와인 가격 예측 모델

1548088 정현정

1. 데이터 전처리 단계
1) 필요한 함수들 import

1-1. 데이터 및 사전 준비 단계

the dataset : predicting the price of wine

1. 필요한 함수들 import하기

```
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt #jupyter notebook에서 그래프 결과물을 바로 볼 수 있다.
import matplotlib
%matplotlib inline
plt.style.use('fivethirtyeight')
plt.rcParams.update({'font.size' : 10})
import itertools

import seaborn as sns

from pylab import rcParams
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from io import StringIO
```

1. 데이터 전처리 단계
2) 데이터 read

2. data 읽어오기

```
tf.logging.set_verbosity(tf.logging.INFO) #Sets the threshold for what messages will be logged.

sess = tf.InteractiveSession()

data = pd.read_csv("winemag-data_first150k.csv") #data set : winemag-data_first150k.csv data
print("shape of the data with all features : ", data.shape)


data.head(5)
```

shape of the data with all features : (150930, 11)

	Unnamed: 0	country	description	designation	points	price	province	region_1	region_2	
0	0	US	This tremendous 100% varietal wine hails from ...	Martha's Vineyard	96	235.0	California	Napa Valley	Napa	Ca Sau
1	1	Spain	Ripe aromas of fig, blackberry and cassis are ...	Carodorum Selección Especial Reserva	96	110.0	Northern Spain	Toro	NaN	T
2	2	US	Mac Watson honors the memory of a wine once ma...	Special Selected Late Harvest	96	90.0	California	Knights Valley	Sonoma	Sau

1. 데이터 전처리 단계
3) Null이 있는 데이터 처리

3. null이 있는 데이터를 처리하기

```
##null값이 있는 모든 행을 삭제 하는 방법
data = data.dropna()
print("shape of the data with all features without null value : ", data.shape)
```

```
shape of the data with all features without null value :  (39241, 11)
```

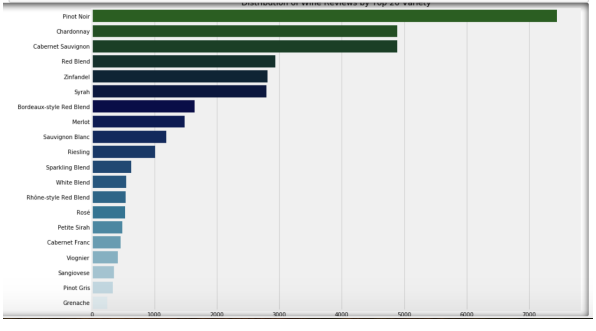
2. 데이터 시각화 단계
1) Variety 중 top20을 막대 그래프로 시각화

2. 데이터 시각화 단계

1. distribution of wine reviews by top 20 variety

Variety: the type of grapes used to make the wine (ie Pinot Noir)

```
print('number of variety list in data : ', data['variety'].nunique())
plt.figure(figsize = (14, 10))
cnt = data['variety'].value_counts().to_frame()[0:20]
#plt.xscale('log')
sns.barplot(x= cnt['variety'], y =cnt.index, data=cnt, palette='ocean',orient='h')
plt.title('Distribution of Wine Reviews by Top 20 Variety');
```



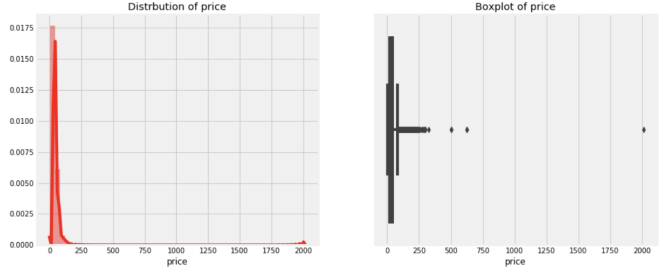
2. 데이터 시각화 단계
2) 와인의 가격 분포를 박스그래프로 시각화

2. Distribution of wine price

```
f, ax = plt.subplots(1,2,figsize=(14,6))
ax1,ax2 = ax.flatten()
sns.distplot(data['price'].fillna(data['price'].mean()),color='r',ax=ax1)
ax1.set_title('Distrbution of price')
sns.boxplot(x = data['price'], ax=ax2)
ax2.set_ylabel('')
ax2.set_title('Boxplot of price')
```

```
/Users/jeonghyeonjeong/anaconda3/lib/python3.6/site-packages/matplotlib/axes/_axes.py:6462: User
Warning: The 'normed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

```
Text(0.5,1,'Boxplot of price')
```

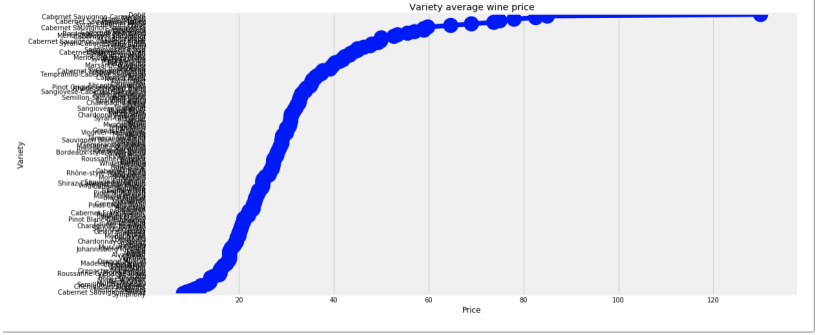


2. 데이터 시각화 단계
3) 와인의 Variety 에 따른 가격 평균 시각화

3. variety wise average wine price

```
cnt = data.groupby(['variety',]).mean()['price'].sort_values(ascending=False).to_frame()

plt.figure(figsize=(16,8))
sns.pointplot(x = cnt['price'] ,y = cnt.index ,color='b',orient='h',markers='o')
plt.title('Variety average wine price')
plt.xlabel('Price')
plt.ylabel('Variety');
```



1. 데이터 전처리 단계
4) 명목형 데이터를 one hot encoding

4. 범주형 명목형 데이터를 one hot encoding

```
import pandas as pd
from pandas import DataFrame, Series
from itertools import cycle
```

```
def dummy_data(data, columns):
    for column in columns:
        data = pd.concat([data, pd.get_dummies(data[column], prefix = column)], axis=1)
        data = data.drop(column, axis=1)
    return data

dummy_colunms = ['country', 'province', 'region_1', 'variety', 'winery']
wine = dummy_data(wine, dummy_colunms)

print('원핫인코딩 후 shape')
print(wine.shape)
```

```
원핫인코딩 후 shape
(23845, 3791)
```

1. 데이터 전처리 단계
4) One hot encoding 결과 조회 및 numpy배열 추출

```
wine_dummy.head(10)
```

	points	price	country_US	province_California	province_New York	province_Oregon	province_Washi
0	96	235.0	1	1	0	0	
2	96	90.0	1	1	0	0	
3	96	65.0	1	0	0	1	
8	95	65.0	1	0	0	1	
9	95	60.0	1	1	0	0	
11	95	48.0	1	0	0	1	
12	95	48.0	1	0	0	1	
14	95	185.0	1	0	0	1	
15	95	90.0	1	0	0	1	
16	95	325.0	1	1	0	0	

10 rows × 3791 columns

```
features = wine.loc[:, 'points' : 'winery_àMaurice']

X = features.values
Y = wine['price'].values
```

1. 데이터 전처리 단계
5) 데이터를 train set과 test set으로 나누기

5. 데이터 train과 test 로 나누기

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state = 0)
```

3. 딥 신경망 모델 구성단계
1) 첫 번째 시도 : logistic regression

3. 신경망 모델 구성단계

1. 첫 번째 시도 : Logistic regression

```
DNNregressor = LogisticRegression(solver = 'liblinear')
DNNregressor.fit(X_train, Y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

```
print("테스트 점수 : {: .2f}".format(DNNregressor.score(X_test, Y_test)))
```

```
테스트 점수 :  0.24
```

3. 딥 신경망 모델 구성단계
2) 두 번째 시도 : keras를 이용하여 Deep Neural Network Regression 모델 만들기

2. 두 번째 시도 : keras을 이용하여 Deep Neural Network Regressor모델 만들기

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

```
import itertools
import os
import math
import numpy as np
import pandas as pd
import tensorflow as tf
import pickle

from sklearn.preprocessing import LabelEncoder
from tensorflow import keras
layers = keras.layers

# This code was tested with TensorFlow v1.7
print("You have TensorFlow version", tf.__version__)
```

3-1. feature 중 descriptions이용

To create a wide representation of our text descriptions we'll use a bag of words model.

참고 사이트 :https://en.wikipedia.org/wiki/Bag-of-words_model

```
vocab_size = 12000
tokenize = keras.preprocessing.text.Tokenizer ( num_words = vocab_size, char_level = False )
tokenize.fit_on_texts (description_train)
```

Then we'll use the texts_to_matrix function to convert each description to a bag of words vector

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

3-2. feature 중 wine variety 이용

```
# Use sklearn utility to convert label strings to numbered index
encoder = LabelEncoder()
encoder.fit(variety_train)
```

```
LabelEncoder()
```

```
variety_train = encoder.transform(variety_train)
```

```
encoder.fit(variety_test)
```

```
LabelEncoder()
```

```
variety_test = encoder.transform(variety_test)
```

```
num_classes = np.max(variety_train) + 1
```

```
# Convert labels to one hot
variety_train = keras.utils.to_categorical(variety_train, num_classes)
variety_test = keras.utils.to_categorical(variety_test, num_classes)
```

```
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)
```

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

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_8 (InputLayer)         (None, 12000)             0         
_________________________________________________________________
input_9 (InputLayer)         (None, 40)                0         
_________________________________________________________________
concatenate_4 (Concatenate)  (None, 12040)             0         
                                                       input_8[0][0]
                                                       input_9[0][0]
_________________________________________________________________
dense_8 (Dense)              (None, 256)               3082496   
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 257       
=================================================================
Total params: 3,082,753
Trainable params: 3,082,753
Non-trainable params: 0
```

```
# Run training
combined_model.fit([description_bow_train, variety_train] + [train_embed], labels_train, epochs=10, batch
```

```
Epoch 1/10
95647/95647 [==============================] - 54s 561us/step - loss: 1293.4685 - acc: 0.0260
Epoch 2/10
95647/95647 [==============================] - 54s 562us/step - loss: 1000.6469 - acc: 0.0339
Epoch 3/10
95647/95647 [==============================] - 53s 557us/step - loss: 873.1452 - acc: 0.0379
Epoch 4/10
95647/95647 [==============================] - 54s 567us/step - loss: 754.5972 - acc: 0.0422
Epoch 5/10
95647/95647 [==============================] - 53s 559us/step - loss: 641.7087 - acc: 0.0448
Epoch 6/10
95647/95647 [==============================] - 54s 566us/step - loss: 534.2462 - acc: 0.0500
Epoch 7/10
95647/95647 [==============================] - 54s 563us/step - loss: 435.9401 - acc: 0.0547
Epoch 8/10
95647/95647 [==============================] - 55s 572us/step - loss: 352.1786 - acc: 0.0603
Epoch 9/10
95647/95647 [==============================] - 55s 573us/step - loss: 284.0254 - acc: 0.0671
Epoch 10/10
95647/95647 [==============================] - 54s 569us/step - loss: 228.9427 - acc: 0.0764

<tensorflow.python.keras.callbacks.History at 0x1c6f132da0>
```

```
#이제 test data를 이용해서 training이 잘 되었는지 확인해본다.
combined_model.evaluate([description_bow_test, variety_test] + [test_embed], labels_test,  batch_size=12
```

```
23912/23912 [==============================] - 6s 265us/step

[518.9948879534204, 0.05842254936007123]
```

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

```
num_predictions = 40
diff = 0

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

```
Predicted:  27.224089 Actual:  18.0

Silky and easygoing on the palate, this comes with slightly sweet flavors of shaved apples, pears and tro
pical fruits.
Predicted:  22.133053 Actual:  19.0

Dark, fruity and common in every way, but sometimes that's all you need. The wine has a clean, ordinar
y cassis and berry nose followed by a juicy palate with dark fruit flavors that show accents of vanilla, spi
ce and chocolate. Totally drinkable and likable.
Predicted:  10.314482 Actual:  8.0
```

```
num_predictions = 40
diff = 0

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

```
Thick and densely extracted, this ripasso has a beautifully tonic and fresh personality with persuasive aromas of dried fruit, black cherry, dark chocolate and spice. In t
he mouth, it offers fruit and crisp flavors backed by smooth tannins and good structure.
Predicted:  33.542866 Actual:  24.0

It's rare to see a Bordeaux-style blend from New Zealand's South Island, but this is a notable exception that can be even better in other vintages. In 2004, it seems a li
ttle light and delicate, with slightly herbal, tomatoey aromas and cherry and herb flavors. The tannins are well-managed, giving it an easy-drinking quality, and you mig
ht even try this with the sorts of fish dishes you'd serve with Pinot Noir.
Predicted:  20.365192 Actual:  15.0

This field blend of 56% Zinfandel, 41% Alicante Bouchet, 2% Burger and 1% Orange Muscat from Red Head Ranch shows smoked pine wood, plums, incense, sandalwo
od and a touch of sour red fruit on the complex nose. There is solid consistency between the nose and the palate, which is powered by cedar along with smoked black
 cherries and mulberries.
Predicted:  48.493927 Actual:  34.0

This bright Barbaresco aged in oak and chestnut barrel shows focused aromas of cherry, wild berry and white almond backed by spice and leather. The intensity of  th
e aromas is not huge, but the wine is compact and neatly focused overall. Drink it with pheasant or game hen.
Predicted:  58.17409 Actual:  53.0

Melony and fleshy as most Central Valley SBs tend to be. There's not much zap, crispness or life blood here, so catch it now for the minimal amount of lemon and gree
```

```
# Compare the average difference between actual price and the model's predicted price
print('Average prediction difference: ', diff/num_predictions)
```

```
Average prediction difference:  1.2301743717130091
```

4. 참고자료

- 와인 데이터 출처 :
- https://www.kaggle.com/zynicide/wine-reviews

- 와인데이터 시각화 참고자료 :
- https://www.kaggle.com/sudhirnl7/wine-recommender

- Keras를 사용한 예측 모델 참고 자료 :
- https://colab.research.google.com/github/sararob/keras-wine-model/blob/master/keras-wide-deep.ipynb#scrollTo=GNzmKJF0OQUf

감사합니다.