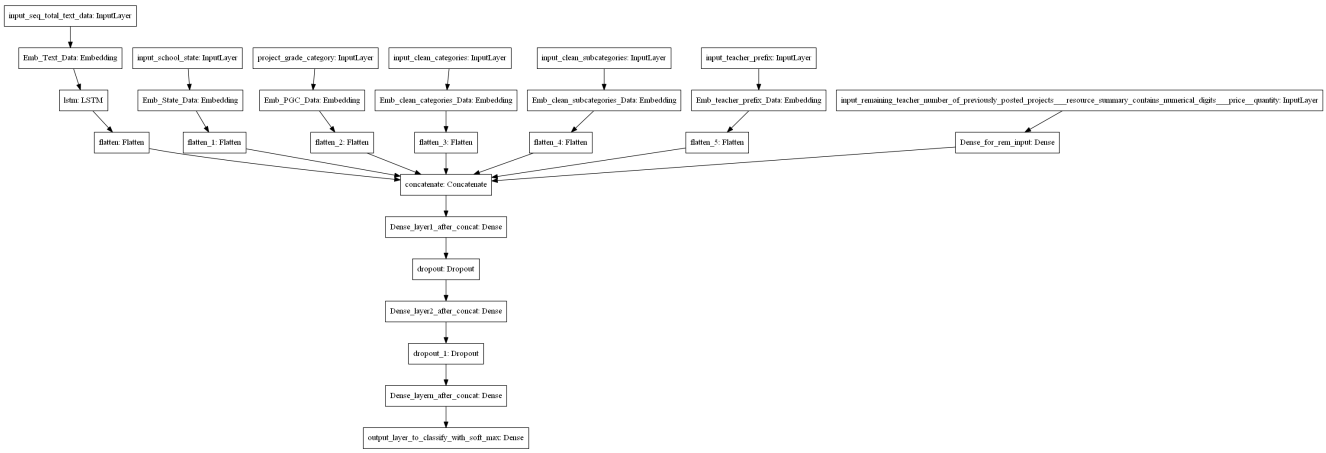
Assignment : 14

1. Preprocess all the Data we have in DonorsChoose Dataset use train.csv
2. Combine 4 essay's into one column named - 'preprocessed_essays'.
3. After step 2 you have to train 3 types of models as discussed below.
4. For all the model use 'auc' as a metric. check this for using auc as a metric
5. You are free to choose any number of layers/hiddden units but you have to use same type of architectures shown below.
6. You can use any one of the optimizers and choice of Learning rate and momentum, resources: cs231n class notes, cs231n class video.
7. For all the model's use TensorBoard and plot the Metric value and Loss with epoch. While submitting, take a screenshot of plots and include those images in .ipynb notebook and PDF.
8. Use Categorical Cross Entropy as Loss to minimize.

Model-1

Build and Train deep neural network as shown below



ref: https://i.imgur.com/w395Yk9.png

- **Input_seq_total_text_data** --- You have to give Total text data columns. After this use the Embedding layer to get word vectors. Use given predefined glove word vectors, don't train any word vectors. After this use LSTM and get the LSTM output and Flatten that output.
- **Input_school_state** --- Give 'school_state' column as input to embedding layer and Train the Keras Embedding layer.
- **Project_grade_category** --- Give 'project_grade_category' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_clean_categories** --- Give 'input_clean_categories' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_clean_subcategories** --- Give 'input_clean_subcategories' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_clean_subcategories** --- Give 'input_teacher_prefix' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_remaining_teacher_number_of_previously_posted_projects._resource_summary_contains_numerical_digits**. ---concatenate remaining columns and add a Dense layer after that.
- For LSTM, you can choose your sequence padding methods on your own or you can train your LSTM without padding, there is no restriction on that.

Below is an example of embedding layer for a categorical columns. In below code all are dummy values, we gave only for referance.

In [0]:

```
# https://stats.stackexchange.com/questions/270546/how-does-keras-embedding-layer-work
input_layer = Input(shape=(n,))
embedding = Embedding(no_1, no_2, input_length=n)(input_layer)
```

```
flatten = Flatten()(embedding)
```

In [1]:

```
%matplotlib inline
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 TfidfVectorizer
from sklearn.model_selection import train_test_split
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 numpy import zeros
from numpy import array
from keras.preprocessing.text import one_hot
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 Input
from keras.layers import Embedding
from keras.layers import LSTM,Bidirectional
from keras.layers.core import Dense, Dropout
from keras.models import Model, load_model
from keras.layers.normalization import BatchNormalization
from keras.callbacks import ReduceLROnPlateau
from keras.preprocessing.text import Tokenizer
from keras.utils import to_categorical

import keras
from tensorboardcolab import *
from keras.regularizers import l2
from keras.layers import LeakyReLU
```

```
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.

In [2]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code

Enter your authorization code:
··········
Mounted at /content/drive
```

In [0]:

```
df = pd.read_csv('/content/drive/My Drive/Applied ML assignments/preprocessed_data.csv')
```

In [4]:

```
resource_data = pd.read_csv('/content/drive/My Drive/LSTM Assignment/resources.csv')
resource_data.columns
project_data = pd.read_csv('/content/drive/My Drive/LSTM Assignment/train_data.csv')
project_data.columns
```

Out[4]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category',
       'project_subject_categories', 'project_subject_subcategories',
       'project_title', 'project_essay_1', 'project_essay_2',
       'project_essay_3', 'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved'],
      dtype='object')
```

In [5]:

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
price_data.head(2)
```

Out[5]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [0]:

```
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [7]:

```
project_data['quantity'].shape
```

Out[7]:

```
(109248,)
```

In [0]:

```
df['quantity'] = project_data['quantity']
#df1['columename']= df2['existing_colume_name']
```

In [9]:

```
df.columns
```

Out[9]:

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

In [0]:

```
#assigning class labels
y=df['project_is_approved']
df.drop(['project_is_approved'],axis=1, inplace=True)
x=df
```

In [85]:

```
#x.columns
#x.drop(['teacher_number_of_previously_posted_projects','price','quantity'],axis=1, inplace=True)
x.columns
#col = ['teacher_prefix', 'school_state', 'project_grade_category',
       #'clean_categories', 'clean_subcategories','essay',
       #'remaining_input']
#x = x[col]
#x.columns
```

Out[85]:

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

In [11]:

```
#Splitting into train and test data
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
#Splitting train data into train and cv(60:20)
X_tr, X_cv, y_tr, y_cv = train_test_split(X_train, y_train, test_size=0.2)
print(X_tr.shape, y_tr.shape)
print(X_cv.shape, y_cv.shape)
```

```
(87398, 9) (87398,)
(21850, 9) (21850,)
(69918, 9) (69918,)
(17480, 9) (17480,)
```

In [0]:

```
#https://stackoverflow.com/questions/21057621/sklearn-labelencoder-with-never-seen-before-values
from sklearn.preprocessing import LabelEncoder
class LabelEncoderExt(object):
    def __init__(self):
        """
        It differs from LabelEncoder by handling new classes and providing a value for it [Unknown]
        Unknown will be added in fit and transform will take care of new item. It gives unknown class i
d
        """
        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 Unknown class
        :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)
```

In [13]:

```
#teacher_prefix
vectorizer = LabelEncoderExt()
vectorizer.fit(X_tr['teacher_prefix'].values)
teacherprefix_ohe_train = vectorizer.transform(X_tr['teacher_prefix'].values)
teacherprefix_ohe_cv = vectorizer.transform(X_cv['teacher_prefix'].values)
teacherprefix_ohe_test = vectorizer.transform(X_test['teacher_prefix'].values)
print(teacherprefix_ohe_cv.shape)
print(teacherprefix_ohe_train.shape)
print(teacherprefix_ohe_test.shape)
```

```
(17480,)
(69918,)
(21850,)
```

In [0]:

```
#Converting categorical features to One hot encoded features
#clean_categories
vectorizer = LabelEncoderExt()
vectorizer.fit(X_tr['clean_categories'].values)
categories_one_hot_train = vectorizer.transform(X_tr['clean_categories'].values)
categories_one_hot_cv = vectorizer.transform(X_cv['clean_categories'].values)
categories_one_hot_test = vectorizer.transform(X_test['clean_categories'].values)

#clean_subcategories
vectorizer = LabelEncoderExt()
vectorizer.fit(X_tr['clean_subcategories'].values)
subcategories_one_hot_train = vectorizer.transform(X_tr['clean_subcategories'].values)
subcategories_one_hot_cv = vectorizer.transform(X_cv['clean_subcategories'].values)
subcategories_one_hot_test = vectorizer.transform(X_test['clean_subcategories'].values)

#school_state
vectorizer = LabelEncoderExt()
vectorizer.fit(X_tr['school_state'].values)
schoolstate_one_hot_train = vectorizer.transform(X_tr['school_state'].values)
schoolstate_one_hot_cv = vectorizer.transform(X_cv['school_state'].values)
schoolstate_one_hot_test = vectorizer.transform(X_test['school_state'].values)


#project_grade_category
vectorizer = LabelEncoderExt()
vectorizer.fit(X_tr['project_grade_category'].values)
project_grade_category_one_hot_train = vectorizer.transform(X_tr['project_grade_category'].values)
project_grade_category_one_hot_cv = vectorizer.transform(X_cv['project_grade_category'].values)
project_grade_category_one_hot_test = vectorizer.transform(X_test['project_grade_category'].values)
```

In [0]:

```
#Concatenating numerical features
rem_input_train = np.concatenate((X_tr['quantity'].values.reshape(-1,1),X_tr['price'].values.reshape(-1
,1),X_tr['teacher_number_of_previously_posted_projects'].values.reshape(-1,1)), axis=1)
rem_input_cv = np.concatenate((X_cv['quantity'].values.reshape(-1,1),X_cv['price'].values.reshape(-1,1)
,X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1,1)), axis=1)
rem_input_test = np.concatenate((X_test['quantity'].values.reshape(-1,1),X_test['price'].values.reshape
(-1,1),X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1,1)), axis=1)
```

In [16]:

```
y_train = to_categorical(y_tr)
y_cv = to_categorical(y_cv)
y_test = to_categorical(y_test)
y_test.shape
```

Out[16]:

```
(21850, 2)
```

In [0]:

```
feature_names[1]
```

In [0]:

```
feature_names[5]
```

In [17]:

```
#Integer encoding Essay column using tokenizer API
#https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/
t = Tokenizer()
t.fit_on_texts(X_tr['essay'])
vocab_size = len(t.word_index) + 1
print("Vocabulary size_train:",vocab_size)
max_length = 400
# integer encode the train data
encoded_docs = t.texts_to_sequences(X_tr['essay'])
essay_pad_train = pad_sequences(encoded_docs, maxlen=max_length, padding='post')
print(essay_pad_train.shape)


# integer encode the cv data
encoded_docs = t.texts_to_sequences(X_cv['essay'])
essay_pad_cv = pad_sequences(encoded_docs, maxlen=max_length, padding='post')
print(essay_pad_cv.shape)


# integer encode the test data
encoded_docs = t.texts_to_sequences(X_test['essay'])
essay_pad_test = pad_sequences(encoded_docs, maxlen=max_length, padding='post')
print(essay_pad_test.shape)
```

```
Vocabulary size_train: 47307
(69918, 400)
(17480, 400)
(21850, 400)
```

In [18]:

```
#Embedding using Glove vectors
embeddings_index = dict()
f = open(r'/content/drive/My Drive/Applied ML assignments/glove.6B.300d.txt')
#with open('/content/drive/My Drive/Applied ML assignments/glove_vectors', 'rb') as f:
  #text = f.read()
for line in f:
  #line.decode(errors='ignore')
  values = line.split()
  word = values[0]
  coefs = np.asarray(values[1:], dtype='float32')
  embeddings_index[word] = coefs
f.close()
print('Loaded %s word vectors.' % len(embeddings_index))
```

```
Loaded 400000 word vectors.
```

In [19]:

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

print("embedding matrix shape",embedding_matrix.shape)
```

```
embedding matrix shape (47307, 300)
```

In [20]:

```
#Flattening the text input data after calculating embedding matrix using glove vectors
```

```
ins = []
concat = []
text_input = Input(shape=(max_length,), name = "text_input")
# max_length = 400 ---->max length of sentence
ins.append(text_input)
e1 = Embedding(vocab_size, 300, weights=[embedding_matrix], input_length=max_length,trainable=False)(te
xt_input)

l1= LSTM(128,kernel_regularizer=l2(0.001),return_sequences=True)(e1)
#l1= LeakyReLU(alpha = 0.3)(l1)
f1= Flatten()(l1)
concat.append(f1)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:541:
The name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:66:
The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4432
: The name tf.random_uniform is deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:190:
The name tf.get_default_session is deprecated. Please use tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:197:
The name tf.ConfigProto is deprecated. Please use tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:203:
The name tf.Session is deprecated. Please use tf.compat.v1.Session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:207:
The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:216:
The name tf.is_variable_initialized is deprecated. Please use tf.compat.v1.is_variable_initialized inst
ead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:223:
The name tf.variables_initializer is deprecated. Please use tf.compat.v1.variables_initializer instead.
```

In [0]:

```
#Combining Categorical features
#https://medium.com/@davidheffernan_99410/an-introduction-to-using-categorical-embeddings-ee686ed7e7f9

cat_vars = ["teacher_prefix","school_state","project_grade_category","clean_categories","clean_subcateg
ories"]

cat_sizes = {}
cat_embsizes = {}
for cat in cat_vars:
    cat_sizes[cat] = X_tr[cat].nunique()#nunique - includes unique elements
    cat_embsizes[cat] = min(50, cat_sizes[cat]//2+1)#embedding size is chosen as half the size of uniqu
e elements + 1
```

In [0]:

```
#Now we iterate over our categorical variables and create an input layer → embedding layer → reshape la
yer
for cat in cat_vars:
    x = Input((1,), name=cat)
    ins.append(x)
    x = Embedding(cat_sizes[cat]+1, cat_embsizes[cat], input_length=1)(x)
    x = Flatten()(x)
    concat.append(x)
```

In [0]:

```
#Converting the remaining input using Dense layer
rem_input_layer =  Input(shape=(3,), name="rem_input_layer")
ins.append(rem_input_layer)
```

```
rem_input_dense = Dense(64, activation='relu')(rem_input_layer)
concat.append(rem_input_dense)
```

In [24]:

```
#After concatenating text input, categorical and remaining numerical features, applying it to the model
from keras.layers import Concatenate
x = Concatenate()(concat)
#X=BatchNormalization()(X)
x= Dense(256,kernel_initializer='glorot_normal',kernel_regularizer=l2(0.002))(x)
#x= LeakyReLU(alpha = 0.3)(x)
x= Dropout(0.6)(x)
x= Dense(128,kernel_initializer='glorot_normal',kernel_regularizer=l2(0.002))(x)
#x= LeakyReLU(alpha = 0.3)(x)
x= Dropout(0.5)(x)
x= Dense(64,kernel_initializer='glorot_normal',kernel_regularizer=l2(0.002))(x)
#x= LeakyReLU(alpha = 0.3)(x)
x= Dropout(0.5)(x)
x= Dense(32,kernel_initializer='glorot_normal',kernel_regularizer=l2(0.002))(x)
#x= LeakyReLU(alpha = 0.3)(x)
x= Dropout(0.5)(x)
#x=BatchNormalization()(x)
x= Dense(16,activation='relu',kernel_initializer='glorot_normal',kernel_regularizer=l2(0.002))(x)
#x= LeakyReLU(alpha = 0.3)(x)
#x= Dropout(0.25)(x)
output=Dense(2, activation='softmax')(x)
model_1 = Model(inputs=ins, outputs=output)
model_1.summary()
#,kernel_initializer='glorot_normal',kernel_regularizer=l2(0.002)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4479
: The name tf.truncated_normal is deprecated. Please use tf.random.truncated_normal instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:148:
The name tf.placeholder_with_default is deprecated. Please use tf.compat.v1.placeholder_with_default in
stead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3733
: 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`.
WARNING:tensorflow:Large dropout rate: 0.6 (>0.5). In TensorFlow 2.x, dropout() uses dropout rate inste
ad of keep_prob. Please ensure that this is intended.
Model: "model_1"
```

Layer (type)	Output Shape	Param #	Connected to
text_input (InputLayer)	(None, 400)	0	
embedding_1 (Embedding)	(None, 400, 300)	14192100	text_input[0][0]
teacher_prefix (InputLayer)	(None, 1)	0	
school_state (InputLayer)	(None, 1)	0	
project_grade_category (InputLa	(None, 1)	0	
clean_categories (InputLayer)	(None, 1)	0	
clean_subcategories (InputLayer	(None, 1)	0	
lstm_1 (LSTM)	(None, 400, 128)	219648	embedding_1[0][0]
embedding_2 (Embedding)	(None, 1, 3)	18	teacher_prefix[0][0]
embedding_3 (Embedding)	(None, 1, 26)	1352	school_state[0][0]
embedding_4 (Embedding)	(None, 1, 3)	15	project_grade_category[0][0]
embedding_5 (Embedding)	(None, 1, 26)	1326	clean_categories[0][0]
embedding_6 (Embedding)	(None, 1, 50)	19550	clean_subcategories[0][0]
rem_input_layer (InputLayer)	(None, 3)	0	

```
flatten_1 (Flatten)             (None, 51200)        0           lstm_1[0][0]
__________________________________________________________________________________________________
flatten_2 (Flatten)             (None, 3)            0           embedding_2[0][0]
__________________________________________________________________________________________________
flatten_3 (Flatten)             (None, 26)           0           embedding_3[0][0]
__________________________________________________________________________________________________
flatten_4 (Flatten)             (None, 3)            0           embedding_4[0][0]
__________________________________________________________________________________________________
flatten_5 (Flatten)             (None, 26)           0           embedding_5[0][0]
__________________________________________________________________________________________________
flatten_6 (Flatten)             (None, 50)           0           embedding_6[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 64)           256         rem_input_layer[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 51372)        0           flatten_1[0][0]
                                                                 flatten_2[0][0]
                                                                 flatten_3[0][0]
                                                                 flatten_4[0][0]
                                                                 flatten_5[0][0]
                                                                 flatten_6[0][0]
                                                                 dense_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 256)          13151488    concatenate_1[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 256)          0           dense_2[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 128)          32896       dropout_1[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 128)          0           dense_3[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 64)           8256        dropout_2[0][0]
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 64)           0           dense_4[0][0]
__________________________________________________________________________________________________
dense_5 (Dense)                 (None, 32)           2080        dropout_3[0][0]
__________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 32)           0           dense_5[0][0]
__________________________________________________________________________________________________
dense_6 (Dense)                 (None, 16)           528         dropout_4[0][0]
__________________________________________________________________________________________________
dense_7 (Dense)                 (None, 2)            34          dense_6[0][0]
==================================================================================================
Total params: 27,629,547
Trainable params: 13,437,447
Non-trainable params: 14,192,100
__________________________________________________________________________________________________
```

In [0]:

```
ins.
```

In [0]:

```
import tensorflow as tf
from sklearn.metrics import roc_auc_score

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

In [26]:

```
import keras
adam = keras.optimizers.Adam(lr=0.001)
model_1.compile(optimizer=adam, loss='categorical_crossentropy',metrics=[auroc])
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793: The name tf.tra
in.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3576
: The name tf.log is deprecated. Please use tf.math.log instead.

WARNING:tensorflow:From <ipython-input-25-4a25250c5bd7>:5: py_func (from tensorflow.python.ops.script_o
```

```
ps) is deprecated and will be removed in a future version.
Instructions for updating:
tf.py_func is deprecated in TF V2. Instead, there are two
    options available in V2.
    - tf.py_function takes a python function which manipulates tf eager
    tensors instead of numpy arrays. It's easy to convert a tf eager tensor to
    an ndarray (just call tensor.numpy()) but having access to eager tensors
    means `tf.py_function`s can use accelerators such as GPUs as well as
    being differentiable using a gradient tape.
    - tf.numpy_function maintains the semantics of the deprecated tf.py_func
    (it is not differentiable, and manipulates numpy arrays). It drops the
    stateful argument making all functions stateful.
```

In [0]:

```
from keras.callbacks import *
es = EarlyStopping(monitor='val_loss', mode='min', patience=10,verbose=1)

batch_size = 512
filepath = '/content/drive/My Drive/Applied ML assignments/Epoch/epochs:{epoch:03d}-val_auc:{val_auroc:.3f}.hdf5'
#earlyStopping = EarlyStopping(monitor='val_loss', patience=10, verbose=0, mode='min')
mcp_save = ModelCheckpoint(filepath, save_best_only=True, monitor='val_auc', mode='max')
reduce_lr_loss = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=1, verbose=1,min_lr=0.001, mode='min')
callbacks=[es, mcp_save, reduce_lr_loss]
#model.fit(Xtr_more, Ytr_more, batch_size=batch_size, epochs=50, verbose=0, callbacks=[earlyStopping, mcp_save, reduce_lr_loss], validation_split=0.25)
```

In [28]:

```
history_1= model_1.fit({'text_input': essay_pad_train, 'school_state': schoolstate_one_hot_train, 'project_grade_category': project_grade_category_one_hot_train,'clean_categories': categories_one_hot_train, 'clean_subcategories':subcategories_one_hot_train, 'teacher_prefix':teacherprefix_ohe_train, 'rem_input_layer':rem_input_train},
          y_train,epochs=20, batch_size=512,verbose=1,
          validation_data=({'text_input': essay_pad_cv, 'school_state': schoolstate_one_hot_cv, 'project_grade_category': project_grade_category_one_hot_cv,'clean_categories': categories_one_hot_cv,'clean_subcategories':subcategories_one_hot_cv, 'teacher_prefix':teacherprefix_ohe_cv, 'rem_input_layer':rem_input_cv},
          y_cv),callbacks=callbacks)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/ops/math_grad.py:1424: where (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
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/tensorflow_backend.py:1033: The name tf.assign_add is deprecated. Please use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1020: The name tf.assign is deprecated. Please use tf.compat.v1.assign instead.

Train on 69918 samples, validate on 17480 samples
Epoch 1/20
69918/69918 [==============================] - 1093s 16ms/step - loss: 3.6809 - auroc: 0.4961 - val_loss: 3.1844 - val_auroc: 0.5000
Epoch 2/20
69918/69918 [==============================] - 1079s 15ms/step - loss: 3.0693 - auroc: 0.4969 - val_loss: 2.9966 - val_auroc: 0.4979
Epoch 3/20
69918/69918 [==============================] - 1081s 15ms/step - loss: 2.9487 - auroc: 0.4902 - val_loss: 2.9057 - val_auroc: 0.4895
Epoch 4/20
69918/69918 [==============================] - 1104s 16ms/step - loss: 2.8859 - auroc: 0.4809 - val_loss: 2.8666 - val_auroc: 0.4806
Epoch 5/20
69918/69918 [==============================] - 1080s 15ms/step - loss: 2.5507 - auroc: 0.4643 - val_loss: 0.9984 - val_auroc: 0.4263
Epoch 6/20
69918/69918 [==============================] - 1076s 15ms/step - loss: 0.8215 - auroc: 0.4935 - val_loss: 0.7143 - val_auroc: 0.5951
Epoch 7/20
```

```
69918/69918 [==============================] - 1144s 16ms/step - loss: 0.7151 - auroc: 0.5242 - val_los
s: 0.6793 - val_auroc: 0.5762
Epoch 8/20
69918/69918 [==============================] - 1088s 16ms/step - loss: 0.6664 - auroc: 0.5741 - val_los
s: 0.6411 - val_auroc: 0.6230
Epoch 9/20
69918/69918 [==============================] - 1089s 16ms/step - loss: 0.6234 - auroc: 0.6225 - val_los
s: 0.6082 - val_auroc: 0.6855
Epoch 10/20
69918/69918 [==============================] - 1099s 16ms/step - loss: 0.5835 - auroc: 0.6848 - val_los
s: 0.5623 - val_auroc: 0.7130
Epoch 11/20
69918/69918 [==============================] - 1096s 16ms/step - loss: 0.6467 - auroc: 0.6089 - val_los
s: 0.5537 - val_auroc: 0.7034
Epoch 12/20
69918/69918 [==============================] - 1097s 16ms/step - loss: 0.5459 - auroc: 0.6901 - val_los
s: 0.5470 - val_auroc: 0.7062
Epoch 13/20
69918/69918 [==============================] - 1050s 15ms/step - loss: 0.5167 - auroc: 0.7101 - val_los
s: 0.5164 - val_auroc: 0.7162
Epoch 14/20
69918/69918 [==============================] - 1053s 15ms/step - loss: 0.4984 - auroc: 0.7155 - val_los
s: 0.4858 - val_auroc: 0.7256
Epoch 15/20
69918/69918 [==============================] - 1086s 16ms/step - loss: 0.4794 - auroc: 0.7208 - val_los
s: 0.4680 - val_auroc: 0.7249
Epoch 16/20
69918/69918 [==============================] - 1104s 16ms/step - loss: 0.4624 - auroc: 0.7269 - val_los
s: 0.4526 - val_auroc: 0.7300
Epoch 17/20
69918/69918 [==============================] - 1097s 16ms/step - loss: 0.4469 - auroc: 0.7313 - val_los
s: 0.4433 - val_auroc: 0.7290
Epoch 18/20
69918/69918 [==============================] - 1099s 16ms/step - loss: 0.4335 - auroc: 0.7365 - val_los
s: 0.4262 - val_auroc: 0.7357
Epoch 19/20
69918/69918 [==============================] - 1096s 16ms/step - loss: 0.4238 - auroc: 0.7423 - val_los
s: 0.4182 - val_auroc: 0.7385
Epoch 20/20
69918/69918 [==============================] - 1186s 17ms/step - loss: 0.4177 - auroc: 0.7425 - val_los
s: 0.4156 - val_auroc: 0.7374
```

In [29]:

```
custom_objects = {"auroc":auroc}
#from keras.models import load_model
#best_model_2 = load_model('/content/drive/My Drive/Applied ML assignments/Epoch/epochs:011-val_acc:0.6
73.hdf5',custom_objects=custom_objects)
result = model_1.evaluate({'text_input': essay_pad_test, 'school_state': schoolstate_one_hot_test, 'pro
ject_grade_category': project_grade_category_one_hot_test,'clean_categories': categories_one_hot_test,'
clean_subcategories':subcategories_one_hot_test, 'teacher_prefix':teacherprefix_ohe_test, 'rem_input_la
yer':rem_input_test},
          y_test,batch_size=512)
```

```
21850/21850 [==============================] - 119s 5ms/step
```

In [30]:

```
print("{} of test data {}". format(model_1.metrics_names[0],result[0]))
print("{} of test data {}". format(model_1.metrics_names[1],result[1]))
```

```
loss of test data 0.4113458995306246
auroc of test data 0.7382636455252072
```