In [1]:

```
from numpy import array
from numpy import asarray
from numpy import zeros
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Embedding
from keras.layers import Input, Embedding, LSTM, Dense, concatenate, Dropout
from keras.models import Model


import pandas as pd
import numpy as np

import pickle
from tqdm import tqdm
import os

from IPython.core.display import display, HTML
display(HTML("<style>.container { width:100% !important; }</style>"))
```

```
Using TensorFlow backend.
```

In [2]:

```
data=pd.read_csv("fully_processed_data.csv", nrows=1000)
data.head(2)
```

Out[2]:

	Unnamed: 0	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_projects	pro
0	0	in	mrs	grades_prek_2	0	
1	1	fl	mr	grades_6_8	7	

Total Text Data

In [3]:

```
docs_essay=list(data.essay.values)
labels=np.array(data.project_is_approved.values)
```

In [4]:

```
#Prepare tokenizer
tokens = Tokenizer()
tokens.fit_on_texts(docs_essay)
vocab_size = len(tokens.word_index) + 1
#Integer encode the documents
encoded_docs_essay = tokens.texts_to_sequences(docs_essay)
#print(encoded_docs)
print(vocab_size)
```

```
9049
```

In [5]:

```
max_len=0
all_lengths=[]
for sent in docs_essay:
    length=len(sent.split())
    all_lengths.append(length)
print(max(all_lengths))
```

```
303
```

In [6]:

```
# pad documents to a max length of 4 words
max_length = max(all_lengths)
padded_docs_essay = pad_sequences(encoded_docs_essay, maxlen=max_length, padding='post')
print(padded_docs_essay)
```

```
[[   1   84   45 ...    0    0    0]
 [   1 1384    2 ...    0    0    0]
 [ 590 5400    5 ...    0    0    0]
 ...
 [  76  610  115 ...    0    0    0]
 [   1    3  835 ...    0    0    0]
 [1550 5387 1028 ...    0    0    0]]
```

In [7]:

```
#Load the whole embedding into memory
embeddings_index = dict()
file = open('glove.6B.300d.txt')
for line in file:
    values = line.split()
    word = values[0]
    coefs = asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
file.close()

print('Loaded %s word vectors.' % len(embeddings_index))

#Create a weight matrix for words in training docs
embedding_matrix = zeros((vocab_size, 300))
for word, i in tqdm(tokens.word_index.items()):
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector #embedding_matrix.shape: (9049, 300)

print(len(embedding_matrix))
print(len(embedding_matrix[0]))
```

```
100%|██████████| 9048/9048 [00:00<00:00, 270988.64it/s]

Loaded 400000 word vectors.
9049
300
```

In [8]:

```
#Get the flattened LSTM output for input text
input_layer1 = Input(shape=(max_length,))
embedding = Embedding(input_dim=vocab_size, output_dim=300, weights=[embedding_matrix], input_length=max_length,
trainable=False)(input_layer1)
lstm_out = LSTM(32, return_sequences=True)(embedding)
flatten_lstm_out = Flatten()(lstm_out)
```

```
WARNING:tensorflow:From /root/anaconda3/lib/python3.7/site-packages/tensorflow/python/framework/op_d
ef_library.py:263: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be re
moved in a future version.
Instructions for updating:
Colocations handled automatically by placer.
```

Categorical data: school_state

In [9]:

```
docs_school_state=list(data.school_state.values)
```

In [10]:

```
#Prepare tokenizer
tokens = Tokenizer()
tokens.fit_on_texts(docs_school_state)
vocab_size = len(tokens.word_index) + 1
#Integer encode the documents
encoded_docs_school_state = tokens.texts_to_sequences(docs_school_state)
#print(encoded_docs)
print(vocab_size)
```

```
49
```

In [11]:

```
max_len=0
all_lengths=[]
for sent in docs_school_state:
    length=len(sent.split())
    all_lengths.append(length)
print(max(all_lengths))
```

```
1
```

In [48]:

```
# pad documents to a max length of 4 words
max_length = max(all_lengths)
padded_docs_school_state = pad_sequences(encoded_docs_school_state, maxlen=max_length, padding='post')
#print(padded_docs_school_state)
```

In [13]:

```
#Get the flattened LSTM output for input text
input_layer2 = Input(shape=(max_length,))
embedding = Embedding(input_dim=vocab_size, output_dim=5, input_length=max_length, trainable=True)(input_layer2)
flatten_school_state = Flatten()(embedding)
```

Categorical data: project_grade_category

In [14]:

```
docs_project_grade_category=list(data.project_grade_category.values)
```

In [15]:

```
#Prepare tokenizer
tokens = Tokenizer()
tokens.fit_on_texts(docs_project_grade_category)
vocab_size = len(tokens.word_index) + 1
#Integer encode the documents
encoded_docs_project_grade_category = tokens.texts_to_sequences(docs_project_grade_category)
#print(encoded_docs)
print(vocab_size)
```

```
10
```

In [47]:

```
#encoded_docs_project_grade_category
```

In [17]:

```
max_len=0
all_lengths=[]
for sent in encoded_docs_project_grade_category:
    length=len(sent)
    all_lengths.append(length)
print(max(all_lengths))
```

```
3
```

In [18]:

```
# pad documents to a max length of 4 words
max_length = max(all_lengths)
padded_docs_project_grade_category = pad_sequences(encoded_docs_project_grade_category, maxlen=max_length, paddin
g='post')
print(padded_docs_project_grade_category)
```

```
[[1 2 3]
 [1 6 7]
 [1 6 7]
 ...
 [1 2 3]
 [1 2 3]
 [1 2 3]]
```

In [19]:

```
#Get the flattened LSTM output for input text
input_layer3 = Input(shape=(max_length,))
embedding = Embedding(input_dim=vocab_size, output_dim=5, input_length=max_length, trainable=True)(input_layer3)
flatten_project_grade_category = Flatten()(embedding)
```

Categorical data: clean_categories

In [20]:

```
docs_clean_categories=list(data.clean_categories.values)
```

In [21]:

```
#Prepare tokenizer
tokens = Tokenizer()
tokens.fit_on_texts(docs_clean_categories)
vocab_size = len(tokens.word_index) + 1
#Integer encode the documents
encoded_docs_clean_categories = tokens.texts_to_sequences(docs_clean_categories)
#print(encoded_docs)
print(vocab_size)
```

```
16
```

In [46]:

```
#encoded_docs_clean_categories
```

In [23]:

```
max_len=0
all_lengths=[]
for sent in encoded_docs_clean_categories:
    length=len(sent)
    all_lengths.append(length)
print(max(all_lengths))
```

```
5
```

In [24]:

```
# pad documents to a max length of 4 words
max_length = max(all_lengths)
padded_docs_clean_categories = pad_sequences(encoded_docs_clean_categories, maxlen=max_length, padding='post')
print(padded_docs_clean_categories)
```

```
[[ 1  2  0  0  0]
 [11 12  5  6  0]
 [ 5  6  0  0  0]
 ...
 [ 1  2  7  0  0]
 [ 1  2  3  4  0]
 [ 1  2  7  0  0]]
```

In [25]:

```
#Get the flattened LSTM output for input text
input_layer4 = Input(shape=(max_length,))
embedding = Embedding(input_dim=vocab_size, output_dim=5, input_length=max_length, trainable=True)(input_layer4)
flatten_clean_categories = Flatten()(embedding)
```

Categorical data: clean_subcategories

In [26]:

```
docs_clean_subcategories=list(data.clean_subcategories.values)
```

In [27]:

```
#Prepare tokenizer
tokens = Tokenizer()
tokens.fit_on_texts(docs_clean_subcategories)
vocab_size = len(tokens.word_index) + 1
#Integer encode the documents
encoded_docs_clean_subcategories = tokens.texts_to_sequences(docs_clean_subcategories)
#print(encoded_docs)
print(vocab_size)
```

```
38
```

In [45]:

```
#encoded_docs_clean_subcategories
```

In [29]:

```
max_len=0
all_lengths=[]
for sent in encoded_docs_clean_subcategories:
    length=len(sent)
    all_lengths.append(length)
print(max(all_lengths))
```

```
4
```

In [30]:

```
# pad documents to a max length of 4 words
max_length = max(all_lengths)
padded_docs_clean_subcategories = pad_sequences(encoded_docs_clean_subcategories, maxlen=max_length, padding='pos
t')
print(padded_docs_clean_subcategories)
```

```
[[16  1  0  0]
 [30 31 23  0]
 [ 5  8 23  0]
 ...
 [ 1  6  0  0]
 [ 3  4  2  0]
 [16  6  0  0]]
```

In [31]:

```
#Get the flattened LSTM output for input text
input_layer5 = Input(shape=(max_length,))
embedding = Embedding(input_dim=vocab_size, output_dim=5, input_length=max_length, trainable=True)(input_layer5)
flatten_clean_subcategories = Flatten()(embedding)
```

Categorical data: teacher_prefix

In [32]:

```
docs_teacher_prefix=list(data.teacher_prefix.values)
```

In [33]:

```
#Prepare tokenizer
tokens = Tokenizer()
tokens.fit_on_texts(docs_teacher_prefix)
vocab_size = len(tokens.word_index) + 1
#Integer encode the documents
encoded_docs_teacher_prefix = tokens.texts_to_sequences(docs_teacher_prefix)
#print(encoded_docs)
print(vocab_size)
```

```
5
```

In [43]:

```
#encoded_docs_teacher_prefix
```

In [35]:

```
max_len=0
all_lengths=[]
for sent in encoded_docs_teacher_prefix:
    length=len(sent)
    all_lengths.append(length)
print(max(all_lengths))
```

```
1
```

In [44]:

```
# pad documents to a max length of 4 words
max_length = max(all_lengths)
padded_docs_teacher_prefix = pad_sequences(encoded_docs_teacher_prefix, maxlen=max_length, padding='post')
#print(padded_docs_teacher_prefix)
```

In [37]:

```
#Get the flattened LSTM output for input text
input_layer6 = Input(shape=(max_length,))
embedding = Embedding(input_dim=vocab_size, output_dim=5, input_length=max_length, trainable=True)(input_layer6)
flatten_teacher_prefix = Flatten()(embedding)
```

teacher_number_of_previously_posted_projects, nrm_price, presence_of_the_numerical_digits

In [38]:

```
teacher_number_of_previously_posted_projects=list(data.teacher_number_of_previously_posted_projects.values)
presence_of_the_numerical_digits=list(data.presence_of_the_numerical_digits.values)
nrm_price=list(data.nrm_price.values)
```

In [39]:

```
numerical_df=data[['teacher_number_of_previously_posted_projects','presence_of_the_numerical_digits','nrm_price']
]
numerical_df.head(5)
```

Out[39]:

	teacher_number_of_previously_posted_projects	presence_of_the_numerical_digits	nrm_price
0	0	0	0.015397
1	7	0	0.029839
2	1	0	0.051628
3	4	0	0.023228
4	1	0	0.006733

In [40]:

```
#Get the dense layer
input_layer7 = Input(shape=(3,))
dense_layer = Dense(3, activation='relu')(input_layer7)
```

Concatenation of all the layers and building the final model

In [41]:

```
x = concatenate([flatten_lstm_out, flatten_school_state, flatten_project_grade_category, flatten_clean_categories
, flatten_clean_subcategories, flatten_teacher_prefix, dense_layer])
x = Dense(64, activation='relu')(x)
x = Dropout(0.3)(x)
x = Dense(64, activation='relu')(x)
x = Dropout(0.3)(x)
x = Dense(64, activation='relu')(x)

output = Dense(1, activation='sigmoid', name='output')(x)
```

```
WARNING:tensorflow:From /root/anaconda3/lib/python3.7/site-packages/keras/backend/tensorflow_backend
.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will
be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
```

In [52]:

```
model = Model(inputs=[input_layer1,input_layer2,input_layer3,input_layer4,input_layer5,input_layer6,input_layer7]
, outputs=output)
model.compile(optimizer='rmsprop', loss='binary_crossentropy')
model.fit([padded_docs_essay, padded_docs_school_state, padded_docs_project_grade_category, padded_docs_clean_cat
egories, padded_docs_clean_subcategories, padded_docs_teacher_prefix, numerical_df],
          [labels],
          epochs=100, batch_size=32)
```

```
Epoch 1/100
1000/1000 [==============================] - 11s 11ms/step - loss: 0.3830
Epoch 2/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.2698
Epoch 3/100
1000/1000 [==============================] - 11s 11ms/step - loss: 0.1693
Epoch 4/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0750
Epoch 5/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0340
Epoch 6/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0214
Epoch 7/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0088
Epoch 8/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0276
Epoch 9/100
1000/1000 [==============================] - 11s 11ms/step - loss: 0.0037
Epoch 10/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0086
Epoch 11/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0126
Epoch 12/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0154
Epoch 13/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0027
Epoch 14/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0190
Epoch 15/100
1000/1000 [==============================] - 10s 10ms/step - loss: 5.8046e-04
Epoch 16/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0129
Epoch 17/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0136
Epoch 18/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0129
Epoch 19/100
1000/1000 [==============================] - 10s 10ms/step - loss: 4.2130e-04
Epoch 20/100
1000/1000 [==============================] - 10s 10ms/step - loss: 2.0099e-04
Epoch 21/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0039
Epoch 22/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0079
Epoch 23/100
1000/1000 [==============================] - 10s 10ms/step - loss: 1.2012e-04
Epoch 24/100
1000/1000 [==============================] - 10s 10ms/step - loss: 2.0339e-04
Epoch 25/100
1000/1000 [==============================] - 10s 10ms/step - loss: 6.9188e-04
Epoch 26/100
1000/1000 [==============================] - 10s 10ms/step - loss: 1.2809e-04
Epoch 27/100
1000/1000 [==============================] - 10s 10ms/step - loss: 8.4513e-05
Epoch 28/100
```

```
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0098
Epoch 29/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0019
Epoch 30/100
1000/1000 [==============================] - 10s 10ms/step - loss: 3.5896e-05
Epoch 31/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0081
Epoch 32/100
1000/1000 [==============================] - 10s 10ms/step - loss: 3.4467e-04
Epoch 33/100
1000/1000 [==============================] - 10s 10ms/step - loss: 6.0592e-06
Epoch 34/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0039
Epoch 35/100
1000/1000 [==============================] - 10s 10ms/step - loss: 4.5638e-04
Epoch 36/100
1000/1000 [==============================] - 10s 10ms/step - loss: 5.7558e-05
Epoch 37/100
1000/1000 [==============================] - 10s 10ms/step - loss: 2.8163e-05
Epoch 38/100
1000/1000 [==============================] - 10s 10ms/step - loss: 2.8566e-04
Epoch 39/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0056
Epoch 40/100
1000/1000 [==============================] - 10s 10ms/step - loss: 5.1308e-04
Epoch 41/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0064
Epoch 42/100
1000/1000 [==============================] - 10s 10ms/step - loss: 8.7524e-04
Epoch 43/100
1000/1000 [==============================] - 10s 10ms/step - loss: 8.5128e-06
Epoch 44/100
1000/1000 [==============================] - 10s 10ms/step - loss: 3.7393e-06
Epoch 45/100
1000/1000 [==============================] - 10s 10ms/step - loss: 1.7961e-06
Epoch 46/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0122
Epoch 47/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0035
Epoch 48/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0321
Epoch 49/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0090
Epoch 50/100
1000/1000 [==============================] - 10s 10ms/step - loss: 2.2915e-05
Epoch 51/100
1000/1000 [==============================] - 10s 10ms/step - loss: 1.2581e-06
Epoch 52/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0025
Epoch 53/100
1000/1000 [==============================] - 10s 10ms/step - loss: 3.8382e-06
Epoch 54/100
1000/1000 [==============================] - 10s 10ms/step - loss: 2.1238e-04
Epoch 55/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0143
Epoch 56/100
1000/1000 [==============================] - 10s 10ms/step - loss: 3.7219e-05
Epoch 57/100
1000/1000 [==============================] - 10s 10ms/step - loss: 7.7506e-07
Epoch 58/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0048
Epoch 59/100
1000/1000 [==============================] - 10s 10ms/step - loss: 1.7131e-06
Epoch 60/100
1000/1000 [==============================] - 10s 10ms/step - loss: 4.5205e-06
Epoch 61/100
1000/1000 [==============================] - 10s 10ms/step - loss: 7.1023e-07
Epoch 62/100
1000/1000 [==============================] - 10s 10ms/step - loss: 3.4628e-07
Epoch 63/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0018
Epoch 64/100
1000/1000 [==============================] - 10s 10ms/step - loss: 1.5088e-04
Epoch 65/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0162
Epoch 66/100
1000/1000 [==============================] - 10s 10ms/step - loss: 1.8372e-07
Epoch 67/100
1000/1000 [==============================] - 10s 10ms/step - loss: 6.7019e-06
Epoch 68/100
1000/1000 [==============================] - 10s 10ms/step - loss: 2.3022e-06
Epoch 69/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0019
```

```
Epoch 70/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0097
Epoch 71/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0012
Epoch 72/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0021
Epoch 73/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0200
Epoch 74/100
1000/1000 [==============================] - 10s 10ms/step - loss: 1.1456e-04
Epoch 75/100
1000/1000 [==============================] - 10s 10ms/step - loss: 7.3046e-06
Epoch 76/100
1000/1000 [==============================] - 10s 10ms/step - loss: 0.0048
Epoch 77/100
1000/1000 [==============================] - 10s 10ms/step - loss: 6.4467e-05
Epoch 78/100
1000/1000 [==============================] - 10s 10ms/step - loss: 3.6286e-05
Epoch 79/100
1000/1000 [==============================] - 10s 10ms/step - loss: 3.0810e-06
Epoch 80/100
1000/1000 [==============================] - 10s 10ms/step - loss: 4.2039e-04
Epoch 81/100
1000/1000 [==============================] - 10s 10ms/step - loss: 8.3117e-04
Epoch 82/100
1000/1000 [==============================] - 10s 10ms/step - loss: 3.3723e-06
Epoch 83/100
1000/1000 [==============================] - 10s 10ms/step - loss: 1.4977e-05
Epoch 84/100
1000/1000 [==============================] - 10s 10ms/step - loss: 6.7526e-06
Epoch 85/100
1000/1000 [==============================] - 10s 10ms/step - loss: 4.2866e-05
Epoch 86/100
1000/1000 [==============================] - 10s 10ms/step - loss: 7.7695e-04
Epoch 87/100
1000/1000 [==============================] - 10s 10ms/step - loss: 1.7581e-05
Epoch 88/100
1000/1000 [==============================] - 10s 10ms/step - loss: 3.1668e-06
Epoch 89/100
1000/1000 [==============================] - 10s 10ms/step - loss: 1.1971e-07
Epoch 90/100
1000/1000 [==============================] - 10s 10ms/step - loss: 1.2983e-07
Epoch 91/100
1000/1000 [==============================] - 10s 10ms/step - loss: 7.7874e-06
Epoch 92/100
1000/1000 [==============================] - 10s 10ms/step - loss: 2.9007e-07
Epoch 93/100
1000/1000 [==============================] - 10s 10ms/step - loss: 1.8689e-07
Epoch 94/100
1000/1000 [==============================] - 10s 10ms/step - loss: 1.9090e-07
Epoch 95/100
1000/1000 [==============================] - 10s 10ms/step - loss: 2.0246e-07
Epoch 96/100
1000/1000 [==============================] - 10s 10ms/step - loss: 1.2149e-07
Epoch 97/100
1000/1000 [==============================] - 10s 10ms/step - loss: 1.2571e-07
Epoch 98/100
1000/1000 [==============================] - 10s 10ms/step - loss: 2.5450e-07
Epoch 99/100
1000/1000 [==============================] - 10s 10ms/step - loss: 1.1602e-07
Epoch 100/100
1000/1000 [==============================] - 10s 10ms/step - loss: 1.1602e-07
```

Out[52]:

```
<keras.callbacks.History at 0x7fc890358780>
```

In [53]:

```
loss, accuracy = model.evaluate([padded_docs_essay, padded_docs_school_state, padded_docs_project_grade_category,
padded_docs_clean_categories, padded_docs_clean_subcategories, padded_docs_teacher_prefix, numerical_df],
          [labels], verbose=0)
print('Accuracy: %f' % (accuracy*100))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-53-32fdb01b0fa9> in <module>
      1 loss, accuracy = model.evaluate([padded_docs_essay, padded_docs_school_state, padded_docs_pr
oject_grade_category, padded_docs_clean_categories, padded_docs_clean_subcategories, padded_docs_tea
cher_prefix, numerical_df],
----> 2           [labels], verbose=0)
      3 print('Accuracy: %f' % (accuracy*100))

TypeError: cannot unpack non-iterable numpy.float64 object
```

In [49]:

```
import pdfkit
pdfkit.from_file('Test Pad.html', 'Test Pad.pdf')
```

```
Loading page (1/2)
Warning: Failed to load file:///mnt/0AD801EDD801D7B9/Donors Assignment/LSTM-for-Donors/custom.css (i
gnore)
Printing pages (2/2)
Done
```

Out[49]:

```
True
```