DSO106C - Machine Learning Lesson 1

Alison Schnoes

Load data packages and data csv into Jupyter Notebook

```
# add because this is what we used during lesson 1
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
```

```
# add per the hands on page
import seaborn as sns
from sklearn.utils import shuffle
Diamonds = shuffle(sns.load_dataset('diamonds'))
```

```
# add data
Diamonds = pd.read_csv('/Users/music/Desktop/Diamonds.csv')
```

Directions from Hands On page

And use the following variables to predict the price of diamonds:

- carat
- cut
- color
- clarity

“You should be aware that in order to run this, you cannot have any non-number values in your dataset. If you do, and you attempt this step, you will end up with this error message: "ValueError: could not convert string to float: 'See notes for:" The “convert string to float” should tip you off that you have a problem with data types, and the “see notes for” section will tell you which variables are causing the error. You can either drop them out, or dummy code them into your dataframe - the choice is yours based on what variables you want to retain in your model.” from Create the Linear Regression Model page 3 Lesson 1 for the next 3 slides

Renaming values into numbers for the cut, color, and clarity

```
# renaming values into numbers for the cut column for Linear Regression Model
print(Diamonds['cut'].unique())
```

```
['Ideal' 'Premium' 'Good' 'Very Good' 'Fair']
```

```
Diamonds = Diamonds.replace(['Fair'],'1')
Diamonds = Diamonds.replace(['Very Good'],'2')
Diamonds = Diamonds.replace(['Good'],'3')
Diamonds = Diamonds.replace(['Premium'],'4')
Diamonds = Diamonds.replace(['Ideal'],'5')
```

```
print(Diamonds['cut'].unique())
```

```
['5' '4' '3' '2' '1']
```

Renaming values into numbers for the cut, color, and clarity

```
## renaming values into numbers for the COLOR column for Linear Regression Model
print(Diamonds['color'].unique())
```

```
['E' 'I' 'J' 'H' 'F' 'G' 'D']
```

```
Diamonds = Diamonds.replace(['E'],'1')
Diamonds = Diamonds.replace(['I'],'2')
Diamonds = Diamonds.replace(['J'],'3')
Diamonds = Diamonds.replace(['H'],'4')
Diamonds = Diamonds.replace(['F'],'5')
Diamonds = Diamonds.replace(['G'],'6')
Diamonds = Diamonds.replace(['D'],'7')
```

```
print(Diamonds['color'].unique())
```

```
['1' '2' '3' '4' '5' '6' '7']
```

Renaming values into numbers for the cut, color, and clarity

```
# renaming values into numbers for the clarity column for Linear Regression Model
print(Diamonds['clarity'].unique())
```

```
['SI2' 'SI1' 'VS1' 'VS2' 'VVS2' 'VVS1' 'I1' 'IF']
```

```
Diamonds = Diamonds.replace(['SI2'],'1')
Diamonds = Diamonds.replace(['SI1'],'2')
Diamonds = Diamonds.replace(['VS1'],'3')
Diamonds = Diamonds.replace(['VS2'],'4')
Diamonds = Diamonds.replace(['VVS2'],'5')
Diamonds = Diamonds.replace(['VVS1'],'6')
Diamonds = Diamonds.replace(['I1'],'7')
Diamonds = Diamonds.replace(['IF'],'8')
```

```
print(Diamonds['clarity'].unique())
```

```
['1' '2' '3' '4' '5' '6' '7' '8']
```

Data Wrangling

```
x = Diamonds[['cut', 'carat', 'color', 'clarity']]
y = Diamonds['price']
```

The x data is the cut, carat, color, clarity columns from the data set.
The y data is the price column from the data set.

I did not use the # rows column, x, y, depth, table, and x columns because it was unclear what they were and/or not in the directions to use them.

Train Test Split

```
# Train Test Split
```

```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = .4, random_state=101)
```

```
print(x_train.shape, y_train.shape)
print(x_test.shape, y_test.shape)
```

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

Splitting the dataset using a 60/40 train/test split as in the textbook

Printing out the shape of the data that I was using for the machine learning algorithm. X_train was 32364 rows and 4 columns. X_test dataset is 21576 fires and 4 columns.

Linear Regression Model

```
#Linear Regression Model
lm = LinearRegression()
lm.fit(x_train, y_train)
```

```
5]: LinearRegression()
```

The output of LinearRegression() let us know that it worked okay after redoing the 3 columns from characters into numbers.

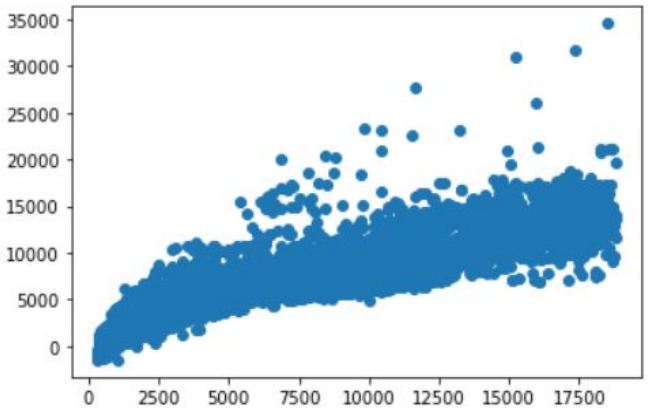
Examine Predictions

```
#Examine Predictions for the diamond price
predictions = lm.predict(x_test)
predictions
```

```
6]: array([1837.80541411, 5565.29003319,   96.49894298, ..., 4678.49619797,
       5541.21156527, 1476.50102026])
```

```
plt.scatter(y_test, predictions)
```

```
7]: <matplotlib.collections.PathCollection at 0x292a64e46a0>
```



Using the array to plot the predictions of the model.

It is pretty straight for the line so the models fit.

Accuracy Score of 86% of the time

```
#Accuracy Score of 86.8%
print("Score:", lm.score(x_test, y_test))
```

```
Score: 0.8679813053205264
```

Errors

```
#Mean Absolute Error
metrics.mean_absolute_error(y_test, predictions)
#High Error rate
```

```
916.3934560513005
```

```
#Mean Squared Error
metrics.mean_squared_error(y_test, predictions)
```

```
2097264.116568206
```

```
#Root Mean Squared Error (RMSE)
np.sqrt(metrics.mean_squared_error(y_test, predictions))
```

```
1448.1933975019379
```

High Error Rate due to using dummy code to string data into numbers to make the linear regression work.

Cross Validation

```
#k-Fold Cross Validation in Python
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
```

```
#Create the folds
kfold = KFold(3, True, 1)
for train, test in kfold.split(x,y):
    print('train: %s, test: %s' % (train,test))
```

```
train: [    0     1     4 ... 53937 53938 53939], test: [    2     3     8 ... 53932 53934 53935]
train: [    1     2     3 ... 53935 53938 53939], test: [    0     4     6 ... 53933 53936 53937]
train: [    0     2     3 ... 53935 53936 53937], test: [    1     5     9 ... 53931 53938 53939]
```

```
C:\Users\music\anaconda3\lib\site-packages\sklearn\utils\validation.py:70: FutureWarning: Pass shuffle=True, random_state=1
as keyword args. From version 1.0 (renaming of 0.25) passing these as positional arguments will result in an error
  warnings.warn(f"Pass {args_msg} as keyword args. From version "
```

```
print(cross_val_score(lm, x,y, cv=3))
# The percentages vary between -93% to 71% using data sets for testing scores
```

```
[ 0.05105015  0.71683548 -0.92644159]
```

Cross Validation

1. Import packages
2. Create the folds for training and test sets.
3. Print each set of train and test data and average together
    a. The percentages vary between -93% to 71% using data sets for testing scores for the models