Marcari Price Suggestion Challenge

```
)6559701009217455969/06559701009217455969/1IzZW5Ix_i0h4CqPlV7PxtQIFFVUb7Tdp?e=download&authuse
```

```
--2020-09-27 15:36:41--  https://doc-08-0c-docs.googleusercontent.com/docs/securesc/hao
Resolving doc-08-0c-docs.googleusercontent.com (doc-08-0c-docs.googleusercontent.com)..
Connecting to doc-08-0c-docs.googleusercontent.com (doc-08-0c-docs.googleusercontent.cor
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-zip-compressed]
Saving to: ‘mercari-price-suggestion-challenge.zip’

mercari-price-sugge     [              <=> ] 403.16M  62.9MB/s    in 6.4s

2020-09-27 15:36:48 (62.9 MB/s) - ‘mercari-price-suggestion-challenge.zip’ saved [42273
```

```
from google.colab import drive
import pickle
import pandas as pd
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
from sklearn.preprocessing import LabelBinarizer,OneHotEncoder,LabelEncoder
from sklearn.preprocessing import StandardScaler
import scipy
import pickle
from sklearn.metrics import mean_squared_error
from keras.preprocessing.sequence import pad_sequences
import keras as ks
from keras.preprocessing.text import Tokenizer
import tensorflow as tf
from keras.models import model_from_json
from google.colab import files
from sklearn.model_selection import train_test_split
drive.mount('/content/drive',force_remount = True)
```

```
Mounted at /content/drive
```

```
#unzipping data inside colab local
!unzip mercari-price-suggestion-challenge.zip -d /content
```

```
Archive:  mercari-price-suggestion-challenge.zip
  inflating: /content/sample_submission.csv.7z
  inflating: /content/sample_submission_stg2.csv.zip
  inflating: /content/test.tsv.7z
```

```
!apt-get install p7zip-full
!p7zip -d train.tsv.7z
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
p7zip-full is already the newest version (16.02+dfsg-6).
0 upgraded, 0 newly installed, 0 to remove and 11 not upgraded.

7-Zip (a) [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs Intel(R) Xe

Scanning the drive for archives:
1 file, 77912192 bytes (75 MiB)

Extracting archive: train.tsv.7z
--
Path = train.tsv.7z
Type = 7z
Physical Size = 77912192
Headers Size = 122
Method = LZMA2:24
Solid = -
Blocks = 1

Everything is Ok

Size:       337809843
Compressed: 77912192
```

```
#Extracting dataframe /from file and splitting in to train and test for giving inputs to two
data = pd.read_csv("train.tsv",sep = '\t')
train, test = train_test_split(data, random_state=123, train_size=0.99)
```

```
def final_fun_1(X):

  #Inorder to do feature engineering hacks , we have to first remove the missing values assoc

  X['name'] = X['name'].replace([np.nan], ' ')
  X['item_description'] = X['item_description'].replace([np.nan,'No description yet'], ' ')
  X['brand_name'] = X['brand_name'].fillna('missing').astype('category')

  #Splitting Category in to subcategories
  X[['Main_category','subcategory1','subcategory2','subcategory3','subcategory4']] = X['categ
  X['Main_category'] = X['Main_category'].fillna('missing').astype('category')
  X['subcategory1'] = X['subcategory1'].fillna('missing').astype('category')
  X['subcategory2'] = X['subcategory2'].fillna('missing').astype('category')
  X['subcategory3'] = X['subcategory3'].fillna('missing').astype('category')
  X['subcategory4'] = X['subcategory4'].fillna('missing').astype('category')
```

```
#feature engineering
X['description_length'] = X['item_description'].str.len()
with open('/content/drive/My Drive/Colab Notebooks/CS 1/dict1.pickle', 'rb') as handle:
  dict1 = pickle.load(handle)
X['Mean_Brand_Price'] = X['brand_name'].map(dict1)
with open('/content/drive/My Drive/Colab Notebooks/CS 1/dict2.pickle', 'rb') as handle:
  dict2 = pickle.load(handle)
X['Mean_Sub_Price'] = X['subcategory2'].map(dict2)
with open('/content/drive/My Drive/Colab Notebooks/CS 1/dict3.pickle', 'rb') as handle:
  dict3 = pickle.load(handle)
X['Median_Brand_Price'] = X['brand_name'].map(dict3)

#Encoding new numerical features with standard scaling
descr_len = np.array(X['description_length']).reshape(-1,1)
with open('/content/drive/My Drive/Colab Notebooks/CS 1/desc_feats.pickle', 'rb') as handle
  desc_feats = pickle.load(handle)
descfeats = desc_feats.transform(descr_len)
mean_brand_X = np.array(X['Mean_Brand_Price']).reshape(-1,1)
with open('/content/drive/My Drive/Colab Notebooks/CS 1/mean_brand.pickle', 'rb') as handle
  mean_brand = pickle.load(handle)
mean_brand = mean_brand.transform(mean_brand_X)
mean_cat_X = np.array(X['Mean_Sub_Price']).reshape(-1,1)
with open('/content/drive/My Drive/Colab Notebooks/CS 1/mean_cat.pickle', 'rb') as handle:
  mean_cat = pickle.load(handle)
mean_cat = mean_cat.transform(mean_cat_X)
median_brand_X = np.array(X['Median_Brand_Price']).reshape(-1,1)
with open('/content/drive/My Drive/Colab Notebooks/CS 1/median_brand.pickle', 'rb') as hand
  median_brand = pickle.load(handle)
median_brand = median_brand.transform(median_brand_X)

#tokenizing name and description column
raw_text = np.hstack([X.item_description.str.lower(), X.name.str.lower()])
with open('/content/drive/My Drive/Colab Notebooks/CS 1/vec.pickle', 'rb') as handle:
  vec = pickle.load(handle)
X["seq_item_description"] = vec.texts_to_sequences(X.item_description.str.lower())
X["seq_name"] = vec.texts_to_sequences(X.name.str.lower())

#label encoding all the categorical columns
with open('/content/drive/My Drive/Colab Notebooks/CS 1/enc.pickle', 'rb') as handle:
  enc = pickle.load(handle)
X['Main_category'] = enc.transform(X['Main_category'])
with open('/content/drive/My Drive/Colab Notebooks/CS 1/enc1.pickle', 'rb') as handle:
  enc1 = pickle.load(handle)
X['subcategory1'] = enc1.transform(X['subcategory1'])
with open('/content/drive/My Drive/Colab Notebooks/CS 1/enc2.pickle', 'rb') as handle:
  enc2 = pickle.load(handle)
X['subcategory2'] = enc2.transform(X['subcategory2'])
with open('/content/drive/My Drive/Colab Notebooks/CS 1/enc3.pickle', 'rb') as handle:
  enc3 = pickle.load(handle)
X.brand_name = enc3.transform(X.brand_name)
```

```
  #Now concatenate all the numerical features and apply scaling method
  numerical_X=np.concatenate((descfeats,mean_brand,mean_cat,median_brand),axis=1)
  with open('/content/drive/My Drive/Colab Notebooks/CS 1/normal_train.pickle', 'rb') as hand
    normal_train = pickle.load(handle)
  normal_X = normal_train.transform(numerical_X)

  #creatng keras dataset
  def data(dataset):
    X = {
        'name': pad_sequences(dataset.seq_name, maxlen=10)
        ,'item_desc': pad_sequences(dataset.seq_item_description, maxlen=75)
        ,'brand_name': np.array(dataset.brand_name)
        ,'Main_category': np.array(dataset.Main_category)
        ,'subcategory1': np.array(dataset.subcategory1)
        ,'subcategory2': np.array(dataset.subcategory2)
        ,'item_condition': np.array(dataset.item_condition_id)
    }
    return X

  X = data(X)
  X['numerical_features'] = normal_X

  json_file = open('/content/drive/My Drive/Colab Notebooks/CS 1/model.json', 'r')
  loaded_model_json = json_file.read()
  json_file.close()
  loaded_model = model_from_json(loaded_model_json)
  # load weights into new model
  loaded_model.load_weights("/content/drive/My Drive/Colab Notebooks/CS 1/model.h5")
  print("Loaded model from disk")

  optimizer = ks.optimizers.Adam(0.002)
  loaded_model.compile(loss="mse", optimizer=optimizer)

  scores = loaded_model.predict(X)
  return scores


pred = final_fun_1(test)


print("Predicted Prices are",np.exp(pred))
```

```
Predicted Prices are [[32.899418]
 [12.242004]
 [27.820223]
 ...
 [18.08508 ]
 [19.384813]
 [20.318356]]
```

```
def final_fun_2(X,Y):
  #Inorder to do feature engineering hacks , we have to first remove the missing values assoc
```

```
    X['name'] = X['name'].replace([np.nan], ' ')
    X['item_description'] = X['item_description'].replace([np.nan,'No description yet'], ' ')
    X['brand_name'] = X['brand_name'].fillna('missing').astype('category')

    #Splitting Category in to subcategories
    X[['Main_category','subcategory1','subcategory2','subcategory3','subcategory4']] = X['categ
    X['Main_category'] = X['Main_category'].fillna('missing').astype('category')
    X['subcategory1'] = X['subcategory1'].fillna('missing').astype('category')
    X['subcategory2'] = X['subcategory2'].fillna('missing').astype('category')
    X['subcategory3'] = X['subcategory3'].fillna('missing').astype('category')
    X['subcategory4'] = X['subcategory4'].fillna('missing').astype('category')

    #feature engineering
    X['description_length'] = X['item_description'].str.len()
    with open('/content/drive/My Drive/Colab Notebooks/CS 1/dict1.pickle', 'rb') as handle:
      dict1 = pickle.load(handle)
    X['Mean_Brand_Price'] = X['brand_name'].map(dict1)
    with open('/content/drive/My Drive/Colab Notebooks/CS 1/dict2.pickle', 'rb') as handle:
      dict2 = pickle.load(handle)
    X['Mean_Sub_Price'] = X['subcategory2'].map(dict2)
    with open('/content/drive/My Drive/Colab Notebooks/CS 1/dict3.pickle', 'rb') as handle:
      dict3 = pickle.load(handle)
    X['Median_Brand_Price'] = X['brand_name'].map(dict3)

    #Encoding new numerical features with standard scaling
    descr_len = np.array(X['description_length']).reshape(-1,1)
    with open('/content/drive/My Drive/Colab Notebooks/CS 1/desc_feats.pickle', 'rb') as handle
      desc_feats = pickle.load(handle)
    descfeats = desc_feats.transform(descr_len)
    mean_brand_X = np.array(X['Mean_Brand_Price']).reshape(-1,1)
    with open('/content/drive/My Drive/Colab Notebooks/CS 1/mean_brand.pickle', 'rb') as handle
      mean_brand = pickle.load(handle)
    mean_brand = mean_brand.transform(mean_brand_X)
    mean_cat_X = np.array(X['Mean_Sub_Price']).reshape(-1,1)
    with open('/content/drive/My Drive/Colab Notebooks/CS 1/mean_cat.pickle', 'rb') as handle:
      mean_cat = pickle.load(handle)
    mean_cat = mean_cat.transform(mean_cat_X)
    median_brand_X = np.array(X['Median_Brand_Price']).reshape(-1,1)
    with open('/content/drive/My Drive/Colab Notebooks/CS 1/median_brand.pickle', 'rb') as hand
      median_brand = pickle.load(handle)
    median_brand = median_brand.transform(median_brand_X)

    #tokenizing name and description column
    raw_text = np.hstack([X.item_description.str.lower(), X.name.str.lower()])
    with open('/content/drive/My Drive/Colab Notebooks/CS 1/vec.pickle', 'rb') as handle:
      vec = pickle.load(handle)
    X["seq_item_description"] = vec.texts_to_sequences(X.item_description.str.lower())
    X["seq_name"] = vec.texts_to_sequences(X.name.str.lower())

    #label encoding all the categorical columns
    with open('/content/drive/My Drive/Colab Notebooks/CS 1/enc.pickle', 'rb') as handle:
      enc = pickle.load(handle)
      enc = pickle.load(handle)
    X['Main_category'] = enc.transform(X['Main_category'])
    with open('/content/drive/My Drive/Colab Notebooks/CS 1/enc1.pickle', 'rb') as handle:
      enc1 = pickle.load(handle)
    X['subcategory1'] = enc1.transform(X['subcategory1'])
    with open('/content/drive/My Drive/Colab Notebooks/CS 1/enc2.pickle', 'rb') as handle:
      enc2 = pickle.load(handle)
    X['subcategory2'] = enc2.transform(X['subcategory2'])
    with open('/content/drive/My Drive/Colab Notebooks/CS 1/enc3.pickle', 'rb') as handle:
      enc3 = pickle.load(handle)
    X.brand_name = enc3.transform(X.brand_name)

    #Now concatenate all the numerical features and apply scaling method
    numerical_X=np.concatenate((descfeats,mean_brand,mean_cat,median_brand),axis=1)
    with open('/content/drive/My Drive/Colab Notebooks/CS 1/normal_train.pickle', 'rb') as hand
      normal_train = pickle.load(handle)
    normal_X = normal_train.transform(numerical_X)

    #creatng keras dataset
    def data(dataset):
      X = {
          'name': pad_sequences(dataset.seq_name, maxlen=10)
          ,'item_desc': pad_sequences(dataset.seq_item_description, maxlen=75)
          ,'brand_name': np.array(dataset.brand_name)
          ,'Main_category': np.array(dataset.Main_category)
          ,'subcategory1': np.array(dataset.subcategory1)
          ,'subcategory2': np.array(dataset.subcategory2)
          ,'item_condition': np.array(dataset.item_condition_id)
      }
      return X

    X = data(X)
    X['numerical_features'] = normal_X

    json_file = open('/content/drive/My Drive/Colab Notebooks/CS 1/model.json', 'r')
    loaded_model_json = json_file.read()
    json_file.close()
    loaded_model = model_from_json(loaded_model_json)
    # load weights into new model
    loaded_model.load_weights("/content/drive/My Drive/Colab Notebooks/CS 1/model.h5")
    print("Loaded model from disk")

    optimizer = ks.optimizers.Adam(0.002)
    loaded_model.compile(loss="mse", optimizer=optimizer)

    scores = loaded_model.predict(X)
    RMSLE = np.sqrt(mean_squared_error(Y, scores))

    return RMSLE


  Y = np.log(test.price+1)
  RMSLE = final_fun_2(test,Y)
  RMSLE = final_run_2(test,y)


  print('Matric value is',RMSLE)
```

```
Matric value is 0.4369601639579909
```