**3.SOURCE CODE**

In [11]:

**import** pandas **as** pd

**import** numpy **as** np

**import** seaborn **as** sns

**import** matplotlib.pyplot **as** plt

**import** warnings

**import** pandas.util.testing **as** tm

In [12]:

data**=**pd.read\_csv("C:\\Users\\Dell\\OneDrive\\Desktop\\diamonds.csv")

In [13]:

data.head(2)

Out[13]:

**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

In [14]:

data.describe()

Out[14]:

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

In [15]:

data.isnull().sum()

Out[15]:

carat 0

cut 0

color 0

clarity 0

depth 0

table 0

price 0

x 0

y 0

z 0

dtype: int64

data.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 53940 entries, 0 to 53939 Data columns (total 10 columns):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # |  | Column | Non-Null Count |  | Dtype |
| 0 |  | carat | 53940 non-null |  | float64 |
| 1 |  | cut | 53940 non-null |  | object |
| 2 |  | color | 53940 non-null |  | object |
| 3 |  | clarity | 53940 non-null |  | object |
| 4 |  | depth | 53940 non-null |  | float64 |
| 5 |  | table | 53940 non-null |  | float64 |
| 6 |  | price | 53940 non-null |  | int64 |
| 7 |  | x | 53940 non-null |  | float64 |
| 8 |  | y | 53940 non-null |  | float64 |
| 9 |  | z | 53940 non-null |  | float64 |

dtypes: float64(6), int64(1), object(3) memory usage: 4.1+ MB

In [17]:

data**=**data.drop(['depth','table','x','y','z'],axis**=**1)

In [18]:

data.head(2)

Out[18]:

**carat cut color clarity price 0** 0.23 Ideal E SI2 326

**1** 0.21 Premium E SI1 326

In [19]:

data.dtypes

Out[19]:

carat float64

cut object

color object

clarity object

price int64 dtype: object

In [20]:

data['price']**=**data.price.astype(float) data.dtypes

Out[20]:

carat float64

cut object

color object

clarity object

price float64 dtype: object

plt.figure(figsize**=**[12,12]) plt.subplot(221)

*#carat weight distribution*

plt.hist(data['carat'],bins**=**20,color**=**'b') plt.xlabel('Carat Weight')

plt.ylabel('Frequency')

plt.title('Distribution of diamond carat weight') plt.subplot(222)

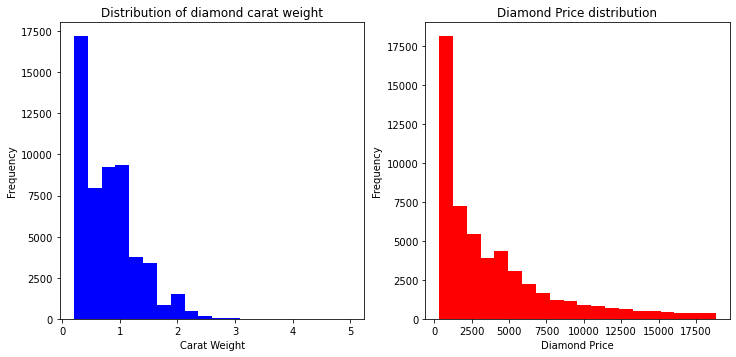
*#distribution of price value*

plt.hist(data['price'],bins**=**20,color**=**'r') plt.xlabel('Diamond Price')

plt.ylabel('Frequency')

plt.title('Diamond Price distribution')

Out[21]:

Text(0.5, 1.0, 'Diamond Price distribution')

In [22]:

data.head(1)

Out[22]:

**carat cut color clarity price 0** 0.23 Ideal E SI2 326.0

In [23]:

**from** sklearn.preprocessing **import** LabelEncoder l1**=**LabelEncoder()

label**=**l1.fit\_transform(data['cut']) l1.classes\_

Out[23]:

array(['Fair', 'Good', 'Ideal', 'Premium', 'Very Good'], dtype=object)

In [24]:

label

Out[24]:

array([2, 3, 1, ..., 4, 3, 2])

In [25]:

data['cut\_label']**=**label

data.head(10)

Out[52]:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **carat** | **cut** | **color** | **clarity** | **price** | **cut\_label** | **clarity\_label** |
| **0** 0.23 | Ideal | 2 | SI2 | 326.0 | 2 | 3 |
| **1** 0.21 | Premium | 2 | SI1 | 326.0 | 3 | 2 |
| **2** 0.23 | Good | 2 | VS1 | 327.0 | 1 | 4 |
| **3** 0.29 | Premium | 6 | VS2 | 334.0 | 3 | 5 |
| **4** 0.31 | Good | 7 | SI2 | 335.0 | 1 | 3 |
| **5** 0.24 | Very Good | 7 | VVS2 | 336.0 | 4 | 7 |
| **6** 0.24 | Very Good | 6 | VVS1 | 336.0 | 4 | 6 |
| **7** 0.26 | Very Good | 5 | SI1 | 337.0 | 4 | 2 |
| **8** 0.22 | Fair | 2 | VS2 | 337.0 | 0 | 5 |
| **9** 0.23 | Very Good | 5 | VS1 | 338.0 | 4 | 4 |

In [53]:

l2**=**LabelEncoder()

label1**=**l2.fit\_transform(data['clarity']) data['clarity\_label']**=**label1

data.head(10)

Out[53]:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **carat** | **cut** | **color** | **clarity** | **price** | **cut\_label** | **clarity\_label** |
| **0** 0.23 | Ideal | 2 | SI2 | 326.0 | 2 | 3 |
| **1** 0.21 | Premium | 2 | SI1 | 326.0 | 3 | 2 |
| **2** 0.23 | Good | 2 | VS1 | 327.0 | 1 | 4 |
| **3** 0.29 | Premium | 6 | VS2 | 334.0 | 3 | 5 |
| **4** 0.31 | Good | 7 | SI2 | 335.0 | 1 | 3 |
| **5** 0.24 | Very Good | 7 | VVS2 | 336.0 | 4 | 7 |
| **6** 0.24 | Very Good | 6 | VVS1 | 336.0 | 4 | 6 |
| **7** 0.26 | Very Good | 5 | SI1 | 337.0 | 4 | 2 |
| **8** 0.22 | Fair | 2 | VS2 | 337.0 | 0 | 5 |
| **9** 0.23 | Very Good | 5 | VS1 | 338.0 | 4 | 4 |

In [28]:

data['color']**=**data['color'].map({'D':1,'E':2,'F':3,'G':4,'H':5,'I':6,'J':7,'NA':8})

In [29]:

data['color'].fillna(0)

Out[29]:

|  |  |
| --- | --- |
| 0 | 2 |
| 1 | 2 |
| 2 | 2 |
| 3 | 6 |
| 4 | 7 |
|  | .. |
| 53935 | 1 |
| 53936 | 1 |
| 53937 | 1 |
| 53938 | 5 |
| 53939 | 1 |

Name: color, Length: 53940, dtype: int64

In [30]:

data['color'].isnull().sum()

Out[30]:

0

data.head(2)

Out[31]:

**carat cut color clarity price cut\_label clarity\_label 0** 0.23 Ideal 2 SI2 326.0 2 3

**1** 0.21 Premium 2 SI1 326.0 3 2

In [32]:

y**=**data['price'] y.head(1)

Out[32]:

0 326.0

Name: price, dtype: float64

In [33]:

x**=**data.drop(['price','cut','clarity'],axis**=**1) x.head(1)

Out[33]:

**carat color cut\_label clarity\_label 0** 0.23 2 2 3

In [34]:

**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 [35]:

len(x\_train)

Out[35]: 43152

In [36]:

len(y\_test)

Out[36]: 10788

In [37]:

len(data)

Out[37]: 53940

In [38]:

43152**+**10788

Out[38]: 53940

In [39]:

data.head()

Out[39]:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **carat** | **cut** | **color** | **clarity** | **price** | **cut\_label** | **clarity\_label** |
| **0** 0.23 | Ideal | 2 | SI2 | 326.0 | 2 | 3 |
| **1** 0.21 | Premium | 2 | SI1 | 326.0 | 3 | 2 |
| **2** 0.23 | Good | 2 | VS1 | 327.0 | 1 | 4 |
| **3** 0.29 | Premium | 6 | VS2 | 334.0 | 3 | 5 |
| **4** 0.31 | Good | 7 | SI2 | 335.0 | 1 | 3 |

data.tail()

Out[40]:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **carat** | **cut** | **color** | **clarity** | **price** | **cut\_label** | **clarity\_label** |
| **53935** | 0.72 | Ideal | 1 | SI1 | 2757.0 | 2 | 2 |
| **53936** | 0.72 | Good | 1 | SI1 | 2757.0 | 1 | 2 |
| **53937** | 0.70 | Very Good | 1 | SI1 | 2757.0 | 4 | 2 |
| **53938** | 0.86 | Premium | 5 | SI2 | 2757.0 | 3 | 3 |
| **53939** | 0.75 | Ideal | 1 | SI2 | 2757.0 | 2 | 3 |

In [43]:

**from** sklearn.preprocessing **import** StandardScaler scaler**=**StandardScaler()

x\_train**=**scaler.fit\_transform(x\_train) x\_test**=**scaler.fit\_transform(x\_test)

In [44]:

**from** sklearn.linear\_model **import** LinearRegression linreg**=**LinearRegression()

linreg.fit(x\_train,y\_train) pred**=**linreg.predict(x\_test)

In [45]:

*#accuracy with linear regression*

**from** sklearn.metrics **import** r2\_score lr**=**r2\_score(y\_test,pred)**\***100

print(lr)

87.76517206528275

In [46]:

*#accuracy with decision tree*

**from** sklearn.tree **import** DecisionTreeRegressor reg**=**DecisionTreeRegressor()

reg.fit(x\_train,y\_train) pred1**=**reg.predict(x\_test)

In [47]:

**from** sklearn.metrics **import** r2\_score dtr**=**r2\_score(y\_test,pred1)**\***100

print(dtr)

97.15579945236499

In [49]:

**from** sklearn.ensemble **import** RandomForestRegressor rf**=**RandomForestRegressor(n\_estimators**=**50)

rf.fit(x\_train,y\_train) pred2**=**rf.predict(x\_test)

In [50]:

**from** sklearn.metrics **import** r2\_score rfr**=**r2\_score(y\_test,pred2)**\***100

print(rfr)

97.75475742017633

**def** prediction():

carat**=**(input('Enter the value of carat: ')) cut**=**int(input('Enter the value of cut: '))

clarity**=**int(input('Enter the value of clarity: ')) color**=**int(input('Enter the value of color: '))

price**=**rf.predict([[carat,cut,clarity,color]])

print("Approximately Price pf Diamonds is:",price,'Rs') pred1**=**prediction()

pred1

Enter the value of carat: 0.23 Enter the value of cut: 2

Enter the value of clarity: 3 Enter the value of color: 2

Approximately Price pf Diamonds is: [3706.02] Rs

In [ ]: