```
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
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
[Errno 2] No such file or directory: './drive/My Drive/LSTM'
/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')
X=X[0:100000]
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_previousl
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)
```

```
#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
```

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

```
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
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 (44576, 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()



# https://github.com/ravi-1654003/LSTM-DonorsChoose/blob/master/LSTM_DonorsChoose.ipynb
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)

state = Input(shape=(1,), name='school_state')
x = Embedding(52, 10, input_length=1)(state)
flatten_2 = Flatten()(x)
```

```
flatten_2 = Flatten()(x)


project_grade_category = Input(shape=(1,), name='project_grade_category')
x = Embedding(5, 10, input_length=1)(project_grade_category)
flatten_3 = Flatten()(x)


clean_categories = Input(shape=(1,), name='clean_categories')
x = Embedding(51, 10, input_length=1)(clean_categories)
flatten_4 = Flatten()(x)


clean_sub_categories = Input(shape=(1,), name='clean_sub_categories')
x = Embedding(393, 10, input_length=1)(clean_sub_categories)
flatten_5 = Flatten()(x)


teacher_prefix = Input(shape=(1,), name='teacher_prefix')
x = Embedding(6, 10, input_length=1)(teacher_prefix)
flatten_6 = Flatten()(x)


remaining_input = Input(shape=(2,), name='remaining_input')
dense_1 = Dense(1, activation='relu',kernel_initializer="he_normal",kernel_regularizer=l2(



x = concatenate([flatten_1,flatten_2,flatten_3,flatten_4,flatten_5,flatten_6,dense_1])

x = Dense(256, activation='relu',kernel_initializer="he_normal",kernel_regularizer=l2(0.00
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,state,project_grade_category,clean_categories,clean_sub_
print(model.summary())
```

```
Model: "model_3"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
essay_input (InputLayer)        (None, 300)          0
__________________________________________________________________________________________________
embedding_13 (Embedding)        (None, 300, 300)     13372800    essay_input[0][0]
__________________________________________________________________________________________________
school_state (InputLayer)       (None, 1)            0
__________________________________________________________________________________________________
project_grade_category (InputLa (None, 1)            0
__________________________________________________________________________________________________
clean_categories (InputLayer)   (None, 1)            0
__________________________________________________________________________________________________
clean_sub_categories (InputLaye (None, 1)            0
__________________________________________________________________________________________________
teacher_prefix (InputLayer)     (None, 1)            0
__________________________________________________________________________________________________
lstm_3 (LSTM)                   (None, 300, 100)     160400      embedding_13[0][0]
__________________________________________________________________________________________________
embedding_14 (Embedding)        (None, 1, 10)        520         school_state[0][0]
__________________________________________________________________________________________________
embedding_15 (Embedding)        (None, 1, 10)        50          project_grade_catego
__________________________________________________________________________________________________
embedding_16 (Embedding)        (None, 1, 10)        510         clean_categories[0][
__________________________________________________________________________________________________
embedding_17 (Embedding)        (None, 1, 10)        3930        clean_sub_categories
__________________________________________________________________________________________________
embedding_18 (Embedding)        (None, 1, 10)        60          teacher_prefix[0][0]
__________________________________________________________________________________________________
remaining_input (InputLayer)    (None, 2)            0
__________________________________________________________________________________________________
flatten_13 (Flatten)            (None, 30000)        0           lstm_3[0][0]
__________________________________________________________________________________________________
flatten_14 (Flatten)            (None, 10)           0           embedding_14[0][0]
__________________________________________________________________________________________________
flatten_15 (Flatten)            (None, 10)           0           embedding_15[0][0]
__________________________________________________________________________________________________
flatten_16 (Flatten)            (None, 10)           0           embedding_16[0][0]
__________________________________________________________________________________________________
flatten_17 (Flatten)            (None, 10)           0           embedding_17[0][0]
__________________________________________________________________________________________________
flatten_18 (Flatten)            (None, 10)           0           embedding_18[0][0]
__________________________________________________________________________________________________
dense_11 (Dense)                (None, 1)            3           remaining_input[0][0
__________________________________________________________________________________________________
concatenate_3 (Concatenate)     (None, 30051)        0           flatten_13[0][0]
                                                                 flatten_14[0][0]
                                                                 flatten_15[0][0]
                                                                 flatten_16[0][0]
                                                                 flatten_17[0][0]
                                                                 flatten_18[0][0]
                                                                 dense_11[0][0]
__________________________________________________________________________________________________
dense_12 (Dense)                (None, 256)          7693312     concatenate_3[0][0]
__________________________________________________________________________________________________
dropout_5 (Dropout)             (None, 256)          0           dense_12[0][0]
__________________________________________________________________________________________________
dense_13 (Dense)                (None, 128)          32896       dropout_5[0][0]
__________________________________________________________________________________________________
dropout_6 (Dropout)             (None, 128)          0           dense_13[0][0]
```

```
________________________________________________________________________________________
dense_14 (Dense)                (None, 64)           8256        dropout_6[0][0]
________________________________________________________________________________________
dense_15 (Dense)                (None, 2)            130         dense_14[0][0]
========================================================================================
Total params: 21,272,867
Trainable params: 21,272,867
Non-trainable params: 0
________________________________________________________________________________________
None
```

```
#https://github.com/ravi-1654003/LSTM-DonorsChoose/blob/master/LSTM_DonorsChoose.ipynb
checkpoint_1 = ModelCheckpoint("model_1.h5",
                             monitor="val_auroc",
                             mode="max",
                             save_best_only = True,
                             verbose=1)
earlystop_1 = EarlyStopping(monitor = 'val_auroc',
                            mode="max",
                            min_delta = 0,
                            patience = 2,
                            verbose = 1)

tensorboard_1 = TensorBoard(log_dir='graph_model_1', batch_size=512)

callbacks_1 = [checkpoint_1,earlystop_1,tensorboard_1]


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


train = [essay_padded_train,x_train_school_ohe,x_train_project_ohe,x_train_clean_cat_ohe,x
cv=[essay_padded_cv,x_cv_school_ohe,x_cv_project_ohe,x_cv_clean_cat_ohe,x_cv_clean_subcat_


import keras
#from keras.optimizers import Adam
#optim=keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)
from keras.optimizers import Adadelta
optim=keras.optimizers.Adadelta(lr=1.0, rho=0.95, epsilon=1e-08, decay=0.0)
#from keras.optimizers import Adagrad
#optim = keras.optimizers.Adagrad(lr=0.01, epsilon=1e-08, decay=0.0)
model.compile(optimizer=optim, loss='categorical_crossentropy', metrics=[auroc])
history_1 = model.fit(train, y_train, batch_size=256, epochs=10, verbose=1,callbacks=callb
     Train on 60000 samples, validate on 20000 samples
     Epoch 1/10
     60000/60000 [==============================] - 125s 2ms/step - loss: 1.6694 - auroc:

     Epoch 00001: val_auroc improved from -inf to 0.70001, saving model to model_1.h5
     Epoch 2/10
     60000/60000 [==============================] - 124s 2ms/step - loss: 0.4976 - auroc:

     Epoch 00002: val_auroc improved from 0.70001 to 0.74172, saving model to model_1.h5
     Epoch 3/10
     60000/60000 [==============================] - 124s 2ms/step - loss: 0.4504 - auroc:
```

```
test=[essay_padded_test,x_test_school_ohe,x_test_project_ohe,x_test_clean_cat_ohe,x_test_c
y_pred=model.predict(test)
a=roc_auc_score(y_test,y_pred)
print("Test auc score",a)
```

```
Test auc score 0.7367849617340079

Epoch 00005: val_auroc did not improve from 0.74296
Epoch 00005: early stopping
```