```
from google.colab import drive
drive.mount('/content/drive')
%cd ./drive/My Drive/LSTM
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=9473

Enter your authorization code:
··********··
Mounted at /content/drive
/content/drive/My Drive/LSTM
```

```
import numpy as np
import pandas as pd
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Input , Dropout
from keras.layers import Flatten
from keras.layers import concatenate
from keras.layers.embeddings import Embedding
from keras.models import Model
from keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from keras.preprocessing.text import Tokenizer
import matplotlib.pyplot as plt
import pickle

import warnings
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer
from sklearn.preprocessing import LabelEncoder
import re
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle
from tqdm import tqdm
import os
```

```
from chart_studio import plotly
import plotly.offline as offline
from keras.layers import LSTM
from keras.preprocessing.text import text_to_word_sequence
import tensorflow as tf
from keras.callbacks import ModelCheckpoint,TensorBoard,ReduceLROnPlateau, EarlyStopping
from keras.layers.normalization import BatchNormalization
from sklearn.feature_extraction.text import TfidfVectorizer
import seaborn as sns
from keras.regularizers import l2
from sklearn.metrics import roc_auc_score
from keras.models import load_model
from IPython.display import Image
from scipy.sparse import hstack
from keras.layers import Conv1D
from sklearn.feature_extraction.text import CountVectorizer
from prettytable import PrettyTable
```

```
Using TensorFlow backend.
```

The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade now or ensure your notebook will continue to use TensorFlow 1.x via the `%tensorflow_version 1.x` magic: more info.

```
X = pd.read_csv('preprocessed_data.csv')
print(X.columns)
X.head(2)
```

```
Index(['school_state', 'teacher_prefix', 'project_grade_category',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay', 'price'],
      dtype='object')
```

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously
0	ca	mrs	grades_prek_2	
1	ut	ms	grades_3_5	

```
Y=X['project_is_approved']
X=X.drop(['project_is_approved'],axis=1)
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.2,stratify=Y)
x_train,x_cv,y_train,y_cv=train_test_split(x_train,y_train,test_size=0.25,stratify=y_train
```

```
x_train.head(2)
```

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

```
(65548, 8) (65548,)
(21850, 8) (21850,)
(21850, 8) (21850,)
```

```
#https://stackoverflow.com/questions/21057621/sklearn-labelencoder-with-never-seen-before-

class LabelEncoderExt(object):
    def __init__(self):
        """
        It differs from LabelEncoder by handling new classes and providing a value for it
        Unknown will be added in fit and transform will take care of new item. It gives un
        """
        self.label_encoder = LabelEncoder()
        # self.classes_ = self.label_encoder.classes_

    def fit(self, data_list):
        """
        This will fit the encoder for all the unique values and introduce unknown value
        :param data_list: A list of string
        :return: self
        """
        self.label_encoder = self.label_encoder.fit(list(data_list) + ['Unknown'])
        self.classes_ = self.label_encoder.classes_

        return self

    def transform(self, data_list):
        """
        This will transform the data_list to id list where the new values get assigned to
        :param data_list:
        :return:
```

```
        """
        new_data_list = list(data_list)
        for unique_item in np.unique(data_list):
            if unique_item not in self.label_encoder.classes_:
                new_data_list = ['Unknown' if x==unique_item else x for x in new_data_list

        return self.label_encoder.transform(new_data_list)
```

```
x_train.columns
```

```
label_encoder = LabelEncoderExt()
label_encoder.fit(x_train['teacher_prefix'].values)
x_train_teacher_ohe=label_encoder.transform(x_train['teacher_prefix'].values)
x_cv_teacher_ohe=label_encoder.transform(x_cv['teacher_prefix'].values)
x_test_teacher_ohe=label_encoder.transform(x_test['teacher_prefix'].values)
```

```
label_encoder = LabelEncoderExt()
label_encoder.fit(x_train['school_state'].values)
x_train_school_ohe=label_encoder.transform(x_train['school_state'].values)
x_cv_school_ohe=label_encoder.transform(x_cv['school_state'].values)
x_test_school_ohe=label_encoder.transform(x_test['school_state'].values)

label_encoder = LabelEncoderExt()
label_encoder.fit(x_train['school_state'].values)
x_train_project_ohe=label_encoder.transform(x_train['project_grade_category'].values)
x_cv_project_ohe=label_encoder.transform(x_cv['project_grade_category'].values)
x_test_project_ohe=label_encoder.transform(x_test['project_grade_category'].values)
```

```
label_encoder = LabelEncoderExt()
label_encoder.fit(x_train['school_state'].values)
x_train_clean_cat_ohe=label_encoder.transform(x_train['clean_categories'].values)
x_cv_clean_cat_ohe=label_encoder.transform(x_cv['clean_categories'].values)
x_test_clean_cat_ohe=label_encoder.transform(x_test['clean_categories'].values)

label_encoder = LabelEncoderExt()
label_encoder.fit(x_train['school_state'].values)
x_train_clean_subcat_ohe=label_encoder.transform(x_train['clean_subcategories'].values)
x_cv_clean_subcat_ohe=label_encoder.transform(x_cv['clean_subcategories'].values)
x_test_clean_subcat_ohe=label_encoder.transform(x_test['clean_subcategories'].values)
```

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
```

```
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(x_train['teacher_number_of_previously_posted_projects'].values.reshape(1,-1

x_train_teacher_no = normalizer.transform(x_train['teacher_number_of_previously_posted_pro
x_cv_teacher_no = normalizer.transform(x_cv['teacher_number_of_previously_posted_projects'
x_test_teacher_no = normalizer.transform(x_test['teacher_number_of_previously_posted_proje

print("After vectorizations")
print(x_train_teacher_no.shape, y_train.shape)
print(x_cv_teacher_no.shape, y_cv.shape)
print(x_test_teacher_no.shape, y_test.shape)
print("="*100)
```

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(x_train['price'].values.reshape(1,-1))

x_train_price_norm = normalizer.transform(x_train['price'].values.reshape(-1,1))
x_cv_price_norm = normalizer.transform(x_cv['price'].values.reshape(-1,1))
x_test_price_norm = normalizer.transform(x_test['price'].values.reshape(-1,1))

print("After vectorizations")
print(x_train_price_norm.shape, y_train.shape)
print(x_cv_price_norm.shape, y_cv.shape)
print(x_test_price_norm.shape, y_test.shape)
print("="*100)
```

```
remaining_train = np.hstack((x_train_price_norm,x_train_teacher_no))
remaining_cv = np.hstack((x_cv_price_norm,x_cv_teacher_no))
remaining_test = np.hstack((x_test_price_norm,x_test_teacher_no))


max_length=300
#https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/
def padded(encoded_docs):
  max_length = 300
  padded_docs = pad_sequences(encoded_docs, maxlen=max_length, padding='post')
```

```
  return padded_docs
#https://stackoverflow.com/posts/51956230/revisions
t = Tokenizer()
t.fit_on_texts(x_train.essay)
vocab_size = len(t.word_index) + 1
# integer encode the documents
encoded_docs = t.texts_to_sequences(x_train.essay)
essay_padded_train = padded(encoded_docs)
encoded_docs = t.texts_to_sequences(x_cv.essay)
essay_padded_cv = padded(encoded_docs)
encoded_docs = t.texts_to_sequences(x_test.essay)
essay_padded_test = padded(encoded_docs)


with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())

# for train
embedding_matrix= np.zeros((vocab_size, 300))
for word, i in t.word_index.items():
    if word in glove_words:
        embedding_vector = model[word]
        embedding_matrix[i] = embedding_vector
print("embedding matrix shape",embedding_matrix.shape)
```

```
embedding matrix shape (46116, 300)
```

```
y_train = to_categorical(y_train, num_classes=2)
y_cv = to_categorical(y_cv, num_classes=2)
y_test = to_categorical(y_test, num_classes=2)


from tensorboardcolab import *
from keras.regularizers import l2
from keras.layers import LeakyReLU
import keras.backend as K
#K.clear_session()



#auc
def auroc(y_true, y_pred):
    return tf.py_func(roc_auc_score, (y_true, y_pred), tf.double)
```

Model 3

```
x_train.head(2)
```

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer=CountVectorizer()
vectorizer.fit(x_train['school_state'])
x_train_school_ohe=vectorizer.transform(x_train['school_state'])
x_cv_school_ohe=vectorizer.transform(x_cv['school_state'])
x_test_school_ohe=vectorizer.transform(x_test['school_state'])
```

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer=CountVectorizer()
vectorizer.fit(x_train['project_grade_category'])
x_train_project_ohe=vectorizer.transform(x_train['project_grade_category'])
x_cv_project_ohe=vectorizer.transform(x_cv['project_grade_category'])
x_test_project_ohe=vectorizer.transform(x_test['project_grade_category'])
```

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer=CountVectorizer()
vectorizer.fit(x_train['clean_categories'])
x_train_clean_cat_ohe=vectorizer.transform(x_train['clean_categories'])
x_cv_clean_cat_ohe=vectorizer.transform(x_cv['clean_categories'])
x_test_clean_cat_ohe=vectorizer.transform(x_test['clean_categories'])
```

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer=CountVectorizer()
vectorizer.fit(x_train['clean_subcategories'])
x_train_clean_subcat_ohe=vectorizer.transform(x_train['clean_subcategories'])
x_cv_clean_subcat_ohe=vectorizer.transform(x_cv['clean_subcategories'])
x_test_clean_subcat_ohe=vectorizer.transform(x_test['clean_subcategories'])
```

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer=CountVectorizer()
vectorizer.fit(x_train['teacher_prefix'])
x_train_teacher_ohe=vectorizer.transform(x_train['teacher_prefix'])
x_cv_teacher_ohe=vectorizer.transform(x_cv['teacher_prefix'])
x_test_teacher_ohe=vectorizer.transform(x_test['teacher_prefix'])
```

```
import numpy as np
remaining_train = hstack((x_train_school_ohe,x_train_project_ohe,x_train_clean_cat_ohe,x_t
remaining_cv = hstack((x_cv_school_ohe,x_cv_project_ohe,x_cv_clean_cat_ohe,x_cv_clean_subc
remaining_test = hstack((x_test_school_ohe,x_test_project_ohe,x_test_clean_cat_ohe,x_test_
```

```
remaining_train = np.expand_dims(remaining_train,2)
remaining_cv = np.expand_dims(remaining_cv,2)
remaining_test = np.expand_dims(remaining_test,2)


K.clear_session()
essay_input = Input(shape=(300,), name='essay_input')

x = Embedding(vocab_size, 300, weights=[embedding_matrix], input_length=300)(essay_input)
lstm_out = LSTM(100,recurrent_dropout=0.5,return_sequences=True)(x)
flatten_1 = Flatten()(lstm_out)



remaining = Input(shape=(101,1), name='remaining')
x = Conv1D(filters=128, kernel_size = 3, padding='valid', kernel_initializer='he_normal',)
x = Conv1D(filters=64, kernel_size = 3, padding='valid', kernel_initializer='he_normal',)(
flatten_2 = Flatten()(x)


x = concatenate([flatten_1,flatten_2])

x = Dense(256, activation='tanh',kernel_initializer="glorot_normal",kernel_regularizer=l2(
x = Dropout(.5)(x)
x = Dense(128, activation='relu',kernel_initializer="he_normal",kernel_regularizer=l2(0.00
x = Dropout(.5)(x)


x = Dense(64, activation='relu',kernel_initializer="he_normal",kernel_regularizer=l2(0.001
final_output = Dense(2, activation='softmax')(x)

model = Model(inputs=[essay_input,remaining], outputs=[final_output])
print(model.summary())
```

```
checkpoint_3 = ModelCheckpoint("model_3.h5",
                               monitor="val_auroc",
                               mode="max",
                               save_best_only = True,
                               verbose=1)
earlystop_3 = EarlyStopping(monitor = 'val_auroc',
                            mode="max",
                            min_delta = 0,
                            patience = 3,
                            verbose = 1,)

tensorboard_3 = TensorBoard(log_dir='graph_3', histogram_freq=0, batch_size=256, write_gra

callbacks_3 = [checkpoint_3,earlystop_3,tensorboard_3]


train = [essay_padded_train,remaining_train]
cv=[essay_padded_cv,remaining_cv]
test=[essay_padded_test,remaining_test]


model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=[auroc])
history_3= model.fit(train, y_train, batch_size=128, epochs=10, verbose=1,callbacks=callba
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl

Train on 65548 samples, validate on 21850 samples
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callbacks.py:112

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callbacks.py:112

Epoch 1/10
65548/65548 [==============================] - 300s 5ms/step - loss: 0.8874 - auroc:

Epoch 00001: val_auroc improved from -inf to 0.50771, saving model to model_3.h5
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callbacks.py:126

Epoch 2/10
65548/65548 [==============================] - 273s 4ms/step - loss: 0.6336 - auroc:

Epoch 00002: val_auroc did not improve from 0.50771
Epoch 3/10
65548/65548 [==============================] - 264s 4ms/step - loss: 0.4929 - auroc:

Epoch 00003: val_auroc did not improve from 0.50771
Epoch 4/10
65548/65548 [==============================] - 262s 4ms/step - loss: 0.4120 - auroc:

Epoch 00004: val_auroc did not improve from 0.50771
Epoch 00004: early stopping
```

```
y_pred=model.predict(test)
a=roc_auc_score(y_test,y_pred)
print("Test auc score",a)
```

```
Test auc score 0.7381877087893663
```

```
try:
  # %tensorflow_version only exists in Colab.
  %tensorflow_version 2.x
except Exception:
  pass

# Load the TensorBoard notebook extension
%load_ext tensorboard
```

```
TensorFlow is already loaded. Please restart the runtime to change versions.
```

```
%reload_ext tensorboard
```

```
%tensorboard --logdir graph_3
```

TensorBoard SCALARS GRAPHS

Show data download links

Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing

0.6

Horizontal Axis

STEP RELATIVE WALL

Runs

Write a regex to filter runs

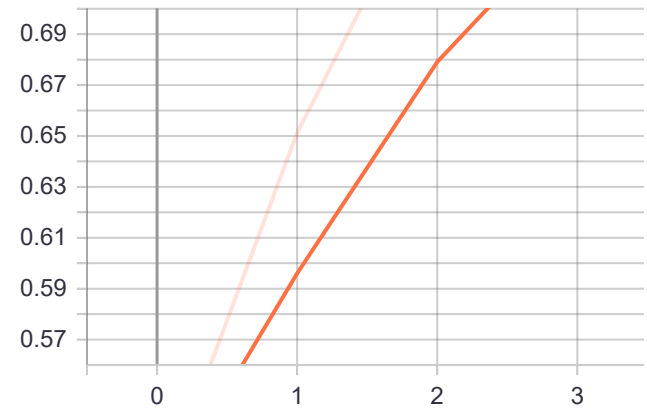
TOGGLE ALL RUNS

graph_3

Filter tags (regular expressions supported)

auroc

auroc



loss

loss