Machine Learning

# Lesson 1 HandsOn

Presented by Richmond Anku

#### Importing packages

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import metrics
import numpy as np
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
```

#### Importing 'Diamonds' dataset

diamonds = pd.read\_csv ('C:/Users/Richmond/Desktop/WOZ-U/Machine Learning/HandsOn/ML repo/Diamonds.csv')
print (diamonds)

Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	\
				_					`
1	0.21	Premium	E	511	59.8	01.0	320	3.89	
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	
53935	0.72	Ideal	D	SI1	60.8	57.0	2757	5.75	
53936	0.72	Good	D	SI1	63.1	55.0	2757	5.69	
53937	0.70	Very Good	D	SI1	62.8	60.0	2757	5.66	
53938	0.86	Premium	Н	SI2	61.0	58.0	2757	6.15	
53939	0.75	Ideal	D	SI2	62.2	55.0	2757	5.83	
	0 1 2 3 4  53935 53936 53937 53938	1 0.21 2 0.23 3 0.29 4 0.31  53935 0.72 53936 0.72 53937 0.70 53938 0.86	0 0.23 Ideal 1 0.21 Premium 2 0.23 Good 3 0.29 Premium 4 0.31 Good 53935 0.72 Ideal 53936 0.72 Good 53937 0.70 Very Good 53938 0.86 Premium	0 0.23 Ideal E 1 0.21 Premium E 2 0.23 Good E 3 0.29 Premium I 4 0.31 Good J 53935 0.72 Ideal D 53936 0.72 Good D 53937 0.70 Very Good D 53938 0.86 Premium H	0 0.23 Ideal E SI2 1 0.21 Premium E SI1 2 0.23 Good E VS1 3 0.29 Premium I VS2 4 0.31 Good J SI2 53935 0.72 Ideal D SI1 53936 0.72 Good D SI1 53937 0.70 Very Good D SI1 53938 0.86 Premium H SI2	0 0.23 Ideal E SI2 61.5 1 0.21 Premium E SI1 59.8 2 0.23 Good E VS1 56.9 3 0.29 Premium I VS2 62.4 4 0.31 Good J SI2 63.3 53935 0.72 Ideal D SI1 60.8 53936 0.72 Good D SI1 63.1 53937 0.70 Very Good D SI1 62.8 53938 0.86 Premium H SI2 61.0	0 0.23 Ideal E SI2 61.5 55.0 1 0.21 Premium E SI1 59.8 61.0 2 0.23 Good E VS1 56.9 65.0 3 0.29 Premium I VS2 62.4 58.0 4 0.31 Good J SI2 63.3 58.0  53935 0.72 Ideal D SI1 60.8 57.0 53936 0.72 Good D SI1 63.1 55.0 53937 0.70 Very Good D SI1 62.8 60.0 53938 0.86 Premium H SI2 61.0 58.0	0       0.23       Ideal       E       SI2       61.5       55.0       326         1       0.21       Premium       E       SI1       59.8       61.0       326         2       0.23       Good       E       VS1       56.9       65.0       327         3       0.29       Premium       I       VS2       62.4       58.0       334         4       0.31       Good       J       SI2       63.3       58.0       335                     53935       0.72       Ideal       D       SI1       60.8       57.0       2757         53936       0.72       Good       D       SI1       63.1       55.0       2757         53937       0.70       Very Good       D       SI1       62.8       60.0       2757         53938       0.86       Premium       H       SI2       61.0       58.0       2757	0       0.23       Ideal       E       SI2       61.5       55.0       326       3.95         1       0.21       Premium       E       SI1       59.8       61.0       326       3.89         2       0.23       Good       E       VS1       56.9       65.0       327       4.05         3       0.29       Premium       I       VS2       62.4       58.0       334       4.20         4       0.31       Good       J       SI2       63.3       58.0       335       4.34                     53935       0.72       Ideal       D       SI1       60.8       57.0       2757       5.75         53936       0.72       Good       D       SI1       63.1       55.0       2757       5.69         53937       0.70       Very Good       D       SI1       62.8       60.0       2757       5.66         53938       0.86       Premium       H       SI2       61.0       58.0       2757       6.15

## Step 1

- Import all required packages
- Import 'diamond' dataset

#### Recoding Cut to CutR

```
def Cut (series):
    if series == "Ideal":
        return 5

if series == "Premium":
        return 4

if series == "Very Good":
        return 3

if series == "Good":
        return 2

if series == "Fair":
        return 1
```

#### Recoding Color to ColorR

```
def Color (series):
    if series == "D":
        return 7

if series == "E":
        return 6

if series == "F":
        return 5

if series == "G":
        return 4

if series == "H":
        return 3

if series == "I":
        return 2
```

# Step 2

- Recode variable columns from string to float
- Recode Cut to CutR
- Recode color to colorR
- Recode clarity to clarityR

### Step 3

- Create X and Y variables
- Subset predictor x variables as an array

### Step 4

• Perform train-test split

### Step 5

- Create linear regression model and
- Fit to training data

#### Creating X and Y variables (subsetting into arrays)

```
x = diamonds[['carat', 'cutR', 'colorR', 'clarityR']]
y = diamonds['price']
```

#### Train-test split: 60/40

```
x_{train}, x_{test}, y_{train}, y_{test} = train_{test_split}(x,y), test_{size} = .4, train_{test_split}(x,y)
```

#### Linear Regression Model

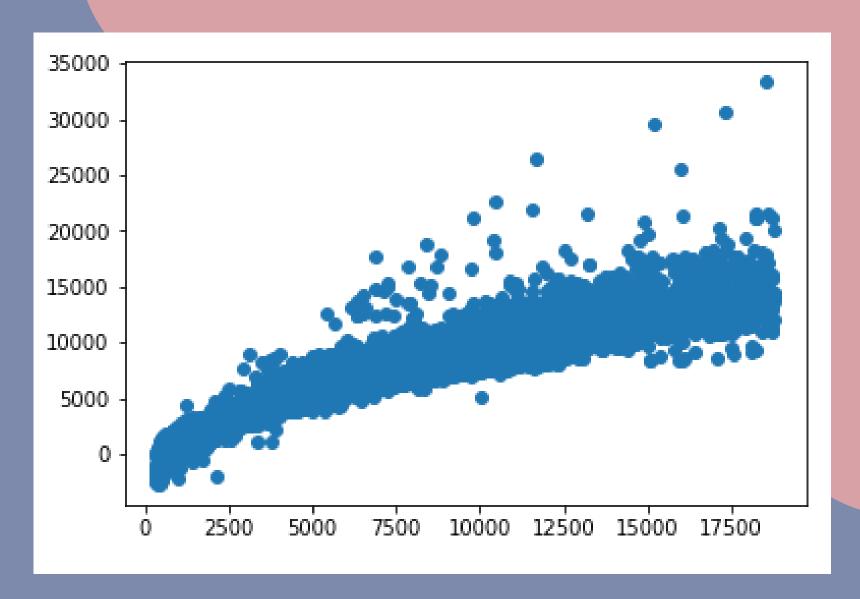
```
lm = LinearRegression()
lm.fit(x_train, y_train)
```

LinearRegression()

## - Step 6

Perform prediction using the model

```
predictions = lm.predict(x_test)
predictions
```





Plot a scatterplot of model predictions

#### **Conclusions**

- The accuracy of predictions of the model is about 90%
- The rmse value is far greater than zero (1240). Indicating that this model may not be a good fit.

### **Cross validation**

- Upon cross validation, the accuracy values vary quite widely across each iteration. The second and third iterations give about 64% and 81% accuracy respectively.
- However, the 4th and 5th iterations are in the negative. This is indicative of the fact that the model does fit certain scenarios.

#### Model Accuracy Score

```
print("Score:", lm.score(x_test, y_test))
#The model predictions are accurate about 90% of the time.
Score: 0.9031757454166052
```

#### **Root Mean Squared Error**

```
np.sqrt(metrics.mean_squared_error(y_test, predictions))
1240.2262724895409
```

#### **Cross Validation**

```
print(cross_val_score(lm, x,y, cv=5))
[ 0.09862808  0.63612892  0.81033106 -16.96778127 -0.9517348 ]
```

```
accuracy = cross_val_score(lm, x,y, cv=5)
print(accuracy)
print("Accuracy of Model with Cross Validation is:", accuracy.mean() * 100)

[ 0.09862808   0.63612892   0.81033106 -16.96778127 -0.9517348 ]
Accuracy of Model with Cross Validation is: -327.48856026922664
```

# Thank you!