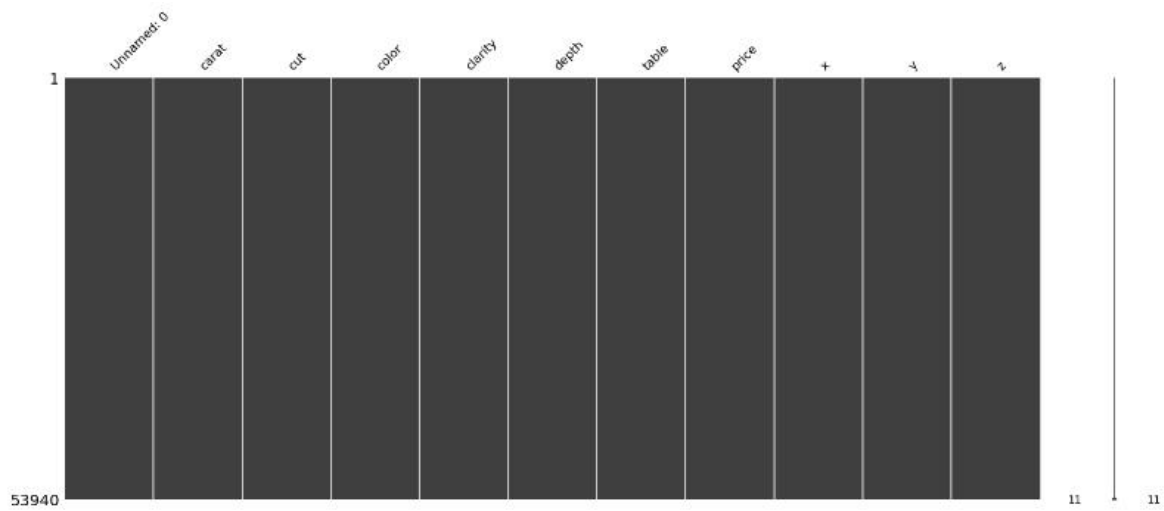
In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
data = pd.read_csv("G:\\VIT SEM\\vit sems\\COURSES\\5th sem\\ml\\project\\diamonds (1).csv")
display(data.head(5))
```

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

In [3]:
```
import missingno as msn
print(data.isnull().sum())
msn.matrix(data)
plt.show()
print(data.shape)
```

```
Unnamed: 0    0
carat         0
cut           0
color         0
clarity       0
depth         0
table         0
price         0
x             0
y             0
z             0
dtype: int64
```



(53940, 11)

In [4]:
```
data.describe()
```

Out[4]:

	Unnamed: 0	carat	depth	table	price	x	y	z
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000
mean	26970.500000	0.797940	61.749405	57.457184	3932.799722	5.731157	5.734526	3.538734
std	15571.281097	0.474011	1.432621	2.234491	3989.439738	1.121761	1.142135	0.705699
min	1.000000	0.200000	43.000000	43.000000	326.000000	0.000000	0.000000	0.000000
25%	13485.750000	0.400000	61.000000	56.000000	950.000000	4.710000	4.720000	2.910000
50%	26970.500000	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	40455.250000	1.040000	62.500000	59.000000	5324.250000	6.540000	6.540000	4.040000
max	53940.000000	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000

In [5]:
```
data = data.drop(["Unnamed: 0"], axis=1)
data.head(5)
```

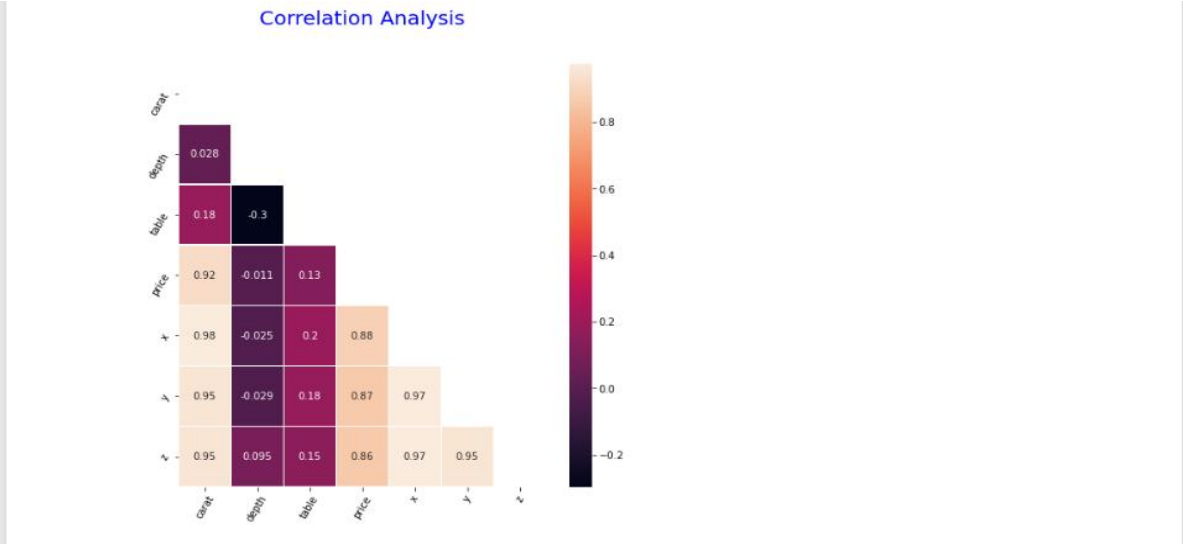
Out[5]:

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

In [6]:
```
s = (data.dtypes =="object")
object_cols = list(s[s].index)
print("Categorical variables:")
print(object_cols)
```

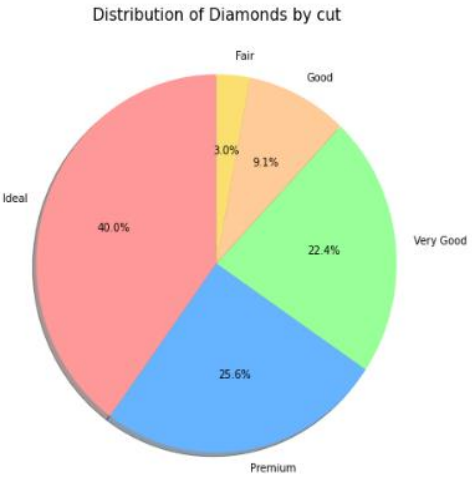
```
Categorical variables:
['cut', 'color', 'clarity']
```

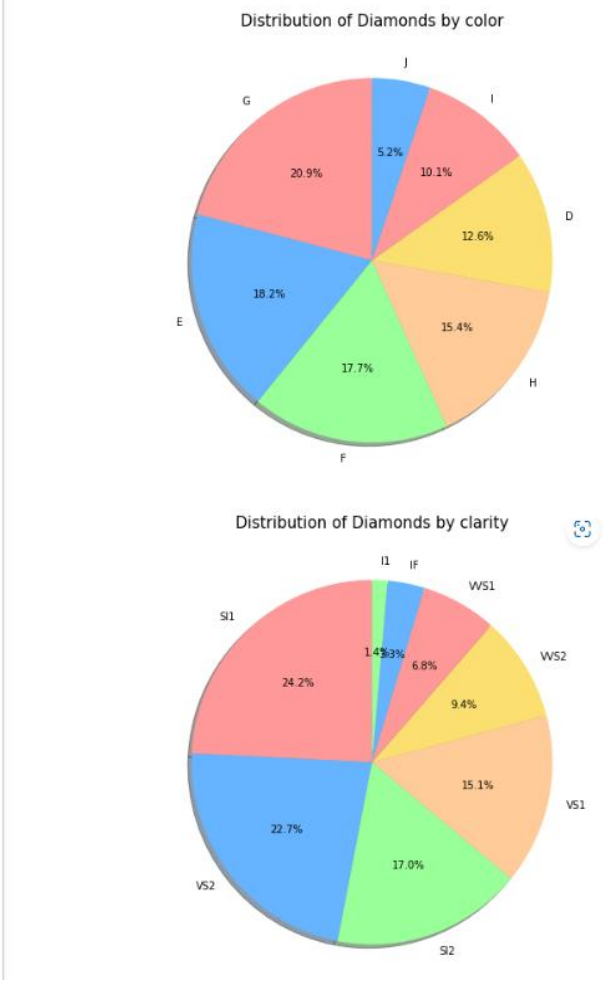
In [7]:
```
import seaborn as sns
corr=data.corr()
plt.figure(figsize=(8,8))
plt.title('Correlation Analysis',color='blue',fontsize=20,pad=40)
mask = np.triu(np.ones_like(corr, dtype = bool))
sns.heatmap(data.corr(), mask=mask, annot=True, linewidths=.5);
plt.xticks(rotation=60)
plt.yticks(rotation = 60)
plt.show()
```



Correlation Analysis: It is used here to see the Strength of the relationship between each feature or variable. By seeing the plot we can say that if carat increases, price value of diamond will also increases.

```
In [8]: def pie_plot(data, col):
            labels = data[col].value_counts().index
            sizes = data[col].value_counts().values
            colors = ['#ff9999','#66b3ff','#99ff99','#ffcc99','#fbdf70']

            plt.figure(figsize = (8,8))
            plt.pie(sizes, labels=labels, shadow = True, startangle=90, colors=colors, autopct='%1.1f%%')
            title=f"Distribution of Diamonds by {col}"
            plt.title(title, color = 'black', fontsize = 15)
        for col in object_cols:
            pie_plot(data, col)
```





The First Pie chart describes about the Distribution of Diamonds by clarity column.In this clarity column, the S11 is having the highest percentage of 24.2% and least is I1 with 1.4%.

The Second Pie chart describes about the Distribution of Diamonds by color column. In this color column, 'G'color has highest percentage 20.9% and the color 'J' has less percentage i.e 5.2% when we compare to other colors.

The Third Pie chart describes about the Distribution of Diamonds by cut column. In this cut column, the ideal shows more percentage(40.0%) compare to other cut and Fair is having less percentage(3%) compare to other cut.

In [9]:
```
linear_vars = data.select_dtypes(include=[np.number]).columns
display(list(linear_vars))
```

```
['carat', 'depth', 'table', 'price', 'x', 'y', 'z']
```

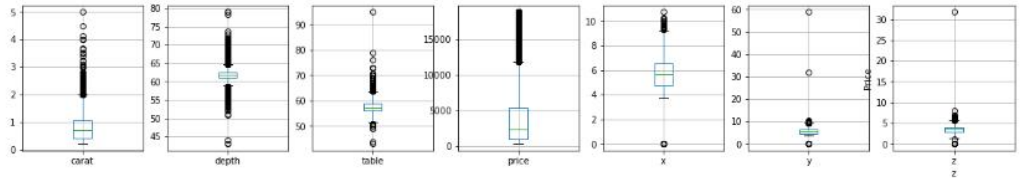
In [10]:
```
data.shape
```

Out[10]: (53940, 10)

In [11]:
```
def databoxplot(data, listvar):
    fig, axes = plt.subplots(nrows=1, ncols=len(listvar), figsize=(20, 3))
    counter=0
    for ax in axes:
        data.boxplot(column=listvar[counter], ax=axes[counter])
        plt.ylabel('Price')
        plt.xlabel(listvar[counter])
        counter = counter+1
    plt.show()
print(data.head(5))
```

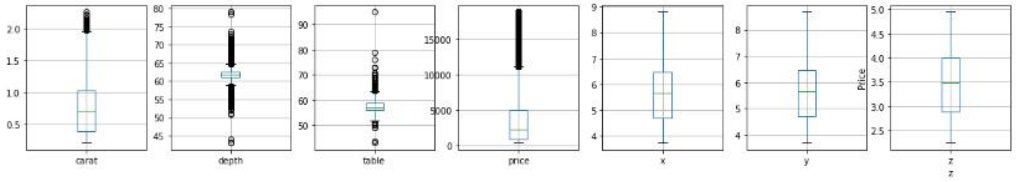
```
   carat      cut color clarity  depth  table  price     x     y     z
0   0.23    Ideal     E     SI2   61.5   55.0    326  3.95  3.98  2.43
1   0.21  Premium     E     SI1   59.8   61.0    326  3.89  3.84  2.31
2   0.23     Good     E     VS1   56.9   65.0    327  4.05  4.07  2.31
3   0.29  Premium     I     VS2   62.4   58.0    334  4.20  4.23  2.63
4   0.31     Good     J     SI2   63.3   58.0    335  4.34  4.35  2.75
```

In [12]:
```
databoxplot(data, linear_vars)
```



In [ ]:

In [13]:
```
def removeoutliers(data, listvars, z):
    from scipy import stats
    for var in listvars:
        data1 = data[np.abs(stats.zscore(data[var])) < z]
    return data1

data = removeoutliers(data, linear_vars,2)
databoxplot(data, linear_vars)
```



In [14]:
```
data.shape
```

Out[14]: (52140, 10)

Databoxplot is a simple way to visualize the shape of our data and to check whether outliers is there or not, we used this databoxplot.

After the databoxplot, we have removed the outliers by using the Z-score.

Z-score means Z degrees + or - 3 or greater than zero is the standard cut-off value for detecting outliers. The distribution of Z scores in a standard normal distribution is depicted in the probability distribution.

Out[14]: (52140, 10)

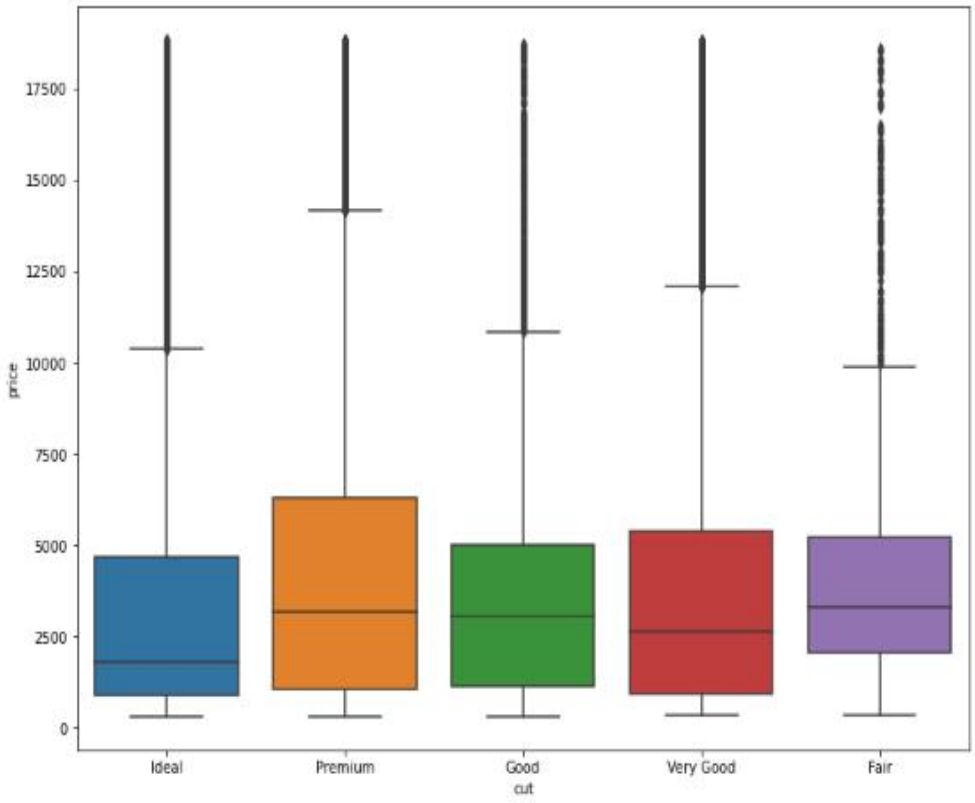
In [15]:
```
display(data.head(5))
data = data.drop(data[data["x"]==0].index)
data = data.drop(data[data["y"]==0].index)
data = data.drop(data[data["z"]==0].index)

data2=data[['cut', 'price']]
data2.head()
```

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

Out[15]:

	cut	price
0	Ideal	326
1	Premium	326
2	Good	327
3	Premium	334
4	Good	335

In [16]:
```
plt.figure(figsize=(12,9))
sns.boxplot(x='cut', y='price', data=data2)
plt.show()
```



In this box plot, we can see that premium has highest price when we compare to others.

In [17]:
```
from sklearn.preprocessing import LabelEncoder
def convert_catg(data1):
    from sklearn.preprocessing import LabelEncoder
    le = LabelEncoder()
    object_cols = list(data1.select_dtypes(exclude=[np.number]).columns)
    object_cols_ind = []
    for col in object_cols:
        object_cols_ind.append(data1.columns.get_loc(col))
        for i in object_cols_ind:
            data1.iloc[:,i] = le.fit_transform(data1.iloc[:,i])
```

In [18]:
```
convert_catg(data)
print(data.head(5))
```

```
   carat  cut  color  clarity  depth  table  price     x     y     z
0   0.23    2      1        3   61.5   55.0    326  3.95  3.98  2.43
1   0.21    3      1        2   59.8   61.0    326  3.89  3.84  2.31
2   0.23    1      1        4   56.9   65.0    327  4.05  4.07  2.31
3   0.29    3      5        5   62.4   58.0    334  4.20  4.23  2.63
4   0.31    1      6        3   63.3   58.0    335  4.34  4.35  2.75
```

In [19]:
```
y=data['price']
y.head(5)
```

Out[19]:
```
0    326
1    326
2    327
3    334
4    335
Name: price, dtype: int64
```

In [20]:
```
x=data.drop(['price', 'x', 'y', 'z'], axis=1)
x.head(5)
```

Out[20]:

	carat	cut	color	clarity	depth	table
0	0.23	2	1	3	61.5	55.0
1	0.21	3	1	2	59.8	61.0
2	0.23	1	1	4	56.9	65.0
3	0.29	3	5	5	62.4	58.0
4	0.31	1	6	3	63.3	58.0

For the diamond dataset, basic data understanding and preprocessing is done which includes converting categorical data like cut, clarity and colour into numerical data and removing unnecessary columns, namely, Unnamed: 0

Training and testing:

In [21]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.8,random_state=42)
```

In [22]:
```
len(x_train)
```

Out[22]:
```
41712
```

In [23]:
```
len(y_test)
```

Out[23]:
```
10428
```

In [24]:
```
len(data)
```

Out[24]:
```
52140
```

In [25]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.fit_transform(x_test)
```

After training and testing the dataset, len(x_train) is 41712, len(y_test) is 10428 and total length of the data is 52140.

Linear Regression:

Linear Regression is one of the simplest algorithms in Machine Learning. As per statistics, the term Regression is defined as a measure of the relation between an output variable and the input variable(s), hence, Linear Regression assumes a linear relationship between the independent (input) and dependent (output) variable.

We have used the variables cut, colour, clarity, x, y, and z to analyse the data. The non-numerical data are cut, colour, and clarity.

```
In [26]: #Lr
         from sklearn.linear_model import LinearRegression
         linreg=LinearRegression()
         linreg.fit(x_train,y_train)
         pred=linreg.predict(x_test)

In [27]: from sklearn.metrics import r2_score
         lrm=r2_score(y_test,pred)*100
         print(lrm)

         86.75964704257704
```

The accuracy for linear regression algorithm is 88.65%.

Decision tree:

Making a training model that can be used to predict the class or value of the target variable is the aim of using a decision tree.

In decision trees, we begin at the tree's base to make predictions. We contrast the root attribute's values with that of the attribute on the record.

As we move up to the next node, we follow the brand that corresponds to the next basis of comparison.

```
In [28]: #dtr
         from sklearn.tree import DecisionTreeRegressor
         reg=DecisionTreeRegressor()
         reg.fit(x_train,y_train)
         pred1=reg.predict(x_test)

In [29]: dtr=r2_score(y_test,pred1)*100
         print(dtr)

         96.61109988248813
```

The accuracy for Decision tree is 96.95%.

Random forest:

Random forest is a supervised learning algorithm.

It creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution.

We use random forest regression to test a non-linear model on the data.

Implementation:

We started fitting models and get their performance error. We are using mean squared error for our performance measure.

```
In [30]: #rfr
from sklearn.ensemble import RandomForestRegressor
rf=RandomForestRegressor(n_estimators=50)
rf.fit(x_train,y_train)
pred2=rf.predict(x_test)

In [31]: rfr=r2_score(y_test,pred2)*100
print(rfr)

98.05411559433279
```

The accuracy for Random forest Regressor is 98.22%.

KNN Regression:

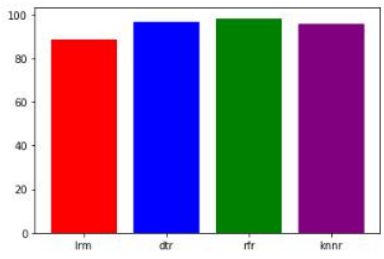
KNN regression is a non-parametric technique that, by averaging the observations in the same neighbourhood, intuitively approximates the relationship between independent variables and the continuous outcome. To forecast the values of any new data points, the KNN algorithm makes use of "feature similarity." In other words, the value given to the new point depends on how much it resembles the points in the training set.

```
In [32]: #knn
from sklearn.neighbors import KNeighborsRegressor
knn=KNeighborsRegressor(n_neighbors=5)
knn.fit(x_train,y_train)
pred3=knn.predict(x_test)

In [33]: knnr=r2_score(y_test,pred3)*100
print(knnr)

95.43984872258227
```

The accuracy for K-Neighbors regressor algorithm is 95.48%.

Comparision of Algorithms:

```
In [34]: print("LinearRegression model",lrm)
print("Decision Tree regressor model",dtr)
print("RandomForest Regressor model",rfr)
print("K-Neighbors Regressor Algorithm",knnr)

LinearRegression model 86.75964704257704
Decision Tree regressor model 96.61109988248813
RandomForest Regressor model 98.05411559433279
K-Neighbors Regressor Algorithm 95.43984872258227

In [35]: method = ['lrm','dtr','rfr','knnr']
accuracy = [88.65,96.89,98.22,95.48]
plt.bar(method,accuracy)
newcolours=["red","blue","green","purple"]
plt.bar(method,accuracy,color=newcolours)
plt.show()
```



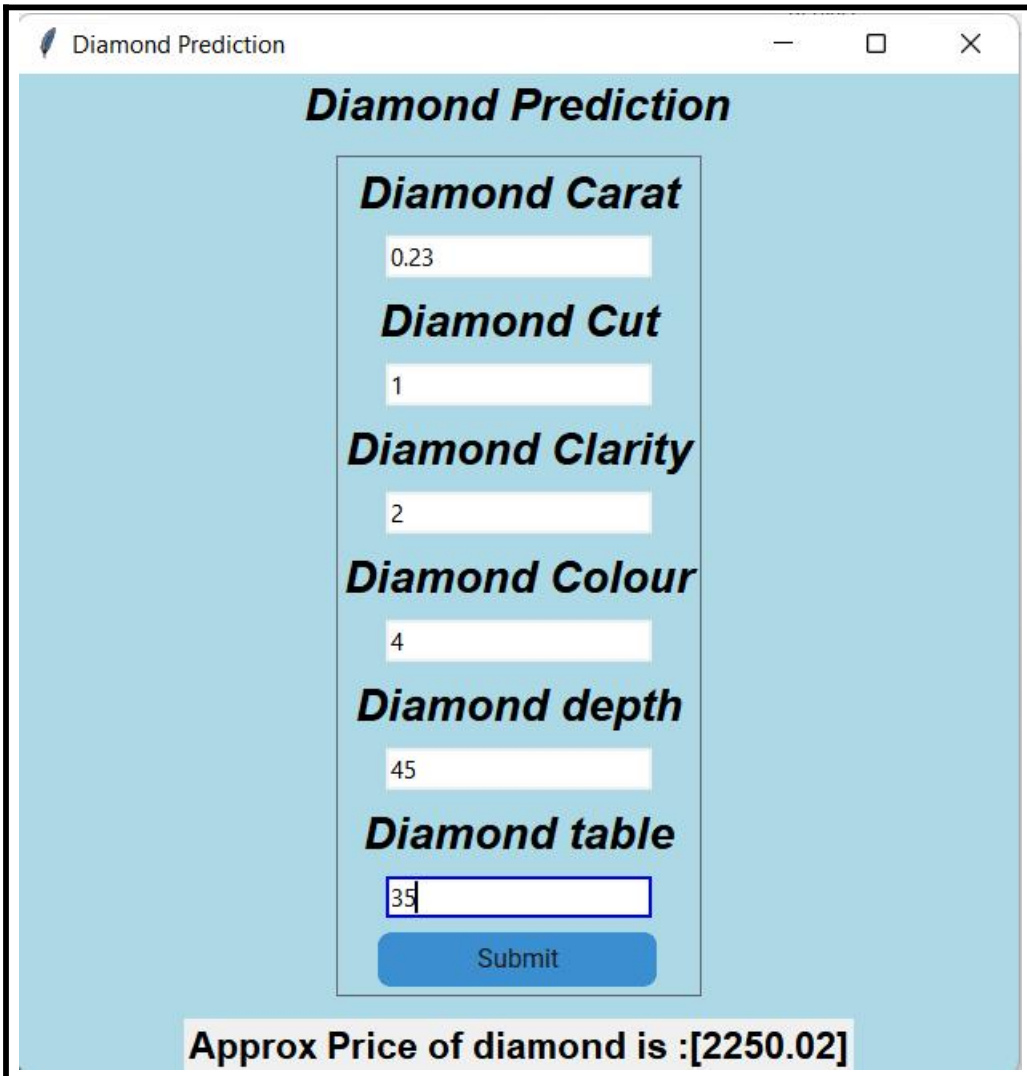
As shown in the Bar plot, the algorithm with best accuracy is Random forest regressor. It has an accuracy of 98.22%.

In [36]:
```
def prediction():
    carat = (input("Enter the value of carat:"))
    cut =   int(input("Enter the value of cut:"))
    claraity = int(input("Enter the value of claraity:"))
    color = int(input("Enter the value of color:"))
    depth = int(input("Enter the value of Depth:"))
    table = int(input("Enter the value of table:"))
    price = rf.predict([[carat,cut,claraity,color,depth,table]])
    print('Approx Price of diamomnd is :',price,'$')

predi = prediction()
predi
```

```
Enter the value of carat:0.23
Enter the value of cut:1
Enter the value of claraity:2
Enter the value of color:4
Enter the value of Depth:45
Enter the value of table:35
Approx Price of diamomnd is : [2250.02] $
```

In [*]:
```
import tkinter.messagebox
from tkinter import *
from tkinter import messagebox
import customtkinter

customtkinter.set_appearance_mode("System")
customtkinter.set_default_color_theme("blue")

window = customtkinter.CTk()
window.config(bg='lightblue')
window.title('Diamond Prediction')
window.geometry("500x500+10+10")

def prediction(carat,cut,claraity,color,depth,table):
    price = rf.predict([[carat,cut,claraity,color,depth,table]])
    print('Approx Price of diamomnd is :',price,'$')
    return price
c = Canvas(window, bg="lightblue", height="400", width="400", highlightthickness=1, highlightbackground="lightblue")
c.place(relx=0.5, rely=0.5, anchor=CENTER)

head = Label(window, text="Diamond Prediction", font=('Aerial 17 bold italic'))
head.config(foreground='black', bg="lightblue")
head.pack()

lab1 = Label(c, text="Diamond Carat", font=('Aerial 17 bold italic'), bg="lightblue")
lab1.pack(pady=4)
val1 = Entry(c, text="Enter Carat", bd=0.2, highlightcolor='blue', highlightthickness=1.5)
val1.pack(pady=4)
lab2 = Label(c, text="Diamond Cut", font=('Aerial 17 bold italic'), bg="lightblue")
lab2.pack(pady=4)
val2 = Entry(c, text="Enter Cut", bd=0.2, highlightcolor='blue', highlightthickness=1.5)
val2.pack(pady=4)
lab3 = Label(c, text="Diamond Clarity", font=('Aerial 17 bold italic'), bg="lightblue")
lab3.pack(pady=4)
val3 = Entry(c, text="Enter Clarity", bd=0.2, highlightcolor='blue', highlightthickness=1.5)
val3.pack(pady=4)
lab4 = Label(c, text="Diamond Colour", font=('Aerial 17 bold italic'), bg="lightblue")
val4 = Entry(c, text="Enter Colour", bd=0.2, highlightcolor='blue', highlightthickness=1.5)
lab4.pack(pady=4,padx=2)
val4.pack(pady=4)
lab5 = Label(c, text="Diamond depth", font=('Aerial 17 bold italic'), bg="lightblue")
val5 = Entry(c, text="Enter depth", bd=0.2, highlightcolor='blue', highlightthickness=1.5)
lab5.pack(pady=4,padx=2)
val5.pack(pady=4)
lab6 = Label(c, text="Diamond table", font=('Aerial 17 bold italic'), bg="lightblue")
val6 = Entry(c, text="Enter table", bd=0.2, highlightcolor='blue', highlightthickness=1.5)
lab6.pack(pady=4,padx=2)
val6.pack(pady=4)
cut_val = IntVar()
clarity_val = IntVar()
depth_val = IntVar()
table_val = IntVar()
carat_val = IntVar()
color_val = IntVar()

def show_msg():
    carat_val = val1.get()
    cut_val = val2.get()
    clarity_val = val3.get()
    color_val = val4.get()
    depth_val = val5.get()
    table_val = val6.get()
    val = prediction(carat_val,cut_val,clarity_val,color_val,depth_val,table_val)
    print(val)
    if not val1.get() or not val2.get() or not val3.get() or not val4.get() or not val5.get() or not val6.get():
        tkinter.messagebox.showinfo("Error", "Please Enter something!")
    else:
       price_ = prediction(carat_val,cut_val,clarity_val,color_val,depth_val,table_val)
       label.configure(text="Approx Price of diamond is :{}".format(price_))
btn = customtkinter.CTkButton(master=c, command=show_msg, text='Submit')
btn.pack(pady=4)
label=Label(window, text="", font=('Helvetica 14 bold'))
label.place(relx=.5, rely=0.97,anchor= CENTER)
window.mainloop()
```

```
Approx Price of diamomnd is : [2250.02] $
[2250.02]
Approx Price of diamomnd is : [2250.02] $
```



Conclusion

The prediction of diamond price is extremely important to know while investing. From this study it can be concluded that data mining techniques such as linear regression, KNN regression, random forest and decision tree can be employed to estimate diamond prices. We used Four algorithms for accuracy comparison- KNN-95.48%, LR- 88.65%, Decision Tree-96.95% and Random Forest-98.22%.

There is 98.22% accuracy from the Random Forest model.

Another models that we could depend on is the Decision Tree and KNN. It also works relatively fine.

By using User interface, we have predicted the Diamond price value using the diamond attributes.

Future works:

Future work shall include more regression models and time series data, in order to extend the precision of the predictions and also we can develop an app for diamond price prediction.

References:

- https://www.kaggle.com
- https://www.wikipedia.com
- https://www.towardsdatascience.com
- G. A. Seber and A. J. Lee. Linear Regression Analysis. Wiley Interscience, 2003.
- G. Hackeling, Mastering machine learning with scikit learning, Packet Publishing, 2014
- Breiman, L., Friedman, J., Olshen, R., Stone, C., 1984. Classification and Regression Trees. Wadsworth Inc., California.
- Nicolas Stier-Moses, Assaf Zeevi, Deconstructing the Price of Diamonds, Columbia Business School, February 24, 2008.
- José M. Peña Marmolejos, Implementing Data Mining Methods to Predict Diamond Prices, Int'l Conf. Data Science,2018.
- Liddicoat, R.T. (ed.) 1993. The GIA Diamond Dictionary (3rd edition), Gemological Institute of America, Santa Monica
- G. M. S. Cardosoa and L. Chambel, A valuation model for cut diamonds, INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH,2005.