Mercari-Sub-II

May 27, 2020

[0]:

```
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import os
import time
from py7zr import unpack_7zarchive
# from pyunpack import Archive
import shutil
import datetime
import math
from contextlib import contextmanager
import scipy
from scipy.sparse import hstack
from sklearn.preprocessing import StandardScaler
from nltk.corpus import stopwords
from tqdm import tqdm
import re
import gc
import pickle
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import KFold

import tensorflow as tf
from tensorflow.keras.layers import Dense,Input
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import LearningRateScheduler
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.models import load_model
```

[0]:

```
os.chdir("/content/drive/My Drive/Kaggle Case Study I")
```

[0]:

```
# Loading the Target Values for Training Data
train_data = pd.read_csv('train/train.tsv',sep = '\t')
train_data = train_data[(train_data.price >= 3) & (train_data.price <= 2000)].reset_index(drop=True)
scaler = StandardScaler()
y_train = scaler.fit_transform(np.log1p(train_data['price'].values.reshape(-1, 1)))

del train_data
gc.collect()
```

[0]:
```
10213
```

[0]:
```
y_train.shape
```

[0]:
```
(1481658, 1)
```

[0]:
```
def build_model(train_shape):
  input_layer = Input(shape=(train_shape,), dtype = 'float32',sparse = True)

  layer1 = Dense(256,activation = "relu",kernel_initializer=tf.keras.initializers.he_uniform(seed = 42))(input_layer)

  layer2 = Dense(64,activation = "relu",kernel_initializer=tf.keras.initializers.he_uniform(seed = 42))(layer1)

  layer3 = Dense(64,activation = "relu",kernel_initializer=tf.keras.initializers.he_uniform(seed = 42))(layer2)

  layer4 = Dense(32,activation = "relu",kernel_initializer=tf.keras.initializers.he_uniform(seed = 42))(layer3)

  output_layer = Dense(1,kernel_initializer=tf.keras.initializers.he_uniform(seed = 42))(layer4)

  model = Model(inputs = input_layer, outputs = output_layer)

  return model
```

[0]:
```
def training_models():
  # Building MLP Models

  # MLP1

  file = open("X_train_tfidf","rb")
  X_train_tfidf = pickle.load(file)
  file.close()
  mlp1 = build_model(X_train_tfidf.shape[1])

  mlp1.compile(optimizer = tf.keras.optimizers.Adam(learning_rate = 0.003),loss␣
↪= "mean_squared_error")

  for i in range(2):
    mlp1.fit(X_train_tfidf,y_train, batch_size= 2**(9 + i), epochs = 1,verbose␣
↪= 1)

  mlp1.save("mlp1.h5")

  # MLP2

  file = open("X_train_binary","rb")
  X_train_binary = pickle.load(file)
  file.close()

  mlp2 = build_model(X_train_binary.shape[1])

  mlp2.compile(optimizer = tf.keras.optimizers.Adam(learning_rate = 0.003),loss␣
↪= "mean_squared_error")

  for i in range(2):
    mlp2.fit(X_train_binary,y_train, batch_size= 2**(9 + i), epochs = 1,verbose␣
↪= 1)

  mlp2.save("mlp2.h5")
```

```
[0]: def predictions():

       file = open("X_test_tfidf","rb")
       X_test_tfidf = pickle.load(file)
       file.close()

       mlp1 = load_model("mlp1.h5")

       y_pred1 = mlp1.predict(X_test_tfidf)[:,0]

       y_pred1 = np.expm1(scaler.inverse_transform(y_pred1.reshape(-1, 1))[:, 0])

       del mlp1, X_test_tfidf
       gc.collect()

       file = open("X_test_binary","rb")
       X_test_binary = pickle.load(file)
```

```
  file.close()

  mlp2 = load_model("mlp2.h5")

  y_pred2 = mlp2.predict(X_test_binary)[:,0]

  y_pred2 = np.expm1(scaler.inverse_transform(y_pred2.reshape(-1, 1))[:, 0])

  del mlp2, X_test_binary
  gc.collect()

  # Generating Emsemble of the above two MLP's

  y_prediction = 0.55 * y_pred1 + 0.45 * y_pred2

  return y_prediction
```

[0]:

```
if __name__ == "__main__":

  training_models()
  print("Training Done.")

  y_prediction = predictions()
  print("Prediction of Ensembles Done")

  # Submitting the Result into a csv file
  test_data = pd.read_csv("test_id.csv")
  submission = pd.DataFrame({'test_id' : test_data.test_id.values,'price' :␣
 ↪y_prediction})
  submission.to_csv("submission.csv", index=False)
  print("Submission Done.")
```

```
2894/2894 [==============================] - 58s 20ms/step - loss: 0.3416
1447/1447 [==============================] - 35s 24ms/step - loss: 0.2025
2894/2894 [==============================] - 58s 20ms/step - loss: 0.3484
1447/1447 [==============================] - 34s 24ms/step - loss: 0.2084
Training Done.
Prediction of Ensembles Done
Submission Done.
```

1 Kaggle Submissions :

[0]:

```
```