Problem Statement:

"Predicting the price of Wine with the Keras Functional API and Tensorflow"

- Building a wide and deep network using Keras (tf.Keras) to predict the price of wine from its description.
- Dataset available at https://github.com/EshitaNandy/Prediction-using-Keras-and-Tensorflow/ (https://github.com/EshitaNandy/Prediction-using-Keras-and-Tensorflow/)

- The overall goal is to create a model that can identify the variety, winery and location of a wine based on a description.
- Prerequisites : Jupyter Notebook, Pandas, Numpy, Scikitlearn and Keras (Tensorflow)

Here are all the imports that we will require to build this model.

In [1]:

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function
```

In [2]:

```
import itertools
import os
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow import keras
layers = keras.layers
```

In [3]:

```
## This code is for testing purpose of proper Tensorflow Execution
print("You have an IT job", tf.__version__)
```

```
You have an IT job 2.1.0
```

In [4]:

```
data = pd.read_csv(r"C:\Users\User\Downloads\KERAS PYTHON\winemag-data-130k-v2.csv")
data.head()
```

Out[4]:

	Unnamed: 0	country	description	designation	points	price	province	region_1	region_
0	0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaI
1	1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaI
2	2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willamett Valle
3	3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaI
4	4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willamett Valle

Do some preprocessing to remove null values and drop unnecessary columns:

In [5]:

```
data = data[pd.notnull(data['country'])]
data = data[pd.notnull(data['price'])]
data = data.drop(data.columns[0], axis =1)
data.head()
```

Out[5]:

	country	description	designation	points	price	province	region_1	region_2	taster_na
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaN	Roger V
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willamette Valley	Paul Gre
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaN	Alexan Peart
4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willamette Valley	Paul Gre
5	Spain	Blackberry and raspberry aromas show a typical...	Ars In Vitro	87	15.0	Northern Spain	Navarra	NaN	Mich Schach

Do some preprocessing to limit wine varieties in the dataset

In [6]:

```
variety_threshold = 500 ## Anything that occurs less than this will be removed
value_counts = data['variety'].value_counts()
to_remove = value_counts[value_counts <= variety_threshold].index
data.replace(to_remove,np.nan, inplace=True)
data = data[pd.notnull(data['variety'])]
data.head()
```

Out[6]:

	country	description	designation	points	price	province	region_1	region_2	taster_na
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaN	Roger V
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willamette Valley	Paul Gre
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaN	Alexan Peart
4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willamette Valley	Paul Gre
7	France	This dry and restrained wine offers spice in p...	NaN	87	24.0	Alsace	Alsace	NaN	Roger V

Split the Data into Training Dataset and Testing Dataset

In [7]:

```
train_size = int(len(data)* 0.8)
print("Train size: %d" % train_size)
print("Train size: %d" % (len(data) - train_size))
```

```
Train size: 82431
Train size: 20608
```

Training and Testing on Features and Labels

In [8]:

```
#Train Features
description_train = data['description'][:train_size]
variety_train = data['variety'][:train_size]

#Train Labels
labels_train = data['price'][:train_size]

#Test Features
description_test = data['description'][train_size:]
variety_test = data['variety'][train_size:]

#Test Labels
labels_test = data['price'][train_size:]
```

Create a tokenizer to preprocess our text description.

In [9]:

```
vocab_size = 12000 # this is a hyperparameter
tokenize = keras.preprocessing.text.Tokenizer(num_words=vocab_size, char_level=False)
tokenize.fit_on_texts(description_train) # only fit on train
```

Wide Feature 1: sparse bag of words(bow) vocab_size vector

In [10]:

```
description_bow_train = tokenize.texts_to_matrix(description_train)
description_bow_test = tokenize.texts_to_matrix(description_test)
```

Wide Feature 2: one-hot vector of variety categories

- Use sklearn utility to convert label strings to numbered index

In [11]:

```
import sklearn
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
encoder.fit(variety_train)
variety_train = encoder.transform(variety_train)
variety_test = encoder.transform(variety_test)
num_classes = np.max(variety_train)+1
```

- Convert labels to one hot

In [12]:

```
variety_train = keras.utils.to_categorical(variety_train, num_classes)
variety_test = keras.utils.to_categorical(variety_test, num_classes)
```

Defining our wide model with the functional API

In [13]:

```
bow_inputs = layers.Input(shape=(vocab_size,))
variety_inputs = layers.Input(shape=(num_classes,))
merged_layer = layers.concatenate([bow_inputs,variety_inputs])
merged_layer = layers.Dense(256, activation='relu')(merged_layer)
predictions = layers.Dense(1)(merged_layer)
wide_model = keras.Model(inputs=[bow_inputs, variety_inputs], outputs=predictions)
```

In [14]:

```
wide_model.compile(loss='mse', optimizer='adam', metrics=['accuracy'])
print(wide_model.summary())
```

```
Model: "model"
________________________________________________________________________
__________________________
Layer (type)                    Output Shape         Param #     Connect
ed to
========================================================================
==========================
input_1 (InputLayer)            [(None, 12000)]      0
________________________________________________________________________
__________________________
input_2 (InputLayer)            [(None, 38)]         0
________________________________________________________________________
__________________________
concatenate (Concatenate)       (None, 12038)        0           input_1
[0][0]
                                                                 input_2
[0][0]
________________________________________________________________________
__________________________
dense (Dense)                   (None, 256)          3081984     concate
nate[0][0]
________________________________________________________________________
__________________________
dense_1 (Dense)                 (None, 1)            257         dense
[0][0]
========================================================================
==========================
Total params: 3,082,241
Trainable params: 3,082,241
Non-trainable params: 0
________________________________________________________________________
__________________________
None
```

Deep Model Feature : Word Embeddings of Wine Description.

In [15]:

```
train_embed = tokenize.texts_to_sequences(description_train)
test_embed = tokenize.texts_to_sequences(description_test)
max_seq_length = 170
train_embed = keras.preprocessing.sequence.pad_sequences(
    train_embed, maxlen=max_seq_length, padding="post")
test_embed = keras.preprocessing.sequence.pad_sequences(
    test_embed, maxlen=max_seq_length, padding="post")
```

Define our Deep Model with the Functional API

In [16]:

```
deep_inputs = layers.Input(shape=(max_seq_length,))
embedding = layers.Embedding(vocab_size, 8, input_length=max_seq_length)(deep_inputs)
embedding = layers.Flatten()(embedding)
embed_out = layers.Dense(1)(embedding)
deep_model = keras.Model(inputs=deep_inputs, outputs = embed_out)
print(deep_model.summary())
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_3 (InputLayer)         [(None, 170)]             0         
_________________________________________________________________
embedding (Embedding)        (None, 170, 8)            96000     
_________________________________________________________________
flatten (Flatten)            (None, 1360)              0         
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 1361      
=================================================================
Total params: 97,361
Trainable params: 97,361
Non-trainable params: 0
_________________________________________________________________
None
```

In [17]:

```
deep_model.compile(loss='mse', optimizer='adam', metrics=['accuracy'])
```

Combine wide and deep into one model

In [18]:

```
merged_out = layers.concatenate([wide_model.output, deep_model.output])
merged_out = layers.Dense(1)(merged_out)
combined_model = keras.Model(wide_model.input + [deep_model.input], merged_out)
combined_model.compile(loss='mse',optimizer='adam', metrics=['accuracy'])
print(combined_model.summary())
```

```
Model: "model_2"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 12000)]      0
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, 38)]         0
__________________________________________________________________________________________________
input_3 (InputLayer)            [(None, 170)]        0
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 12038)        0           input_1[0][0]
                                                                 input_2[0][0]
__________________________________________________________________________________________________
embedding (Embedding)           (None, 170, 8)       96000       input_3[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 256)          3081984     concatenate[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 1360)         0           embedding[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 1)            257         dense[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 1)            1361        flatten[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 2)            0           dense_1[0][0]
                                                                 dense_2[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 1)            3           concatenate_1[0][0]
==================================================================================================
Total params: 3,179,605
Trainable params: 3,179,605
Non-trainable params: 0
__________________________________________________________________________________________________
None
```

Run Training

In [19]:

```
# Training
combined_model.fit([description_bow_train, variety_train] + [train_embed], labels_train
, epochs=10, batch_size=128)

# Evaluation
combined_model.evaluate([description_bow_test, variety_test] + [test_embed], labels_tes
t, batch_size=128)
```

```
Train on 82431 samples
Epoch 1/10
82431/82431 [==============================] - 71s 861us/sample - loss: 14
87.0239 - accuracy: 0.0000e+00
Epoch 2/10
82431/82431 [==============================] - 67s 818us/sample - loss: 12
72.5073 - accuracy: 0.0000e+00
Epoch 3/10
82431/82431 [==============================] - 87s 1ms/sample - loss: 116
8.8411 - accuracy: 0.0000e+00
Epoch 4/10
82431/82431 [==============================] - 76s 924us/sample - loss: 10
69.7310 - accuracy: 0.0000e+00
Epoch 5/10
82431/82431 [==============================] - 62s 746us/sample - loss: 96
4.1559 - accuracy: 0.0000e+00
Epoch 6/10
82431/82431 [==============================] - 59s 715us/sample - loss: 85
5.4663 - accuracy: 0.0000e+00- loss: 827.7758 - accura
Epoch 7/10
82431/82431 [==============================] - 58s 707us/sample - loss: 74
8.1084 - accuracy: 0.0000e+00
Epoch 8/10
82431/82431 [==============================] - 60s 730us/sample - loss: 63
9.2863 - accuracy: 0.0000e+00
Epoch 9/10
82431/82431 [==============================] - 51s 614us/sample - loss: 54
0.9703 - accuracy: 0.0000e+00
Epoch 10/10
82431/82431 [==============================] - 53s 640us/sample - loss: 45
4.9728 - accuracy: 0.0000e+00
20608/20608 [==============================] - 35s 2ms/sample - loss: 146
1.2688 - accuracy: 0.0000e+00
```

Out[19]:

```
[1461.2687954162218, 0.0]
```

Generating predictions on our trained model

In [20]:

```
predictions = combined_model.predict([description_bow_test, variety_test] + [test_embed
])
```

Compare predictions with actual value for the first few items in our test dataset

In [21]:

```
num_predictions = 40
diff = 0

for i in range(num_predictions):
    val = predictions[i]
    print(description_test.iloc[i])
    print('Predicted Price: ',val[0], 'Actual Price: ', labels_test.iloc[i], '\n')
    diff += abs(val[0] - labels_test.iloc[i])
```

```
While aromatically subdued, this is a refreshing, easy-drinking white blen
d, with pleasant lemon and apple flavors. A hint of orange peel and a bris
k, lime-accented acidity highlight the midpalate. Drink now.
Predicted Price:  13.683121 Actual Price:  10.0

A new product from the Tommasi family (celebrated for its Amarone and othe
r hearty red wines), this is light and fresh, with aromas of citrus, stone
fruit and white flowers.
Predicted Price:  14.357279 Actual Price:  13.0

From Italian tenor Andrea Bocelli and his family (vintners in Tuscany for
130 years) comes this bright, cheerful sparkler. It shows crisp acidity, e
ase, tonic effervescence and overall harmony.
Predicted Price:  10.050094 Actual Price:  19.0

“Bolle,” which is Italian for “bubbles,” plays up its cheerful and playful
side with bright aromas of citrus, peach, passion fruit and green apple. I
t washes down easily thanks to the wine's natural acidity.
Predicted Price:  7.1370635 Actual Price:  15.0

Pale red in color, this fruity rosé was one of the originals in the curren
t pink Vinho Verde trend. It is still one of the best. It has a lively, ri
pe strawberry note accompanied by a touch of caramel and bright acidity.
Predicted Price:  13.071145 Actual Price:  10.0

This wine smells subtly of hickory and resin but is otherwise neutral on t
he nose. It feels creamy, smooth and woody, with oak leading the flavor pr
ofile, with other notes of melon and honey. This is a plump, oaky, honeyed
style of Chardonnay.
Predicted Price:  23.106493 Actual Price:  14.0

Notes of cedar and violet meld with a ripe, juicy black-fruit flavor in th
is bold, deeply concentrated Cabernet. Softer and riper than the Rezerva,
it has a slightly grapey midpalate. It's full-bodied in style, with a fram
e of lush, furry tannins.
Predicted Price:  43.85795 Actual Price:  9.0

With flavors of buttered toast, vanilla, sweet mango and orange jam flavor
s, this is as rich, ripe and oaky as they come. Finishing with a honeyed a
ccent, it's made to appeal to consumer's who enjoy the popular style.
Predicted Price:  22.854666 Actual Price:  23.0

This is mentholated and spicy, with a brandied cherry note as the dominant
aroma. Tight but not too severe, it would pair well with meat. Its flavors
of raspberry and dry plum are tasty, and the finish hits hard with tightne
ss featuring dry raspberry and cherry skin. A blend of Cabernet Sauvignon,
Cab Franc, Merlot and Malbec.
Predicted Price:  19.661966 Actual Price:  35.0

Cool and green smelling, with briny notes and whiffs of green pepper and s
ea foam, this shows intensity, while the flavors veer toward lime, passion
fruit and green herb. It's nervy and tasty on the finish.
Predicted Price:  11.388911 Actual Price:  16.0

On the nose, there are cider, oak and baked corn notes. The palate is full
of oak and resin flavors that prevent the fruit from emerging immediately.
It tastes of hickory, apple and melon on the palate and finish.
Predicted Price:  24.228235 Actual Price:  22.0

This is elaborate, but too soft for a great Napa Cabernet. That absence of
structure accentuates its ripeness, giving it tastes of candied cherry and
```

```
licorice. On the plus side, it's dry and smooth, with lots of smoky new Fr
ench oak. Drink now.
Predicted Price:  76.11724 Actual Price:  25.0

Although it's not very aromatic, the nose on this wine is easygoing, with
mild peach and apple aromas. It feels fresh and medium bodied, with apple,
peach and melony flavors, the latter perhaps due to the blend's 14% white
Carmenère. It shows mellow oak notes and latent acidity on the finish.
Predicted Price:  21.57949 Actual Price:  10.0

The underlying wine here is fine, crisp and minerally, with green apple fr
uit. Lots of new oak, however, sticks out, in the form of buttered toast a
nd butterscotch.
Predicted Price:  47.59995 Actual Price:  34.0

This may be a little sweet, hot and heavy, but it has such rich flavors of
berry, currant, spice and mocha that it's easy to drink with ribs, chicke
n, burgers or vegetarian lasagna.
Predicted Price:  20.728733 Actual Price:  29.0

You'll find good, ripe and savory flavors of blackberry, currant and spic
e. A touch of oak and a sweet licorice note give this 100% Cabernet a cand
ied appeal. It can't quite escape its rustic mouthfeel, but it's a nice gl
ass of red wine.
Predicted Price:  42.091408 Actual Price:  23.0

This is the first vintage of this wine to include estate fruit, along with
fruit from Klipsun, Bacchus and Red Willow. A complex medley of herb and s
pice aromas accented by purple flowers is followed by bold yet still refin
ed dark-fruit flavors that linger on the finish. It has the tannin to go t
he distance. Best after 2020.
Predicted Price:  62.27015 Actual Price:  60.0

Prune, raisin and gamy black-fruit aromas fold in a note of iodine. This U
co Valley Malbec features tough but manageable tannins in front of a savor
y tasting palate with roasted plum and berry flavors. A note of salt lends
complexity to a rubbery, tannic finish. Drink through 2023.
Predicted Price:  39.831573 Actual Price:  39.0

This is a darkly fruity and spicy wine from a site overlooking the Marin C
oastal range, often shrouded in fog. Planted to five clones, the site prov
ides plenty of cardamom, clove, forest floor and black tea while allowing
for a wildness of plum tart and crusty bread. The oak imprint is light.
Predicted Price:  73.64102 Actual Price:  49.0

Concentrated boysenberry and blackberry flavors give this dry and full-bod
ied wine a lot to like. They melt in the mouth and seem to coat the tongue
with a ripeness that's rich but not sweet. Throw in dark chocolate, black
currant and a creamy in mouthfeel, and it's an outstanding wine.
Predicted Price:  30.293474 Actual Price:  40.0

Velvety and showing a bite of oak, this wine blends grapes from Spring Hil
l Vineyard and Nobles Vineyard, with a small addition from Horseshoe Bend,
a coastal site that contributes earthy salinity. Bright red fruit is layer
ed and succulent, with an underlying richness that contrasts against the s
izzling acidity.
Predicted Price:  44.6724 Actual Price:  35.0

Dark berry, blue flower, brown spice, espresso and balsamic aromas take sh
ape in the glass. The structured palate offers black cherry, clove, coffee
and licorice alongside close-grained tannins that give the finish a youthf
```

```
ul grip. Drink 2019–2028.
Predicted Price:  74.442604 Actual Price:  75.0

This is a terrific Chardonnay at a great price, offering subtle notes of t
oasted grain upfront, followed by bright lemon-lime fruit and crisp acidit
y. It's medium-bodied, so it won't overpower whatever food you serve it wi
th and the laser beam of citrusy acidity on the finish is wrapped in a gau
zy sheath of pleasant nuttiness. Drink now–2020, possibly longer.
Predicted Price:  35.024696 Actual Price:  21.0

Concentrated aromas of blackberry and iodine are moderately complex. A tig
ht layered palate with juicy acidity propels toasty yet bright flavors of 
black cherry and plum towards a finish with juicy acidity and natural spic
y flavors. Drink through 2021.
Predicted Price:  32.23911 Actual Price:  36.0

This full-bodied wine is saturated with ripe and concentrated blackberry a
nd black-grape flavors. A layered and nicely viscous texture clings to the
sides of the mouth and holds all that gorgeous fruit through the finish. I
t's almost all fruit, dark chocolate and a touch of clove, with little obv
ious oak, for a pure and memorable character.
Predicted Price:  31.202011 Actual Price:  32.0

Another good example of how Lake County grows excellent big reds. Deliciou
s, rich, almost chocolaty, this nevertheless dry wine is packed with flavo
r. It's concentrated, very smooth despite massive tannins and just a pleas
ure to taste. Tempting now, but best after 2020.
Predicted Price:  23.758106 Actual Price:  20.0

Black currant stars in this mountain wine, juicy and brushy in big, pillow
y tannin. Still tightly wound, it shows complexity and plenty of natural s
tructure, begging for more time in the bottle to allow for the gunpowder, 
smoky oak and dark, lush black fruit to marry further. It should show best
2020 through 2023.
Predicted Price:  148.95503 Actual Price:  65.0

Fresh apricot, melon and grapefruit abound from nose to finish in this bri
sk yet delectably fruity Riesling. It's dry and lithe in body yet penetrat
ing and juicy. Zesty lime acidity lingers brightly on the finish.
Predicted Price:  34.524307 Actual Price:  34.0

This balanced, rich wine is produced by a family company dating to the 19t
h century. It has tannins, signs of wood aging and layers of black fruits.
Still young, it will develop well into a solid, concentrated wine. Drink f
rom 2020.
Predicted Price:  18.297495 Actual Price:  16.0

Dark and fruity, this juicy wine is packed with black cherry and highlight
ed with tobacco and coffee. The tannins are substantial, and hint at clean
earth. It's big and full through the middle, but tails off a bit in the fi
nish.
Predicted Price:  36.48348 Actual Price:  75.0

This partnership between the Rhône's Michel Chapoutier and his American im
porter seems remarkably consistent. Peppery clove aromas presage a medium-
bodied, silky wine that combines spice with black olives on the midpalate.
It's long, finely textured and espresso-tinged on the finish.
Predicted Price:  45.273273 Actual Price:  60.0

Zesty lime and green melon notes resonate throughout this ethereally light
but scintillating dry Riesling. Dusty mineral and pollen notes lend an ear
```

```
then dimension to crisp, green plum and herb on the palate. It's spine tin
gling and revitalizing yet lingers persistently on the finish.
Predicted Price:  33.287254 Actual Price:  54.0

A lavish oak component works well in this dark, intense wine to enhance th
e concentrated blackberry and black cherry flavors. Full-bodied and richly
tannic in texture, the wine is very showy and delicious.
Predicted Price:  26.239561 Actual Price:  26.0

This gentle, complex white offers lovingly crafted waves of tropical lyche
e and golden-apple minerality, with bites of wet stone, gravel and lemon.
Balanced in acidity and a lightness of being, it's meant for the table or
kitchen, to sip happily while making a meal.
Predicted Price:  19.647402 Actual Price:  30.0

A blend of Merlot (56%) and Cabernet Franc, this wine opens with aromas of
flowers, huckleberry, cardamom and green herbs, along with light barrel ac
cents. The flavors are dense and rich but far from over the top, supported
by chewy tannins and a fruit-filled finish that sails into the distance. B
est after 2020.
Predicted Price:  54.481853 Actual Price:  64.0

It's good to see this Margaret River stalwart return to the States. The 20
15 DJL is a slightly nutty, toasty wine, plump and silky in texture. Pear,
melon, vanilla and citrus notes finish long.
Predicted Price:  33.013363 Actual Price:  20.0

The wine has powerful tannins which are balanced with equally rich fruit.
It is firm, in the Saint-Estèphe style, while never losing sight of a rich
fruity future. The wine, from an estate at the heart of the Pez plateau, w
ill develop well. Drink from 2024.
Predicted Price:  29.655989 Actual Price:  45.0

Siran is in the south of the Margaux appellation just where the gravel out
crops rise above the river. It shows its fine position in this tannic win
e. It has all the black-currant fruit buried for the moment under the stru
cture. It is a serious wine that will repay aging. Drink from 2022.
Predicted Price:  45.5263 Actual Price:  30.0

Lovely from the first sniff on, this is packed with mineral-drenched fruit
flavors. Pear flesh and skin dominate, along with suggestions of cucumber
and white melon. The leesy minerality is a great palate cleanser.
Predicted Price:  26.43218 Actual Price:  20.0

Though labeled simply Willamette Valley, the vineyard is set in the Eola H
ills AVA. Complex aromas meld sassafras, Asian spices, baking spices and d
elicate notes of leaf and stem. The fruit is lightly brambly with an accen
t on raspberry flavors. The wine is well-proportioned throughout.
Predicted Price:  41.512466 Actual Price:  90.0
```

Compare the average difference between Actual price and the Model's Predicted price

In [23]:

```
print('Average Prediction Difference: ', diff/num_predictions)
```

```
Average Prediction Difference:  13.613453662395477
```

So, the average prediction difference is around 13Dollars in every bottle of wine which is really a good value to move ahead with.