Diamonds Project

Sarah Saadeh
Tamara elheet
Ashraf abdel habrumman

Introduction

- In this project we will deal with the diamonds dataset that contains the prices and other attributes of almost 54,000 diamonds .
- Our goal is try to analyze diamonds by their cut, color, clarity, price and other attributes using some algorithms.
- Obviously this a regression dataset and we will use different models of regression to find the best accuracy.

Describing Data(1)

- Diamonds dataset are uploaded by Shivam Agrawal in 53940 si epahs atad ehT .2017rows and 10 columns. And there are no null values. Also, we have 3 categorical data and rest of data are numerical .
- The dataset contains 10 features: carat, cut, color, clarity, depth, table, price, x (length) and y (width).
- We will explain each feature at the next slide.

Describing Data(2)

1- Index counter ... ,3 ,2 ,1

2- Carat: weight of the diamond.

3- Cut: Quality of the diamond (Fair, Good, Very Good, Premium, Ideal)

4- Color: Color of the diamond, D the best and J the worst.

5- Clarity: How obvious inclusions are within the diamond.

Describing Data(3)

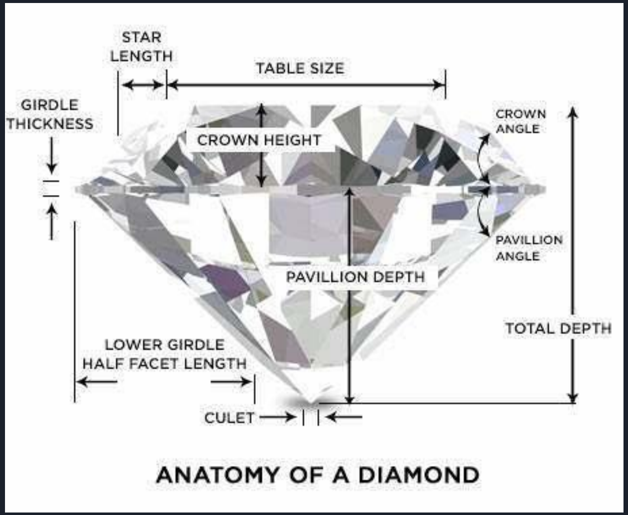
6- Depth: The height of the diamond.

7- Table: The width of the diamond's table expressed as a percentage of its average diameter.

8- Price: the price of the diamond.

9- x: the diamond length in mm.

10- y: the diamond width in mm.



Data visualization(1)

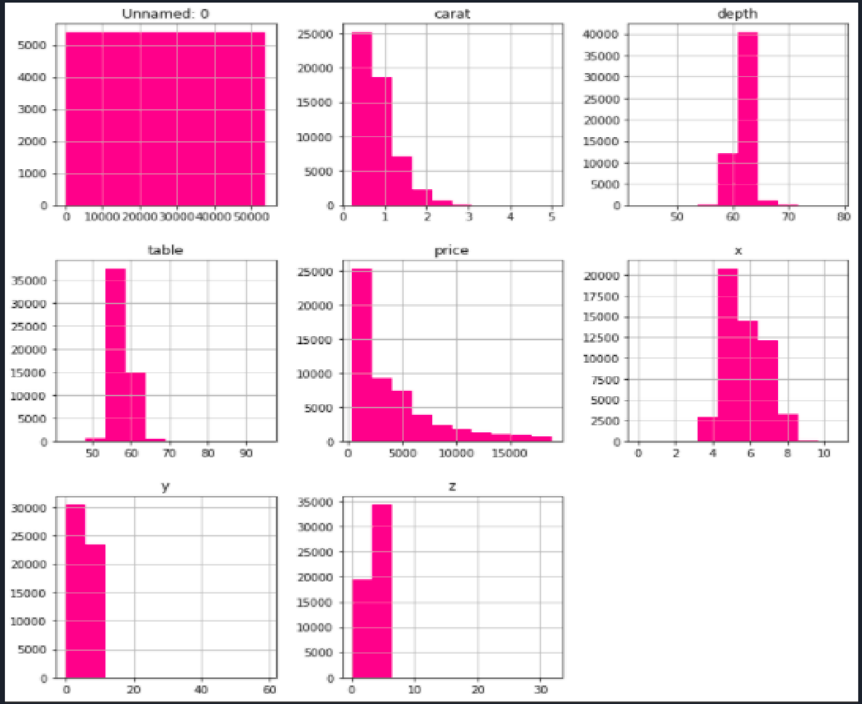
Before we start visualizing all the data we will mainly look at the big picture. Now to get the knowledge that we need about the data we can present it in a histogram for each numerical attribute we have .

To get a deeper look we visualize. When we visualized we worked on a test set to avoid harming the training set.

Data visualization(2)

Here is how we visualize our data:

1. presenting the data in histograms

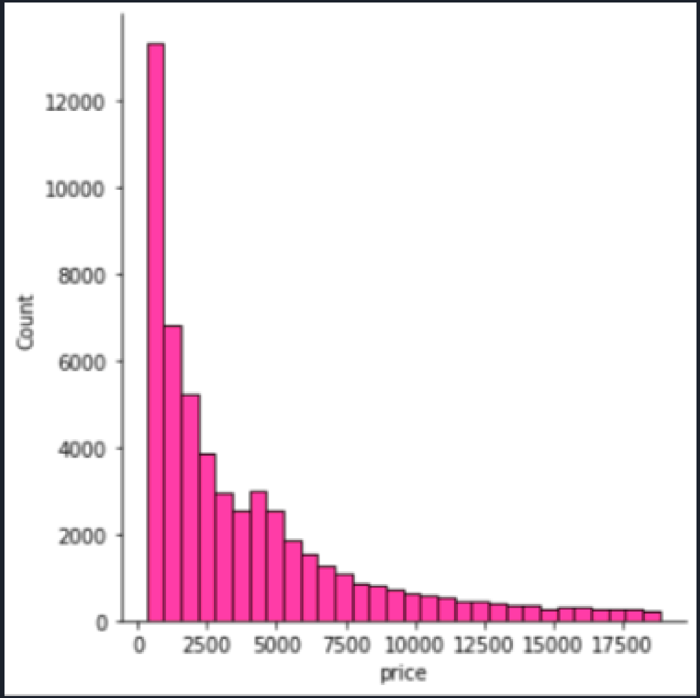


Data visualization(3)

2. We found it better to use the heatmap diagram to present the diamonds correlation.

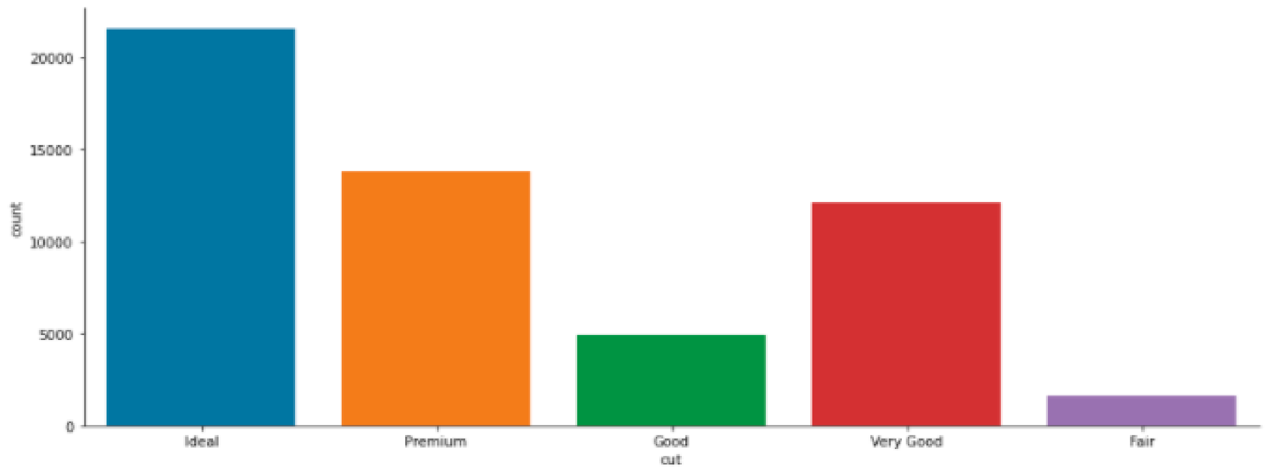
Data visualization(4)

3. Using displot to present the prices of the diamonds:

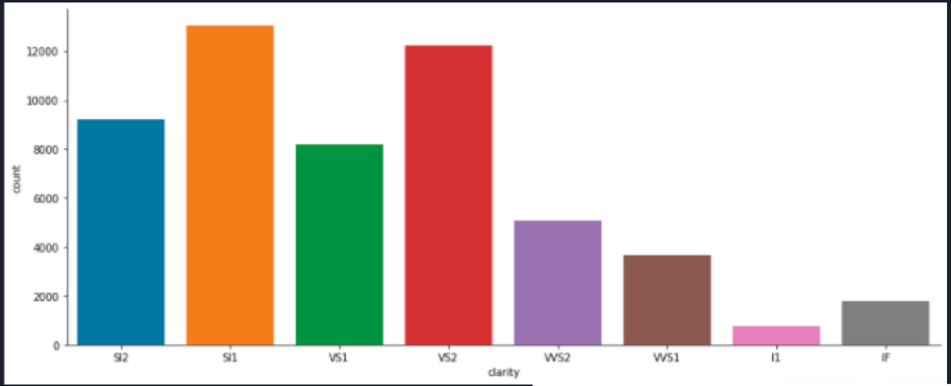


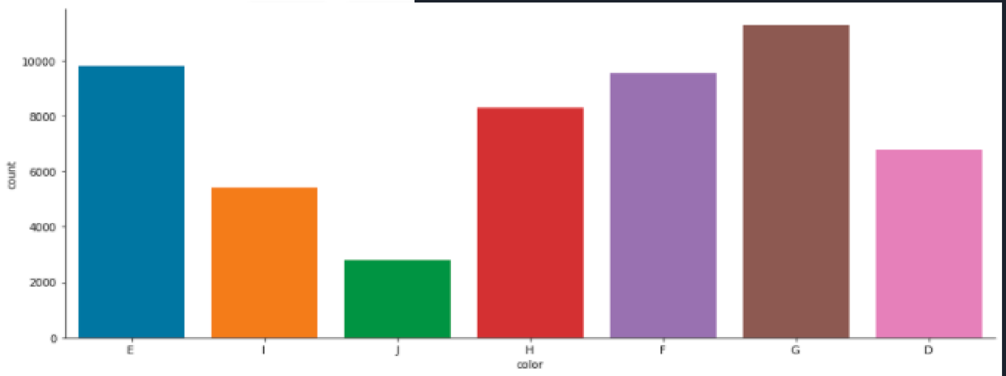
Data visualization(4)

4. Display plots for the cut, color and the clarity of the diamonds.



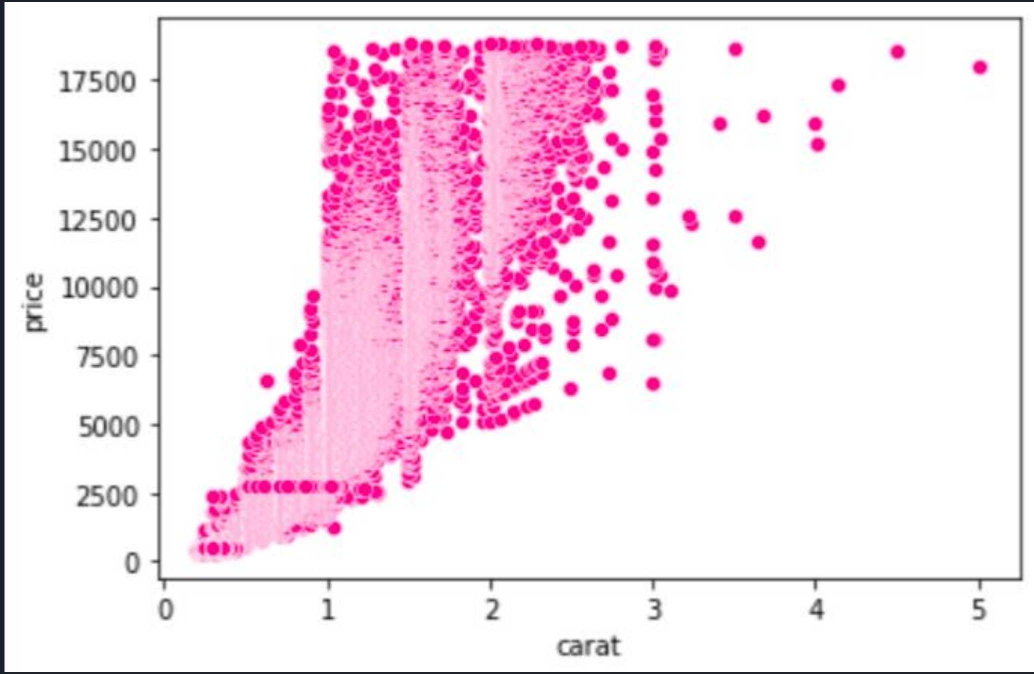
Data visualization(4)





Data visualization(5)

4. plotting the carat vs. its price:



Preparing data for ML:

What we did is that we basically performed some data cleaning since most of the ML algorithms do not work with missing features.

We applied what we learned in this step, we got rid of corresponding districts, the whole attribute and we set the missing values to some values.

We also handled the text and categorical attributes.

Features Engineering (1)

- Volume:

- Volume = x * y * z

- Reduced Features by create Volume

- and Volume has a very High correlation with price more than x, y, z

```python
diamonds['volume'] = diamonds['x'] * diamonds['y'] * diamonds['z']
#diamonds.head()
```

```python
diamonds.drop(['x','y','z'], axis=1, inplace= True)
```

```python
df1 = diamonds.drop("Unnamed: 0", axis = 1)
for col in df1.select_dtypes("object"):
    print(col,len(df1[col].unique()), df1[col].unique())
    print("")
```

```python
diamonds_labels = diamonds['price'].copy()
diamonds = diamonds.drop(["Unnamed: 0","price"], axis=1)
diamonds_num = diamonds.drop(['cut','color', 'clarity'], axis=1)
num_attributes = list(diamonds_num)
cut_attribute = ["cut"]
clarity_attribute = ["clarity"]
color_attribute = ["color"]
full_transformer= ColumnTransformer([
                                  ('std_scaler', StandardScaler(),num_attributes),
                                  ('ordinal_encoder_cut',OrdinalEncoder(categories=[['Fair', 'Good', 'Very Good', 'Premium', 'Ideal']]),cut_attribute),
                                  ('ordinal_encoder_clarity', OrdinalEncoder(categories=[['I1', 'SI2', 'SI1', 'VS2', 'VS1', 'VVS2', 'VVS1', 'IF']]), clarity_attribute),
                                  ('ordinal_encoder_color' , OrdinalEncoder(categories=[['D','E','F','G','H','I','J']] ), color_attribute)
])
diamonds_prepared = full_transformer.fit_transform(diamonds)
diamonds_prepared[0:5]
```

Remove the outliers

After removing the outliers

```
z_scores = stats.zscore(diamonds_prepared_df)

abs_z_scores = np.abs(z_scores)
filtered_entries = (abs_z_scores < 3).all(axis=1)
filtered_entries
diamonds_prepared_df = diamonds_prepared_df[filtered_entries]
diamonds_labels = diamonds_labels[filtered_entries]
diamonds_prepared_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 52550 entries, 0 to 53939
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   carat     52550 non-null  float64
 1   depth     52550 non-null  float64
 2   table     52550 non-null  float64
 3   volume    52550 non-null  float64
 4   cut       52550 non-null  float64
 5   clarity   52550 non-null  float64
 6   color     52550 non-null  float64
dtypes: float64(7)
memory usage: 3.2 MB
```

Before removing the outliers

```
0   Unnamed: 0  53940 non-null  int64
1   carat       53940 non-null  float64
2   cut         53940 non-null  object
3   color       53940 non-null  object
4   clarity     53940 non-null  object
5   depth       53940 non-null  float64
6   table       53940 non-null  float64
7   price       53940 non-null  int64
8   x           53940 non-null  float64
9   y           53940 non-null  float64
10  z           53940 non-null  float64
```

Training and Evaluating on the Training Set: (1)

Now that we framed the problem, explored the data and sampled a training set and a test set, we started to actually train our training set. what we did here is that we choose 5 ML models to train as follows :

Training and Evaluating on the Training Set: (2)

We trained a LinearRegression model, RandomForest, SVM, KNeighborsclass, and decisionTreeRegressor models :

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
```

Training and Evaluating on the Training Set: (3)

Now, to get better results we calculated the rmse to check which model has the least error percentage for the ***training set*** and it went as follow:

```
Random Forest Regressor
    Training Set
MSE: 38860.6465913013
MAE: 103.60018419785929
RMSE: 197.13103913717217

Support Vector Machine
    Training Set
MSE: 7919203.855097151
MAE: 1513.5730266423752
RMSE: 2814.1080034528086

Linear Regression
    Training Set
MSE: 1370809.8232883147
MAE: 844.0006271518904
RMSE: 1170.8158793287332

Decision Tree Regression
    Training Set
MSE: 982.2888179058492
MAE: 12.078555570658331
RMSE: 31.341487168062866

K Nearset Neighbor
    Training Set
MSE: 982.2888179058492
MAE: 12.078555570658331
RMSE: 31.341487168062866
```

Training and Evaluating on the Training Set: (4)

Now, to get better results we calculated the rmse to check which model has the least error percentage for the ***test set*** and it went as follow:

```
Random Forest Regressor
    Test Set
MSE: 258647.5522164276
MAE: 264.50487621010984
RMSE: 508.57403808730504

Support Vector Machine
    Test Set
MSE: 7853061.9716119645
MAE: 1509.3788490749012
RMSE: 2802.3315242154995
```

```
Linear Regression
    Test Set
MSE: 1359022.3745276358
MAE: 840.6513699081141
RMSE: 1165.771150152394

Decision Tree Regression
    Test Set
MSE: 465941.1028138985
MAE: 341.63940162807904
RMSE: 682.5987861210263

K Nearset Neighbor
    Test Set
MSE: 465941.1028138985
MAE: 341.63940162807904
RMSE: 682.5987861210263
```

Training and Evaluating on the Training Set: (5)

Validation:

```
     Random Forest Regressor validation
Scores: [537.20652183 570.32599181 540.6624197  519.55239979 511.68668253
 503.79082675 513.80482355 502.0680367  534.77460966 527.03038218]
Root Mean squared error: 526.0902694488013
Standard deviation: 19.607374599866677

     Support Vector Machine validation
Scores: [2817.35472838 2951.4053786  2945.38336013 2982.24277767 2983.84544058
 2989.76992557 2886.11078919 2748.14549017 2909.65882854 2869.81156317]
Root Mean squared error: 2908.3728281989843
Standard deviation: 75.4594392163723

     Linear Regression validation
Scores: [1151.11932898 1153.53428977 1196.70900618 1181.22629638 1181.80016306
 1203.97282095 1176.22731581 1118.83678608 1190.45535844 1159.16991238]
Root Mean squared error: 1171.3051278035296
Standard deviation: 24.362941790468614

     Decision Tree Regression validation
Scores: [713.59554472 752.88090501 736.90579171 687.40406822 679.74126513
 698.03477738 676.6136876  709.26715283 712.33915194 670.43677223]
Root Mean squared error: 703.7219116757935
Standard deviation: 25.37068173752465

     K Nearset Neighbor validation
Scores: [859.08252087 900.83453743 913.30692873 842.98447556 901.39916188
 874.63671477 826.35446462 882.79478776 852.55876804 850.4899458 ]
Root Mean squared error: 870.4442305457203
Standard deviation: 27.26572923721974
```

Training and Evaluating on the Training Set: (6)

After training and testing the accuracy of the models, we figured out that the best results were given by the decision tree of : 0.9999 and the random forest regression as well 0.9974, so these two are the best models. The worst model was the SVM with accuracy of 0.4720

```
Support Vector Machine
Testing Accuracy: 0.46796404669887937
Training Accuracy: 0.4720301217513977


Random Forest Regressor
Testing Accuracy: 0.9824769246047065
Training Accuracy: 0.9974091775859177


Linear Regression
Testing Accuracy: 0.9079277908154664
Training Accuracy: 0.9086087050230833


Decision Tree Regression
Testing Accuracy: 0.9684330239957494
Training Accuracy: 0.9999345112315475


K Nearset Neighbor
Testing Accuracy: 0.031525531240088805
Training Accuracy: 0.911376919940193
```

Training and Evaluating on the Training Set : (7)

The K Nearest Neighbor usually used for classification dataset.

As we can see, the rmse in the Random Forest Regression is the least among the other models and its equal to 508. This means that the best model we got is the random forest regression model.

Fine tuning: (1)

For our final step, we performed fine-tuning on the Random Forest Regression since it was our best model. The goal of doing this step is to simply improve the accuracy .

We used the Grid search to fine-tune our project as follows :

Fine tuning:(2)

```python
from sklearn.model_selection import GridSearchCV

param_grid = [
    {'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]},
    {'bootstrap': [False], 'n_estimators': [3, 10], 'max_features': [2, 3, 4]}
    ]
forest_reg = RandomForestRegressor()
grid_search = GridSearchCV(forest_reg, param_grid, cv=5, return_train_score=True)
grid_search.fit(X_train,y_train)
pd.DataFrame(grid_search.cv_results_)
```

Fine tuning: (3)

```
cvres = grid_search.cv_results_
for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
    print(np.sqrt(-mean_score), params)
```

```
nan {'max_features': 2, 'n_estimators': 3}
nan {'max_features': 2, 'n_estimators': 10}
nan {'max_features': 2, 'n_estimators': 30}
nan {'max_features': 4, 'n_estimators': 3}
nan {'max_features': 4, 'n_estimators': 10}
nan {'max_features': 4, 'n_estimators': 30}
nan {'max_features': 6, 'n_estimators': 3}
nan {'max_features': 6, 'n_estimators': 10}
nan {'max_features': 6, 'n_estimators': 30}
nan {'max_features': 8, 'n_estimators': 3}
nan {'max_features': 8, 'n_estimators': 10}
nan {'max_features': 8, 'n_estimators': 30}
nan {'bootstrap': False, 'max_features': 2, 'n_estimators': 3}
nan {'bootstrap': False, 'max_features': 2, 'n_estimators': 10}
nan {'bootstrap': False, 'max_features': 3, 'n_estimators': 3}
nan {'bootstrap': False, 'max_features': 3, 'n_estimators': 10}
nan {'bootstrap': False, 'max_features': 4, 'n_estimators': 3}
nan {'bootstrap': False, 'max_features': 4, 'n_estimators': 10}
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: RuntimeWarning: invalid value encountered in sqrt
  This is separate from the ipykernel package so we can avoid doing imports until
```

Conclusion:

We really did enjoy working on this project because it help us to conclude and work on everything we have learned in this course so far . We believe that we will be able to work on more ML project in the future with more efficiency and experience.

Thank you!!