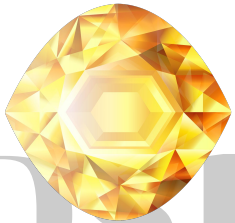
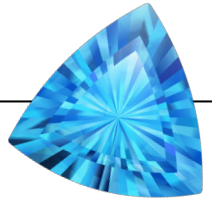


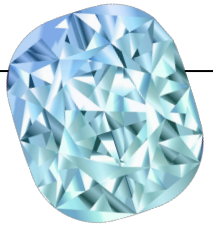
DS106-01-09-ML

Lesson 1 Hands-On



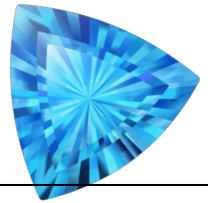
Heather Walker - 2022-11-06

Hands-On Steps



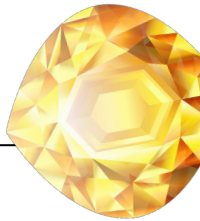
01

Data Wrangling



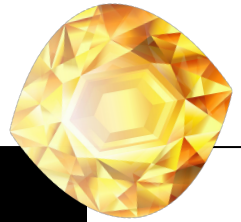
02

Building the Model



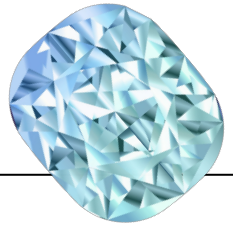
03

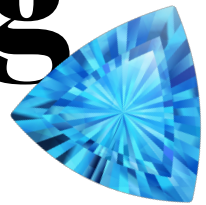
Checking the Error



01

Data Wrangling



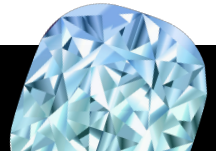


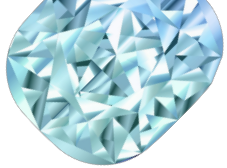
Predictive Variables

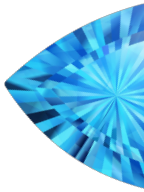
The predictive variables for this dataset are **carat**, **cut**, **color**, and **clarity**. The target variable is **price**. The first thing I notice during *data exploration* is that those 4 predictive variables are not all numeric.

For linear regression, you cannot have any non-number values in your dataset.

Variable	Data Type
carat	float64
cut	category
color	category
clarity	category





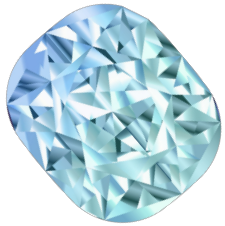


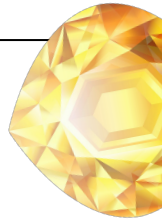
Predictive Variables

I **recoded** the variables to numeric categorical values.

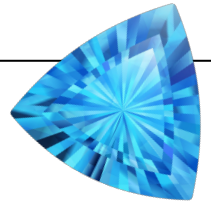
Variable	Data Type	Numeric Categories
carat	float64	n/a
cut	category	0, 1, 2, 3, 4
color	category	5-11 (7 categories)
clarity	category	12-19 (8 categories)

02

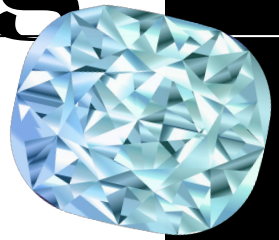




Building the Model



Modeling with train_test_split()



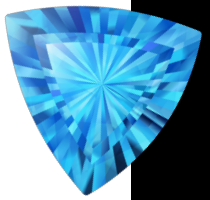
I used a 60/40 train/test split.

```
x_train, x_test, y_train, y_test =
train_test_split(x,y, test_size = .4)

# to see the shape of the data
print(x_train.shape, y_train.shape)
print(x_test.shape, y_test.shape)
```

```
(32364, 4) (32364,)
(21576, 4) (21576,)
```

Modeling with Linear Regression



```
# run linear regression model
lm = LinearRegression()
lm.fit(x_train, y_train)
```

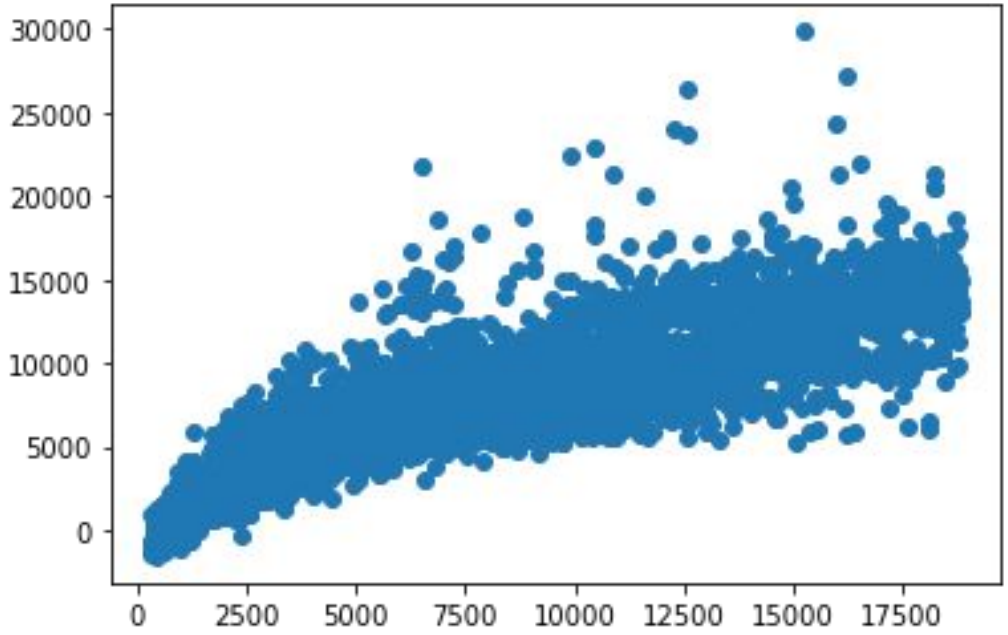
Output:

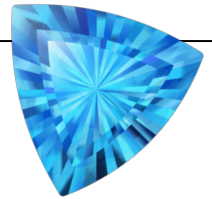
```
LinearRegression()
```

Predictions: Plot & Accuracy

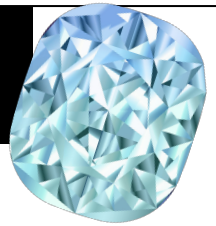
I plotted the predictions on a *scatter plot* for visual inspection.

The *Accuracy Score* of the linear regression model was about **86%**.





03



Checking the Error

Error Scores

959.06556...	2170871.98...	1473.3879...
Mean Absolute Error (MAE)	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)

k-fold cross validation

(with 5 iterations)

```
kfold = KFold(n_splits= 5, shuffle=True)
for train, test in kfold.split(x,y):
    print('train: %s, test: %s' %
(train,test))
```

```
train: [    1     2     3 ... 53936 53938 53939], test: [    0     4    17 ... 53927 53934 53937]
train: [    0     2     4 ... 53937 53938 53939], test: [    1     3     7 ... 53917 53926 53932]
train: [    0     1     3 ... 53936 53937 53938], test: [    2    12    16 ... 53929 53931 53939]
train: [    0     1     2 ... 53936 53937 53939], test: [    5     8    10 ... 53930 53935 53938]
train: [    0     1     2 ... 53937 53938 53939], test: [    6     9    13 ... 53922 53933 53936]
```

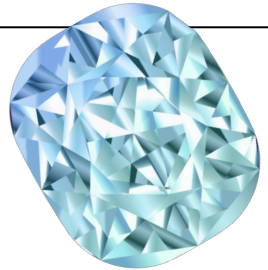
Cross Validation

Cross Validation with 5 iterations

These 5 models have around **86% accuracy** as well.

- 0.86317226
- 0.86687445
- 0.86153088
- 0.8590567
- 0.86875965

Thanks!



CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon** and infographics & images by **Freepik**