Chetan uppara

CSCI Final Project



Data Set and the goal of the project

For my final project I've picked a data set that uses a dataset about diamonds. This dataset contains 10 variables. The goal of the project is to to predict the price of the diamond with the given variables and see how else the price changes.

# Index counter	# carat Carat weight of the diamond		=	Describe cut quality of the diamond. Quality in increasing order Fair, Good, Very Good, Premium, Ideal		△ color ☐ Color of the diamond, with D being the best and J the worst		A clarity How obvious inclusions are within the diamond: (in order from best to worst, FL = flawless, I3= level 3 inclusions) FL,IF,		# depth depth %: The height of a diamond, measured from the culet to the table, divided by its average girdle diameter		# table = table%: The width of the diamond's table expressed as a percentage of its average diameter		# price = the price of the diamond	
		4		Ideal Premium	40% 26%	G E	21% 18%	SI1 VS2	24%	1			1 - 61.20 ± 13,230		
1	53.9k		5.01	Other (18598)	34%	Other (32851)	61%	Other (28617)	53%	43	79	43	95	326	18.8
1		0.23		Ideal		E		SI2		61.5		55		326	
2		0.21		Premium		Е		SI1		59.8		61		326	
3		0.23		Good		Е		VS1		56.9		65		327	
4		0.29		Premium		I		VS2		62.4		58		334	
5		0.31		Good		J		SI2		63.3		58		335	
6		0.24		Very Good		J		VVS2		62.8		57		336	
7		0.24		Very Good		I		VVS1		62.3		57		336	
8		0.26		Very Good		Н		SI1		61.9		55		337	
9		0.22		Fair		E		VS2		65.1		61		337	
10		0.23		Very Good		Н		VS1		59.4		61		338	
11		0.3		Good		J		SI1		64		55		339	
12		0.23		Ideal		J		VS1		62.8		56		340	
13		0.22		Premium		F		SI1		60.4		61		342	
14		0.31		Ideal		J		SI2		62.2		54		344	
15		0.2		Premium		E		SI2		60.2		62		345	
16		0.32		Premium		E		I1		60.9		58		345	
17		0.3		Ideal		I		SI2		62		54		348	





Hypothesis

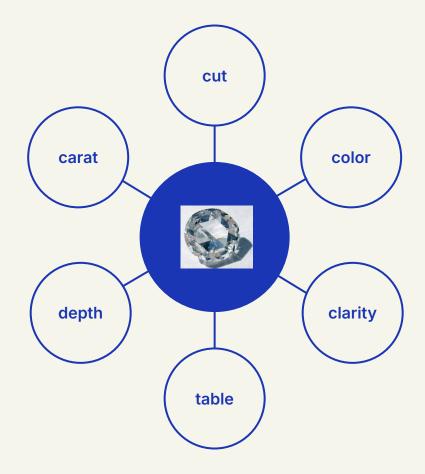
Null Hypothesis

where Data set has a correlation where the clarity, carrots, weight, and quality of the diamond can increase the value of the diamond

Alternate Hypothesis

where the data set does not correlate between total depth percentage and the price of the diamonds

Variables



Data preprocessing

- 1. Replaced the null values with the median.
- I used get dummies (enumerate the values) for my categorical columns.
- 3. I didn't standardize the data.
- There is small amount of multicollinearity, but the variables are related.

```
def clean_data(data_frame):
    number_columns=["carat", "depth", "table", "x", "y", "z"]
    catagory_columns=["cut", "color", "clarity"]

for col in number_columns:
    data_frame[col] = data_frame[col].fillna(data_frame[col].median())

df = pd.get_dummies(data_frame, columns=catagory_columns, drop_first=True)|

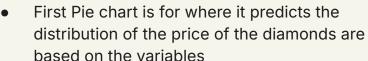
df = df.replace([np.inf, -np.inf], np.nan).fillna(0)

df = df.astype(int)

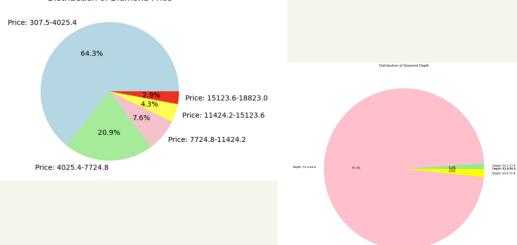
return df
```

```
Correlation Heatmap of Features
  Unnamed: 0 - 1 0.40.0340.10.3-10.340.390.390.300.0451-0.050.00805501090205.0405040301027-9.0-10.106026.003.094.09
        carat -0.41 10.018.150.850.840.820.80.028.140.140.093.160.050701 D.10.140.1-0.082036.240.065.040.130.1
        depth-0.0B40181-0.29.00.D010708210.1-B.02-D.20.0280-20901.802.70280200
        table -0.10.150.29 1 0.130.20.190.120.170.5 0.340.120000504050405089012201070307.
                                                                                                     0.8
        price -0.30.89.010.13 1 0.870.860030000309709.6066-D.002408.659909708
           - 0.6
    cut Good-0.045026.130-070003040604504110.240.140.100000069925009506203050830560400
     cut Ideal -0.130.140.020.50.090.1-10.160.1-10.2611-0.480.440.0101001.435.902.1099.103.8.30.083.
                                                                                                     - 0.4
 cut Premium-0.0502120.20.34.096.120.10.06b.1-9.44 1 0.30.0-108001.0003.90.006.917.0540.200600.02020.054062
cut Very Good -0.03.00328, 12006,6 12021014,1-7.44,3 110.02406132,504401359801303,50466680,6051515
      color E 9.058.10.0029075.10.1-30.1-30.102007090.D D802 11-0.2-20.240.20.1-30.1-0.04500.60650 27702 8001.512
                                                                                                     - 0.2
      color F 9.030905070-D80405024.95.04070046006900400.2030.22110.240.20.1-50.10.01-9.402092.301.003.80107013
      color G 9.035 01.000 703 90 9 60 2030 2030 1080 2050 950 950 950 25.240.24
                                                                                                    - 0.0
       clarity | F 0.0709.0820292078.050.1-20.1-20.1-20.1-20.03.0-30.05040-3330.4950109007.80.06.70.103.0.2.11 -0.-10.08400780.10.050.06
                                                                                                    - -0.2
                                                                                                    - -0.4
                                                ery Good
                                                                   color
```

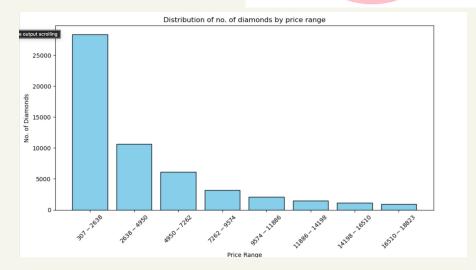
Visualization the data



- 97.3% of the diamond depth is about 54-65mm,
 1.1% of the diamond depth is about 50-5mm7,
 and 1.6% of the diamond depth is about
 64-71mm
- Distribution of no. of diamonds by price range



Distribution of Diamond Price



Confidential

Copyright ©

Machine Learning Models

```
def linear_model(ycolumn):
    X = diamond_df.drop(columns=[ycolumn,"Unnamed: 0"])
    y = diamond_df[ycolumn]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
    lin_reg = LinearRegression()
    lin_reg.fit(X_train, y_train)
    y_pred = lin_reg.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    return mse, r2
```

Linear Regression model

```
def decision_tree(ycolumn):
    X = diamond_df.drop(columns=[ycolumn,"Unnamed: 0"])
    y = diamond_df[ycolumn]

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
    model = tree.DecisionTreeClassifier(max_depth=9, random_state=42, min_samples_leaf=3)
    model = model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    return mse, r2
```

Decision tree model

Results

Linear Regression model

MSE: 2141792.6468459307, R2: 0.8626700641808112

Decision Tree model

MSE: 2495368.551415153, R2: 0.8399990757668726